Accounting and Auditing on AIX 5L

- A comprehensive guide to setting up accounting and auditing on AIX 5L
- Advanced Accounting and Workload Manager integration
- Resource planning and billing techniques

Octavian Lascu
Rajeev Palanki
Sorin Todorescu
Tirapat Ua-arak

Redbooks

ibm.com/redbooks

International Technical Support Organization

Accounting and Auditing on AIX

December 2004

SG24-6396-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (December 2004)

This edition applies to Version 5, Release 3 of AIX (product number 5765-G03).

Contents

Notices vii
Trademarks viii

Preface ix
The team that wrote this redbook. ix
Become a published author x
Comments welcome. xi

Chapter 1. Chapter 1. Introduction 1
1.1 The need for auditing and accounting. 2
1.2 Auditing 2
1.3 Accounting. 3
1.4 Advanced Accounting overview 3

Chapter 2. Auditing on AIX. 5
2.1 General concepts of AIX auditing 6
2.2 Configuring and using auditing 7
2.2.1 Select the auditable events. 7
2.2.2 Collecting information 8
2.3 Setting up auditing. 11
2.3.1 Configuration files 11
2.3.2 Command files 17
2.3.3 Output files 20
2.3.4 Data collection. 21
2.3.5 Starting and stopping auditing. 24
2.4 Recommendations for auditing 26
2.4.1 Using the audit subsystem for a quick security check. 27
2.4.2 Disk space consideration 29
2.4.3 Performance 30
2.4.4 Audit limitations. 30
2.5 Configuration examples. 31
2.5.1 A real-time modification monitor 31
2.5.2 More audit scenarios. 32
2.6 Common mistakes. 39

Chapter 3. Accounting on AIX 41
3.1 General concepts about accounting 42
3.2 Quick setup of the accounting subsystem. 43
3.2.1 Starting the accounting system. 44

3.2.2 Stopping the accounting subsystem 49
3.2.3 Long login user name support in AIX 5.3 49
3.3 Accounting internals 50
3.4 Collecting and reporting data 52
3.4.1 Collecting data 52
3.4.2 Reporting data. 66
3.5 Observing the system 91
3.5.1 The system activity 91
3.5.2 Connect-time usage 103
3.5.3 Who is connected to the system 103
3.5.4 CPU usage 105
3.5.5 Disk usage 106
3.5.6 Printer usage. 109
3.6 Troubleshooting potential accounting errors 110
3.6.1 Fixing tacct errors 110
3.6.2 Fixing wtmp errors. 111
3.6.3 Fixing incorrect file permissions 112
3.6.4 Fixing qacct access file errors 113
3.6.5 Fixing runacct errors 113
3.6.6 Updating an out-of-date holidays file. 113
3.6.7 Fixing date change errors 114
3.6.8 Restarting the runacct command 115
3.6.9 Recommendations 115
3.7 Accounting files 116
3.7.1 Accounting commands 117
3.7.2 Accounting data files 119
3.7.3 Report and summary files 120
3.7.4 Accounting file formats 124

Chapter 4. Accounting and the Workload Manager 125
4.1 Overview 126
4.2 WLM concepts. 126
4.2.1 Definitions 126
4.2.2 Class hierarchy 127
4.2.3 Class attributes 128
4.3 Administering WLM 129
4.3.1 WLM configuration: the six-step process 130
4.3.2 WLM administration tools 130
4.3.3 Setting up WLM. 131
4.3.4 Introduction to WLM commands and WebSM. 138
4.4 WLM performance monitoring tools 141
4.5 WLM accounting 145
4.5.1 Process accounting using WLM classes 146

4.5.2 Displaying WLM class accounting information 147
4.5.3 WLM application programming interface (API) 148
4.6 Resource control using WLM 148

Chapter 5. Advanced Accounting 151
5.1 Managing Advanced Accounting 152
5.1.1 Controlling the advanced accounting 152
5.1.2 Using SMIT to control Advanced Accounting 154
5.2 Accounting data file and e-mail notification 155
5.2.1 Accounting data file 155
5.2.2 E-mail notification 157
5.3 Interval accounting 157
5.3.1 System interval 158
5.3.2 Process interval 158
5.4 Transaction accounting 159
5.5 Data aggregation 160
5.5.1 System-level data aggregation 160
5.5.2 Project-level data aggregation 161
5.6 Projects and policies 162
5.6.1 Projects 163
5.6.2 Policies 164
5.6.3 Load, unload, and query policies 169
5.6.4 Manual loading of a project 171
5.7 Reporting and analysis 173
5.8 Testing example 177
5.8.1 Using no interval, aggregation, project, and policy 177
5.8.2 System interval on 182
5.8.3 Aggregation for each user ID 184
5.8.4 Loading both the project and the policy 186
5.8.5 Aggregation based on application record 194

Appendix A. Security audit events in AIX 5.3 199

Appendix B. The accounting files 217
The acct file format 218
The tacct file format 219
The utmp file format 219
The ctmp file format 223
The accrec file format 224
The files in the bos.acct package 225

Appendix C. Accounting records in Advanced Accounting 229

Abbreviations and acronyms 239

Related publications 241
IBM Redbooks 241
Other publications 241
Online resources 242
How to get IBM Redbooks 242
Help from IBM 242

Index 243

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

developerWorks®
AIX 5L™
AIX®
DFS™
HACMP™
IBM®
Micro-Partitioning™
POWER5™
pSeries®
PTX®
Redbooks™
Redbooks (logo) ™
RS/6000®

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Preface

Auditing and Accounting on AIX 5L is your comprehensive guide to setting up, maintaining, and troubleshooting the advanced auditing and accounting features on your AIX® 5L™ systems. This IBM® Redbook guides you through the steps to develop, monitor, troubleshoot, and optimize best practices for auditing and accounting in your environment.

In this redbook, you will find an overview of what auditing and accounting can do for you, how to set up an auditing system, procedures for creating the right accounting system for your environment, and a summary of available third-party accounting systems that can plug into the AIX suite.

You will also be able to decide how much accounting and auditing you need to do on your system, how to size the subsystems to handle your requirements, and find a list of general rules to help prevent common mistakes and fix what may have already gone wrong.

This redbook is useful for system administrators, system security officers, companies needing to bill clients for system resource use, and any others looking for a flexible system to monitor system resources.

This redbook also provides information about the new AIX 5L V5.3 feature Advanced Accounting. This new feature is provided in addition to traditional accounting and it is based on mainframe technology such as interval accounting and transaction accounting. Advanced Accounting supports LPAR (logical partitioning), WLM (Workload Management) and Micro-Partitioning™.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Octavian Lascu is a Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of pSeries® clusters and Linux®. Before joining the ITSO, Octavian worked in IBM Global Services Romania as a software and hardware Services Manager. He holds a Master's degree in Electronic Engineering from the Polytechnical Institute in Bucharest and is also an IBM Certified Advanced Technical Expert in AIX/PSSP/HACMP™. He has worked with IBM since 1992.

Rajeev Palanki is a Software Engineer at IBM Software Labs, Bangalore in India. He has been with IBM for four years and holds an engineering degree in Computer Science from Bhopal University, India. He is currently involved with the IBM Java™ Development Kit and has worked extensively on AIX. His areas of expertise include the Java Virtual Machine and AIX system internals. He has co-authored articles about IBM developerWorks® and the Java diagnostics guide.

Sorin Todorescu is a Software Engineer at Alcatel Romania in Timisoara, Romania, where he manages the IT team of the Technical Center Department. He has more than 10 years of experience in the IT field and holds a Master's degree in Electronic Engineering from Politechnical University of Timisoara. His areas of expertise include system administration for UNIX® systems, including AIX, Solaris, and Linux; and performance tuning, problem determination, installation, and maintenance for UNIX machines. He has administered AIX systems since 1993.

Tirapat Ua-arak is a System Services Representative at IBM Thailand. He holds a Bachelor's degree in Electronics Engineering from King Mongkut's Institute of Technology, Thailand. He has 10 years of experience in supporting RS/6000® and pSeries servers. He is also a certified Specialist in pSeries AIX System Support and AIX Certified Advanced Technical Expert.

Thanks to the following people for their contributions to this project:

Scott Vetter
International Technical Support Organization, Austin Center

Betsy Thaggard
International Technical Support Organization, Austin Center

Dino Quintero
IBM Poughkeepsie

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will team with IBM technical professionals, Business Partners, and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us because we want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 003 Internal Zip 2834
  11400 Burnet Road
  Austin, Texas 78758-3493

1

Chapter 1. Introduction

This chapter provides an overview of the auditing and accounting subsystems on AIX. We discuss the need for accounting and auditing in enterprises and the value these activities bring to the business.

We define the concepts of traditional accounting and briefly describe the principles of the auditing subsystem. We also introduce the features provided by advanced accounting and its relevance to the dynamic nature of systems today.

Subsequent chapters provide detailed descriptions of the accounting and auditing features.

1.1 The need for auditing and accounting

With the dynamic nature of business today and the emphasis on efficient management of resources, it is vital for enterprises to effectively manage IT costs.

It is critical for organizations to accurately determine the utilization of computing services and evaluate the costs that are expended for the utilized resources. This is useful in achieving optimal resource allocation and control.

Most of the today's enterprises are driven by the Services Model in the current business scenario, and are billing users of the systems based on the amount of computing services being used.

There is an increasing need for organizations to account for work units (in terms of resource utilization) and evaluate the cost on the basis of work units utilized. The accounting subsystem on AIX provides an effective means to monitor resource utilization and provide accounting data and utilities for charge-back.

It is also imperative for businesses to ensure that security policies for the system are in place and prevent any possible violations of the policy. All information pertinent to security must be recorded in order to detect any potential security breach or threat.

1.2 Auditing

The auditing subsystem on AIX provides system administrators the features to record information pertinent to system security. This information is essential for the system administrators to prevent potential violations of the system security policy.

Any occurrence on the system relevant to system security is considered an auditable event. The set of auditable events on the system defines which occurrences can actually be audited and the granularity of the auditing provided.

The central concept of auditing is to detect any occurrence of an auditable event, record information pertaining to the set of auditable events, and process the information to examine audit trails and generate periodic reports.

An auditing subsystem also provides features to monitor the audit trail in real time for generating alerts to immediate security threats.

1.3 Accounting

The accounting subsystem provides features for monitoring system resource utilization and billing users for the use of resources. Accounting data can be collected on a variety of system resources: processors, memory, disks, and such.

Accounting data provides valuable information to:

- Develop effective charge-back policies.
- Assess the adequacy of the current resources.
- Effectively balance and control resource allocation.
- Forecast future needs.

The accounting subsystem ties up with the AIX Workload Manager (WLM); thus the resource usage per WLM class can be monitored and controlled.

The information in this book should help you understand how to implement the accounting utility in your system:

- Collecting and reporting system data
- Collecting accounting data
- Reporting accounting data
- Accounting commands
- Accounting files

1.4 Advanced Accounting overview

Advanced Accounting incorporates several features that provide the accounting needs for entities that cannot be accounted by the traditional accounting system. The traditional accounting system is based on time-share environments where UNIX processes and users are considered work units. The advanced accounting system provides mechanisms that are viable for dynamic systems that exploit features such as LPARs and application transactions.

AIX 5.3 introduces several new features, including interval accounting, data aggregation, and transactional accounting, which provide a robust, flexible, and scalable accounting environment to the end user.

Advanced Accounting provides usage-based information for a variety of system resources so that you can develop comprehensive charge-back strategies. You can collect accounting data on resources such as disks, network interfaces, virtual devices, file systems, processors, and memory. Interval accounting gives you the ability to view this data over system administrator–defined time intervals to develop chronological views. This has several potential applications, including capacity planning.

2

Auditing on AIX

This chapter describes AIX auditing, the auditing mechanisms, and how to implement accounting procedures for recording security-related information. The following topics are included:

- General concepts of AIX auditing
- Configuring the auditing subsystem
- Auditing scenarios
- Recommendations for auditing
- Common errors

2.1 General concepts of AIX auditing

The auditing system is intended to record security-related information and to alert you about potential or actual violations of system security policy. For example, you can detect who has modified security-sensitive files on the system, learn about unsuccessful **login** or **su** attempts (it could be someone trying to get unapproved root privileges), or create your own security-related triggers. You have to plan whether you want a real-time alert, or to collect data for analyzing it later, or both in the same time.

Also you should consider protecting the stored data (auditing-related) from intruders, or your system may be hacked, possibly without leaving a trace about what happened. Auditing is a measure you have to take in conjuction with other methods to protect your systems from intruders and to trace any intruding action.

The audit subsystem performs the following functions:

- Detecting events
- Collecting event information
- Processing audit records

An auditable event is any change in the security state of the system, a violation of the system access control or security policy, or both. For instance, modifying an authentication file and accessing the system are considered to be auditable events. This kind of event generates an audit record, passed to the kernel logging routines for storage or analysis.

A complete audit record consists of:

- The audit header that contains information common to all events, such as the name of the event and the success or failure of the event.
- The audit tail, that contains event-specific information, such as any additional information relevant to security auditing.

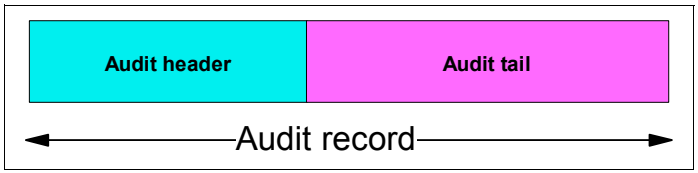

Figure 2-1 The structure of an audit record

The audit logger constructs the audit record and appends it to the kernel audit trail. From there, it can be written in one or both of the following modes:

BIN mode Written in two binary files.
STREAM mode Written synchronously via an audit pseudo-device.

2.2 Configuring and using auditing

For each record, the event logger uses an audit header and an audit tail. The audit header identifies the user and processes to which the event belongs and the time of the event. The structures of audit headers are defined in the /usr/include/sys/audit.h file. The code that detects the event supplies the event type and the return code, or status of that event and, optionally, event-specific information.

Event-specific information consists of object names (files being accessed, failed logon attempts), subroutine parameters, and other modified parameters, which together form the audit tail. If the content of the audit tail is specific to the process, the format of the information of the audit tail is specific to each event. The format of the audit tail is defined in the /etc/security/audit/events file. The events are defined symbolically and may be up to 16 bytes long.

2.2.1 Select the auditable events

We have to select the audit events to be recorded by the event logger. The two types of event auditing, event auditing and object auditing, are described in this section.

Event auditing

The events defined in the system (/etc/security/audit/events file) form the base events class. There are approximately 130 different base events built into AIX. The base events can be grouped, forming an audit class. The audit class names are assigned arbitrarily. The processes belongs to users, so for each user in the system we can define one or more audit classes to be audited for that user. Each process run by that user is tagged with its audit classes. Usually we assign classes to user names rather than to basic events.

Example 2-1 Example of event class and assigning audit class to a user

```
start:
        binmode = on
...

bin:
        trail = /audit/trail
...
stream:
        cmds = /etc/security/audit/streamcmds

classes:
...more classes
```

```
general=USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,FS_Chroot,PORT_Locke
d,PORT_Change,FS_Mkdir,FS_Rmdir
files=FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_Link,FILE_Unlink,FILE_Rename,FILE_Owner,FI
LE_Mode,FILE_Acl,FILE_Privilege,DEV_Create
... more classes
users:
        root = general,files
```

The content of a configuration file is shown in Example 2-1 on page 7. The definition of the class *files*, and the assignment of the classes *general* and *files* to the user `root` is highlighted.

Note: When you assign a class to a user, the assignment becomes effective the next time the user logs on.

Object auditing

In practice, these objects are files. Objects are not associated with users. Three operations can be audited on files: read, write, execute.

Example 2-2 Example of object auditing in the /etc/security/audit/object file

```
/etc/security/passwd:
r = "S_PASSWD_READ"
w = "S_PASSWD_WRITE"
```

Note: Event auditing is *always* associated with user ID. You can audit one user for the *general* class, another user for the *general* and *files* classes. The classes (or just basic events) are associated with users in the /etc/security/audit/config file.

Auditing objects refer to individual files. Objects are not associated with any user ID. For defining the files and events that you want to monitor for each specific object (file), you have to modify the /etc/security/audit/objects file.

2.2.2 Collecting information

We have to decide which kernel logging mode to use:

- BIN mode
- STREAM mode

The BIN mode

The BIN mode records the audit events to alternating two temporary BIN files, then appends them to a single audit trail file as shown in Figure 2-2 on page 9.

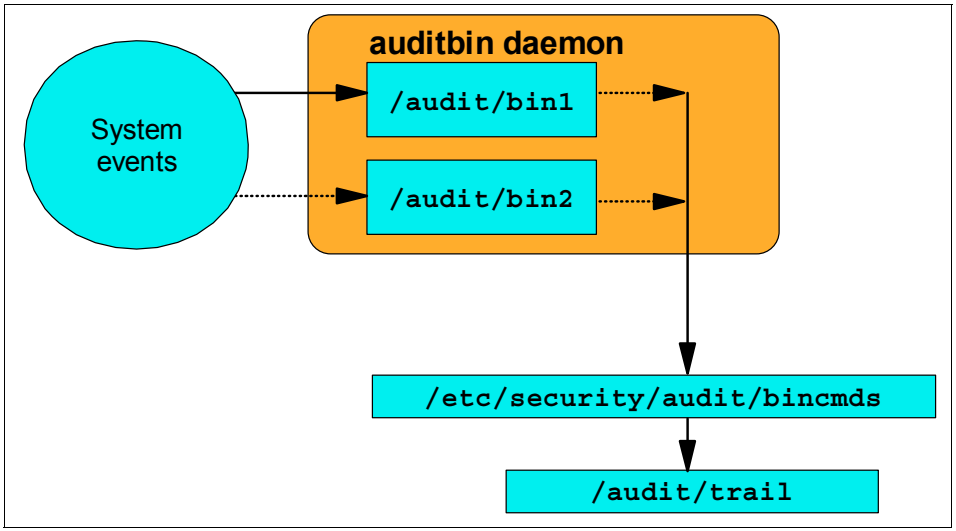

Figure 2-2 Data collection in BIN mode

The alternating BIN mechanism (/audit/bin1 and /audit/bin2) is used to ensure that the audit susbsystem always has something to write to while the audit records are processed. When the audit subsystem switches to the other bin, it empties the first bin content to the /audit/trail file.

When time comes to switch the bin again, the first bin will be available. This mechanism ensures the decoupling of the storage and analysis of the data from the data generation.

Typically, the **`auditcat`** program is used to read the data from the bin that the kernel is not writing to at the moment. To make sure that the system never runs out of space for the audit trail (the output of the **`auditcat`** program), the *freespace* parameter can be specified in the /etc/security/audit/config file. If the system has less than the amount of disk blocks (512-byte) specified here, it generates a syslog message.

To set up the BIN mode:

- Add `binmode = on` in the /etc/security/audit/config file.
- Check the bin stanza of the /etc/security/audit/config file and verify the path, trail, and bin attributes.

The STREAM mode

The STREAM mode writes the audit records to a circular buffer that can be read by a /dev/audit device file, as shown in Figure 2-3 on page 10. When the kernel reaches the end of the buffer, it simply wraps to the beginning.

The **auditstream** command is used to read the /dev/audit audit device file. The **auditselect** command is used to select the events we are interested in, using SQL-like statements, and the **auditpr** command is used to convert the binary format into a human-readable form. The **auditstream** command provides the -c flag for selecting specific events. For more elaborate event selection, use the **auditselect** command.

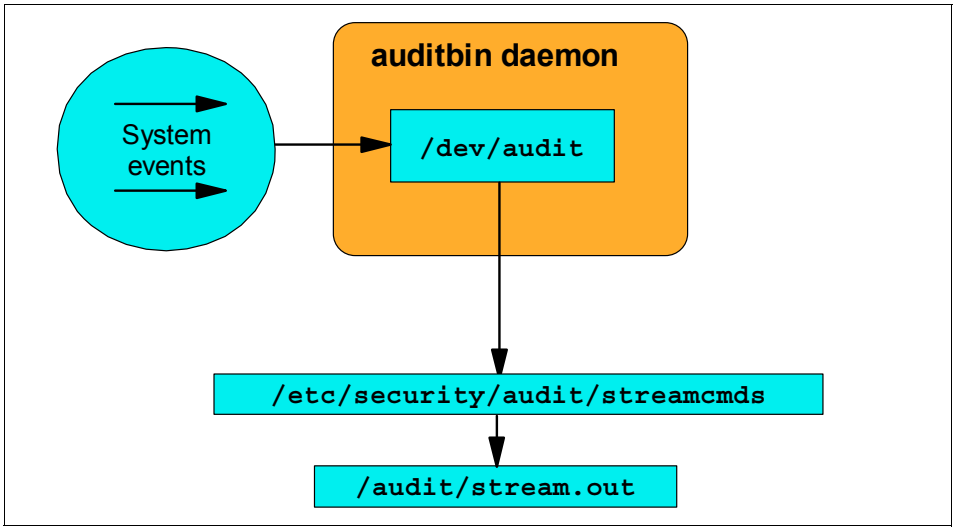

Figure 2-3 Data collection in STREAM mode

To set up the STREAM mode:

- Add `streammode = on` in the /etc/security/audit/config file.
- Check the `stream` stanza in the /etc/security/audit/config file and verify the path, trail and bin attributes.

Note: The two modes can be used simultaneously.

2.3 Setting up auditing

The files hereafter are involved in configuring, starting, and querying the audit subsystem. A short description of each file follows.

2.3.1 Configuration files

The configuration files are located in the /etc/security/audit directory.

config This is an ASCII stanza file that contains audit system configuration information. This file contains five stanzas: start, bin, stream, classes, and users. The stanzas are highlighted in Example 2-3.

– The `start` stanza defines the data collection method to use: BIN or STREAM. Use the value `on` to activate one or both of modes, or `off` to disable one of the modes, as in Example 2-3.

Example 2-3 Example of config file

```
[#][/etc/security/audit]> more config
start:
   binmode = on
   streammode = on
bin:
   trail = /audit/trail
   bin1 = /audit/bin1
   bin2 = /audit/bin2
   binsize = 10240
   cmds = /etc/security/audit/bincmds
   freespace = 65536
stream:
   cmds = /etc/security/audit/streamcmds
classes:
general=USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,FS_C
hroot,PORT_Locked,PORT_Change,FS_Mkdir,FS_Rmdir
objects=S_ENVIRON_WRITE,S_GROUP_WRITE,S_LIMITS_WRITE,S_LOGIN_WRITE,S_PASSWD_REA
D,S_PASSWD_WRITE,S_USER_WRITE,AUD_CONFIG_WR
```

some more class definitions follow...

```
users:
root = general,files
```

– The `bin` stanza defines the BIN data collection mode:

- `trail` gives the location of the trail audit record for the bin mode.
- The *bin1* and *bin2* lines specify the location of the two binary files.

- The `binsize` line is the size of temporary files in bytes, before switching to the other binary temporary file.
- The `cmds` line gives the location of the backend program (command) for the BIN mode.

– The `stream` stanza defines the STREAM data collection mode. There is only one line to define, the cmds line, which defines the backend command program for the STREAM mode.

– The `classes` stanza defines the sets of audit events. The following sets are predefined classes in the config file:

general	Refers to general commands such as **su** and password change.
objects	Refers to files such as /etc/security/paswd (read or write) and others such as /etc/group, /etc/limits, and so forth for write operations.
SRC	Refers to System Resource Controller (SRC) activity: start, stop, adding, or deleting entries.
kernel	Refers to kernel-related activities.
files	Refers to file-related operations: open, close, link, unlink, and so forth.
svipc	Refers to System V related activities: shared memory, semaphores, or system message exchanges.
mail	Refers to mail exchange operations: send, receive, or write.
cron	Refers to cron activities: start or stop the daemon, and add or delete entries in the crontab.
tcpip	Refers to TCP/IP user-level operations, such as configure, route, connect, access, and so forth, and kernel-level operations on sockets: open, close, connect, send, or receive.
lvm	Refers to LVM operations such as add, delete, extend, and so forth.

There is a special event class called ALL, implicitly defined, that contains all of the audit events defined on the system.

A total of 32 classes is supported. You should not attempt to define more than 31 audit classes (31 + the ALL class).

– The `user` stanza declares the audit class to be audited for a user. See in Example 2-3 on page 11, which shows that the user `root` is audited for the `general` and `files` classes.

oconfig This file is the copy of the config file. The copy is made when we start the audit subsystem (**audit start** command).

events This is an ASCII stanza file that contains information about audit events. The file contains just one stanza, auditpr, which lists all of the audit events in the system. The auditpr stanza also contains formatting information that the **auditpr** command needs to write an audit tail for each event.

Example 2-4 Example of the /etc/security/audit/event file

```
[#][/etc/security/audit]> cat events
....
auditpr:
```

...other rows precede

```
* kernel proc events

*       fork()
        PROC_Create = printf "forked child process %d"

*       exit()
        PROC_Delete = printf "exited child process %d"

*       exec()
        PROC_Execute = printf "euid: %d egid: %d epriv: %x:%x name %s"
```

... other rows follow

The auditpr stanza of the event file has lines in the following format:

audit_event = format_command

The format command can have the following parameters:

Parameter	Description
(empty)	The event has no tail.
printf *format*	The tail is formatted according to the string supplied for the *format* parameter. The %x symbols within the string indicate places for the audit trail to supply data.
Program -i n *arg* ...	The tail is formatted by the program that is specified by the Program parameter. The -i *n* parameter is passed to the program as its first parameter, indicating that the output is to be indented by *n* spaces. Other formatting information can be specified with the arg parameter. The audit event name is passed as the last parameter. The tail is written to the standard input of the program.

For a complete definition of the formatting parameters, see the /etc/security/audit/events file description in*AIX 5L Version 5.3 Files Reference,* SC23-4895.

The format parameter is commonly used with the printf format specification, as we show in the following examples. For this, we activate the audit subsystem, using the STREAM mode in two distinctive ways: one without printing the tail, the other one with printing the tail. This means to use `streammode = on` in the `start` stanza of the /etc/security/audit/config file, and use the following content for the `/etc/security/audit/streamcmds` file:

To avoid printing the tail part of audit records:

```
# /usr/sbin/auditstream | auditpr > /audit/stream.out &
```

To print the tail part of the audit records stored inhe /audit/stream.out file:

```
# /usr/sbin/auditstream | auditpr -v > /audit/stream.out &
```

In Example 2-5, we use the STREAM mode and **auditpr** command without the -v option (used to print the tail of the command) to store the audit records.

Example 2-5 Example of events without the tail part

```
[#][/audit]> tail -f stream.out
FS_Chdir        user3    OK          Tue Oct 26 11:10:49 2004 tsm
S_ENVIRON_WRITE user3    FAIL        Tue Oct 26 11:10:49 2004 tsm
S_PASSWD_READ   user3    OK          Tue Oct 26 11:10:50 2004 su
S_PASSWD_READ   user3    OK          Tue Oct 26 11:10:50 2004 su
USER_SU         user3    OK          Tue Oct 26 11:10:53 2004 su
FS_Chdir        user3    OK          Tue Oct 26 11:10:53 2004 su
FS_Chdir        user3    OK          Tue Oct 26 11:10:56 2004 ksh
```

In Example 2-6, we use the STREAM mode and **auditpr** command with the -v option (used to print the tail of the command) to store the audit records.

Example 2-6 Example of events with the tail part

```
[#][/audit]> tail-f stream.out
FS_Chdir        user3    OK          Tue Oct 26 11:10:49 2004 tsm
        change current directory to: /home/user3
S_ENVIRON_WRITE user3    FAIL        Tue Oct 26 11:10:49 2004 tsm
        audit object write event detected /etc/security/environ
S_PASSWD_READ   user3    OK          Tue Oct 26 11:10:50 2004 su
        audit object read event detected /etc/security/passwd
S_PASSWD_READ   user3    OK          Tue Oct 26 11:10:50 2004 su
        audit object read event detected /etc/security/passwd
USER_SU         user3    OK          Tue Oct 26 11:10:53 2004 su
        root
```

```
FS_Chdir        user3    OK          Tue Oct 26 11:10:53 2004 su
        change current directory to: /
FS_Chdir        user3    OK          Tue Oct 26 11:10:56 2004 ksh
        change current directory to: /tmp
```

The highlighted lines represent the tail part of the event. In Example 2-7 we consider the FS_Chdir event.

Example 2-7 The print format of the tail for FS_Chdir event

```
[#][/etc/security/audit]> cat events
...
*       chdir()
        FS_Chdir = printf "change current directory to: %s"
...
```

You can recognize the definition of the tail, where %s was substituted with the directory. To verify the other events, search the tail definition in /etc/security/audit/events and compare with Example 2-7.

Refer to Appendix A, “Security audit events in AIX 5.3” on page 199 for the complete list of auditable events defined in AIX5L V5.3.

objects

This file is an ASCII stanza file that contains information about the audited objects (files). This file contains one stanza for each audited file. The stanza name is in fact the path name of the audited file.

Example 2-8 Example of object file

```
[#][/etc/security/audit]> more objects
/etc/security/environ:
        w = "S_ENVIRON_WRITE"

/etc/security/group:
        w = "S_GROUP_WRITE"

/etc/security/limits:
        w = "S_LIMITS_WRITE"

/etc/security/login.cfg:
        w = "S_LOGIN_WRITE"

/etc/security/passwd:
        r = "S_PASSWD_READ"
        w = "S_PASSWD_WRITE"
```

```
/etc/security/user:
        w = "S_USER_WRITE"
```

Note: The lines that start with an asterisk (*) are comments.

bincmds

This file is an ASCII template that contains the BIN mode commands that are invoked when the audit system is initialized. The path name of this file is defined in the stream stanza of the /etc/security/audit/config file.

Example 2-9 Example of bincmds file

```
[#][/etc/security/audit]> cat bincmds
/usr/sbin/auditcat -p -o $trail $bin
```

We use the -p flag to compress the data appended to the audit trail file pointed by the $trail variable. The $bin variable points to the input file, which is one of the two binary files.

streamcmds

This file is an ASCII template that contains the STREAM mode commands that are invoked when the audit system starts. The path name of this file is defined in the stream stanza of the /etc/security/audit/config file.

To read all records from the audit device that have audit events in the authentication class, format them, and display them on the system console, include the lines in Example 2-10 in the /etc/security/audit/streamcmds file.

Example 2-10 Example of streamcmds file

```
/usr/sbin/auditstream -c authentication | \
/usr/sbin/auditpr -t0 -v > /dev/console &
```

This command enables timely auditing of user authentication events. The **auditpr** command specifies when header titles are displayed. The default title consists of an optional message (the -m flag) followed by the name of each column of output. The option -t0 (zero) ignores any title.

Note: The **auditstream** command, in STREAM mode, should run in the background; do not forget the ampersand (&) at the end on the command line.

2.3.2 Command files

The command (binary) files involved in accounting are located in the /usr/sbin directory:

audit

Controls system auditing through its keywords (start, shutdown, on, off, panic, query).

auditbin

The auditbin daemon in the audit subsystem that manages the temporary bin1 and bin2 files. The command also delivers bins of data records to backend commands for processing. The program starts only if the attribute `binmode = on` exists in the /etc/security/audit/config file. It creates a zero-size file, /audit/auditb, showing that auditbin is running.

auditcat

This is one of several backend commands that process the audit data records. For instance, to configure the system to append audit bin data to the audit trail file, add this line to the /etc/security/audit/bincmds file:

```
# /usr/sbin/auditcat  -o $trail $bin
```

When the auditbin daemon calls the **auditcat** command, the daemon replaces the *$bin* string with the path name of the current bin file, and replaces the *$trail* string with the name of the default audit trail file.The command accepts the -p flag, which is used to compress the audit records added to the audit trail. Compression is done through Huffman encoding.

auditconv

Converts audit records that were generated by previous versions of the operating system into the format used by AIX V4 and higher of the operating system. For instance, to convert the old audit file pre_v4_auditbin, storing the results in converted_auditbin, enter the following command:

```
# /usr/sbin/auditconv pre_v4_auditbin converted_auditbin
```

auditmerge

Used to merge binary audit trails. This is especially useful if there are audit trails from several systems that must be combined:

```
# auditmerge trail.system1 trail.system2 | auditpr -v -hhelrRtpc
```

The **auditmerge** command concatenates the two audit trail files coming from system1 and system2 machines and pipes the output to the **auditpr** command to be printed. The -v flag of the **auditpr** command prints the audit tail of the event in addition to the standard audit information that the kernel delivers for every event.

auditselect

The **auditselect** command selects audit records that match identified criteria and writes the records to standard output. We can filter the audit trail to obtain specific records for analysis or select specific records for long-term storage.

To select bin-collected data records that match USER_SU or USER_Login audit events, add the **auditselect** command to /etc/security/audit/bincmds as in Example 2-11.

Example 2-11 Example of auditselect command

```
/usr/sbin/auditselect -e "event== USER_SU || event== \
USER_Login" $bin >> /audit/trail.login
```

While auditing is enabled, the records for each initiation of a user session are read from the current bin file and written to the /audit/trail.login file.

The **auditselect** command has three valid logical operator values:

&&(And) The logical operator *term1 && term2* means that the expression term1 and term2 is true.

||(Or) The logical operator *term1 || term2* means that the expression term1 or term2 is true.

!(Not) The logical operator *!term1* means that term1 is not true.

Aside from the logical operator value, there is also a relational operator value and a different field. For a detailed discussion of the **auditselect** command, refer to *AIX 5L Version 5.3, Commands Reference, Volume 1, a - c,* SC23-4888.

auditstream

This command reads audit records from the /dev/audit file (the audit device file) and copies the records to standard output in binary format. You can select a subset of the audit records by specifying audit classes (defined in the /etc/security/audit/config file) with the -c flag; otherwise, all currently enabled audit classes are copied. Audit stream data can be displayed and processed as it is generated.

For example, the command output can be piped to an audit backend command for further processing or redirected to a file. Both the **auditselect** command, which selects data records according to defined criteria, and the **auditpr** command, which formats the records for viewing or for printing, are examples of backend commands.

Example 2-12 Example of auditstream command

```
/usr/sbin/auditstream | /usr/sbin/auditselect -e "event == \
USER_Login || event == USER_SU" | \
/usr/sbin/auditpr -v > /dev/lp0 &
```

This command reads the /dev/audit device file, and formats and writes all user su and login events to the line printer.

auditpr

This command reads audit records, in binary or stream format, from standard input and sends formatted records to standard output. See Example 2-13.

Example 2-13 Example of auditpr command

```
[#][/]> /usr/sbin/auditpr -v < /audit/trail
event           login    status      time                        command
-------------------------------------------------------------------------
FS_Chdir        root     OK          Tue Oct 05 12:58:26 2004 ksh
FILE_Unlink     root     OK          Tue Oct 05 12:59:03 2004 vi
FILE_Unlink     root     OK          Tue Oct 05 12:59:12 2004 vi
FS_Chdir        root     OK          Tue Oct 05 12:59:34 2004 ksh
FS_Chdir        root     OK          Tue Oct 05 12:59:37 2004 ksh
FILE_Unlink     root     OK          Tue Oct 05 12:59:40 2004 vi
FILE_Unlink     root     OK          Tue Oct 05 12:59:59 2004 vi
CRON_Start      root     OK          Tue Oct 05 13:00:00 2004 cron
FS_Chdir        root     OK          Tue Oct 05 13:00:00 2004 cron
FILE_Unlink     root     OK          Tue Oct 05 13:00:02 2004 vi
FILE_Unlink     root     OK          Tue Oct 05 13:00:04 2004 vi
FILE_Unlink     root     OK          Tue Oct 05 13:02:38 2004 vi
FILE_Unlink     root     OK          Tue Oct 05 13:02:44 2004 vi
FILE_Unlink     root     OK          Tue Oct 05 13:02:44 2004 vi
TCPIP_connect   root     OK          Tue Oct 05 13:20:15 2004 telnetd
FILE_Write      root     OK          Tue Oct 05 13:20:15 2004 telnetd
```

The **auditpr** command reads the system audit trail file and prints the audit records with default header titles and fields. The audit trail file must be a valid audit bin or record. Example 2-14 shows how to select the fields we want to print from audit bin records.

Example 2-14 using auditpr command with parameters

```
[#][/audit]> auditpr -h e,l,r,t,R,c < trail|more
event           login    real     time                      status      command
-------------------------------------------------------------------------------
FS_Chdir        root     root     Tue Oct 05 12:58:26 2004 OK          ksh
FILE_Unlink     root     root     Tue Oct 05 12:59:03 2004 OK          vi
FILE_Unlink     root     root     Tue Oct 05 12:59:12 2004 OK          vi
FS_Chdir        root     root     Tue Oct 05 12:59:34 2004 OK          ksh
```

```
FS_Chdir        root     root     Tue Oct 05 12:59:37 2004 OK          ksh
FILE_Unlink     root     root     Tue Oct 05 12:59:40 2004 OK          vi
FILE_Unlink     root     root     Tue Oct 05 12:59:59 2004 OK          vi
CRON_Start      root     root     Tue Oct 05 13:00:00 2004 OK          cron
FS_Chdir        root     root     Tue Oct 05 13:00:00 2004 OK          cron
FILE_Unlink     root     root     Tue Oct 05 13:00:02 2004 OK          vi
FILE_Unlink     root     root     Tue Oct 05 13:00:04 2004 OK          vi
FILE_Unlink     root     root     Tue Oct 05 13:02:38 2004 OK          vi
FILE_Unlink     root     root     Tue Oct 05 13:02:44 2004 OK          vi
FILE_Unlink     root     root     Tue Oct 05 13:02:44 2004 OK          vi
TCPIP_connect   root     root     Tue Oct 05 13:20:15 2004 OK          telnetd
FILE_Write      root     root     Tue Oct 05 13:20:15 2004 OK          telnetd
FS_Chdir        root     root     Tue Oct 05 13:20:21 2004 OK          tsm
FILE_Unlink     root     root     Tue Oct 05 13:20:24 2004 OK          ksh
```

The -h option of the **auditpr** command enables you to select what fields to display and the order in which to display them. Here is a brief description of each field that we used:

- The e field gives you the audit event.
- The l field gives you the user's login name that generated the audit event.
- The r field gives you the user's real name that generated the audit event.
- The t field gives you the time the record was written.
- The R field gives you the audit status.
- The c field gives you the command name.

For a detailed description of these commands and their parameters, see *AIX 5L Version 5.3, Commands Reference, Volume 1, a - c,* SC23-4888.

2.3.3 Output files

The files that are generated by the audit subsystem are placed in the /audit directory. Because an audit can produce lot of data and the /audit directory is by default created in **/** (root) filesystem, we recommended creating a separate file system for the audit files, large enough to keep the audit trail and the audit bin files.

/audit/bin1	The first temporary BIN file.
/audit/bin2	The second temporary BIN file.
/audit/trail	The trail file, used to store the audit records that are temporarily stored in the bin1 and bin2 files.
/audit/auditb	The indicator file showing to audit modules that audit is activated in the BIN mode. It is created by the auditbin daemon when it starts and it has the size zero.
/audit/stream.out	The STREAM files used in STREAM mode.

We have to select the system events that we want to be audited from the events file. The selected events should use to detect activities that compromise or violate the security of the system.

The file /etc/security/audit/events contain one stanza: auditpr and list events on the system and the corresponding format to write the tail of each event by **auditpr** command:

```
auditpr[]
* these are comments
AuditEvent = FormatCommand
```

If you have coded new events in your application code or kernel extension, you should add new events (and associated formatting instructions) to the events file.

If necessary, group the selected events into classes in the /etc/security/audit/config file. A special event class is the class ALL - containing all events on the system. Use it with caution because activating this class generates a large volume of data.

Assign the defined classes (or events):

- To objects, if you audit files, in the /etc/security/audit/objects file.
- To users, in /etc/security/audit/config.

2.3.4 Data collection

The selection of a data collection method depends on how you intend to use the data. You can configure the system to use both of the data collection methods at the same time.

The BIN collection method enables the storage of a large amount of data in the audit trail. If this mode is selected, the auditbin daemon starts when the audit subsystem starts.

The kernel receives at startup two file descriptors corresponding to the two BIN files defined in the config file. It suspends the calling process and starts writing into the first file. When the first file reaches its maximum bin size, the auditbin daemon substitutes *$bin* and *$trail* variables in the bincmds file; empties the first file, appending its content to the trail file (trail attribute of the config file); switches to the second file descriptor and starts to fill the second file; and reactivates the calling process. The process repeats.

The data from the bin file is appended to the trail file using the command stored in the bincmds file. See the highlighted lines in Example 2-15 on page 22.

Example 2-15 Config file for the BIN mode

```
[#][/etc/security/audit]> head -20 config

start:
        binmode = on
        streammode = on

bin:
        trail = /audit/trail
        bin1 = /audit/bin1
        bin2 = /audit/bin2
        binsize = 10240
        cmds = /etc/security/audit/bincmds
        freespace = 65536
...
[#][/etc/security/audit]> cat /etc/security/audit/bincmds
/usr/sbin/auditcat -p -o $trail $bin
[p630n02][/etc/security/audit]>
```

The STREAM collection method enables processing of data as it is collected. The audit records are written to a circular buffer within the kernel, and may be retrieved by readingthe /dev/audit device file, allowing real-time detection and monitoring.

Another use of the STREAM method is to create an audit trail file written to disk immediately. Another use is writing the audit stream into a program that stores the trail on a remote system, which enables central processing while protecting the original audit information from tampering at the original host. See the highlighted lines for the STREAM mode in Example 2-16.

In order to configure one method or another (or both), modify the binmode attribute, streammode attribute, or both in the /etc/security/audit/config file.

Example 2-16 Config file for STREAM mode

```
[#][/etc/security/audit]> cat config

start:
        binmode = on
        streammode = on

stream:
        cmds = /etc/security/audit/streamcmds
...
[#][/etc/security/audit]> cat  /etc/security/audit/streamcmds
/usr/sbin/auditstream | auditpr -v > /audit/stream.out &
[#][/etc/security/audit]>
```

The audit logger knows now that it has to use both BIN mode and STREAM mode. The backend commands are located in /etc/security/audit/bincmds for the BIN mode and in /etc/security/audit/streamcmds for the STREAM mode if not specified otherwise in the cmds attribute of the bin or stream stanza.

Other attributes specific to bin and stream stanzas are:

trail

Specifies the path name of the audit trail file. When this is defined, the auditbin daemon can substitute the path name of the audit trail file for the *$trail* string in the backend commands that it calls.

bin1

Specifies the path name that the auditbin daemon uses for its primary bin file. If the *$bin* string is the parameter value, the auditbin daemon substitutes the name of the current bin file.

bin2

Specifies the path name that the auditbin daemon uses for its secondary bin file. If the *$bin* string is the parameter value, the auditbin daemon substitutes the name of the current bin file.

binsize

Specifies a decimal integer string that defines the threshold size (in bytes) of each audit bin. If the binsize parameter is set to 0, no bin switching occurs, and all bin collection goes to bin1.

cmds

Specifies the path name of the file that contains the audit backend commands called by the auditbin daemon. The file contains command lines, each composed of one or more backend commands with input and output that can be piped together or redirected. See “bincmds” on page 16 for more information.

bytethreshold

Specifies the decimal integer string that defines the approximate number of bytes written to an audit bin before a synchronous update is performed. If the bytethreshold is set to 0, this function is disabled. Both bytethreshold and eventthreshold can be used simultaneously.

eventthreshold

Specifies a decimal integer string that defines the maximum number of events written to an audit bin before a synchronous update is performed. If the

eventthreshold is set to 0, this function is disabled. Both attributes, eventthreshold and bytethreshold, can be used simultaneously.

freespace

Specifies a decimal integer string that defines the recommended number of 512-byte free blocks in the file system where the audit trail file is located. If the free space of file system is below this value, audit generates a warning message through the syslog subsystem every time that the audit bin is switched. The default value is 65536 blocks (64 megabytes). The maximum possible value is 4194303 (about 2 GB of free disk space). If this value is set to 0, no warning message is generated.

Note: The Auditing chapter of *AIX 5L Version 5.3, Security Guide,* SC23-4907, recommends the following:

- The freespace parameter in the bin stanza should be configured, at minimum, to a value that equals 25% of the disk space dedicated to the storage of the audit trails.
- The bytethreshold and binsize parameters should each be set to 65536 bytes.
- The binmode parameter in the start stanza in /etc/security/audit/config should be set to panic.

2.3.5 Starting and stopping auditing

Invoke the **audit** command for starting, stopping or querying the auditing.

audit start

This is the correct way of starting the auditing. When you start the audit, it creates the /audit directory automatically if one does not already exist, and the file /audit/auditb of zero size, showing that the audit is activated. If there is no error, the command returns immediately without a message.

audit shutdown

Stop the auditing. It empties the internal buffers into the BIN records or /dev/audit file, then removes the /audit/auditb file used as “active” indicator for the audit modules. When executing the command, the system responds with:

```
auditing reset
```

audit on (panic)

Restarts the auditing system after a suspension, if the system is properly configured (for example, if the **audit start** command was used initially and the configuration is still valid). If auditing is already started when the

command runs, only bin data collection can be changed. When executed, the system responds with:

```
auditing enabled
```

Note: If you specify the panic option, the system shuts down if bin data collection is enabled but cannot be written to a bin file.

audit off

Suspends the auditing system, but leaves the configuration valid. The system responds with:

```
auditing disabled
```

Note: Data collection pauses until the **audit on** command is run.

audit query

Displays the status of auditing. See Example 2-17.

Example 2-17 Example of system response for audit query command

```
auditing on
audit bin manager is process 24150
audit events:
general -
USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,FS_Chroot,PO
RT_Locked,PORT_Change,FS_Mkdir,FS_Rmdir
/etc/security/group:
w = S_GROUP_WRITE
/etc/security/environ:
w = S_ENVIRON_WRITE
/etc/security/limits:
w = S_LIMITS_WRITE
/etc/security/passwd:
r = S_PASSWD_READ
w = S_PASSWD_WRITE
/etc/security/login.cfg:
w = S_LOGIN_WRITE
/etc/security/user:
w = S_USER_WRITE
/etc/security/audit/config:
w = AUD_CONFIG_WR
```

To start the audit process at system startup: After configuring the audit files, add the following line to the system initialization file (the /etc/rc file):

```
/usr/sbin/audit start 1>&- 2>&-
```

Or you can add this to the /etc/inittab the line:

```
audit:2:once:/usr/sbin/audit start 1>&- 2>&-
```

Note: We recommend using the **mkitab** command to modify the /etc/inittab file:

```
# mkitab "audit:2:once:/usr/sbin/audit start 1>&- 2>&-"
```

The audit process will start as configured each time the system is initialized. To shut down the audit subsystem properly at system shutdown, add the following to the /etc/shutdown file (or you can run this command manually):

```
/etc/sbin/audit shutdown
```

2.4 Recommendations for auditing

The AIX 5.3 documentation recommends care when running auditing during normal system operations, because collecting and handling the audit data may add extra load to the system (specially when using the ALL event class).

However, a system administrator should know how to configure, start, stop, and display at least basic audit information. These tasks should be practiced and learned so that in case you suspect that your system is attacked, you should be able to configure the system for collecting the audit data.

On systems with little memory or CPU power, we do not recommend starting the audit subsystem automatically; instead, have it ready to be launched, especially if you plan to audit the ALL class. Also, we recommend configuring the auditing to avoid the use of the ALL class, because this may make it even more difficult to detect an intruder or a security breach in the large amount of data generated by auditing the ALL class.

We recommend preparing in advance a list of critical objects (files) or events to be audited, before a critical situation actually arises. Thus, you can avoid unnecessary load, unless you start the audit subsystem.

On a critical server we recommend defining a minimal set of audit events and objects and letting the audit subsystem activate at system startup.

Note: Make sure you test the auding subsystem in a maintenance window, or during off-peak hours, and try to identify as accurately as possible the load that may be induced by the auditing subsystem.

2.4.1 Using the audit subsystem for a quick security check

The **watch** command can be used to monitor a single suspicious program without setting up the entire audit subsystem. This command records either all events that are generated by the specified program, or just the ones thar\t you specify. For example, to see all FILE_Open events when running vi /etc/hosts, type the first two lines in Example 2-18.

The /tmp/vi.watch file stores all FILE_Open events for the editor session. The -e flag is used to specify what event is audited, and the -o flag stores the output generated by the watch command in a file.

Note: The **watch** command can work only if the audit subsystem is disabled. Use **audit shutdown** or **audit off** before using the **watch** command.

Example 2-18 Example of using the watch command for FILE_OPEN event

```
[#][/audit]> watch -eFILE_Open -o /tmp/vi.watch vi /etc/hosts
[#][/audit]> cat /tmp/vi.watch

***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ -------------------------------
FILE_Open       root     OK          Mon Oct 25 14:51:11 2004 vi
        flags: 0 mode: 0 fd: 4 filename /usr/share/lib/terminfo/x/xterm
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ -------------------------------
FILE_Open       root     FAIL        Mon Oct 25 14:51:11 2004 vi
        flags: 0 mode: 0 fd: 4 filename //.exrc
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ -------------------------------
FILE_Open       root     OK          Mon Oct 25 14:51:11 2004 vi
        flags: 1282 mode: 600 fd: 4 filename /var/tmp/Ex56786
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ -------------------------------
FILE_Open       root     OK          Mon Oct 25 14:51:11 2004 vi
        flags: 2 mode: 0 fd: 4 filename /var/tmp/Ex56786
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ -------------------------------
FILE_Open       root     OK          Mon Oct 25 14:51:11 2004 vi
        flags: 1282 mode: 600 fd: 4 filename /var/tmp/Ex56786
***** WATCH *****
event           login    status      time                     command
```

```
-------------- -------- ----------- ----------------------- ------------------------------
FILE_Open      root     OK          Mon Oct 25 14:51:11 2004 vi
        flags: 2 mode: 0 fd: 4 filename /var/tmp/Ex56786
***** WATCH *****
event          login    status      time                    command
-------------- -------- ----------- ----------------------- ------------------------------
FILE_Open      root     OK          Mon Oct 25 14:51:11 2004 vi
        flags: 0 mode: 0 fd: 5 filename /etc/hosts
***** WATCH *****
event          login    status      time                    command
-------------- -------- ----------- ----------------------- ------------------------------
FILE_Open      root     OK          Mon Oct 25 14:51:11 2004 vi
        flags: 0 mode: 0 fd: 5 filename /usr/lib/nls/msg/en_US/ex.cat
```

Example 2-19 shows all of the events generated by the **ls** command.

Example 2-19 Using watch to see all events generated by the ls command

```
[#][/audit]> watch ls
auditb      bin1        bin2        stream.out  trail
***** WATCH *****
event          login    status      time                    command
-------------- -------- ----------- ----------------------- ------------------------------
AUD_Proc       root     OK          Mon Oct 25 14:47:36 2004 watch
        pid: 0 cmd: 4
***** WATCH *****
event          login    status      time                    command
-------------- -------- ----------- ----------------------- ------------------------------
PROC_SetUserIDs root     OK          Mon Oct 25 14:47:36 2004 watch
        effect: 0, real: 0, saved: -1, login: -1
***** WATCH *****
event          login    status      time                    command
-------------- -------- ----------- ----------------------- ------------------------------
TCB_Exec       root     OK          Mon Oct 25 14:47:36 2004 watch
        filename: /usr/bin/ls
***** WATCH *****
event          login    status      time                    command
-------------- -------- ----------- ----------------------- ------------------------------
PROC_Execute   root     OK          Mon Oct 25 14:47:36 2004 ls
        euid: 0 egid: 0 epriv: ffffffff:ffffffff name /usr/bin/ls
***** WATCH *****
event          login    status      time                    command
-------------- -------- ----------- ----------------------- ------------------------------
PROC_Load      root     OK          Mon Oct 25 14:47:36 2004 ls
        file: /usr/lib/nls/loc/en_US
***** WATCH *****
event          login    status      time                    command
-------------- -------- ----------- ----------------------- ------------------------------
PROC_LoadMember root     OK          Mon Oct 25 14:47:36 2004 ls
```

```
        file: /usr/lib/libi18n.a, member: shr.o
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------------------
PROC_Load       root     OK          Mon Oct 25 14:47:36 2004 ls
        file: /usr/lib/nls/loc/en_US
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------------------
PROC_Load       root     OK          Mon Oct 25 14:47:36 2004 ls
        file: /usr/lib/nls/loc/en_US
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------------------
```

2.4.2 Disk space consideration

Auditing all events can produce large amounts of data that are difficult to process. A single command, such as **ls**, would result in several events, as we can see in Example 2-19 on page 28.

Each record in the audit trail takes about 50 to 150 bytes depending on the mode used and whether the verbose mode flag is specified. This means that 1 MB of data could contain about 6800 audit records.

Even if we focus on specific events to audit, there will still be a lot of data. The volume of data can be reduced using the **auditselect** command combined with the **auditpr** command. The **auditselect** command can be used to pull data from a specific time period, for a specific user, for a specific event, or any combinations of these. For instance:

```
/usr/sbin/auditselect -f /audit/pick /audit/trail | /usr/sbin/auditpr -v
```

The /audit/pick file contains the event selection conditions for the **auditselect** command, and reads as follows:

```
command == rlogin && \
time >= 08:00:00 && time <= 17:00:00 && \
data >= 04/01/04 && date <= 04/12/04
```

This command reports the use of the **rlogin** command within the specific time interval (8:00am - 5:00pm, between April 1st and April 12th). The -v flag of the **auditpr** command prints the tails of the selected audit events.

2.4.3 Performance

Each event does a preliminary check to see whether auditing is turned on. If on, it informs the kernel that audit is enabled, and starts recording and auditing information for each command that has auditing enabled.

The kernel logger service automatically writes the auditing information to the audit trail file after the command is completed. This is true for all events and objects. Although there is a slight difference in the amount of time each event takes to complete the audit logging, the difference is still so small that is almost negligible.

Overall, regardless of whether auditing is enabled, the system will experience an additional load, based on actual auditing. The reason is that there are some programs that have coded their own set of audit events. The additional load will not actually happen if the audited events do not occur.

2.4.4 Audit limitations

In AIX 5.3, the two bin files are no longer world-writable, thus closing a possible security leak.[1]

Some events are generated by commands. For instance, the creation of a user (the **mkuser** command) generates the USER_Create event. However, it is possible to create a new user without using the **mkuser** command, thus such an event would be triggered. You can create a new user by using an editor to modify the corresponding security file (/etc/password).

Some events are recorded in the trail or stream.out files, without being explicitly configured. This is the case for cron and TCP/IP events. You can exclude these events using the **auditselect** command.

Note: Many database products do not interact properly with the audit subsystem. Such products open their files through a database monitor, and the audit subsystem is unaware which user is requesting which data from the database monitor. The audit subsystem only sees the database monitor accessing its files.

These comments are not meant to denegrate or discourage the audit subsystem, which can be very effective, especially in environments where root and system

[1] In previous releases of AIX, the BIN files used by the BIN mode were world-writable. Even if the /audit directory was not accessible by others, it could be read and searched by the system group. Therefore, a user from the system group could introduce or remove audit events.

access is tightly controlled. Two styles of usage might be considered for these environments:

- Monitor-specific objects and events that are relevant for the security of the system and write local program to report usage.
- Be prepared to record objects and events for use after the fact in investigating problems. This implies some planning and effort in order to manage the large amount of data that is susceptible to being collected.

2.5 Configuration examples

This section presents some audit configuration examples, for real-time monitoring of the modification of critical system files, and additional auditing scenarios.

2.5.1 A real-time modification monitor

Assume that we want to monitor file access for some critical files in real-time (for example, all files in the /etc directory). The event audited for the files in the /etc directory is FILE_Write (present in /etc/security/audit/events).

In the file config, define a class called `filemon` as follows:

```
Class:
   filemon = FILE_Write
```

Construct the /etc/security/audit/objects file audited for this class as follows:

```
   find /etc -type f | awk '{printf("%s:\n\tw = FILE_Write\n\n",$1)}' >>
/etc/security/audit/objects
```

The precedent command does the following: for every file of the /etc directory (find /etc -type f), print the name of the file, go to the next line, print a tab and `w = FILE_Write` followed by two blank lines. The output is redirected into the /etc/security/audit/objects file. The file objects should look like this:

```
/etc/consdef:
      w = FILE_Write
/etc/csh.cshrc:
      w = FILE_Write
/etc/csh.login:
   w = FILE_Write
/etc/dlpi.conf:
   w = FILE_Write
```

Set up the STREAM mode and configure the users to be audited in /etc/security/audit/config:

```
start:
   binmode = off
   streammode = on
stream:
   cmds = /etc/security/audit/streamcmds
classes:
   filemon = FILE_write
users:
   root = filemon
afx = filemon
...
```

The STREAM mode is on, and the stream stanza is pointing to /etc/security/audit/streamcmds, containing the backend command.

Considering that the output is sent to the console, the content of the /etc/security/audit/streamcmds is as follows:

```
/usr/sbin/auditstream | /usr/sbin/auditselect -e "event == FILE_Write" |
auditpr  -hhelpPRtTc -v > /dev/console &
```

The **auditstream** command reads the /dev/audit special file and pipes the output to the **auditselect** command, which filters only the FILE_Write events. The **auditpr** command converts the binary format into human-readable form, also printing the tail of the event (the -v flag).

To activate the auditing:

```
# /usr/sbin/acct/audit on
```

2.5.2 More audit scenarios

The audit system should be able to create an audit trail of all auditable events. Your log file may contain just enough information, or it might contain too much information. Always remember that each record should contain information that would help you construct a scenario of what actually happened for a given time.

Initially, you need to set up auditing in order to know what is happening on your system. Auditing can be configured for one user, for all users, for several classes, or for all system activity. Now that you have the audit record, you have to assess what you can use it for.

At this point, you are now ready to read the data collected. There are two repositories for auditing. The default is the /audit/trail file for BIN data collection, and the /audit/stream.out file for STREAM mode data collection.

Here is the scenario for event auditing. This record was generated when user3 performed the following actions:

- Connected to the server (in this case, the server IP label is p630n02, or IP address is 192.168.100.31) via network (TCP/IP) using the **telnet** command (**telnet p630n02**).
- Logged in as user3 from the login prompt.
- Attempted to change user3's password using the **passwd** command; the attempt was not successful.
- Attempted to change user3's password again (same command). This time, the attempt was successful.
- Attempted to switch user (the **su** command) to root, but used a wrong password.
- Attempted to switch user to user root again using the **su** command. This attempt was successful.

Assumptions:

- The audit data collection mode is set up for both BIN and STREAM modes.
- Both the default bin and stream stanzas are used.
- We audit the general class for user3.

Event auditing: BIN mode

To read the output, you can use this command:

```
# auditpr -v < /audit/bin1
```

Because the bin1 file is too small, the content of this file has not yet been transferred to the trail file. The audit records are still kept in this file, so we use it to display the audit records. The following lines are displayed:

Example 2-20 Event auditing: output of the bin file

```
[#][/audit]> auditpr -v < bin1
event           login    status      time                     command
------------------------------------------------------------------------

TCPIP_connect   root     OK          Tue Oct 26 09:52:10 2004 telnetd
        TCP/IP ::ffff:9.12.6.176 telnet/tcp open
FS_Chdir        user3    OK          Tue Oct 26 09:52:16 2004 tsm
        change current directory to: /home/user3
S_ENVIRON_WRITE user3    FAIL        Tue Oct 26 09:52:16 2004 tsm
        audit object write event detected /etc/security/environ
PASSWORD_Change user3    FAIL        Tue Oct 26 09:52:34 2004 passwd
        user3
```

```
PASSWORD_Change user3    OK          Tue Oct 26 09:52:43 2004 passwd
        user3
S_PASSWD_READ   user3    OK          Tue Oct 26 09:52:50 2004 su
        audit object read event detected /etc/security/passwd
S_PASSWD_READ   user3    OK          Tue Oct 26 09:52:50 2004 su
        audit object read event detected /etc/security/passwd
USER_SU         user3    FAIL        Tue Oct 26 09:52:51 2004 su
        root
S_PASSWD_READ   user3    OK          Tue Oct 26 09:52:53 2004 su
        audit object read event detected /etc/security/passwd
S_PASSWD_READ   user3    OK          Tue Oct 26 09:52:53 2004 su
        audit object read event detected /etc/security/passwd
USER_SU         user3    OK          Tue Oct 26 09:52:56 2004 su
        root
FS_Chdir        user3    OK          Tue Oct 26 09:52:56 2004 su
        change current directory to: /
```

This has five columns:

event Gives the event name defined in the events file.

login Gives the login ID.

status States whether the execution of an event is successful. Valid values are OK, FAIL, FAIL_AUTH, FAIL_PRIV, FAIL_ACCESS, and FAIL_DAC.

time Gives the date and time the event was executed.

commands Gives the command used that triggered the event.

The status column has six valid values:

1. The OK value means that there was a successful execution of an event.
2. The FAIL value means that there was an unsuccessful execution of an event. This is the default FAIL value.
3. The FAIL_AUTH value indicates that authentication was denied. The user may have tried to log on and failed authentication by giving an incorrect password, or tried to log on to a console where they do not have permission.
4. The FAIL_PRIV value indicates lack of privilege.
5. The FAIL_ACCESS value indicates lack of access.
6. The FAIL_DAC value indicates that the event failed because of a discretionary access control (DAC) denial. Access Control Lists are a form of information repository that contain data relative to the rights of access (permissions) to shared resources (objects). ACLs are categorized on DAC mechanism.

Observe the PASSWORD_Change and USER_SU events and the status of each event. The second line of each event is in fact the tail portion. The format of this comes from the printf information in the the events file (/etc/security/audit/events). If the **auditpr** command used the events file, the tail portion can be modified (depending on what you want to record).

When user3 attempted to change the password and entered a wrong password, the PASSWORD_Change event was triggered with a FAIL status. The second line shows the user whose password user3 wanted to change.

The second time user3 attempted to change the password, the attempt was successful. This time the PASSWORD_Change event was again triggered, but with the OK status.

The same thing happened when user3 tried to switch user to user root. The only difference is that now the USER_SU event was triggered, and the second line gives you the user ID to which user user3 wants to switch (**su**).

Event auditing: STREAM mode

To read the output, you can use the **cat** command.

Example 2-21 Event auditing: output of the stream.out file

```
[#][/audit]> tail -f stream.out
event           login    status      time                      command
------------------------------------------------------------------------
```

... telnet connection

```
TCPIP_connect   root     OK          Tue Oct 26 09:52:10 2004 telnetd
FS_Chdir        user3    OK          Tue Oct 26 09:52:16 2004 tsm
S_ENVIRON_WRITE user3    FAIL        Tue Oct 26 09:52:16 2004 tsm
```

... unsuccessful password change

```
PASSWORD_Change user3    FAIL        Tue Oct 26 09:52:34 2004 passwd
```

...successful password change

```
PASSWORD_Change user3    OK          Tue Oct 26 09:52:43 2004 passwd
```

... su to root unsuccessful

```
S_PASSWD_READ   user3    OK          Tue Oct 26 09:52:50 2004 su
S_PASSWD_READ   user3    OK          Tue Oct 26 09:52:50 2004 su
USER_SU         user3    FAIL        Tue Oct 26 09:52:51 2004 su
```

... su to root successful

```
S_PASSWD_READ   user3    OK          Tue Oct 26 09:52:53 2004 su
```

```
S_PASSWD_READ   user3    OK          Tue Oct 26 09:52:53 2004 su
USER_SU         user3    OK          Tue Oct 26 09:52:56 2004 su
FS_Chdir        user3    OK          Tue Oct 26 09:52:56 2004 su
```

As in the BIN mode, five columns are displayed. Take a look at the events PASSWORD_Change and USER_SU and the status of each event. The events in this record (STREAM) should be interpreted the same way that you interpreted the BIN mode record.

Notice the different content of the bin file and the stream.out file.

The reason for this is that in BIN mode we used the **auditpr** command *and* the -v option. The -v option displays the tail of each audit record using the format specification in the /etc/security/audit/events file. You can see the content of the streamcmds file, where **auditpr** is used without the -v option:

```
#[/etc/security/audit]> cat streamcmds
/usr/sbin/auditstream | auditpr > /audit/stream.out &
#[/etc/security/audit]>
```

Note: Aside from the formatting difference, do not forget that BIN mode can store records for a long term, but STREAM mode records are overwritten whenever you start auditing.

Object auditing: STREAM mode

This is the scenario for object auditing. This record was generated when user3 performed the following:

- Connected to the server (in this case, the server IP label is p630n02, or IP address is 192.168.100.31) via network (TCP/IP) using the telnet command (**telnet p630n02**).
- Logged in as `user3` from the login prompt. user3 belongs to the `staff` group (see Example 2-22).

Example 2-22 User user3 properties

```
[p630n02][/]> lsuser user3
user3 id=216 pgrp=staff groups=staff home=/home/user3 shell=/usr/bin/ksh
auditclasses=General,PROC_Kill login=true su=true rlogin=true daemon=true
admin=false sugroups=ALL admgroups= tpath=nosak ttys=ALL expires=0 auth1=SYSTEM
auth2=NONE umask=22 registry=files SYSTEM=compat logintimes= loginretries=0
pwdwarntime=0 account_locked=false minage=0 maxage=0 maxexpired=-1 minalpha=0
minother=0 mindiff=0 maxrepeats=8 minlen=0 histexpire=0 histsize=0 pwdchecks=
dictionlist= fsize=2097151 cpu=-1 data=262144 stack=65536 core=2097151
rss=65536 nofiles=2000 time_last_login=1098891181 tty_last_login=/dev/pts/0
host_last_login=9.12.6.176 unsuccessful_login_count=0 roles=
```

- Attempted to switch user to `root` with the correct password (using the **su** command).
- Opened the /etc/security/passwd file using **vi /etc/security/passwd**, then closed it without saving.
- Opened the /etc/security/passwd file again, then saved the /etc/security/password file and exited from vi editor.

We assume that:

- STREAM mode data collection is on.
- The default bin and stream stanza are used.
- The file /etc/security/objects has the following stanza:

```
/etc/security/passwd:
        r = "S_PASSWD_READ"
        w = "S_PASSWD_WRITE"
```

  This enables you to audit the object (file) /etc/security/passwd, generating S_PASSWD_READ and S_PASSWD_WRITE events if the file is read or written.

- In the /etc/security/audit/config file, user3 has assigned the class General for auditing, as in the precedent examples:

```
[p630n02][/etc/security/audit]> cat config
....
users:
        user3 = General
        root = General
[p630n02][/etc/security/audit]>
```

We have edited the file /etc/security/audit/streamcmds to include the -v option. The file should look like this:

```
[p630n02][/]> cat /etc/security/audit/streamcmds
/usr/sbin/auditstream | auditpr -v > /audit/stream.out &
[p630n02][/]>
```

This option captures not only the header but also the tail of the event, writing it to the stream.out file.

Example 2-23 on page 38 shows the output for the STREAM mode data collection method that used object auditing, using the -v parameter for **auditpr** command in the streamcmds file.

Example 2-23 Object auditing: content of the stream.out file

```
event           login    status      time                      command
-------------------------------------------------------------------------
```

... telnet connection

```
TCPIP_connect   root     OK          Tue Oct 26 09:45:39 2004 telnetd
        TCP/IP ::ffff:9.12.6.176 telnet/tcp open
FS_Chdir        user3    OK          Tue Oct 26 09:45:45 2004 tsm
        change current directory to: /home/user3
S_ENVIRON_WRITE user3    FAIL        Tue Oct 26 09:45:45 2004 tsm
        audit object write event detected /etc/security/environ
```

... the **su** command

```
S_PASSWD_READ   user3    OK          Tue Oct 26 09:45:54 2004 su
        audit object read event detected /etc/security/passwd
S_PASSWD_READ   user3    OK          Tue Oct 26 09:45:54 2004 su
        audit object read event detected /etc/security/passwd
USER_SU         user3    OK          Tue Oct 26 09:45:58 2004 su
        root
FS_Chdir        user3    OK          Tue Oct 26 09:45:58 2004 su
        change current directory to: /
```

... open, read, and close without saving /etc/security/passwd

```
FILE_Unlink     user3    OK          Tue Oct 26 09:46:08 2004 vi
        filename /var/tmp/Ex20582
S_PASSWD_READ   user3    OK          Tue Oct 26 09:46:08 2004 vi
        audit object read event detected /etc/security/passwd
S_PASSWD_READ   user3    OK          Tue Oct 26 09:46:08 2004 vi
        audit object read event detected /etc/security/passwd
FILE_Unlink     user3    OK          Tue Oct 26 09:46:12 2004 vi
        filename /var/tmp/Ex20582
```

...open, read, save, and close /etc/security/passwd

```
FILE_Unlink     user3    OK          Tue Oct 26 09:46:21 2004 vi
        filename /var/tmp/Ex20584
S_PASSWD_READ   user3    OK          Tue Oct 26 09:46:21 2004 vi
        audit object read event detected /etc/security/passwd
S_PASSWD_READ   user3    OK          Tue Oct 26 09:46:21 2004 vi
        audit object read event detected /etc/security/passwd
S_PASSWD_WRITE  user3    OK          Tue Oct 26 09:46:23 2004 vi
        audit object write event detected /etc/security/passwd
```

```
FILE_Unlink     user3    OK          Tue Oct 26 09:46:23 2004 vi
        filename /var/tmp/Ex20584
```

Observe the first highlighted S_PASSWD_READ event. This event was triggered when user3 tried to read the /etc/security/passwd file using the vi editor.

On user3's second attempt, another **vi** command was issued, and the same event, which is S_PASSWD_READ, was triggered.

Look at the third highlighted line. Previously, user3 read the file, and this time wrote something on that file. At this point, we do not know whether the actual contents of the file were changed. We only know that user3 exited from vi and saved the file.

user3 is a member of the staff group; why did the system allow user3 to view and save the /etc/security/passwd file?

This is because user3 actually did an **su** to user `root`; that is why the USER_SU event was logged. He was able to log on successfully; that is why the FS_Chdir was logged. Note that if a user logs on to the system, the user is directed to its home directory (as specified in user's properties). This is the reason why he was able to change directory to the root (/) directory.

Note that the example is actually a combination of user auditing (we still use the General class for the user3 in /etc/security/audit/config) and object auditing (the file /etc/security/passwd is audited for read and write access).

2.6 Common mistakes

It is not the intent of this section to cover all problem determination techniques. We just present some common encountered mistakes and the corresponding error messages.

Using audit commands in the wrong order can confuse the auditing system. The audit subsystem can be reset by stopping the audit subsystem (**audit shutdown** command) and erasing everything in the /audit directory (excepting trail, stream.out, and bin files, such as bin1 and bin2).

Certain errors may appear when running audit start.

- If you receive the following error message:

  ```
  ** failed setting kernel audit objects
  ```

  This occurs when there is a syntax error in the /etc/security/audit/objects file.

- If you receive the following error message:

  ```
  auditbin: ** failed backend command
  /etc/auditcat -p -o /audit/trail -r /audit/bin1
  ```

  This error can be corrected by removing or renaming the BIN files. It is sometimes helpful to run **audit shutdown** again and then to retry audit start.

- It is necessary to have the user stanza in the /etc/security/audit/config file; otherwise the following error appears when you start auditing:

  ```
  ** cannot find "users" stanza in "/etc/security/audit/config"
  ** failed setting kernel audit objects
  ```

  If it is not obvious that the stanza is missing, verify that each of the classes is defined on a single continuous line.

- If you receive the following error message:

  ```
  ** /audit is not a directory
  ```

  This is caused by the presence of an audit file audit in the (/) directory. This file should be renamed or erased, allowing the creation of an /audit directory with the **audit start** command.

- Starting the audit subsystem with the **audit on** command gives a zero return value. This means that the command was executed successfully, but the audit subsystem is not actually started. Use the **audit start** command instead.

- Restarting the audit subsystem sets /audit/stream.out to zero size. If you want to preserve it, save it someplace else, then issue the **audit start** command.

- If you receive the following error message:

  ```
  ** auditing enabled already
  A system call received a parameter that is not valid.
  ```

  Audit is already started and is running, and you are trying to start it again.

- If you receive the following error message:

  ```
  ** cannot find "streammode" keyword in "start" stanza
  in"/etc/security/audit/config"
  A system call received a parameter that is not valid.
  ```

  The streammode line of the /etc/security/audit/config was probably erased. If the file is present, check the contents for extra characters.

3

Accounting on AIX

This chapter introduces general concepts about accounting in AIX and a quick setup procedure. We also present how to control the accounting subsystem, collect data, generate reports, and understand these reports.

We describe some of the common errors and how to fix these errors. In the final part of this chapter, we summarize the commands, data, and format files that are involved in the accounting subsystem.

Note: A new Advanced Accounting subsystem is available beginning with AIX 5.3. This is presented in Chapter 5, “Advanced Accounting” on page 151.

3.1 General concepts about accounting

The accounting system enables you to collect data and generate summaries and reports about the use of various system resources.

Accounting can be used to monitor system resources such as processors, memory, and disks. By monitoring these resources we can detect shortages and thus be able to react for eliminating bottlenecks and for forecasting future resource needs.

If more than one system is used in a pool, and if there is no load-balance mechanism in place, we may detect overcharged systems or, on the contrary, some insufficiently used systems. In such cases, we do not need to add (buy) new systems or resources, but to redistribute users or charge to the unused machines, saving money.

Another kind of data collected by the accounting system is *connect-time* usage accounting, which lets us know how many users are connected to a system and for how long. The connect time data enables us to detect unused accounts, which have to be invalidated (for security reasons) or even erased to save resources. Also, the connect-time usage may enable the discovery of suspect activities (such as too many unsuccessful logon attempts) that signal that security measures should be adopted.

We also can define prime and non-prime hours and company holidays to reflect the company schedule for the current year. The same data may serve to bill users for using resources such as processors, memory, disks, printers, and any other chargeable service (using the **chargefee** command).

The accounting system lets us to distinguish between prime time and non-prime time (holidays are considered non-prime time). This feature allows charging different rates for the use of the system in prime and non-prime times.

The data collected by the accounting subsystem is used to automatically generate reports, such as daily and weekly reports. The reports can be generated at any time, using accounting-specific commands. The accounting subsystem provides tools that enable us to observe how the system reacts at a particular moment in time (for instance, when executing a specific command or task).

How does this work?

- We need to prepare the system or systems with some directories and files with the correct ownership and rights; define prime time, non-prime time, and holidays; and mark the resources we want to account for.

- Next, we start the accounting system by executing an activation command and verifying that this command is executed automatically at system startup or when the system is switched into multiuser mode.
- We also have to be prepared to stop the accounting system when the system is shut down (in order to prevent accounting data file corruption or data loss).
- We have to schedule the data collect (adding crontab entries for the desired operations), to generate automatic reports, and to perform file maintenance operations (to avoid oversized files and running out of storage space).
- If all of this is carefully planned, the accounting subsystem will collect the desired data, automatically generating daily and monthly reports with little or no assistance. If there are errors, a mail is sent to designated users (pointed by an environment variable).

We also provide a quick setup guide for the AIX accounting subsystem.

Conventions used in this chapter:

- References to the accounting commands do not indicate the path to these commands. All accounting commands are located in /usr/sbin/acct directory and linked to the /usr/lib/acct directory.
- References to the accounting report and summary files show the path starting from the /var/adm/acct directory.
- Some data files, reports, and summary accounting files, if noted like Spacct*i*_*mmdd*, have the following meaning: the *i* is an integer, *mm* is the month (01...12), and *dd* the day (01...31).
- The version of AIX used at the time of writing this book is:

```
[#][/etc/security/audit]> oslevel -r
5300-00
```

  We recommend that you use the latest maintenance level and fixes. Some future fixes may also change some of the options presented in this book, so make sure you always use the latest version of AIX manuals.

3.2 Quick setup of the accounting subsystem

The following is an overview of the steps needed to set up the basic accounting system. Refer to the commands and files noted in these steps for more specific information.

You must have the accounting package installed on your system in order to use it. Use the **lslpp** command (Example 3-1 on page 44) to check for the bos.acct package, which contains the accounting system.

Example 3-1 How to verify that the accounting package is installed

```
[#][/]> lslpp -L bos.acct
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  bos.acct                   5.3.0.1    C     F    Accounting Services
```

The files that are installed by the software package bos.acct are listed in “The files in the bos.acct package” on page 225.

If the bos.acct package is not installed on your system, install it (from a NIM server or from the installation media). In the Example 3-2 we use the **installp** command and an NFS mounted directory for the installation media. The flag -a applies, the -c flag commits the software media, and -d points to the installation media (in our case, the current directory).

Example 3-2 Installing the bos.acct data software package

```
[#][/mnt/aix53/installp/ppc]> installp -acX -d. bos.acct
```

For further details about how to install packages, refer to *AIX 5L Version 5.3, Installation Guide and Reference,* SC23-4887.

3.2.1 Starting the accounting system

1. Log on to the system as user `root`.
2. Update the execution environment with the following variable:

   ```
   export MAILCOM='mail root'
   ```

   If the shell you use for the root user is `ksh`, you can update your /.profile file with the MAILCOM variable such as in the preceding example. Use the right syntax to set the MAILCOM variable depending on the shell the root user account uses.

   There are two commands, **runacct** and **ckpacct**, that use the MAILCOM environment variable to send the errors to the configured users. For more than one user, modify the environment variable:

   ```
   export MAILCOM='mail root adm'
   ```

   In this case, the error messages from **runacct** or **ckpacct** commands (if any) are send to users `root` and `adm`, using the **mail** command.

   The /usr/sbin/acct directory is not in the root user's path by default. If you want to execute the accounting commands without using the full path, update the PATH environment variable:

   ```
   export PATH=/usr/sbin/acct:$PATH
   ```

3. Create wtmp and pacct files in the /var/adm directory. The wtmp file is used to collect connect time accounting data, and the pacct file is used to collect process accounting data. Use the **nulladm** command to ensure that each file has the correct access permission—read (r) and write (w) permission for the file owner and group, and read permission for others—by typing:

```
# cd /var/adm
# /usr/sbin/acct/nulladm wtmp pacct qacct
```

4. Update the /etc/acct/holidays file to include the hours you designate as prime time and to reflect your holiday schedule for the year. For example, if you are located in the United States, just check the year and you may leave the file as it is. If not, follow the steps below.

   Note: Comment lines can appear anywhere in the file as long as the first character in the line is an asterisk (*).

   To define prime time, fill in the fields on the first data line (the first line that is not a comment), using a 24-hour clock. This line consists of three four-digit fields, in the following order:

   – Current year (YYYY)

   – Beginning of prime time (hhmm)

   – End of prime time (hhmm)

   Leading blanks are ignored. You can enter midnight as either 0000 or 2400. For example, to specify the year 2004, with prime time beginning at 8:00 a.m. and ending at 5:00 p.m., enter:

```
2004 0800  1700
```

   To define the company holidays for the year, use the next data line. Each line contains four fields, in the following order:

```
Day of the year
Month
Day of the month
Description of holiday
```

   The day-of-the-year field contains the number of the day on which the holiday falls and must be a number from 1 through 365 (366 on leap year). For example, February 1st is day 32. The other three fields are for information only and are treated as comments.

   A two-line example follows:

```
1  Jan  1  New Year's Day
332  Nov 28  Thanksgiving Day
```

5. If you are using long user names (new in AIX 5.3), make sure you create the /var/adm/acct/sumx and the /var/adm/acct/nitex directories.

   If you do not use the -X option to start the daily and monthly procedures (if you are not using long login user names) and if you do not have the nitex and sumx directories, you may get the error in Example 3-3 via e-mail.

Example 3-3 Message received from lastlog command when used with -X flag

```
From daemon Thu Oct  7 04:01:03 2004
Received: (from daemon@localhost) by p630n02 (AIX5.3/8.11.6p2/8.11.0) id
i97913H25554 for root; Thu, 7 Oct 2
004 04:01:03 -0500
Date: Thu, 7 Oct 2004 04:01:03 -0500
From: daemon
Message-Id: <200410070901.i97913H25554@p630n02>
To: root
Status: O

Please create directory /var/adm/acct/nitex for use with long user name files.
Please create directory /var/adm/acct/sumx for use with long user name files.
```

   This is generated by the **lastlogin** command, called with the -X flag, when the two directories previously mentioned do not exist.

   If you are not using long user names (longer that 8 characters), as defined in the sys0 properties (**lsattr -El sys0|grep max_logname**):

```
max_logname   9     Maximum login name length at boot time        True
```

   We recommend that you edit the /usr/sbin/acct/runacct file and comment out the following line:

```
# lastlogin -X
```

   Replace it with:

```
lastlogin
```

6. Turn on process accounting by adding the following line to the /etc/rc file (or, if the line exists, but is commented, uncomment it):

```
/usr/bin/su - adm -c /usr/sbin/acct/startup
```

   The /etc/rc file is used at normal system startup, when the system is entering into multiuser mode (init 2). The startup procedure records the time when accounting was turned on and cleans up the previous day's accounting files.

> **Note:** The **startup** command is executed as user adm.

7. Identify each file system you want to include for accounting by adding the last line in Example 3-4 to the corresponding stanza in the /etc/filesystems file.

Example 3-4 Sample stanza in /etc/filesystems

```
/work:
        dev             = /dev/lv00
        vfs             = jfs
        log             = /dev/loglv00
        mount           = true
        options         = rw
        account         = true
```

Disk data accounting can be performed only on local disks or directories. For more information about how to set up disk data accounting on a local directory level for NIS (Network Information Service) users, refer to “Disk-usage accounting” on page 60.

8. Specify the data file that is used to store the accounting records for the printing subsystem by adding the following line to the corresponding queue stanza in the /etc/qconfig file:

```
acctfile = /var/adm/qacct
```

To make sure the file exists and has the right permissions and ownership use:

```
/usr/sbin/acct/nulladm /var/adm/qacct
```

Note: If you do not create the qconfig file using **nulladm**, the **qdaemon** will create the file with the wrong permissions, ownership, or both.

Do not forget to run the **/usr/lib/lpd/digest** command on the /etc/qconfig file. When the digest is completed, any changes to the /etc/qconfig file are reflected in the /etc/qconfig.bin file. To list, use:

```
enq -d
```

For more information, refer to “Printer usage accounting” on page 63.

9. As adm user, create the /var/adm/acct/nite, /var/adm/acct/sum, and /var/adm/acct/fiscal directories. These will be used to collect daily and monthly records:

```
su - adm
cd /var/adm/acct
mkdir nite sum fiscal
```

If the max_logmax option (long login user name support) is activated and you plan to use long login names for the users, create `nitex` and `sumx` directories instead of the nite and sum directories:

```
su - adm
```

```
cd /var/adm/acct
mkdir nitex sumx fiscal
```

10. Set daily accounting procedures to run automatically from root's crontab by editing the /var/spool/cron/crontabs/root file to include the **dodisk**, **ckpacct**, and **runacct** commands. For example:

```
0 2 * * 4 /usr/sbin/acct/dodisk
5 * * * * /usr/sbin/acct/ckpacct
0 4 * * 1-6 /usr/sbin/acct/runacct
              2>/var/adm/acct/nite/accterr
```

The first line starts disk accounting at 2:00 a.m. (0 2) each Thursday (4). The second line starts a check of the integrity of the active data files at 5 minutes past each hour (5 *) every day (*). The third line runs the daily accounting procedure at 4:00 a.m. (0 4) every Monday through Saturday (1-6). If these times do not fit the hours your system operates, adjust your entries.

If you use long login user names, use the -X flag:

```
0 2 * * 4 /usr/sbin/acct/dodisk -X
5 * * * * /usr/sbin/acct/ckpacct -X
0 4 * * 1-6 /usr/sbin/acct/runacct -X 2>/var/adm/acct/nite/accterr
```

11. Set the monthly accounting summary to run automatically by including the **monacct** command in the /var/spool/cron/crontabs/root file. For example, for long login user names, use:

```
15 5 1 * * /usr/sbin/acct/monacct
```

or

```
15 5 1 * * /usr/sbin/acct/monacct -X
```

Make sure you schedule this procedure early enough to allocate enough time for the report to finish. In the previous example, the procedure starts at 5:15 a.m. on the first day of each month.

Notes:

- The accounting system was started using the **startup** command executed by user adm.
- The other procedures, such as disk accounting data, the daily and monthly procedure, and the maintenance routine are launched using the root account.
- For simplicity (we do not have procedures using different accounts), we have used the root's crontab to generate the reports and summaries.
- For detailed information, see *AIX 5L Version 5.3, System Management Guide: Operating System and Devices,* SC23-4910.

3.2.2 Stopping the accounting subsystem

1. Delete the root's crontab entries belonging to the accounting system:

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk
   5 * * * * /usr/sbin/acct/ckpacct
   0 4 * * 1-6 /usr/sbin/acct/runacct 2>/var/adm/acct/nite/accterr
   ```

   Alternately, you may use **crontab -e** to invoke vi to edit the crontab entries.

2. Delete or comment out the line responsible for starting the accounting system from /etc/rc file:

   ```
   /usr/bin/su - adm -c /usr/sbin/acct/startup
   ```

3. Comment out the lines `acctfile = /var/adm/qacct` in /etc/qconfig and `account = true` in /etc/filesystem.

4. Stop the accounting system by executing this command:

   ```
   /usr/sbin/acct/shutacct
   ```

Note: When stopping or rebooting the system, you should avoid using the **reboot** or **halt** commands. These commands do not properly stop the accounting subsystem and may cause inconsistency in the accounting data files or loss of accounting data. Use the **shutdown** command instead.

5. For cleaning up the system, run the **/usr/sbin/acct/remove** command. (See the files erased by this command in 3.4.1, "Collecting data" on page 52.)

3.2.3 Long login user name support in AIX 5.3

A new feature of the AIX 5.3 is the support for long login user names. This means that user accounts can be created having login names longer than 8 characters. Use the **lsattr** command to verify whether this feature is activated:

```
[#][/]> lsattr -E -l sys0|grep max_logname
max_logname     255 Maximum login name length at boot time            True
```

To activate this feature, if not active, use:

```
[#][/]> chdev -l sys0 -a max_logname=255
sys0 changed
```

If the feature is activated and you really use long login file names, when you use accounting commands you should check to see whether the -X option is available. This option allows the treatment of long login file names by the accounting system. In such case, add the -X option to all automatic procedures:

```
#=================================================================
#      PROCESS ACCOUNTING:
#  runacct at 11:10 every night
```

```
#  dodisk at 11:00 every night
#  ckpacct every hour on the hour
#  monthly accounting 4:15 the first of every month
#=================================================================
10 23 * * 0-6 /usr/lib/acct/runacct -X 2>/usr/adm/acct/nite/accterr >
/dev/null
0 23 * * 0-6 /usr/lib/acct/dodisk -X > /dev/null 2>&1
0 * * * * /usr/lib/acct/ckpacct -X > /dev/null 2>&1
15 4 1 * * /usr/lib/acct/monacct -X > /dev/null 2>&1
#=================================================================
```

The daily report and summary files are generated in the /var/adm/acct/nitex and /var/adm/acct/sumx directories. These directories should be created with `adm.adm` ownership.

3.3 Accounting internals

1. What files are involved in the accounting system?

   There are three kind of files involved in the accounting system:

   – The *command* files, which are stored in the directory /usr/sbin/acct. The same file names can be found in /usr/lib/acct directory; they are actually soft linked to the files in /usr/sbin/acct (for SystemV compatibility).

   – The *data* files, stored in the /var/adm directory:

   - The pacct file for process accounting
   - The wtmp file for connect time, reboots, date change, and init multiuser level changes accounting
   - The dtmp file for disk accounting
   - The fee file for extra services provided to the users
   - The qacct file for printing accounting

   – The *report and summary* files, stored in the /var/adm/acct directory:

   - The nite subdirectory contains files that the **runacct** command reuses daily.
   - The sum subdirectory contains the cumulative summary files that the **runacct** command updates daily and the daily reports that are generated by the same **runacct** procedure.
   - The fiscal subdirectory contains the monthly summary files that the **monacct** command creates.

Report and summary files are more numerous; for details, refer to 3.7, “Accounting files” on page 116 and to 3.7.3, “Report and summary files” on page 120.

There are also the daily report files /var/adm/acct/sum/rprt*mmdd* and the monthly report file in /var/adm/acct/fiscal/fiscrpt*mm* (in ASCII format).

2. How do I display the binary files that are generated by the accounting system?

   The accounting system uses the following binary formats:

   – The *wtmp* binary file format. The files /var/adm/wtmp and /etc/utmp are such files. Use the **fwtmp** command to examine the content of these files. See Example 3-7 on page 57 about using **fwtmp** to display wtmp file content.

   – The *tacct* binary file format. The file /var/tmp/acct/sum/tacct is such a file. Use the **prtacct** command to display it. See Example 3-20 on page 90.

   – The *qacct* binary file in *accrec* format. The **pac** command displays the content of the qacct file. Example 3-42 on page 110 shows **pac** command sample output.

3. How do I manage the large files that are generated by the accounting subsystem?

   The accounting subsystem prevents the generated files from growing too large. For example, the pacct file can grow rapidly so the accounting system uses the **ckpacct** program to split this file and check whether there is enough disk space available. However, this does not happen with the wtmp file, which is emptied during the daily run of the **runacct** command.

4. What are the most common issues related to the accounting system?

   The most common issues are related to insufficient disk space in the /var file system, corrupted accounting binary data files, date changes in the system causing wtmp file inconsistencies, and system stops.

   We recommend using the **shutdown** command to stop the system if the accounting system is activated, as this properly shuts down the accounting system so that no data is lost and no accounting data file is damaged.

5. Why can I not see the program currently with the **acctcom** command?

   Because **acctcom** shows only terminated processes. The exit routine of the process actually writes the accounting data to the pacct file. The **acctcom** command reads that file. If your process is still running, it does not appear yet in the pacct file.

   To see the running processes, use the **ps** command.

6. How do I modify the default value of $0.02 per printed page used by the accounting system?

   Use the **pac** command with the -p flag.

3.4 Collecting and reporting data

We activate the accounting subsystem using the **/usr/sbin/acct/startup** command. Ensure that the proper line exists in the multiuser level init file /etc/rc so that the accounting is reactivated when restarting the machine.

New entries should be added to the root's crontab to generate daily and monthly reports. The procedures for starting, stopping, and generating automatic reports are described in 3.2, "Quick setup of the accounting subsystem" on page 43.

Usually the raw accounting data is stored in binary files, specific for each resource. All of these resources are converted, on a daily basis, into total accounting reports (ranged by user) and merged into daily reports.

The total accounting reports (called monthly or fiscal reports) are generated at the end of the fiscal period or at the end of the month. The other features described above are accessible by commands and described later in this chapter.

3.4.1 Collecting data

There are several types of accounting data that you may want to collect:

Connect-Time Accounting
: This accounting data is about what users used the system, when and for how long they stayed logged in, and other events such as system reboots, date changes, or multiuser level changes.

Process and Command Accounting
: Any terminated process logs information about itself, such as the user ID of the process, the command that started the process, the CPU and memory used, and more.

Disk-Usage Accounting
: The data stores are records containing the user ID and the local disk block used.

Printer-Usage Accounting
: The data record stores the user name, the number of pages printed, and an estimated cost.

Fee Accounting Fee charges per user for extra services.

The accounting subsystem is started using the **startup** command and is stopped by the **shutacct** command. These commands, **shutacct**, **startacct**, and as we will see later, **turnacct**, are interfaces for another basic command that is actually used to start the accounting system: the **accton** command.

Figure 3-1 shows how these commands interact. When starting the accounting system, **startup** writes a message to wtmp, activates the kernel accounting routines to write to the pacct file to ensure that it is writable, and erases some old files.

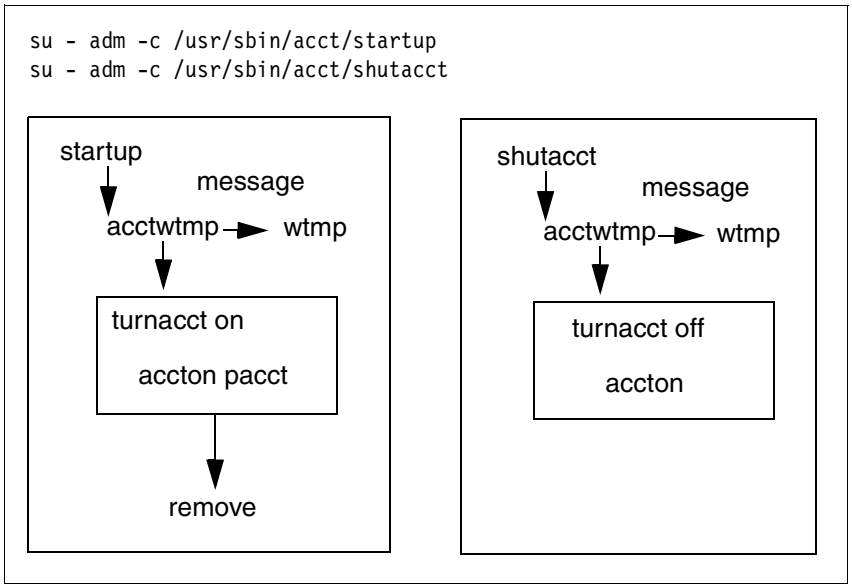

Figure 3-1 How the startup and shutacct commands work

When stopping the accounting, **shutacct** writes a message to the wtmp file and stops the kernel routines to write to the pacct file.

The **startup** script calls **acctwtmp** to write the starting event to the file /var/adm/wtmp, calls the **turnacct** script with the argument on, and finally the **remove** script.

The **shutacct** script calls **acctwtmp** to write the event to the /var/adm/wtmp file and calls the **turnacct** script with the argument off.

The **remove** command does some cleanup when the accounting system is started:

```
rm -f /var/adm/acct/sum/wtmp*
```

```
rm -f /var/adm/acct/sum/pacct*
rm -f /var/adm/acct/nite/lock*
rm -f /var/adm/acct/sumx/wtmp*
rm -f /var/adm/acct/sumx/pacct*
rm -f /var/adm/acct/nitex/lock*
```

The **turnacct** command is another interface to the **accton** command, and it is used to start, stop, and switch the pacct files using the on, off, or switch arguments. The switch argument, used by the **turnacct** command, moves the file pacct into pacct*i*, where *i* is an integer that starts from 1 and is incremented for each such call, and enables the accounting subsystem to write into a new empty pacct file. In fact, the **turnacct** command is another interface to the same **accton** command.

Basically, the accounting system is started using **accton** *file* (which usually is **accton pacct**). To stop the accounting, execute **accton** with no arguments. We use **statup** and **shutacct** commands to properly start or stop the accounting system.

Connect-time accounting

The following files are involved in connect-time accounting:

- /etc/utmp
- /var/adm/wtmp
- /etc/security/lastlog
- /etc/security/failedlogin

By default, the file /var/adm/wtmp does not exist, so you have to create it when starting the accounting system using the **nulladm** command. This file stores persistent records across reboots about connect-time accounting. The file /etc/utmp has the same structure as the file /var/adm/wtmp, but it is voided when the machine restarts.

Connect-time accounting is activated if the /var/adm/wtmp file exists, has proper ownership and rights, and the accounting system has been started. The following information is recorded in the wtmp file:

- The user process creation (login, init process) and termination
- Boot records (**reboot** and **shutdown** commands)
- Changes in run levels
- The turn on and off of the accounting system, if this is done using the **startup** and **shutacct** commands
- Time adjustments, such as using the **date** command to change the system date time

Login information

The utmp, wtmp, failedlogin, and lastlog files contain records about users who are connected (or trying to connect) to the system.

When a user attempts to logs in, the **`login`** program writes entries in these two files:

- /etc/utmp, which contains a record of users logged into the system
- /var/adm/wtmp (if it exists), which contains connect-time accounting records

If the login is successful, the login process prints information about the last login of the user ID in /etc/security/lastlog, then updates this file with the current login information. The /etc/security/lastlog file is an ASCII file that contains stanzas with the last login attributes for users.

On an invalid login attempt, due to an incorrect login name or password, the login program makes an entry in the /etc/security/failedlogin file, which contains a record of unsuccessful login attempts.

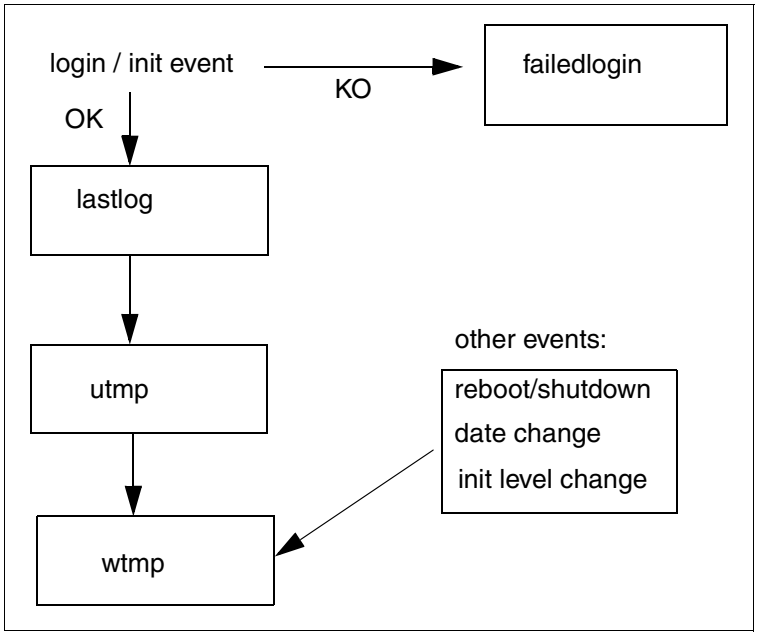

Figure 3-2 Connect-time accounting process

The records in these files follow the utmp format, which is defined in the utmp.h header file. See “The utmp file format” on page 219.

The following commands write in /var/adm/wtmp: **init**, **login**, **shutdown**, **reboot**, **acctwtmp** and others such as **telnet**, **rlogin**, **xterm**/**aixterm**, and **ttymon** etc., that exec the **login** command.

When a user logs on to a system, the **login** program looks for the user ID entry in the /etc/security/lastlog file. If the user ID is found, the login program prints to the standard output:

- User name
- Time stamp of the last login
- Login terminal
- Host name that initiated the login session

Example 3-5 Lookup in lastlogin for a Telnet session

```
[#][/]> telnet 192.168.100.32
\Trying...
Connected to 192.168.100.32.
Escape character is '^]'.

telnet (p630n02)
...
AIX Version 5
(C) Copyrights by IBM and by others 1982, 2004.
login: root
root's Password:
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 5.3!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*******************************************************************************
Last unsuccessful login: Mon Oct 18 18:00:10 CDT 2004 on ssh from 9.12.6.177
Last login: Tue Oct 19 14:49:53 CDT 2004 on /dev/pts/2 from p630n03

[#][/]>
```

The login program writes the new login records in the lastlog file.

Example 3-6 Example of entry for the root user in the /etc/security/lastlog file

```
root:
        time_last_login = 1097610493
        tty_last_login = ssh
        host_last_login = 9.12.6.177
```

```
        unsuccessful_login_count = 0
        time_last_unsuccessful_login = 1097266466
        tty_last_unsuccessful_login = ftp
        host_last_unsuccessful_login = ::ffff:9.12.6.176
```

The next file to be updated is /etc/utmp. This file contains the connect-time information about the current users logged on to the system. When the user logs out, the same record is written without the user name.

The last file to be updated is /var/adm/wtmp (if it exists) with the same record as the utmp file. When the user logs out, this file is updated with the same record but with null user name and host fields.

This file records the date change events (the execution of the **date** command), system restarts (**shutdown** and **reboot** commands), changes in run levels, and starting and stopping the accounting subsystem (if done using the **startup** and **shutacct** commands).

Example 3-7 Entries in wtmp file when a user logs on an exit

login in utmp:

```
freeware pts/3 pts/3 7 18202 0000 0000 1097617621 9.12.6.176 Tue Oct 12
16:47:01 CDT 2004
```

login in wtmp, the same as in utmp:

```
freeware pts/3 pts/3 7 18202 0000 0000 1097617621 9.12.6.176 Tue Oct 12
16:47:01 CDT 2004
```

logout in wtmp:

```
freeware pts/3 pts/3 7 25640 0000 0000 1097617574 9.12.6.176 Tue Oct 12
16:46:14 CDT 2004
         pts/3 pts/3 6 18202 0000 0000 1097617609 9.12.6.176 Tue Oct 12
16:46:49 CDT 2004
```

The records that are written in these files have the same structure, defined in “The utmp file format” on page 219. When a user tries to log on to a system and does not succeed, the **login** program writes a record of the same type (utmp) in the /etc/security/failedlogin file.

The binary files wtmp and utmp can be read using the **/usr/sbin/acct/fwtmp** command, which converts the binary utmp and wtmp files to ASCII format.

Example 3-8 Using fwtmp to display the contents of the wtmp binary file

```
[p630n02][/var/adm]> fwtmp < wtmp
               openacct 9     0 0000 0000 1097830800              Fri Oct 15 04:00:00 CDT 2004
```

```
root    pts/0  pts/0    7 18064 0000 0000 1097609868 9.12.6.176    Tue Oct 12 14:37:48 CDT 2004
root    pts/1  pts/1    7 17384 0000 0000 1097703849 9.12.6.176    Wed Oct 13 16:44:09 CDT 2004
tester1 pts/2  pts/2    7 27672 0000 0000 1097793534 9.12.6.177    Thu Oct 14 17:38:54 CDT 2004
root    pts/3  pts/3    7 25698 0000 0000 1097619235 192.168.100.1 Tue Oct 12 17:13:55 CDT 2004
root    pts/4  pts/4    7 21750 0000 0000 1097701122 192.168.100.1 Wed Oct 13 15:58:42 CDT 2004
tester2 pts/5  pts/5    7 28286 0000 0000 1097793550 9.12.6.177    Thu Oct 14 17:39:10 CDT 2004
root    pts/6  pts/6    7 20148 0000 0000 1097613620 192.168.100.1 Tue Oct 12 15:40:20 CDT 2004
root    pts/10 pts/10   7 29942 0000 0000 1097786162 9.12.6.143    Thu Oct 14 15:36:02 CDT 2004
root    pts/11 pts/11   7 30338 0000 0000 1097786615 9.12.6.176    Thu Oct 14 15:43:35 CDT 2004
```

The following fields are displayed:

- User login name
- Line identification number (from the /etc/inittab file)
- Device name (for example, console or tty23)
- Type of entry
- Process identification number
- Process termination status
- Process exit status
- Session starting time (numeric)
- Host machine name
- Starting date and time (in date/time format)

Note: If the user name (used to log on to the system) does not exist, the user name that will be logged in /etc security/failedlogin is UNKNOWN_USER. This is to avoid recording user passwords if, by mistake, you typed in the password instead of the user name. See the highlighted lines in Example 3-9.

Example 3-9 Example of /etc/security/failedlogin file

```
[p630n02][/etc/security]> fwtmp < failedlogin |head
root     pts/1 7 11798 0000 0000 1096987087 192.168.100.1       Tue Oct  5 09:38:07 CDT 2004
root     pts/0 7 15510 0000 0000 1097006142 9.12.6.177          Tue Oct  5 14:55:42 CDT 2004
tester1  pts/4 7 22030 0000 0000 1097091171 9.12.6.177          Wed Oct  6 14:32:51 CDT 2004
tester2  pts/5 7 21892 0000 0000 1097098847 9.12.6.177          Wed Oct  6 16:40:47 CDT 2004
root     pts/6 7 25436 0000 0000 1097162357 192.168.100.1       Thu Oct  7 10:19:17 CDT 2004
root     FTP   7 27056 0000 0000 1097163047 ::ffff:9.12.6.177   Thu Oct  7 10:30:47 CDT 2004
root     pts/0 7 27090 0000 0000 1097164620 9.12.6.177          Thu Oct  7 10:57:00 CDT 2004
UNKNOWN_ pts/0 7 27090 0000 0000 1097164622 9.12.6.177          Thu Oct  7 10:57:02 CDT 2004
root     pts/7 7 18548 0000 0000 1097167668 192.168.100.34      Thu Oct  7 11:47:48 CDT 2004
UNKNOWN_ pts/7 7 22578 0000 0000 1097168647 node6               Thu Oct  7 12:04:07 CDT 2004
[p630n02][/etc/security]>
```

When a user logs out, the **init** program writes logout records into these two files, wtmp and utmp. The records have the same format, utmp, and differ from login records in that they have a blank user name record.

System reboots

When the system shuts down by executing the **shutdown** command, this command calls the **shutacct** command in order to properly turn off the accounting system, which calls the **acctwtmp** command. This command writes a record in the /var/adm/wtmp. See Example 3-10 for shutdown and system boot entries in the wtmp file.

Note: When the system reboots, it creates a new /etc/utmp file. This is why the /var/adm/wtmp file is necessary, to log the connect time accounting records across reboots. When accounting is activated, the wtmp file should be created with the **nulladm** command as described in “Quick setup of the accounting subsystem” on page 43.

Example 3-10 Shutdown and system boot in wtmp

```
shutdown tty0 0 0 0000 0000 1097609325 Tue Oct 12 14:28:45 CDT 2004
system boot 2 0 0000 0000 1097609800 Tue Oct 12 14:36:40 CDT 2004
```

Date change and init level changes

When the system time is modified using the **date** command, or the multiuser level of the system is changed, a record is written to the /etc/utmp and /var/adm/wtmp files.

Example 3-11 Date change, init level change, and reboot records

Date change in wtmp:

```
   old time      3     0 0000 0000 1097615253 Tue Oct 12 16:07:33 CDT 2004
   new time      4     0 0000 0000 1097616088 Tue Oct 12 16:21:28 CDT 2004
```

Run level change in wtmp:

```
run-level 2 1 0 0062 0123 1097609800 Tue Oct 12 14:36:40 CDT 2004
```

Process and command accounting

Two conditions must be met to start the process accounting: A file called /var/adm/pacct with proper ownership and rights must exist, and the accounting subsystem must be started.

As `root`, create the file /var/adm/pacct to set up the accounting system:

```
   /usr/sbin/acct/nulladm /var/adm/pacct
```

Note: The ownership of the pacct file is adm.adm. The **nulladm** command even when executed as root, ensures the correct ownership and rights for the files that it creates. The **nulladm** program is actually a shell script that does the following:

```
cp /dev/null file_to_process
chmod 644 file_to_process
chown adm file_to_process
```

Here, *file_to_process* is passed as argument to the **nulladm** command. This provides the correct access (644 and owner adm) to the pacct and wtmp files.

A record is written to the pacct file each time a process finishes. It includes:

- Process type (for example, child process)
- The exit status indicating how the process terminated
- The user ID number
- The group ID number
- Terminal from which the process originated
- Start, user, system, and CPU time
- The mean memory used
- The total number of I/O characters transferred
- The total number of 1024-byte blocks read or written
- The name of the command used to start the process

The fields of the records written to pacct file are those defined in the tacct format, as described in “The tacct file format” on page 219.

Disk-usage accounting

Disk accounting data can be generated only for local disks or directories. No accounting data is generated if you try to do disk accounting on an NFS mounted directory.

Disk-accounting data is collected by the **dodisk** command, launched by the cron daemon:

```
0 2 * * 4 /usr/sbin/acct/dodisk
```

The disk-usage records are stored temporarily in the /var/adm/dtmp file, in ASCII format, and in the /var/adm/acct/nite/dacct file. This last file is in binary tacct format file and the structure of this file is described in “The tacct file format” on page 219.

Note: The **dodisk** command overwrites the files /var/adm/dtmp and /var/adm/acct/nite/dacct, so be aware if you launch it twice in the same day.

The dtmp file is an intermediate ASCII data file and is used to generate the binary dacct file. Figure 3-3 presents a diagram of the disk accounting process.

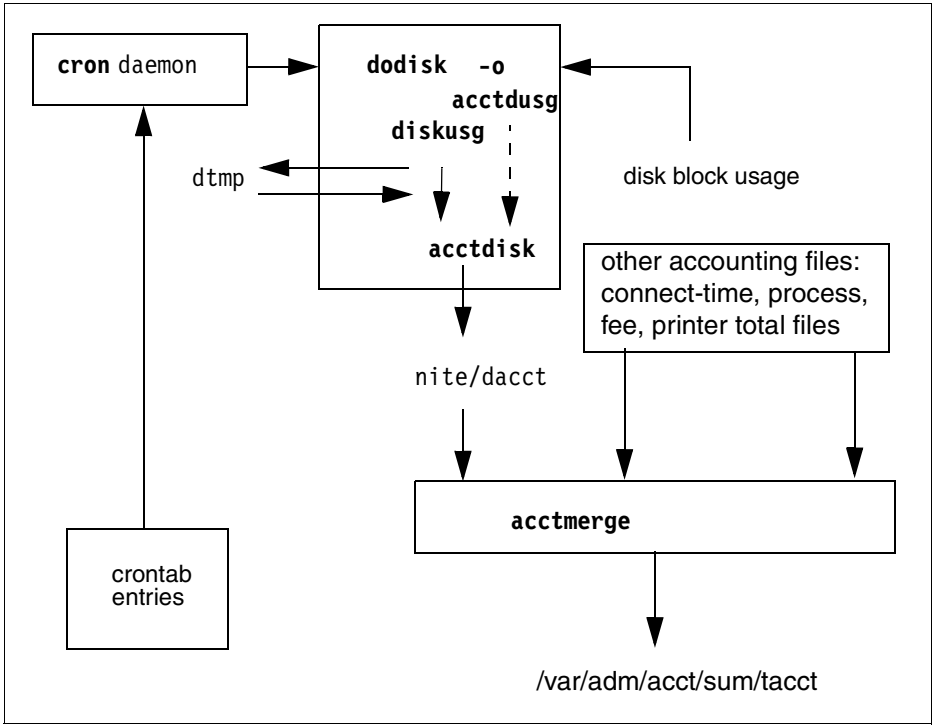

Figure 3-3 Collecting disk block accounting data

The final dacct disk accounting data file is generated in two steps:

1. Generating the /var/tmp/dtmp file using **diskusg** or **acctdusg**.

   a. First, if the -o flag is not used, the **diskusg** command generates accounting data in the dtmp file. Disk usage accounting is performed only for the filesystems that have the acct = true attribute in the /etc/filesystems file. Use this command to perform disk accounting for local data (on local disks) and local users in /etc/passwd.

   b. If the -o flag is used with **dodisk**, the command uses **find** ***args*** **-print** to generate the input for the **acctdusg** command, which writes the output in the dtmp file. The ***args*** parameter is a list of directories for which we want to generate disk accounting. If empty, the / (root) directory is used. The **acctdusg** command is slower but more thorough than the **diskusg** command; on the other hand, it can be used to do disk accounting at directory level.

Note: If the **dodisk -o** command is used:

- If you execute the command as adm, you may get error messages due to access permission to certain files. This is normal because the user adm does not have rights in certain directories.
- If you execute the command as root, you still get error messages complaining about file access in the /proc file system.

```
[#][/]> dodisk -o
find: 0652-023 Cannot open file /proc/10608.
find: 0652-023 Cannot open file /proc/14714.
find: 0652-023 Cannot open file /proc/21872.
/proc/25726/fd/8: A file or directory in the path name does not
exist.
```

  This happens because the **find** command, inside **dodisk**, is executed from the (/) directory, without excluding /proc.

  Normally, to avoid the /proc directory, the **find** command should be executed excluding this:

```
find /-name proc -prune -o -print
```

Both commands, **diskusg** and **acctdusg,** accept the -p argument, so we can use a different password file than the local one, /etc/passwd, such as one that is generated with the **ypcat passwd** command. Example 3-12 on page 63 shows how to create a password file from NIS password map.

2. In the second (final) step, **acctdisk** converts the dtmp generated file into a binary tacct file in the nite/dacct file, or in nitex/dacct file if the -X flag is used. The command **acctmerg**, launched by the **runacct** daily procedure, merges the disk data into the total tacct daily file.

The **dodisk** command accepts zero or more parameters:

- The -X flag. If used, it checks for the existence of the nitex and sumx directories. The result file, dacct, is put in nitex directory. This is used for providing long login user name support.
- The -o flag. Disk usage accounting is performed using the **acctdusg** command at directory level. This is possible because **acctdusg** is fed with file names generated by the **find** ***args*** **-print** command, where ***args*** are local directories. This is useful for remotely mounted filesystems or disk accounting at directory level.
  - The -o flag and arguments: **dodisk -o** ***args***. The *args* parameters should be directories and the output of the **find** ***args*** **-print** command is passed (piped) to the **acctdusg** command.

– The -o flag and no arguments: **dodisk -o**. The / (root) is used as parameter for the find command: **find / -print**.

If the -o flag is not present, then the **diskusg** command is used to generate disk usage statistics at filesystem level.

– **dodisk** without any arguments: in this case, the /etc/filesystem is checked for the attribute `account = true`. Disk accounting data is generated only for those filesystems that have this attribute.

– **dodisk** ***args*** (without the -o flag). In this case, the *args* parameters should be directories and are passed as arguments to the **diskusg** command. The filesystem stanza with the attribute `account = yes` is ignored.

Note: Both commands, **diskusg** and **acctdusg**, work only for *local* filesystems and directories. If you have a remotely mounted filesystem, you cannot get disk accounting statistics using these two commands.

In conclusion, use **dodisk -o** if you want disk usage statistics at directory level for local filesystems. For local usage at filesystem level, use **dodisk** and no -o flag. If long user login names are activated, add the -X flag. If NIS is used to store user names, modify the following lines in **dodisk** script as shown:

- If **diskusg** is used:

```
diskusg $XFLAG $args > dtmp to
diskusg -p password_file $XFLAGS $args > dtmp
```

- If **acctdusg** is used:

```
find $dir -print | acctdusg $XFLAG to
find $dir -print | acctdusg -p password_file $XFLAG
```

In both cases, *password_file* is the file that is used to generate the NIS passwd file, including its path. If you not have access to this file, use the NIS passwd map to generate one (Example 3-12).

Example 3-12 Creating a password file from NIS table passwd.byname

```
ypcat passwd|awk -F: 'OFS=":"{print($1,"",$3,$4,$5,$6,$7)}' > password_file
```

This command does not show the encrypted password in your *password_file*.

Printer usage accounting

The printer accounting records related to printer usage are stored in the /var/adm/qacct file. The **qdaemon** command writes a record to the qacct file, containing:

- The login user name
- The number of pages printed

- The number of time the printer was used by the user
- The total cost for the pages printer (default $0.02 per page)

The structure of the qacct file is defined in the /usr/include/sys/accrec.h header file. For details, see “The accrec file format” on page 224.

Printer usage accounting is done only for the queues that have the `acct = on` attribute defined in the queue definition stanza.

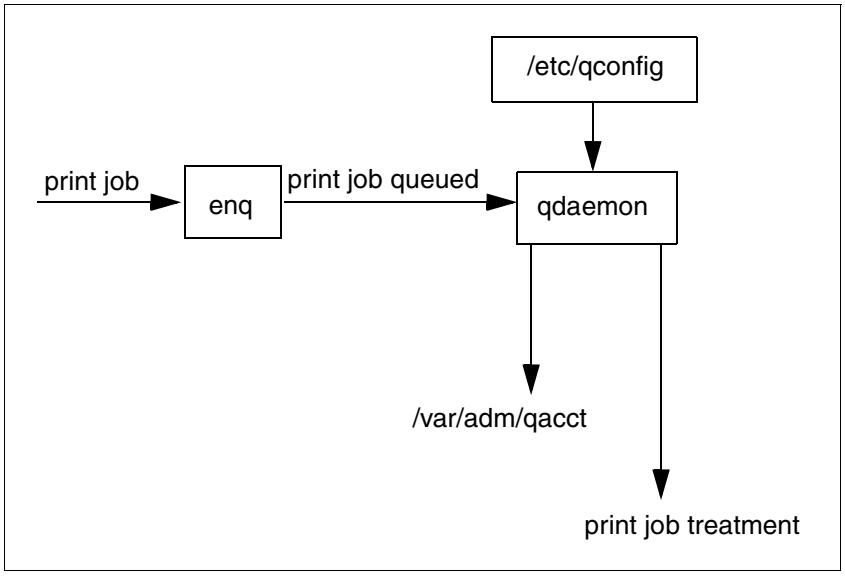

Figure 3-4 Generating print accounting data

You should know that printer accounting does not work in all printing environments. Therefore, a more in-depth explanation of how the queuing system calculates the number of pages printed by a job is presented in the following paragraph. You will then be able to verify whether the system will collect correct printer-usage data.

The queuing component that calculates the number of pages printed by a job is the **`/usr/lib/lpd/pio/etc/pioformat`** program. The **`pioformat`** program is the device-independent formatter driver for the queuing system. It is called by /usr/lib/lpd/piobe, and it dynamically loads, links, and drives the appropriate device-dependent formatter to process the job’s specific data stream type (such as PostScript, ASCII, GL, or PCL).

In fact, **`pioformat`** calculates the number of pages printed only if it is responsible for transforming the input data stream into the printer’s data stream type. In other

words, **pioformat** does not calculate the number of pages printed if it has been called with the pass-through option or is using the pass-through formatter.

In fact, printer accounting functions correctly if you are using a PCL or ASCII printer. If your printer is a PostScript printer, you will not be able to gather printer usage data. That is the case even if you are printing an ASCII file, because the transformation from ASCII to PostScript is done with the **enscript** command, not the **pioformat** command.

Printer accounting also does not work if the queue's backend processor does not call **pioformat**, even if it is a PCL or ASCII printer. That is the case when you define a remote printer with no local filtering. The backend processor for such print queues is /usr/lib/lpd/rembak, which does not call **pioformat**.

Fee accounting

If services are provided to the user that are not among those managed by the accounting system (such as installing a new bundle or recovering files) you still may use the accounting system to charge that user for the service by using the **chargefee** command. Such an entry consists of:

- User login name
- User ID
- The number of units charged to the user

The total for the accounting records is stored in the /var/adm/fee file. This is in fact an ASCII file that contains the total figures for the accounting records.

Launch the command **chargefee** ***username units*** to produce the total accounting record, where *username* is the login use name for which we want to charge *units* units, like this:

```
[#][/]> chargefee tester2 10
[#][/]>
```

The command tests the existence of the user in the /etc/passwd file. If the user is not there, it checks whether NIS is used, and if it is, it checks for the user in the map passwd.byname. If the user does not exist, it gives an error:

```
[#][/]> chargefee tester10 10
ypmatch: 1831-150 Cannot match key tester10 in map passwd.byname.
        Reason: no such key in map.
chargefee: 0850-001 cannot find login name tester10
```

If the user is found in /etc/passwd or in the map passwd.byname (if NIS is running), then the system responds with the command prompt, silently adding the record to the /var/adm/fee file.

3.4.2 Reporting data

The accounting subsystem can automatically generate daily reports and summary files using the **runacct** procedure, and monthly reports using **monacct** procedure. These procedures are started by the **cron** daemon. However, we can generate reports and summaries at any time using the data accounting files and commands.

Generating daily reports using the runacct command

After the accounting data is collected in the accounting data files, the records are processed and converted into reports and summaries. The daily routine that automatically generates these reports and summaries is the **runacct** command. This command is usually launched via cron, but, if there are errors, the reports will not be generated; thus you must fix these errors and the command can be launched manually. If the environment variable *MAILCOM* is defined and has the value *mail root adm*, and if there are errors, an e-mail is sent to the root and adm users.

Dealing with big numbers

First, a few words about numbers. Some collected data is large enough that it is hard to print or display in a convenient form. We prefer to have tables with aligned rows and columns that are easy to follow.

To solve this problem, the accounting commands automatically convert records into scientific notation when numbers become large. A number is represented in scientific notation in the following format:

```
Basee+Exp
Basee-Exp
```

This is the number equal to the base number multiplied by 10 to the +exp or -exp power. For example:

```
The scientific notation 1.345e+9 is equal to 1.345x10^9, or 1,345,000,000.
The scientific notation 1.345e-9 is equal to 1.345x10^-9 or, 0.000000001345
```

Sample report

Example 3-13 on page 68 shows a daily report generated by the **runacct** procedure. The report is actually generated by the **prdaily** procedure, which gets data from different files, merges them, and formats them properly. The daily report has five parts:

1. The content of the nite/reboot file, noted as `the reboot file` in Example 3-13 on page 68, showing the restart of the system, changes in the multiuser run level, starting and stopping of the accounting system, date changes, or other messages written to the wtmp file.

2. The content of the nite/lineuse file (marked `the lineuse file` in Example 3-13 on page 68), generated by acctcon1, showing summary statistics about system line usage. The report contains the following information:

   LINE Console, tty, or pty in use

   MINUTES Total number of minutes the line was in use

   PERCENT Percentage of time in the accounting period that the line was in use

   # SESS Number of new login sessions started

   # ON Same as # SESS

   # OFF Number of logouts plus interrupts made on the line

3. The third part of the report prints the tacct records containing process, disk, printer, fee, and total connect time statistics, using the **`prtacct`** command called inside **`prdaily`** and the file sum/daytacct. These fields are displayed:

   UID User ID

   LOGIN NAME User name

   CPU (PRIME/NPRIME)
   Total CPU time for all of the user's processes in minutes

   KCORE (PRIME/NPRIME)
   Total memory used by running processes, in kilobyte-minutes

   CONNECT (PRIME/NPRIME)
   Total connect time (how long the user was logged in) in minutes

   DISK BLOCKS Average total amount of disk space used by the user on all filesystems for which accounting is enabled

   FEES Total fees entered with the **`chargefee`** command

   # OF PROCS Total number of processes belonging to this user

   # OF SESS Number of distinct login sessions for this user

   # DISK SAMPLES Number of times disk samples were run during the accounting period; value is zero if no DISK BLOCKS are owned

4. The `command summary` shows the content of the nite/daycms file. The following fields are displayed:

   COMMAND NAME The command that was executed

   NUMBER CMDS The number of times the command executed

TOTAL KCOREMIN	Total memory used by running the command, in kilobyte-minutes
TOTAL CPU-MIN	Total CPU time used by the command in minutes
TOTAL REAL-MIN	Total real time elapsed for the command in minutes
MEAN SIZE-K	Mean size of memory used by the command per CPU minute
MEAN CPU-MIN	Mean number of CPU minutes per execution of the command
HOG FACTOR	Measurement of how much the command dominates the CPU while it is active: the ratio of TOTAL CPU-MIN over TOTAL REAL-MIN
CHARS TRNSFD	Number of characters transferred by the command with system reads and writes
BLOCKS READ	Number of physical block reads and writes performed by the command

5. The content of the nite/loginlog file. A 0000-00-00 in the loginlog file means that the user never logged in. This part displays two fields:

The first field	In YY-MM-DD, indicates the most recent login for the specified user
The second field	The name of the user account

Example 3-13 Example of daily report generated by runacct command

```
Thu Oct  7 04:00:03 CDT 2004 DAILY REPORT FOR AIX Page 1
```

First part: the reboot file

```
from Wed Oct  6 04:00:00 CDT 2004
to   Thu Oct  7 04:00:01 CDT 2004
2       date changes
1       openacct
2       accting off
2       system boot
2       run-level 2
1       Just testing acctwtmp
2       AIX, acctg
1       runacct
1       acctcon1
```

Second part: the lineuse file

```
TOTAL DURATION: 1440 MINUTES

LINE    MINUTES PERCENT # SESS  # ON    # OFF
```

```
ftp19244        0       0       0       0       2
ftp24762        0       0       0       0       2
pts/0   1440    100     2       1       1
pts/2   1440    100     1       1       1
pts/3   1440    100     1       1       1
pts/4   1440    100     2       1       1
pts/5   679     47      2       1       1
TOTALS  6439    --      8       5       9
...
Thu Oct  7 04:00:03 CDT 2004 DAILY USAGE REPORT FOR AIX Page 1
```

Third part: process accounting

```
LOGIN   CPU   CPU    KCORE KCORE  CONNECT CONNECT DISK      FEES    # OF   # OF # DISK
UID  NAME    PRIME NPRIME PRIME NPRIME PRIME   NPRIME  BLOCKS            PROCS  SESS SAMPLES

0    TOTAL    1    1      1788  8086   2179    4260    2.983e+06 0       142195 8    14
0    root     1    1      1784  8086   2014    2940    1.284e+06 0       8740   6     1
1    daemon   0    0      0     0      0       0       8         0       19     0     1
2    bin      0    0      0     0      0       0       1.160e+06 0       0      0     1
3    sys      0    0      0     0      0       0       8         0       0      0     1
4    adm      0    0      0     0      0       0       27448     0       0      0     1
5    uucp     0    0      0     0      0       0       2000      0       0      0     1
6    invscout 0    0      0     0      0       0       8         0       0      0     1
7    nuucp    0    0      0     0      0       0       8         0       0      0     1
200  snapp    0    0      0     0      0       0       8         0       0      0     1
201  ipsec    0    0      0     0      0       0       25256     0       0      0     1
202  tester1  0    0      5     0      147     660     48        0       4260   1     1
203  tester2  0    0      0     0      18      660     32        0       129176 1     1
204  tester3  0    0      0     0      0       0       8         0       0      0     1
2000 freeware 0   0      0     0      0       0       483720    0       0      0     1

Thu Oct  7 04:00:03 CDT 2004 DAILY COMMAND SUMMARY Page 1
```

Fourth part: the command summary

```
                                    TOTAL COMMAND SUMMARY
COMMAND      NUMBER TOTAL    TOTAL   TOTAL    MEAN     MEAN    HOG    CHARS     BLOCKS
NAME         CMDS   KCOREMIN CPU-MIN REAL-MIN SIZE-K   CPU-MIN FACTOR TRNSFD    READ

TOTALS       142195 9874.08  1.62    5157.84  6082.21  0.00    0.03   5.576e+09 165741.00

gcc-3.3.     2      7615.77  0.16    0.23     46941.48 0.08    69.69  5.685e+08 0.00
smitty       53     809.41   0.26    123.11   3155.47  0.00    0.21   2.715e+07 0.00
vi           30     802.88   0.18    49.65    4461.72  0.01    0.36   3.466e+07 0.00
java         1      259.00   0.02    1.09     12432.00 0.02    1.91   5.296e+06 0.00
pg           38     213.91   0.19    33.81    1120.61  0.01    0.56   4.307e+07 0.00
dfpp         4      96.07    0.18    0.18     525.50   0.05    99.57  34016.00  0.00
sysck        7      26.98    0.04    0.04     664.04   0.01    93.41  2.129e+06 0.00
ksh          650    18.34    0.07    1429.73  269.82   0.00    0.00   1.155e+07 0.00
```

```
restbyna     9      4.49     0.12     0.59     37.22    0.01     20.61    1.99e+09  0.00
readaacc     37     4.31     0.20     14.24    21.52    0.01     1.41     6.495e+07 0.00
diff         2      3.52     0.02     0.05     170.94   0.01     44.13    4.437e+07 0.00
sendmail     87     3.09     0.00     13.03    912.31   0.00     0.03     5.961e+06 0.00
diskusg      7      2.65     0.01     0.11     307.88   0.00     7.57     2.202e+08 4106.00
v3fshelp     1      2.08     0.01     1.11     148.00   0.01     1.27     3.371e+08 82304.00
lscfg        2      1.96     0.01     0.01     269.00   0.00     93.33    3.122e+06 0.00
installp     20     1.64     0.00     0.94     701.56   0.00     0.25     664076.00 0.00
projctl      58     1.40     0.00     0.02     299.00   0.00     22.78    692868.00 0.00
...
```

Fifth part: the lastlogin file

```
0000-00-00  adm
0000-00-00  bin
0000-00-00  daemon
0000-00-00  freeware
0000-00-00  guest
0000-00-00  hennie
0000-00-00  invscout
0000-00-00  ipsec
0000-00-00  lp
0000-00-00  lpd
0000-00-00  nobody
0000-00-00  nuucp
0000-00-00  root
0000-00-00  snapp
0000-00-00  sorintodorescu
0000-00-00  sshd
0000-00-00  sys
0000-00-00  tester1
0000-00-00  tester2
0000-00-00  tester3
0000-00-00  uucp
```

The diagram in Figure 3-5 on page 71 shows how the daily reports are generated by the accounting subsystem.

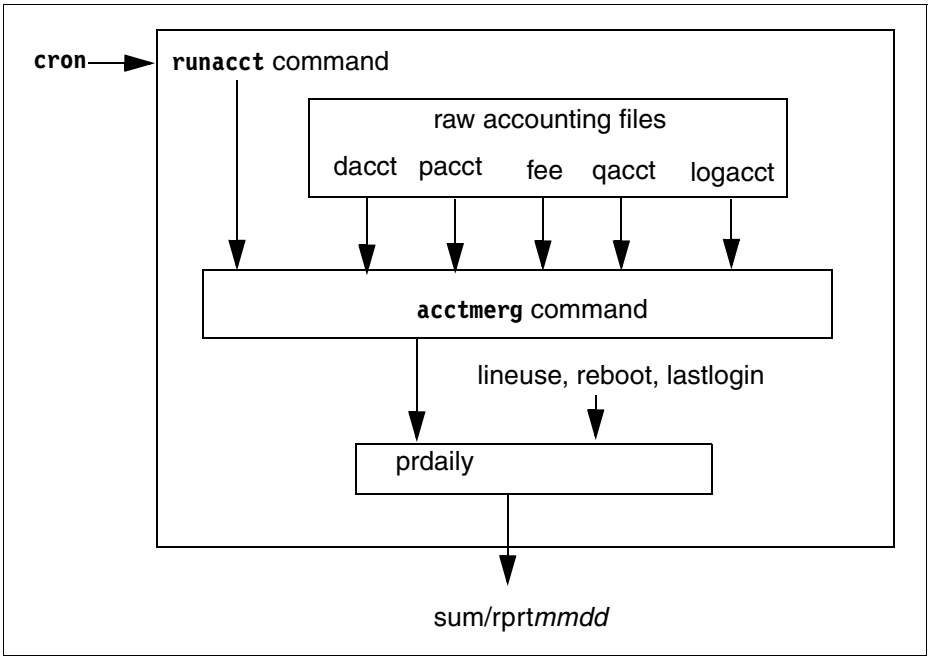

Figure 3-5 Generating daily accounting reports

The daily procedure, **runacct**, is initiated by the **cron** daemon and produces a total binary daily report file, /var/adm/acct/nite/dayacct. The same **runacct** procedure prepares summary files to produce /var/adm/acct/sum/rprt*mmdd* daily reports.

The **runacct** command creates two lock files: lock and lock1 in the /var/adm/acct/nite directory, preventing two runacct commands from running simultaneously. The lastdate file prevents more than one invocation per day.

The running states of the runacct command

The **runacct** command has several stages. When a stage is complete, the name of the next stage is written in the /var/adm/acct/nite/statefile. Because this procedure is crucial to the accounting process, it is worth explaining in detail. Table 3-1 shows the possible states of the **runacct** command.

Table 3-1 The runacct command states

State	Description
SETUP	Moves the active accounting files to working files and restarts the active files.

WTMPFIX	Verifies the integrity of the wtmp file, correcting date changes if necessary.
CONNECT1	Calls the **acctcon1** command to produce connect session records.
CONNECT2	Converts connect session records into connect total accounting records (ctacct) using **acctcon2** command.
PROCESS	Converts process accounting records into tacct format using **acctprc1** and **acctprc2** commands.
MERGE	Merges the total connect time data with total process data into a single file.
FEES	Converts the output of the chargefee command into tacct format and merges it with the connect and process total accounting records.
DISK	Merges disk accounting records with connect, process, and fee tacct records
QUEUEACCT	Sorts the queue (printer) accounting records, converts them into tacct format, and merges them with other tacct records.
MERGETACCT	Merges the daily total accounting records in the daytacct report file with the summary total accounting records in the /var/adm/acct/sum/tacct report file.
CMS	Produces command summaries in the file /var/adm/acct/sum/cms and last login information.
USEREXIT	If the /var/adm/siteacct shell file exists, calls it at this point to perform site-dependent processing.
CLEANUP	Deletes temporary files and exits.

1. The WTMPFIX state

   The **wtmpfix** command is called to verify and fix possible incoherences in the file wtmp that could cause **acctcon1** and **acctcon2** commands (which run in the CONNECT1 and CONNECT2 steps) to fail. The program checks the login names to ensure that it consists only of alphanumeric characters, a $ (dollar sign), or spaces. If the login name is invalid, the **wtmpfix** command changes the login name to INVALID and writes a diagnostic message to standard error. Another check is done to correct date and time stamp inconsistencies and writes the corrected records to standard output. If the date and time stamps are not consistent, then the **acctcon1** and **acctcon2** commands will fail. When the system date is changed using the **date** command, two change date

records are written to the /var/adm/wtmp file. The first record is the old date, denoted by the old time string. The old time string is placed in the line field and the OLD_TIME flag is placed in the type field. The second record is the new date, denoted by the new time string. The new time string is placed in the line field and the NEW_TIME flag is placed in the type field. See Example 3-10 on page 59 and Example 3-11 on page 59 for reboot date change entries in the wtmp file.

Errors that are encountered are written to the wtmperror file. If the **wtmpfix** command cannot treat some errors, the **runacct** program exits.

2. The CONNECT1 state

   In this state, **acctcon1** is called to produce the connect records file (in ctmp format, described in "The ctmp file format" on page 223), lineuse file, and reboots files.

3. The CONNECT2 state

   The **acctcon2** script is called to transform the nite/ctmp file into a binary ctacct file.

4. The PROCESS state

   The process accounting files are transformed into tacct format files. For each process accounting file (/var/adm/Spacct*i.ddmm*) the script calls the **acctprc1** command. This program reads the process accounting files and generates a text output file. The **acctprc1** command gets the user's login name from the nite/ctmp file. The output of the program is piped to the **acctprc2** command, which transforms that data into tacct format, generating the file nite/ptacct*i.ddmm*.

5. The MERGE state

   In this state, the ctacct (total connection accounting file) and ptacct (total process accounting file) files are merged into a single total accounting file: nite/daytacct.

6. The FEES state

   The extra services file, fee, is merged into the total accounting file, nite/daytacct.

7. The DISK state

   The disk accounting file, nite/dacct file, generated by the **dodisk** command, is merged into the day's total accounting file nite/daytacct.

8. The QUEUEACCT state

   The queue accounting file is merged into the nite/daytacct file, then emptied.

9. The MERGEACCT state

   The script merges each day's tacct file into the sum/tacct.*ddmm* file. If the sum/tacct file gets corrupted or lost, it can be recreated easily.

10. The CMS state

    In this stage, **runacct** does command summary reports using the **acctcms** command, and generates the last login information by using the **lastlogin** command.

11. The USEREXIT state

    This step enables you to use your own accounting procedure stored in /var/adm/siteacct. If this file exists, it is run. No additional exit test is done.

12. The CLEANUP state

    The script erases or empties the files used to prepare the daily report, then writes the *complete* state to the statefile file.

    The following files are erased:

    – Spacct*.*mmdd*
    – nite/lock*
    – nite/ptacct*.*mmdd*
    – nite/wtmp.*ddmm*
    – nite/wtmperror
    – nite/active*mmdd*

    The file fee is emptied using the **nulladm** command and the nite/tmpwtmp file is moved to nite/owtmp.

Figure 3-6 on page 75 shows the **runacct** command information flow.

runacct

```
  Init some vars.
  Test for -X flag. If used, check for the sumx/nitex directories. Continue
  if exist, exit if not.
  Erase -X flag from the positional parameters ($@) and call dowork
  procedure.


  dowork ()
     Check for other runacct running. If yes, exit, if not, continue.

     Check for enough space in /var/adm. If more than 500 blocks, continue,
     if not, exit.

     Check for the number of positional parameters:

     If 0 - the usual way of calling runacct by cron each day, without
     parameters. Check if not already run for the current date.Update
     lastdate, statefile and active files and continue.

     If 1 parameter - the date - restarts runacct for date at current state.
     Update activefile and continue.

     If 2 parameters - the date and state - restart runacct for date and
     specified state.Update active and state files and continue.

     If more than 2 parameters - exit.

     Enter the while -test case- loop. We treat the states here:

               SETUP             DISK
               WTMPFIX           QUEUEACCT
               CONNECT1          MERGEACCT
               CONNECT2          CMS
               PROCESS           USEREXIT
               MERGE             CLEANUP
               FEES              COMPLETE

     Exit if loop is the name of the state completed is COMPLETE. Before any
     exit, remove the lock files.
```

Figure 3-6 The runacct flow

Files generated in the nite directory

active This file logs the runacct progress execution, warnings, and errors encountered.

accterr Log file for runacct error and warning messages.

cms Total daily command summary file. Created by the **acctcms** command, this file is the ASCII version of the sum/cms tacct binary file. This file is used and initialized by the **monacct** command.

ctmp ASCII temporary connect time record file, generated by the **acctcon1** command.

dacct Disk usage accounting records, in binary tacct format, generated by the **dodisk** command.

daytacct Binary tacct file, with total accounting records for the previous day.

daycms ASCII daily command summary file version of the sum/daycms binary file. The **runacct** command invokes **prdaily**, which invokes **acctcms** to create this file.

lastdate Stores in *mmdd* format the last date the **runacct** was executed.

log Diagnostic output produced by **acctcon1** when called inside **runacct**.

reboots Lists the system reboots, date changes, and multiuser init-level changes since the last run of the **runacct** command.

wtmperror Error messages produced by the **wtmpfix** command during **runacct** execution, by **acctcon1**.

lineuse Connect-time line use statistics, produced by **acctcon1**.

owtmp The previous day's wtmp file, checked by **wtmpfix**.

statefile The runacct execution state file.

The files generated in the sum directory

cms Binary tacct active command summary file.

cmsprev The previous day's sum/cms file.

daycms The previous day's command summary file.

rpr*mmdd* Daily reports for the months *mm* and day *dd*.

tacct The cumulated tacct binary file. Updated daily by **runacct** and recreated by **monacct**.

tacct*mmdd* The tacct binary file for the month *mm* and day *dd*.

tacctprev Previous day's tacct file.

Generating monthly reports using the monacct command

The **monacct** command, started by the **cron** daemon, prints the summary account per month or fiscal period. The only parameter accepted is a number: the month or fiscal period for which the accounting is processed. If no parameter is given, it defaults to the current month. If used this way, it should be launched by **cron** on the first day of each month.

Example of activating **monacct** on the first of each month at 5:15:

```
15 5 1 * * /usr/sbin/acct/monacct
```

It produces the following files in the /var/adm/acct /fiscal directory:

cms*n*	The total command summary file in tacct binary format
fiscrpt*n*	The total monthly report, in the same ASCII format as the daily report
tacct*n*	Total accounting file for the fiscal period *n* in tacct binary format

The **monacct** command restarts the sum files in /var/adm/acct/sum directory.

Example 3-14 shows a monthly report, which has the same sections as the daily report, with data for the current month.

Example 3-14 Example of monthly (fiscal) report generated by monacct command

```
Fri Oct  8 17:54:12 EDT 2004  Page 1

      LOGIN   CPU    CPU  KCORE  KCORE CONNECT CONNECT DISK      FEES # OF     # OF   # DISK
UID   NAME    PRIME  NPRIME PRIME NPRIME PRIME NPRIME  BLOCKS          PROCS    SESS   SAMPLES

0     TOTAL   1      4     324   1433   28608  78754   4.810e+07 0    92115     82     30
0     root    1      4     324   1431   28608  78754   4.771e+07 0    92108     82      6
2     bin     0      0     0     0      0      0       391152    0    0         0      6
3     sys     0      0     0     0      0      0       144       0    0         0      6
4     adm     0      0     0     0      0      0       48        0    0         0      6
5     uucp    0      0     0     0      0      0       96        0    0         0      6
205   sshd    0      0     0     2      0      0       0         0    7         0      0


Fri Oct  8 17:54:12 EDT 2004 TOTAL COMMAND SUMMARY FOR FISCAL 10 Page 1

                                    TOTAL COMMAND SUMMARY
COMMAND    NUMBER TOTAL    TOTAL   TOTAL     MEAN    MEAN    HOG    CHARS      BLOCKS
NAME       CMDS   KCOREMIN CPU-MIN REAL-MIN  SIZE-K  CPU-MIN FACTOR TRNSFD     READ

TOTALS     92115  1756.68  4.71    116539.94 372.85  0.00    0.00   6.903e+09  47989.00
```

```
dfpp       32    651.25      1.11  1.11      587.87  0.03    99.37   524632.00  0.00
X          9263  348.77      0.49  5.05      716.96  0.00    9.63    1.066e+08  0.00
backbyna   4     258.92      1.00  1.92      260.00  0.25    51.91   1.363e+09  0.00
diskusg    28    116.24      0.05  0.31      2439.19 0.00    15.26   7.805e+08  47915.00
nimesis    4     80.71       0.33  191.84    242.14  0.08    0.17    1535.00    0.00
smitty     3     44.68       0.01  9.10      3177.59 0.00    0.15    3.897e+06  0.00
compress   6413  33.14       0.14  1.81      240.58  0.00    7.60    2.06e+08   0.00
egrep      554   29.13       0.11  0.13      254.19  0.00    88.18   5.521e+07  0.00
ksh        2876  28.97       0.13  41092.50  223.84  0.00    0.00    3.667e+07  0.00
ftpd       5     19.32       0.21  22.63     92.50   0.04    0.92    2.688e+09  0.00
auditcat   6408  17.37       0.27  5.86      63.53   0.00    4.67    3.46e+08   0.00


Fri Oct  8 04:00:02 EDT 2004 LAST LOGIN Page 1

00-00-00  adm           00-00-00  ipsec         00-00-00  snapp
00-00-00  bin           00-00-00  lp            00-00-00  sshd
00-00-00  bubu1         00-00-00  lpd           00-00-00  sys
00-00-00  daemon        00-00-00  nobody        00-00-00  testuser
00-00-00  guest         00-00-00  nuucp         00-00-00  uucp
00-00-00  imnadm        00-00-00  p630n01       04-10-08  root
```

Figure 3-7 and Figure 3-8 on page 79 show the **monacct** command information flow.

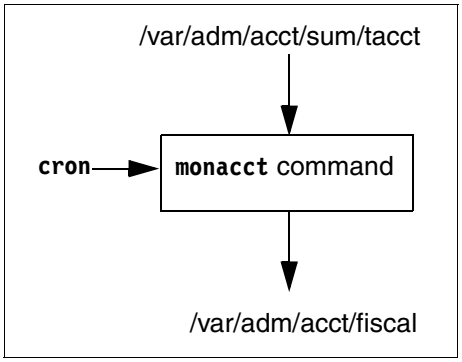

Figure 3-7 Generate the monthly (fiscal) reports

The **monacct** command erases the daily tacct files /sum/tacct*mmdd* and sum/rprt_*mmdd* files, and restarts the sum/tacct and sum/cms files.

```
monacct

    init variables
    moves sum/tacct file to fiscal/tacct_mm file
    deletes the old sum/tacct files
    recreates the sum/tacct file with nulladm command
    moves the command summary file sum/cms to fiscal/cms_mm
    recreates sum/cms with nulladm command
    produce the monthly report with prtacct in fiscal/fiscrpt_mm
    add to fiscal/fiscrpt_mm the command summary
    add to fiscal/fiscrpt_mm the sum/loginlog file
    let place to to do any charging fee
```

Figure 3-8 The monacct command flow

Connect-time report

The connect-time records are stored in /var/adm/wtmp file. When launched by the **cron** daemon, the **runacct** command uses two other commands to process the connect-usage records, such as login, logout, and system-shutdown records (see Figure 3-9 on page 80):

- The **acctcon1** command reads data from the standard input, where wtmp file content is redirected. It converts the series of logins and logouts from the wtmp file into a sequence of login sessions to the standard output.
- The **acctcon2** command converts the sequence of sessions that it reads from the standard input, from the **acctcon1** command into connect-time total accounting records directed to standard output.
- The output of the **acctcon2** command is merged with other total accounting records by the **acctmerg** command.

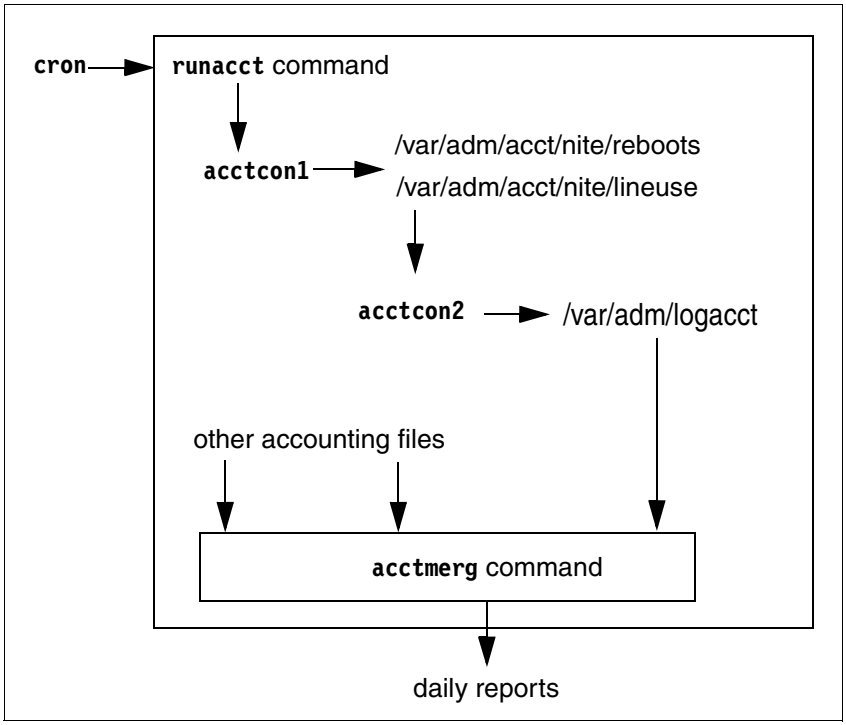

Figure 3-9 Generation and merging of connect-time report

The following list contains a summary of the commands that are used to process the information about connect-time sessions:

Data files used:

/etc/utmp The active connect-session database in utmp binary format. Voided by the reboot of the system.

/var/adm/wtmp The cumulative connect-session database in utmp binary format. Stores the connect records across reboots of the system.

/etc/security/failedlogin
The failed login sessions in utmp binary format.

/**etc/security/lastlog** Last login database in ASCII format.

Commands used:

ac Displays total connect session records.

acctcon1 Summarizes connect sessions from wtmp file.

acctcon2	Produces total accounting records using input from the **acctcon1** command.
acctwtmp	Writes records to the wtmp file.
fwtmp	Displays records from wtmp, utmp, or failedlogin files (binary files in utmp format).
last	Displays login information.
lastlogin	Displays last date a user logged in to a system.
prctmp	Displays binary files in ctmp file format.
wtmpfix	Fixes errors in the wtmp file, such as errors provoked by date changes or invalid user names.
who	Displays who is logged in on a system.

Process report

Process accounting records are stored in /var/adm/pacct*i* files. The **runacct** command calls two other commands to process the records (Figure 3-10 on page 82):

- The **acctprc1** command translates the user ID into a login user name and writes ASCII records to the standard output containing chargeable items, such as CPU time, mean memory size, and I/O data for each process. If no other file is specified, the **acctprc1** command uses the numeric user IDs for translating the user names (as they appear in /etc/passwd).
- The **acctprc2** command transforms the records it gets from **acctprc1** from standard input into total accounting records, summarizes by user name, and sorts these records. The total accounting records generated at the standard output are added to the daily report by the **acctmerg** command.

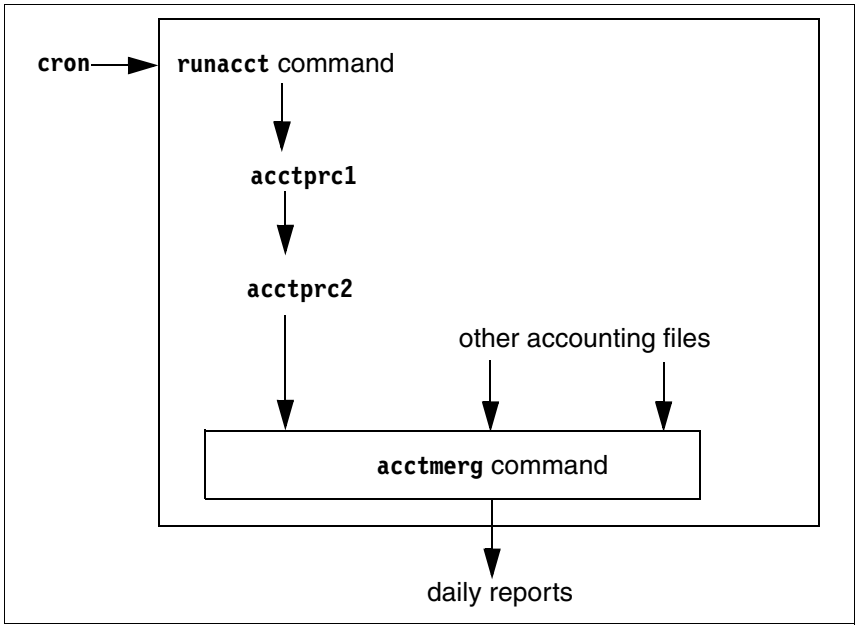

Figure 3-10 Generation and merging of process report

Process accounting data also provides information that you can use to monitor system resource usage. The **acctcms** command summarizes resource use by command name. This provides information about how many times each command was run, how much processor time and memory was used, and how intensely the resources were used.

This parameter is known as the “hog factor” and is the ratio between the CPU time of the process and the total time of the process. The **acctcms** command produces long-term statistics on system utilization, providing information about total system usage and the frequency with which commands are used.

Data files used:

pacct	The active process data file
pacct*i*	The rotated pacct files.
Spacct*immdd*	The pacct*i* files produced by runacct for the specified month *mm* and day *dd.*

Commands used with pacct file:

acctcom	Displays process accounting records from pacct file.
acctcms	Displays command-usage records from accounting records.

lastcomm	Displays information about the last command executed.
acctprc1	Reads acct format records, adds login names to the corresponding user ID, and writes the ASCII record to the standard output.
acctprc2	Summarizes and converts the acctprc1 output to tacct format.
prtacct	Displays binary tacct file records.

Using the acctcom command

The **acctcom** command handles the same data as the **acctcms** command, but provides detailed information about each *terminated* process. Information about *active* processes can be examined using the **ps** command.

You can display all process accounting records or select records of particular interest. Selection criteria include the load imposed by the process, the time period when the process ended, the name of the command, the user or group that invoked the process, the name of the WLM class the process belonged to, and the port at which the process ran.

Unlike other accounting commands, **acctcom** can be run by all users. If no input file is provided (pacct*i* file or standard input), the command reads the **pacct** file. The input file is in acct file format, or tacct file format if the -t flag is used. The output can be ASCII (-a flag) or binary (Example 3-15).

Other flags are available for sorting the output, and displaying the summary of prime or non-prime time commands. For a detailed description of the **acctcom** command, refer to *AIX 5L Version 5.3, Commands Reference, Volume 1, a - c,* SC23-4888.

Example 3-15 Example of acctcom file output

```
COMMAND                      START    END          REAL      CPU        MEAN
NAME       USER     TTYNAME  TIME     TIME       (SECS)    (SECS)     SIZE(K)
#accton    root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#bsh       root     ?        04:00:00 04:00:00     0.03      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#mv        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#cp        root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#acctwtmp  root     ?        04:00:00 04:00:00     0.00      0.00        0.00
#fwtmp     root     ?        04:00:00 04:00:00     0.00      0.00        0.00
```

```
#awk       root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#sed       root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#fwtmp     root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#cp        root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#chmod     root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#chown     root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#bsh       root     ?        04:00:00 04:00:00     0.02      0.00         0.00
#acctwtmp  root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#fwtmp     root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#awk       root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#fwtmp     root     ?        04:00:00 04:00:00     0.00      0.00         0.00
#dspmsg    root     ?        04:00:00 04:00:00     0.00      0.00         0.00
```

Description of the fields displayed by the **acctcom** command:

COMMAND NAME The command name, preceded by a # if running with superuser rights

USER The login name of the user who launched the command

TTYNAME The controlling terminal, if any

START TIME The start time of the command

END TIME The end time of the command

REAL (SECS) The number of seconds the command was active

CPU (SECS) The number of CPU seconds used by the command

MEAN SIZE(K) The average memory size of the command in KB

Using the acctcms command

The **acctcms** command reads each file given as parameter (usually in tacct file format), sorts and adds all records for identically named processes, and writes the records to standard output. By default, the output file is in binary format.

In Example 3-16, the **acctcms** command was used to summarize the command record entries from pacct1, pacct2, and pacct3; sort the entries after the total CPU time using the -c flag; and print the output in ASCII format using the -a flag.

Example 3-16 Example of acctcms command

```
[#][/var/adm]> acctcms -a -c pacct1  pacct2  pacct3 |head -20
                                  TOTAL COMMAND SUMMARY
COMMAND    NUMBER TOTAL     TOTAL   TOTAL    MEAN      MEAN    HOG    CHARS     BLOCKS
NAME       CMDS   KCOREMIN  CPU-MIN REAL-MIN SIZE-K    CPU-MIN FACTOR TRNSFD    READ

TOTALS     12674  5.114e+08 2574.24 8222.95  198673.69 0.20    31.31  8.072e+11 7.00

prog1      2      3.081e+07 1809.10 1000.32  17032.14  904.55  180.85 8.297e+08 0.00
```

```
prog2       3      2.446e+08 301.04  942.43   812409.19 100.35  31.94   1.884e+06 0.00
date        3867   54.31     48.49   104.44   1.12      0.01    46.43   3.991e+07 0.00
compress    179    30079.55  39.41   101.91   763.17    0.22    38.68   1.161e+11 0.00
uncompress  177    9900.82   37.84   95.92    261.67    0.21    39.45   1.163e+11 0.00
tar         812    304.00    33.88   102.46   8.97      0.04    33.06   5.703e+11 0.00
yes         5      364.77    2.88    56.87    126.77    0.58    5.06    3.04e+09  0.00
dfpp        12     285.42    0.55    0.83     519.43    0.05    65.88   289136.00 0.00
find        7      5.01      0.11    0.44     43.83     0.02    26.21   6.472e+06 0.00
java        3      975.56    0.08    95.99    12045.45  0.03    0.08    2.134e+07 0.00
ksh         1334   20.41     0.06    178.50   323.85    0.00    0.04    6.49e+06  0.00
sshd        16     37.73     0.06    1288.81  616.54    0.00    0.00    2.919e+06 0.00
```

Using the lastcomm command

Use the **lastcomm** command to display information, in reverse chronological order, about all previously executed commands that are recorded in the /var/adm/pacct file. For a detailed description of the **lastcomm** command, refer to *AIX 5L Version 5.3, Commands Reference, Volume 3, i - m,* SC23-4890.

Example 3-17 Example of lastcomm command output

```
<adm>[/var/adm/acct/sum]> lastcomm |head -10
ping               tester1  pts/14     0.01 secs Thu Oct 14 17:11
lastcomm         X adm      pts/0      0.01 secs Thu Oct 14 17:11
head               adm      pts/0      0.01 secs Thu Oct 14 17:11
df                 tester1  pts/14     0.01 secs Thu Oct 14 17:11
w                  tester1  pts/14     0.01 secs Thu Oct 14 17:11
ls                 tester1  pts/14     0.01 secs Thu Oct 14 17:11
sleep              tester1  pts/14     0.01 secs Thu Oct 14 17:10
ping               tester1  pts/14     0.01 secs Thu Oct 14 17:10
df                 tester1  pts/14     0.01 secs Thu Oct 14 17:10
w                  tester1  pts/14     0.01 secs Thu Oct 14 17:10
```

Description of fields displayed by the **lastcomm** command in Example 3-17:

- The name of the command
- The flag showing additional information about the command:
  - S if executed in superuser mode
  - F ran after a fork, but without an ensuing exec
  - C if ran in PDP-11 compatibility mode
  - D if terminated with a generation of a signal
  - X if terminated with a signal (as shown in Example 3-17)
- The login name of the user executing the command
- The controlling terminal, if any
- The number of CPU seconds the command has used
- The start time of the command

Daily command summary

The daily command summary report is generated by the **runacct** command. The ASCII version of this file is /var/adm/acct/nite/daycms, and the cumulative version, in binary tacct format, for the current month or fiscal period is in /var/adm/acct/sum/daycms (Example 3-18).

This report shows how the system resources are used by a command. Use it to obtain the most used commands on the system and how they charge your system.

Example 3-18 Example of daily command report file: nite/daycms

COMMAND NAME	NUMBER CMDS	TOTAL KCOREMIN	TOTAL CPU-MIN	TOTAL REAL-MIN	MEAN SIZE-K	MEAN CPU-MIN	HOG FACTOR	CHARS TRNSFD	BLOCKS READ
TOTALS	62292	1.401e+07	396.28	24290.48	35348.51	0.01	1.63	9.437e+09	21.00
java	2	8.988e+06	49.12	199.49	182966.48	24.56	24.62	4.84e+08	0.00
ctest	6	5.017e+06	337.49	192.29	14864.80	56.25	175.51	6.046e+09	0.00
sshd	39	2435.60	6.92	9943.77	352.06	0.18	0.07	1.389e+09	0.00
x	28	673.10	1.61	24.15	418.17	0.06	6.66	2.401e+08	0.00
smitty	27	237.34	0.08	68.45	3131.95	0.00	0.11	1.24e+07	0.00
dfpp	4	96.16	0.18	0.18	524.50	0.05	99.44	139968.00	0.00
ksh	319	89.17	0.53	9406.75	166.78	0.00	0.01	1.515e+07	0.00
topas	9	86.82	0.26	135.87	340.17	0.03	0.19	4.05e+06	0.00
xlcentry	4	13.79	0.00	0.01	5295.10	0.00	23.81	2.819e+06	0.00
vi	47	4.93	0.01	73.98	591.06	0.00	0.01	2.731e+06	0.00
ssh	1	4.14	0.01	61.38	589.00	0.01	0.01	2.102e+06	0.00
sendmail	79	3.05	0.00	13.04	899.69	0.00	0.03	5.972e+06	0.00
projctl	54	2.18	0.00	0.01	1193.29	0.00	13.21	522841.00	0.00
acctprc2	11	1.86	0.01	0.01	339.24	0.00	87.50	2.639e+06	0.00

Note: The data is sorted by the `TOTAL KCOREMIN` column.

Description of each column of the daily command summary:

COMMAND NAME The command name.

NUMBER CMDS The number of times the command was run.

TOTAL KCOREMIN Memory measurement in kilobyte segments.

TOTAL CPU-MIN Total CPU time the program accumulates, in minutes.

TOTAL REAL-MIN Total real time that the program accumulates, in minutes.

MEAN SIZE-K Mean memory size, in kilobytes.

MEAN CPU-MIN Mean CPU time per invocation of the command, in minutes (`TOTAL CPU-MIN` / `NUMBER CMDS`).

HOG FACTOR	CPU usage factor: (`TOTAL CPU-MIN / TOTAL REAL-MIN`).
CHARS TRNSFD	Total number of characters read and written by the process.
BLOCKS READ	Total number of physical blocks read and written by the process.

Last login report

Using this report you may obtain when a login account was last used. This enables you to find unused accounts and delete them, saving system resources.

Disk-usage report

Disk-usage records stored in the /var/adm/acct/nite/dacct file and merged by the **`acctmerg`** command into the daily accounting reports (Figure 3-11).

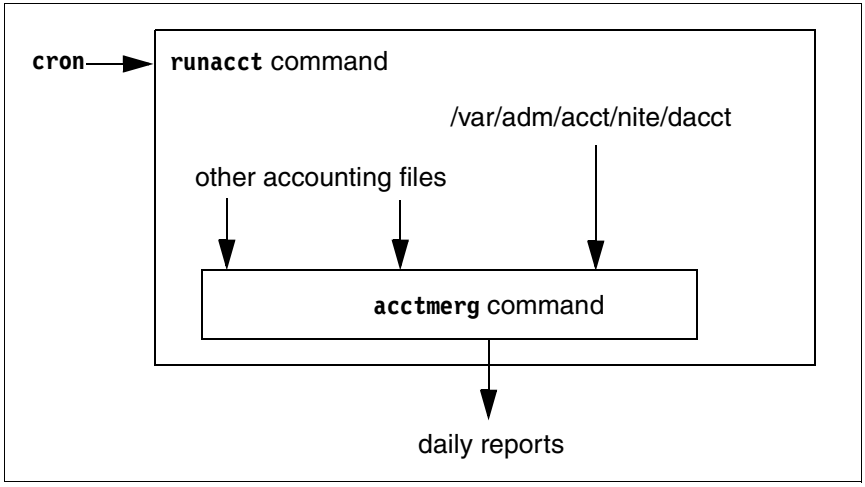

Figure 3-11 Generation and merging of disk-usage report

Data files used:

dtmp	Temporary ASCII file produced by the **`dodisk`** command; contains ASCII disk usage records
nite/dacct	Binary tacct file produced by **`dodisk`**

Commands used:

dodisk	Daily procedure that generates the disk usage file dacct in tacct binary file format.
diskusg	Procedure in **`dodisk`** that generates dtmp temporary file.

acctdusg Procedure in **dodisk** that generates dtmp temporary file. Called if -o flag is used. Enables accounting at directory level.

acctdisk Procedure in **dodisk** that converts dtmp ASCII file to tacct binary file.

Printer-usage report

Printer-usage records are stored in the /var/adm/qacct file and added to the daily report with the **acctmerg** command (Figure 3-12).

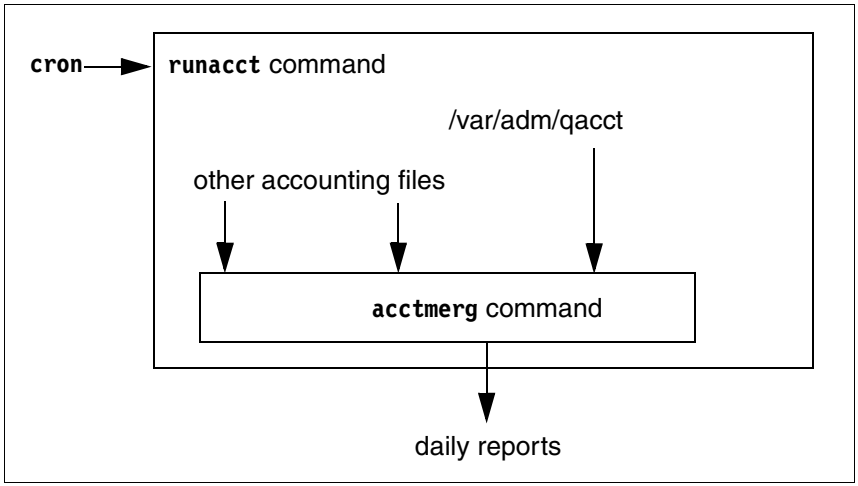

Figure 3-12 Generation and merging of printer-usage report

Data file used:

qacct Data file containing printer accounting records

Command used:

pac Command that displays the printer accounting records from the qacct file

Fee report

The fee report is produced by the **chargefee** command. If you want to bill a user for a service provided, use the **chargefee** command:

```
chargefee user units
```

This charges the user with a number of units for the extra service provided. The command generates an ASCII total accounting record stored in /var/adm/fee file.

This file is added to the daily reports with the **acctmerg** command (Figure 3-13 on page 89).

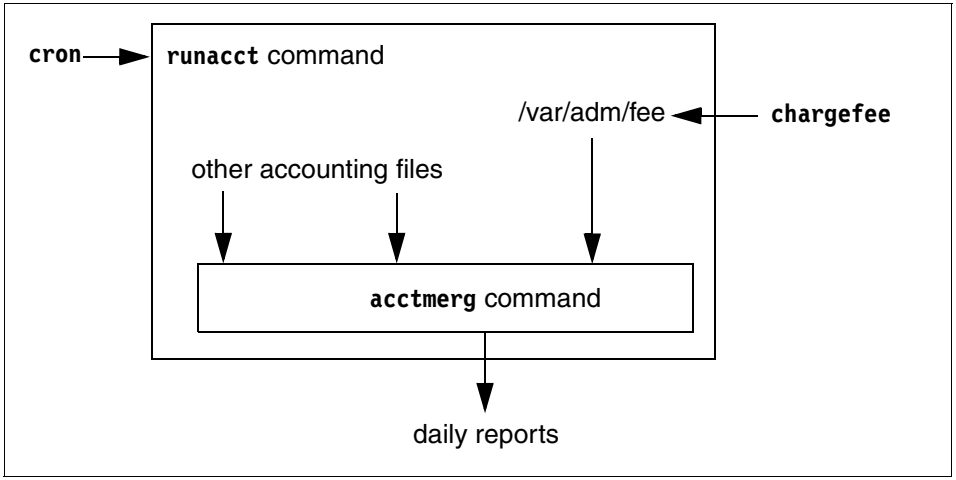

Figure 3-13 Generation and merging fee report

Data file used:

fee Total accounting records (in ASCII) generated by the **chargefee** command

Command used:

chargefee Command charging a user with a number of units. Writes an ASCII tacct in the file /var/adm/fee.

Example 3-19 shows charging the user tester2 (with 20 units billed) and the total accounting information in ASCII format (the /var/adm/fee file).

Example 3-19 The /var/adm/fee file (charging user tester2)

```
[#][/]> chargefee tester2 20
[#][/]> cat /var/adm/fee
203 tester2 0 0 0 0 0 0 0 0 0 0 0 0 20 0 0 0
[#][/]>
```

Generating reports about system activity

Daily reports and monthly reports are generated by commands launched by the **cron** daemon. You can generate your own reports at any time using the **prtacct** command. The command can read any files in tacct format, select the headers to print (-f flag), or produce verbose output, printing floating-point numbers in higher precision notation in order to produce formatted output.

Such total accounting files (tacct) are daily reports about connect time, process time, disk usage, and printer usage (Example 3-20). For input files, you may use the one of the following files in the sum directory:

- tacct, the total accounting file for the current month or fiscal period
- tacctprev, the total accounting file for the current month or fiscal period as it was a day before
- tacct*mmdd,* the total accounting file for the month *mm* and day *dd*

Example 3-20 Example of a report generated using prtacct command

```
[#][/var/adm/acct/sum]> prtacct -f 1,2,3,5,7,9  tacct1006

Fri Oct  8 16:33:11 CDT 2004  Page 1

             LOGIN    CPU       KCORE     BLKIO     RD/WR
UID          NAME     PRIME     PRIME     PRIME     PRIME

0            TOTAL    9         139       8.588e+09 1.828e+06
0            root     9         139       8.588e+09 1.828e+06
1            daemon   0         0         0         0
2            bin      0         0         0         0
3            sys      0         0         0         0
4            adm      0         0         134       0
5            uucp     0         0         0         0
6            invscout 0         0         0         0
7            nuucp    0         0         0         0
200          snapp    0         0         0         0
201          ipsec    0         0         0         0
```

Our example shows only the fields 1,2,3,5,7,and 9 of the records from the total accounting file for October 8. The tacct record is described in “The tacct file format” on page 219.

Maintaining process data files

The size of the process accounting file (pacct) may grow rapidly, and processing a big pacct file is slow. Another issue may arise when there is no space left in the /var filesystem. Activate the **ckpacct** command via **crontab** to prevent such situations:

```
5 * * * * /usr/sbin/acct/ckpacct
```

In this example, the **cron** daemon activates the **ckpacct** command at five minutes past every hour, every day. The **ckpacct** command verifies the size of the pacct file. If the size of the active data exceeds 1000 blocks, **ckpacct** invokes the **turnacct switch** command, which moves the pacct active file into the pacct*i* file, stops the accounting system (using **accton** with no parameter) and restarts the

accounting system using **accton pacct**, recreating a new, empty active pacct file. No mail is send when **ckpacck** rotates files.

The **ckpacct** command takes only one optional parameter: the block size of the active file. It rotates the pacct file if the pacct file exceeds the number of blocks that are specified by the parameter, thus creating rotated pacct*i* files.

The **ckpacct** command also verifies the free blocks in the /var filesystem. If less than 500 blocks are available, it automatically turns off the accounting using the **turnacct off** command. If, at the next run it finds 500 blocks available, the accounting is reactivated using the **turnacct on** command.

Note: If the *MAILCOM* environment variable is set to mail root adm, for each event related to the size of /var filesystem, a mail is send to root and adm users, who will continue to receive warning mails (that /var/adm is still low on space) until 500 free blocks are available and accounting is turned on again.

3.5 Observing the system

Besides strict accounting procedures, some additional commands are useful for monitoring the system. These utilities are presented in this section.

3.5.1 The system activity

The utilities that are most commonly used for observing the system activity are **iostat**, **sar**, the **sadc**, **sa1**, and **sa2** data collectors, **vmstat**, **ps**, **time**, and **timex**.

The iostat command

This is another utility delivered with the bos.acct package, used for monitoring system input/output device loading. **iostat** works the same way as **vmstat**:

```
iostat flags device interval count
```

Invoked alone, **iostat** provides statistics concerning the time since the system was booted. Each subsequent report covers the time since the previous report. All statistics are reported each time **iostat** is run. The report consists of a tty and CPU header row followed by a row of tty and CPU statistics. On multiprocessor systems, CPU statistics are calculated systemwide as averages among all processors.

Example 3-21 Using the iostat command with interval and count parameters

```
[#][/var/adm/acct/sum]> iostat 2 100

System configuration: lcpu=4 drives=5
```

```
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.5        195.5               0.0      0.1       99.9       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
dac1             0.0       0.0       0.0          0         0
dac1-utm         0.0       0.0       0.0          0         0
dac0             0.0       0.0       0.0          0         0
dac0-utm         0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        531.0               0.0      0.1       99.9       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
dac1             0.0       0.0       0.0          0         0
dac1-utm         0.0       0.0       0.0          0         0
dac0             0.0       0.0       0.0          0         0
dac0-utm         0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        554.0               0.1      0.0       99.9       0.0
```

The following headers are used in the tty and CPU utilization report:

tin Total number of characters read by the system for all ttys.

tout Total number of characters written by the system to all ttys.

% user Percentage of CPU utilization that occurred while executing at the user level (application).

% sys Percentage of CPU utilization that occurred while executing at the system level (kernel).

% idle Percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.

% iowait Percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.

physc The number of physical processors consumed, displayed only if the partition is running with shared processor.

% entc Percentage of entitled capacity consumed, displayed only if the partition is running with shared processor.

The following headers are used in the disk utilization report:

% tm_act Percentage of time the physical disk was active (bandwidth utilization for the drive).

Kbps	Amount of data transferred (read or written) to the drive in KB per second.
tps	The number of transfers per second that were issued to the physical disk. A transfer is an I/O request to the physical disk. Multiple logical requests can be combined into a single I/O request to the disk. A transfer is of indeterminate size.
Kb_read	Total number of KB read.
Kb_wrtn	Total number of KB written.

Read more about this command in *AIX 5L Version 5.3, Commands Reference, Volume 3, i - m,* SC23-4890.

The sar command

Use the **sar** command (system activity reporter) to display system activity. The operating system contains several counters for CPU, buffers, disks, system calls, and other system activities.

For the command to read counters in the operating system at *n* intervals of *m* seconds:

```
sar n m
```

For the command to report all available statistics:

```
sar -A
```

Example 3-22 shows how a system passes from an idle state when a compilation is launched. We use **sar 2 100** to display global processor statistics.

Example 3-22 Global processor statistics (using sar)

```
[#][/]> sar 2 100

AIX p630n02 3 5 000685BF4C00    10/08/04

System configuration: lcpu=4

15:22:54    %usr    %sys    %wio   %idle
15:22:56       0       0       0     100
15:22:58       0       0       0     100
15:23:00       0       0       0     100
15:23:02       0       1       3      96
15:23:04       0       0       1      99
15:23:06       0       0       0     100
15:23:08       0       0       0     100
15:23:10       0       0       0     100
15:23:12       1       1       0      97
```

```
15:23:14      18       4       0      77
15:23:16      20       3       0      76
15:23:18      18       4       0      77
15:23:20      21       2       0      76
15:23:22      17       5       0      78
15:23:24      16       5       1      79
15:23:26      18       4       0      78
15:23:28      16       5       1      78
15:23:30      19       4       0      77
15:23:32      18       4       1      78
15:23:34      18       3       0      78
15:23:36       4       2       1      92
15:23:38       0       0       0     100
15:23:40       0       0       0     100
15:23:42       0       0       0     100
[#][/]>
```

After the date and time of the sample, the following values are displayed:

%idle Percentage of time the CPU was idle with no outstanding disk I/O requests

%sys Percentage of time the CPU spends executing system calls

%usr Percentage of time the CPU spends in execution at user or application level

%wio Percentage of time the CPU was idle, during which the system had outstanding disk NFS requests

Example 3-23 shows the same compilation, this time using **sar -P ALL 1 100** to show individual processor statistics.

Example 3-23 Example of sar command with statistics per processor

```
[#][/]> sar -P ALL 1 100

AIX p630n02 3 5 000685BF4C00    10/15/04

System configuration: lcpu=4

12:10:16 cpu    %usr    %sys    %wio   %idle

12:09:17  0       67      17       2      14
          1        0       0       0     100
          2        1       1       0      98
          3        0       0       0     100
          -       17       4       0      78
12:09:18  0       57      17       4      22
          1        0       0       0     100
```

```
          2        1       0       0      99
          3        0       0       0     100
          -       14       4       1      80
12:09:19  0       13      16       1      70
          1       18       6       0      76
          2        5      19       0      76
          3       45      10       0      45
          -       20      13       0      67
12:09:20  0       47      11       1      41
          1        0       0       0     100
          2        2       0       2      96
          3       23       2       0      75
          -       18       3       1      78
12:09:21  0       64      18       3      15
          1        0       0       0     100
          2        0       1       0      99
          3        0       0       0     100
          -       16       5       1      78
```

The same values are displayed, plus one additional value: the CPU field, showing the processor for which the statistics are displayed.

The sadc data collector

The system uses **sadc**, **sa1**, and **sa2** procedures to store the data coming from these counters to disk. The **sadc** command is invoked like this:

```
sadc n m outfile
```

The **sadc** samples the counter data *m* times with an interval of *n* seconds between samples, writing the result in *outfile* in binary format. If no outfile is given, the output is the standard output.

Note: The time interval between samples should be greater than five seconds; otherwise, the **sadc** itself may affect the data collected.

The shell scripts **sa1** and **sa2** are used more frequently than the **sadc** command.

The sa1 data collector

The **sa1** script collects and stores data into binary file /var/adm/sa/sa*dd* (where *dd* is the current day). It accepts the same arguments, *n* and *m*, as **sadc**.

Example 3-24 Example of crontab entries for sa1 and the generated binary files

```
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3 &
[#][/]> ls -l /var/adm/sa
total 3088
```

```
-rw-r--r--   1 root     system       140364 Oct 05 17:40 sa05
-rw-r--r--   1 root     system       467880 Oct 06 17:40 sa06
-rw-r--r--   1 root     system       467880 Oct 07 17:40 sa07
-rw-r--r--   1 root     system       356176 Oct 08 17:40 sa08
-rw-r--r--   1 root     system        97524 Oct 11 14:40 sa11
```

The sa2 data collector

The **sa2** script collects and stores data in text file /var/adm/sar*dd.*

Example 3-25 Example of crontab entries for sa2 command and text files generated

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -ubcwyaqvm &
[#][/]> ls -l /var/adm/sa/sar*
-rw-r--r--   1 root     system         1927 Oct 05 18:05 /var/adm/sa/sar05
-rw-r--r--   1 root     system         4818 Oct 06 18:05 /var/adm/sa/sar06
-rw-r--r--   1 root     system         4818 Oct 07 18:05 /var/adm/sa/sar07
-rw-r--r--   1 root     system         4405 Oct 08 18:05 /var/adm/sa/sar08

AIX p630n02 3 5 000685BF4C00    10/05/04

System configuration: lcpu=4

15:00:00    %usr    %sys    %wio   %idle
16:00:00       0       0       0     100
17:00:00       1       3       1      95

Average        1       2       0      97

System configuration: lcpu=4

15:00:00 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
16:00:00       0       0       0       0       0       0       0       0
17:00:00     508    4062      87       0       0       0       0       0

Average      254    2031      87       0       0       0       0       0

System configuration: lcpu=4

15:00:00 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
16:00:00     224       1       0    0.02    0.02    2715      32
17:00:00   24601   12187   12186    0.03    0.03 2130374 2128414

Average    12413    6095    6094    0.03    0.03 1066607 1064286

System configuration: lcpu=4

15:00:00 cswch/s
16:00:00     452
```

```
17:00:00    9581

Average     5017

System configuration: lcpu=4

15:00:00 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
16:00:00       0       0       3       0       0       0
17:00:00       0       0      12       0       0       0

Average        0       0       7       0       0       0

System configuration: lcpu=4

15:00:00  iget/s lookuppn/s dirblk/s
16:00:00       0          2        0
17:00:00       0          4        0

Average        0          3        0

System configuration: lcpu=4

15:00:00 runq-sz %runocc swpq-sz %swpocc
16:00:00     1.0       1     1.0       0
17:00:00     1.5      10     1.0       0

Average      1.5       5     1.0       0

System configuration:

15:00:00  proc-sz     inod-sz     file-sz     thrd-sz
16:00:00  76/262144   0/281       730/1023    249/524288
17:00:00  77/262144   0/281       738/1023    250/524288

System configuration: lcpu=4

15:00:00   msg/s  sema/s
16:00:00    0.01    0.00
17:00:00    0.01    0.00

Average     0.01    0.00
```

The vmstat command

The **vmstat** command also comes in the bos.acct package. The command reports statistics about kernel threads, virtual memory, disks, traps, and CPU activity. These system-wide statistics (among all processors) are calculated as averages for values expressed as percentages, and otherwise as sums. Used

with programs such as MRTG (Multi Router Traffic Grapher), it can provide visual representation of the systemwide charge of a system.

MRTG is freeware that is widely used by the sysadmin community. You can find information about MRTG at:

http://www.mrtg.org

If the **vmstat** command is invoked without flags, the report contains a summary of the virtual memory activity since system startup (Example 3-26).

Example 3-26 vmstat without flags

```
[#][/]> vmstat

System configuration: lcpu=4 mem=8192MB

kthr    memory              page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
36  7 204306 1802061   0   0   0   0    0   0  39 3835 1741  2 13 82  3
```

The command can also be invoked with parameters:

```
vmstat interval count
```

The *interval* parameter specifies the amount of time in seconds between each report. The first report contains statistics for the time since system startup. Subsequent reports contain statistics collected during the interval since the previous report. The *count* parameter can only be specified with the *interval* parameter, and cannot be zero. If the *count* parameter is specified, its value determines the number of reports generated and the number of seconds apart. If the *interval* parameter is specified without the *count* parameter, reports are generated continuously (Example 3-27).

Example 3-27 Example of vmstat command with interval and count parameters

```
[#][/var/adm/acct/sum]> vmstat 2 100

System configuration: lcpu=4 mem=8192MB

kthr    memory              page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre      re  pi  po  fr   sr  cy  in sy   cs us sy id wa
 0  0 204311 1802056   0   0   0   0    0   0   4  183  92  0  0 99  0
 0  0 204313 1802054   0   0   0   0    0   0   8  146  96  0  0 99  0
 0  0 204313 1802054   0   0   0   0    0   0   5  143  97  0  0 99  0
 0  0 204313 1802054   0   0   0   0    0   0   6  141  95  0  0 99  0
```

Description of the headers used:

kthr Kernel thread state changes per second over the sampling interval.

- **r** The number of kernel threads placed in the run queue.
- **b** The number of kernel threads placed in wait queue (awaiting resource, awaiting input/output).

memory Information about use of virtual and real memory. Virtual pages are considered active if they have been accessed. A page is 4096 bytes.

- **avm** Shows the active virtual pages.
- **fre** Shows the size of the free list.

A large portion of real memory is utilized as a cache for file system data. It is not unusual for the size of the free list to remain small.

page Information about page faults and paging activity. These are averaged over the interval and given in units per second.

- **re** The pager input/output list.
- **pi** The ages paged in from paging space.
- **po** The ages paged out to paging space.
- **fr** The ages freed (page replacement).
- **sr** The ages scanned by page-replacement algorithm.
- **cy** The clock cycles by page-replacement algorithm.

faults Trap and interrupt rate averages per second over the sampling interval.

- **in** Device interrupts.
- **sy** System calls.
- **cs** Kernel thread context switches.

cpu Breakdown of percentage usage of CPU time.

- **us** User time.
- **sy** System time.
- **id** CPU idle time.
- **wa** CPU idle time during which the system had outstanding disk/NFS I/O request(s). See detailed description above.
- **pc** The number of physical processors consumed. Displayed only if the partition is running with shared processor.

ec The percentage of entitled capacity consumed. Displayed only if the partition is running with shared processor.

Starting with AIX 5.3, the **vmstat** command reports the number of physical processors consumed (pc), and the percentage of entitlement consumed (ec), in the micro-partitioning and simultaneous multithreading environments. These metrics will only be displayed on micro-partitioning and simultaneous multithreading environments. Read more about this command in *AIX 5L Version 5.3, Commands Reference, Volume 6, v - z,* SC23-4893.

The ps command

This command belongs to the bos.rte.control package and is used to observe a process running in the memory. AIX 5.3 has two versions of this command: the AIX version and the System V version. The use of this command is complex and beyond the scope of this book. Read more about this command in *AIX 5L Version 5.3, Commands Reference, Volume 4, n - r,* SC23-4891.

The time and timex commands

The precedent commands **sar**, **vmstat**, and **ps** can be used to observe globally the systemwide resource consumption while executing a command. To observe only the resources used by a command during execution, use **time** and **timex** commands.

time

The **time** command is invoked as in Example 3-28.

Example 3-28 Example of time command

```
[#][/work/rsync-2.6.3]> time make
...compiling some stuff

real    0m23.19s
user    0m18.38s
sys     0m1.07s
```

The command runs the program given as an argument, and prints CPU time statistics about the program in the argument. The CPU statistics that are printed by the command are:

- The real time elapsed between the beginning and the end of the program
- Total CPU user time
- Total CPU system time

Note: The exit status of the **time** command is the exit status of the specified command. Otherwise, the **time** command exits with specific values. Refer to *AIX 5L Version 5.3, Commands Reference, Volume 5, s - u,* SC23-4892 for further information about the **time** command.

timex

The **timex** command performs the same way as the time command. Using flags, it prints accounting statistics about the command and all of its child processes.

When used with the -s flag, it reports total system activity during the execution of the command. The parameters are the same as those of the **sar** command (Example 3-29).

Example 3-29 Example of timex -s command

```
[#][/work/rsync-2.6.3]> timex make
... compiling some stuff
real 23.71
user 18.56
sys  1.12

AIX p630n02 3 5 000685BF4C00    10/08/04
System configuration: lcpu=4

15:36:30    %usr    %sys    %wio   %idle
15:36:54      20      25       0      55

System configuration: lcpu=4

15:36:30 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
15:36:54       0       0       0       0       0       0       0       0

System configuration: lcpu=4 mem=8192MB

15:36:30   slots cycle/s fault/s  odio/s
15:36:54  130692    0.00 11033.53   62.08

System configuration: lcpu=4

15:36:30 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
15:36:54       2       0  318110       0       0       0

System configuration: lcpu=4

15:36:30 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
15:36:54   58487     345   39826    7.70    7.83 1338507  768001
```

```
System configuration: lcpu=4

15:36:30 cswch/s
15:36:54   26635

System configuration: lcpu=4

15:36:30  iget/s lookuppn/s dirblk/s
15:36:54       2       1012        0

System configuration: lcpu=4

15:36:30 runq-sz %runocc swpq-sz %swpocc
15:36:54     1.9      92

System configuration:

15:36:30  proc-sz     inod-sz     file-sz     thrd-sz
15:36:54  101/262144  17/7489     769/1877    260/524288

System configuration: lcpu=4

15:36:30   msg/s  sema/s
15:36:54    0.00    0.00
```

When used with the -p parameter, **timex** prints process accounting records for a command and all of its children, as shown in Example 3-30.

Example 3-30 Example of timex -p command

```
[#][/work]> timex -p make
... compiling some stuff

real 23.90
user 18.56
sys  1.11

START AFT: Fri Oct  8 15:39:01 CDT 2004
END BEFOR: Fri Oct  8 15:39:25 CDT 2004
COMMAND             START    END          REAL      CPU    CHARS   BLOCKS
NAME       TTYNAME  TIME     TIME       (SECS)   (SECS)   TRNSFD     READ  USER
#cc1       pts/0   15:39:01 15:39:01     0.27     0.25    398656        0  root
#as        pts/0   15:39:01 15:39:01     0.05     0.00    257024        0  root
#gcc       pts/0   15:39:01 15:39:01     0.36     0.00      3724        0  root
#cc1       pts/0   15:39:02 15:39:02     0.50     0.47    409792        0  root
#as        pts/0   15:39:02 15:39:02     0.05     0.02    375296        0  root
#gcc       pts/0   15:39:02 15:39:02     0.58     0.00      3724        0  root
```

```
#cc1       pts/0   15:39:02 15:39:02     0.50     0.47    411648        0  root
.............. Omitted lines ...........
```

3.5.2 Connect-time usage

Use the **ac** command to display connect-time records. To use this command, the accounting system should be active and the /var/adm/wtmp file must exist. Connect time is given in hours and is rounded to hundredths (Example 3-31).

Example 3-31 Using **ac** *to display connect time information*

Total connect time for all users:

```
[#][/]> ac
        total   304.58
```

Total connect time for user tester1:

```
[#][/]> ac tester1
        total    45.06
```

Total connect time for user tester1 and root:

```
[#][/]> ac root tester1
        total   265.19
```

Individual connect time and the sum of each individual connect time for the users specified by the -p parameter:

```
[#][/]> ac -p root tester1
        root 220.15
        tester1 45.06
        total   265.21
```

For a complete description of the **ac** command, refer to *AIX 5L Version 5.3, Commands Reference, Volume 1, a - c,* SC23-4888.

3.5.3 Who is connected to the system

The **who** command is used to display the users who are currently logged on to the system, using by default the /etc/utmp file.

Example 3-32 Sample output of the who command

```
<adm>[/var/adm/acct/sum]> who
root        pts/0       Oct 12 14:37     (9.12.6.176)
root        pts/1       Oct 13 16:44     (9.12.6.176)
root        pts/3       Oct 12 17:13     (192.168.100.1)
root        pts/4       Oct 13 15:58     (192.168.100.1)
```

```
root        pts/5       Oct 14 13:52     (192.168.100.1)
root        pts/6       Oct 12 15:40     (192.168.100.1)
root        pts/7       Oct 14 15:08     (192.168.100.1)
root        pts/8       Oct 14 13:33     (192.168.100.1)
root        pts/9       Oct 14 13:46     (192.168.100.1)
root        pts/10      Oct 14 15:36     (9.12.6.143)
root        pts/11      Oct 14 15:43     (9.12.6.176)
root        pts/12      Oct 14 14:14     (node6)
root        pts/13      Oct 14 15:45     (9.12.6.177)
tester1     pts/14      Oct 14 15:46     (9.12.6.177)
tester2     pts/15      Oct 14 15:47     (9.12.6.177)
```

Example 3-32 on page 103 displays these fields: name, line, time, host. The general output format of the **who** command is as follows:

name The user's login name.

state Indicates whether the line is writable by everyone (see the -T flag).

line Identifies the line name as found in the /dev directory.

time Represents the time when the user logged in.

activity Represents the hours and minutes since activity last occurred on that user's line. A dot (.) here indicates line activity within the last minute. If the line has been quiet more than 24 hours or has not been used since the last system startup, the entry is marked as old.

pid Identifies the process ID of the user's login shell.

term Identifies the process termination status (see the -d flag). For more information about termination values, refer to the wait subroutine or to the /usr/include/sys/signal.h file.

exit Identifies the exit status of ended processes (see the -d flag).

hostname Indicates the name of the machine the user is logged in from.

Using the **who** command with the -u flag displays the user name, tty, login time, line activity, process ID, and host name of each current user (Example 3-33).

Example 3-33 Using the who command with the -u flag

```
[#][/]> who -u
root        pts/0       Oct 18 15:45     1:11     17238  (new_gw)
root        pts/1       Oct 18 17:56      .       10802  (tot198.itso.ibm.com)
[#][/]>
```

It is possible to use a different file, such as /var/tmp/wtmp, in which case you should specify the name of the file as a parameter. (Example 3-34).

Example 3-34 Example of use of the who command with the /var/adm/wtmp file

```
[#][/work]> who /var/adm/wtmp |head
root         pts/0        Oct 12 14:37     (9.12.6.176)
root         pts/1        Oct 15 14:27     (9.12.6.177)
root         pts/2        Oct 15 14:52     (9.12.6.177)
root         pts/3        Oct 12 17:13     (192.168.100.1)
root         pts/4        Oct 13 15:58     (192.168.100.1)
root         pts/5        Oct 15 12:07     (9.12.6.176)
root         pts/6        Oct 12 15:40     (192.168.100.1)
root         pts/7        Oct 15 08:55     (9.12.6.176)
root         pts/9        Oct 15 12:07     (9.12.6.176)
root         pts/11       Oct 14 15:43     (9.12.6.176)
```

Read more about the **who** command in *AIX 5L Version 5.3, Commands Reference, Volume 6, v - z,* SC23-4893.

3.5.4 CPU usage

You can obtain CPU usage information using the /var/adm/pacct file and these commands:

accprc1 Reads records from standard input that are in the acct format, adds the login names that correspond to user IDs, then writes an ASCII record to standard output.

accprc2 Reads (from standard input) the records written by **acctprc1**, summarizes them by user ID and name, and writes the sorted summaries to standard output as total accounting (binary) records.

prtacct Formats and displays the binary tacct file generated by **acctprc2**.

The accounting system must be activated to use these three commands (Example 3-35).

Example 3-35 Using the acctprc1, acctprc2, and prtacct commands

```
[#][/tmp]> acctprc1 < /var/adm/pacct > out1.file
[#][/tmp]> head out1.file
0          root       0.000000        0.000001  0       0       0
0          root       0.000000        0.000001  1716    0       0
0          root       0.000000        0.000001  0       0       0
0          root       0.000000        0.000001  0       0       0
0          root       0.000000        0.000001  0       0       0
```

```
0          root       0.000000        0.000001  0       0       0
0          root       0.000000        0.000001  62208   0       0
0          root       0.000000        0.000001  648     0       0
0          root       0.000000        0.000001  31080   0       0
0          root       0.000000        0.000001  10288   0       0
[#][/tmp]> acctprc2 < out1.file > out.tacct
[#][/tmp]> file out.tacct
out.tacct: data or International Language text

[#][/tmp]> prtacct -f 1,2,3,4,5  out.tacct |head -20

Fri Oct  8 13:46:01 CDT 2004  Page 1

            LOGIN    CPU       CPU       KCORE
UID         NAME     PRIME     NPRIME    PRIME

0           TOTAL    6         1         1134
0           root     5         0         1072
1           daemon   0         0         0
4           adm      0         0         0
202         tester1  0         0         3
203         tester2  0         1         58
205         sshd     0         0         2
```

In Example 3-35 on page 105, the records of the out1.file generated by **acctprc1** are in acct file format and contain the user ID, login name, prime CPU time, non-prime CPU time, the total number of characters transferred (in 1024-byte units), the total number of blocks read and written, and main memory size (in 64-byte units) for each process.

The second file, generated by **acctprc2** and called out.tacct, is in tacct binary format. The **prtacct** command is used to display certain fields of this binary file.

3.5.5 Disk usage

The following utilities are provided for displaying disk usage: **diskusg**, **acctdisk**, **acctdusg**, **dacct**, and **acctmerge**.

Using the diskusg command

You can use **diskusg**, **acctdisk**, and **acctmerge** to generate your own set of disk accounting data. The **diskusg** command collects data. **acctdisk** reads the output lines of the **diskusg** or **acctdusg** commands from standard input, converts each individual record into a total accounting record, and writes the records to standard output. The **acctmerge** command is used for printing the data. As we have seen in “Disk-usage accounting” on page 60, **diskusg** can be used only for local disk accounting data and only for filesystem block accounting.

The **diskusg** command, when executed, prints (see Example 3-36):

- The user ID
- The corresponding login name
- The number of blocks charged to the user

Example 3-36 The diskusg command (data to the standard output)

```
[#][/]> diskusg /dev/lv00
   0    root            224112
   3    sys             8
 207    tester9         32
[#][/]>
```

Notes:

- If there are disk blocks belonging to a user ID without equivalent login names in /etc/passwd, they should *not* be reported. This situation may occur if you have removed a user account from the system, or if you have restored files from a backup created on another system under another (not existing on local) user ID.
- Use the -u flag in this case. The -u flag writes a record for each file charged to a user ID without a corresponding login entry (consisting of filesystem, inode number, and user ID). The -v flag writes a list of all files charged to a user ID without a corresponding login entry to the standard output. (See Example 3-37.)

Example 3-37 Disk blocks belonging to nonexistent user IDs

```
[#][/]> diskusg -u file.out /dev/lv00
   0    root    224112
   3    sys     8
 207    tester9 32

[#][/]> head file.out
/dev/lv00 16407 210
/dev/lv00 16409 210
/dev/lv00 16410 210
/dev/lv00 16411 210
/dev/lv00 16412 210
/dev/lv00 16413 210
/dev/lv00 16414 210
/dev/lv00 16415 210
/dev/lv00 16416 210
/dev/lv00 16417 210

[#][/]> diskusg -v /dev/lv00|head
The uid is not valid.
```

```
        File system: /dev/lv00, inode: 16407, uid: 210
The uid is not valid.
        File system: /dev/lv00, inode: 16409, uid: 210
The uid is not valid.
        File system: /dev/lv00, inode: 16410, uid: 210
The uid is not valid.
        File system: /dev/lv00, inode: 16411, uid: 210
... more lines
```

The **diskusg** command accepts the -p flag as a parameter, enabling use of another password file than the default /etc/passwd. You may use if NIS is used and you do not have access to the file that generates the passwd.byname map (Example 3-38).

Example 3-38 Disk accounting data using a different password file

```
[#][/work]> diskusg -p /work/passwd /dev/lv00
   0    root     224112
   3    sys      8
 207    tester9  32
 210    newuser  22616
```

A limitation of **diskusg**: By default, it can handle only 5000 entries in the password file. If there are more than that, use the -U flag to overcome this limit.

Another flag accepted is -X, used in conjunction with long login user names.

Using the acctdusg command

Use this command in conjunction with the **find** command to print disk accounting data related to local directories on your system. The same flags as for **diskusg**, -X, -u, and -p, are accepted (Example 3-39).

Example 3-39 Example of using acctdusg command

```
[#][/work]> find . -print |acctdusg
00000000000     root             221616
00000000003     sys                   8
00000000207     tester9              32
```

The output displays:

- the user ID
- the user name
- the number of 512-byte disk blocks belonging to that user

Using an existing dacct file and acctmerg command

The output of the **dodisk** and **acctdusg** commands can be redirected to a file and this file can be converted to a tacct binary format using **acctdisk** command:

Example 3-40 Example of generating binary tacct file using acctdisk command

```
[#][/work]> find . -print |acctdusg > /var/adm/dtmp
[#][/work]> acctdisk < /var/adm/dtmp > /var/adm/acct/nite/dacct
```

Example 3-40 shows how to get the data manually, but usually disk-usage records are stored in the file /var/adm/acct/nite/dacct, and are collected by the **dodisk** command, which is launched by the **cron** daemon. We can use the **acctmerg** command, usually launched automatically by **runacct** command, to display the disk-usage records (see Example 3-41).

Example 3-41 Example of acctmerg command used to show disk usage

```
[#][/]> acctmerg -a1,2,13  -h -v < /var/adm/acct/nite/dacct
            LOGIN    DISK
UID         NAME     BLOCKS

0           root     1.372e+06
1           daemon   8.000e+00
2           bin      1.552e+06
3           sys      8.000e+00
4           adm      1.256e+03
5           uucp     2.000e+03
6           invscout 8.000e+00
7           nuucp    8.000e+00
200         snapp    8.000e+00
201         ipsec    2.526e+04
202         tester1  4.800e+01
203         tester2  3.200e+01
204         tester3  8.000e+00
```

The preceding example prints in ASCII format the fields 1, 2 and 13 (-a1.2.13) of the records stored in /var/adm/acct/dacct, along with a header (-h) and using scientific conventions for the 1KB blocks used by each user (-v).

For a complete description of the **acctmerg** command, refer to *AIX 5L Version 5.3, Commands Reference, Volume 1, a - c,* SC23-4888.

3.5.6 Printer usage

Use the **pac** command to display printer or plotter accounting records. You must have system accounting activated as described in “Quick setup of the accounting

subsystem” on page 43, and the acctfile = *filename* attribute must be present in the queue definition stanza.

Example 3-42 Using the pac command

To display the accounting printer information for all users:

```
[#][/]> pac
  Login               pages/feet   runs               price
root                       14.00    7                USD .28
tester1                     4.00    2                USD .08
```

To display the accounting printer information for one user:

```
[#][/]> pac tester
  Login               pages/feet   runs               price
tester                      0.00    0                USD .00
```

The following parameters are displayed in these examples:

- The login user name of the user that used the printer
- The number of pages (or feet) printed
- The number of times the printer was used
- The price for the printed pages

By default, the **pac** command charges $0.02 per unit. You can modify this with the -p parameter. There are many other parameters, such as sorting users alphabetically, specifying another queue configuration file, and so on. For a complete description of the **pac** command, refer to *AIX 5L Version 5.3, Commands Reference, Volume 4, n - r,* SC23-4891.

This command does not void, erase, or rotate the qacct file, so you can use it without restrictions. Instead, the qacct file is voided when the **runacct** command is run in the QUEUEACCT stage.

3.6 Troubleshooting potential accounting errors

A system crash, a filesystem out of space, or inconsistencies in any binary accounting file are situations that can arise. This section presents some procedures for fixing potential (or actual) problems.

3.6.1 Fixing tacct errors

The integrity of the /var/adm/acct/sum/tacct file is important, especially if you are using the accounting system to charge users for system resources. Occasionally, tacct records appear to contain negative numbers, duplicate user numbers, or a user number of 65,535.

How to fix a damaged tacct file:

1. Move to the /var/adm/acct/sum directory by typing:

   ```
   cd /var/adm/acct/sum
   ```

2. Use **prtacct** to check the total accounting file, tacctprev, by typing:

   ```
   prtacct tacctprev
   ```

3. The **prtacct** command formats and displays the tacctprev file so that you can check connect time, process time, disk usage, and printer usage. If the tacctprev file looks correct, change the latest tacct.*mmdd* file from a binary file to an ASCII file. In the following example, the **acctmerg** command converts the tacct.mmdd file to an ASCII file named tacct.new:

   ```
   acctmerg -v < tacct.mmdd > tacct.new
   ```

   Note: The **acctmerg** command with the -a flag also produces ASCII output. The -v flag produces more precise notation for floating-point numbers.

   The **acctmerg** command is used to merge the intermediate accounting record reports into a cumulative total report (tacct). This cumulative total is the source from which the **monacct** command produces the ASCII monthly summary report. Since the **monacct** procedure removes all tacct.*mmdd* files, recreate the tacct file by merging these files.

4. Edit the tacct.new file to remove the bad records and write duplicate user number records to another file by typing:

   ```
   acctmerg -i < tacct.new > tacct.mmdd
   Create the tacct file again by typing:
   acctmerg tacctprev < tacct.mmdd > tacct
   ```

3.6.2 Fixing wtmp errors

The /var/adm/wtmp file might cause problems in the day-to-day operation of the accounting system. When the date is changed and the system is in multiuser mode, date change records are written to the /var/adm/wtmp file. When a date change is encountered, the **wtmpfix** command adjusts the time stamps in the wtmp records. Some combinations of date changes and system restarts may slip past **wtmpfix** and cause the **acctcon1** command to fail and the **runacct** command to send mail to the root and adm accounts listing incorrect dates.

To fix a damaged wtmp file:

1. Move to the /var/adm/acct/nite directory by typing:

   ```
   cd /var/adm/acct/nite
   ```

2. Convert the binary wtmp file to an ASCII file that you can edit by typing:

   ```
   fwtmp <wtmp.mmdd> wtmp.new
   ```

   The **fwtmp** command converts wtmp from binary to ASCII.

3. Edit the ASCII wtmp.new file to delete damaged records or all records from the beginning of the file until the needed date change by typing:

   ```
   vi wtmp.new
   ```

4. Convert the ASCII wtmp.new file back to binary format by typing:

   ```
   fwtmp -ic <wtmp.new> wtmp.mmdd
   ```

5. If the wtmp file is beyond repair, use the **nulladm** command to create an empty wtmp file:

   ```
   /usr/sbin/acct/nulladm /var/adm/wtmp
   ```

6. The **nulladm** command creates the specified file with read and write permissions for the file owner and group, and read permissions for other users. It ensures that the file owner and group are adm.

3.6.3 Fixing incorrect file permissions

To use the accounting system, file ownership and permissions must be correct. The adm administrative account owns the accounting command and scripts, except for the **/var/adm/acct/accton** command, which is owned by root.

1. Check the file permissions of the /var/adm/acct files and directories:

   ```
   [#[/var/adm/acct]> ls -la /var/adm/acct
   total 16
   drwxrwxr-x   5 adm      adm             256 Oct 05 14:26 .
   drwxrwxr-x  13 root     adm            4096 Oct 05 17:05 ..
   drwxr-xr-x   2 adm      adm             256 Oct 05 14:26 fiscal
   drwxr-xr-x   2 adm      adm             256 Oct 05 14:26 nite
   drwxr-xr-x   2 adm      adm             256 Oct 05 14:26 sum
   ```

2. Adjust file permissions with the **chown** command, if necessary. The permissions are 755 (all permissions for owner, and read and execute permissions for all others).

3. Also, the directory itself should be write-protected from others. If necessary, change the file permission and the ownership of the files as follows:

   ```
   cd /var/adm/acct
   chown adm.adm . sum/* nite/* fiscal/*
   ```

4. To prevent tampering by users trying to avoid charges, deny write permission for others on these files. Change the **accton** group owner to adm, and permissions to 710 (that is, no permissions for others). Processes owned by adm can execute the **accton** command, but ordinary users cannot do that.

5. The /var/adm/wtmp file must also be owned by adm. If /var/adm/wtmp is owned by root, you see the following message during startup:

   ```
   /var/adm/acct/startup: /var/adm/wtmp: Permission denied
   ```

   To correct the ownership of the /var/adm/wtmp file, change ownership to the adm user and group by typing this command:

   ```
   chown adm.adm /var/adm/wtmp
   ```

3.6.4 Fixing qacct access file errors

If you activate the print queue accounting but you forget to create the qacct file with the **nulladm** command, the print queue system does that for you but with the wrong file access permissions:

```
<adm>[/var/adm]> ls -l qacct
--w-r-xr--   1 root     printq          268 Oct 15 09:32 qacct
```

The **runacct** command, if run by adm, cannot erase the qacct file. Change permissions and ownership for adm.

3.6.5 Fixing runacct errors

The **runacct** command usually processes files that are very large. The procedure consumes considerable system resources while it is taking place, so **runacct** is normally run at hours when it can take over the machine and not disturb anyone.

The **runacct** command is a script with several stages. The stages enable you to restart the command where it stopped without having to rerun the entire script. As it completes one stage, it writes the name of the next in the state file.

When **runacct** encounters problems, it sends error messages to different destinations depending on where the error occurred. Usually it sends a date and a message to the console, directing you to look in the active*mmdd* file (such as active0621 for June 21st), which is in the /usr/adm/acct/nite directory. When the **runacct** command aborts, it moves the entire active file to active*mmdd* and appends a message describing the problem.

3.6.6 Updating an out-of-date holidays file

You should update the holiday file if you receive a mail message or if the /var/adm/acct/accterr file contains entries like this:

```
***UPDATE /etc/acct/holidays WITH NEW HOLIDAYS***
```

The /usr/lib/acct/holidays (symbolic link to /etc/acct/holidays**)** file is out of date after the last holiday listed has passed or when the year changes. Update the

out-of-date holidays file by editing the /var/adm/acct/holidays file. If the list of holidays is too long, the **acctcon1** command generates an error, and you must shorten your list. You are safe with 20 or fewer holidays. If you want to add more holidays, just edit the holidays file each month.

3.6.7 Fixing date change errors

Errors are expected to appear when changing the date of the system (such as when using Daylight Saving Time). Processing the /var/adm/wtmp file might produce some warnings mailed to root. The wtmp file contains information collected by the **/etc/init** and **/bin/login** commands and is used by accounting scripts primarily for calculating connect time (the length of time a user is logged in). Unfortunately, date changes confuse the program that processes the wtmp file. As a result, the **runacct** command sends mail to root and adm complaining of errors after a date change since the last time accounting was run.

1. Determine whether you received any errors.

   The **acctcon1** command outputs error messages that are mailed to adm and root by the **runacct** command. For example, if **acctcon1** stumbles after a date change and fails to collect connect times, adm might get mail similar to:

   ```
   Mon Jan 6 11:58:40 CST 1992
   acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
   new: Mon Jan 6 11:57:59 1992
   acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
   new: Mon Jan 6 11:57:59 1992
   acctcon1: bad times: old: Tue Jan 7 00:57:14 1992
   new: Mon Jan 6 11:57:59 1992
   ```

2. Adjust the wtmp file by typing:

   ```
   /usr/sbin/acct/wtmpfix wtmp
   ```

   The **wtmpfix** command examines the wtmp file for date and time-stamp inconsistencies and corrects problems that could make **acctcon1** fail. However, some date changes slip by **wtmpfix** (see 3.6.2, “Fixing wtmp errors” on page 111).

3. Run accounting right before shutdown or immediately after startup.

   Using **runacct** at these times minimizes the number of entries with bad times. The **runacct** command continues to send mail to the root and adm accounts until you edit the **runacct** script, find the WTMPFIX section, and comment out the line where the file log gets mailed to the root and adm accounts.

3.6.8 Restarting the runacct command

You might encounter situations when **runacct** does not complete. There are many reasons why this can happen; among the most common are filesystems out of space, damaged wtmp or pacct files, or system stops.

The first step is checking the following files for errors:

- /var/adm/acct/nite/active*ddmm*
- /var/adm/acct/nite/accterr
- /var/adm/acct/nite/statefile

The last file mentioned, statefile, should contain the word COMPLETE. If not, and if **runacct** is not running, there was a problem.

Perform any action you consider necessary to eliminate errors. Then delete the lock and lastdate files from the /var/adm/acct/nite directory.

Finally, restart the **runacct** command as follows:

1. If you have discovered errors the next day after **runacct** was scheduled to run and you want it to run for a specific date, restart the command for a specific date by typing:

   ```
   nohup runacct 1022 2>>/var/adm/acct/nite/accterr&
   ```

   This command restarts the **runacct** command for October 22, redirecting errors into the accterr file.

2. Type `runacct` for starting the command at the current date, from the state written in the statfile file.

3. If you want to restart **runacct** at a specific date at a specific state, use:

   ```
   nohup runacct 1022 MERGE 2>>/var/adm/acct/nite/accterr&
   ```

3.6.9 Recommendations

- Do not attempt to modify the implicit names of data accounting files (pacct, qacct, fee, wtmp) unless you know what you are doing. Many files are involved in the accounting system so there may be consequences.
- Remember to digest your /etc/qconfig file with **enq -d** if you have modified this file for printing accounting.
- Do not use different file names for printing accounting inside the /etc/qconfig file.
- Remember the limitations of the printing accounting system.
- Remember that disk accounting can only be performed on local disks.

- Remember to use the -X flag for the accounting scripts if you use long login user names.
- Do not attempt to use accounting commands to see processes still running. Use the **ps** command instead. The process accounting information is written in the pacct file *only* when the process exits.
- Remember that some commands erase data accounting files.
- Remember to verify the **runacct** command for -X problems.
- Do not attempt to use **halt** or **reboot** to stop your system. Use **shutdown** instead.

3.7 Accounting files

The accounting data files reside in the following directories:

- The binary files and shell commands are locate in /usr/sbin/acct. For compatibility with System V, the same files reside in /usr/lib/acct, and they are soft-linked to the accounting files from /usr/sbin/acct (Example 3-43).

Example 3-43 The /usr/lib/acct directory

```
[#][/usr/lib/acct]> ls -l
total 0
lrwxrwxrwx   1 root     adm             17 Oct 04 18:01 ac -> /usr/sbin/acct/ac
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 acctcms -> /usr/sbin/acct/acctcms
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 acctcon1 -> /usr/sbin/acct/acctcon1
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 acctcon2 -> /usr/sbin/acct/acctcon2
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 acctdisk -> /usr/sbin/acct/acctdisk
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 acctdusg -> /usr/sbin/acct/acctdusg
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 acctmerg -> /usr/sbin/acct/acctmerg
lrwxrwxrwx   1 root     adm             21 Oct 04 18:01 accton -> /usr/sbin/acct/accton
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 acctprc1 -> /usr/sbin/acct/acctprc1
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 acctprc2 -> /usr/sbin/acct/acctprc2
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 acctwtmp -> /usr/sbin/acct/acctwtmp
lrwxrwxrwx   1 root     adm             24 Oct 04 18:01 chargefee -> /usr/sbin/acct/chargefee
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 ckpacct -> /usr/sbin/acct/ckpacct
lrwxrwxrwx   1 root     adm             17 Oct 04 18:01 diskusg -> /usr/sbin/diskusg
lrwxrwxrwx   1 root     adm             21 Oct 04 18:01 dodisk -> /usr/sbin/acct/dodisk
lrwxrwxrwx   1 root     adm             20 Oct 04 18:01 fwtmp -> /usr/sbin/acct/fwtmp
lrwxrwxrwx   1 root     adm             18 Oct 04 18:01 holidays -> /etc/acct/holidays
lrwxrwxrwx   1 root     adm             24 Oct 04 18:01 lastlogin -> /usr/sbin/acct/lastlogin
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 monacct -> /usr/sbin/acct/monacct
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 nulladm -> /usr/sbin/acct/nulladm
lrwxrwxrwx   1 root     adm             21 Oct 04 18:01 prctmp -> /usr/sbin/acct/prctmp
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 prdaily -> /usr/sbin/acct/prdaily
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 prtacct -> /usr/sbin/acct/prtacct
```

```
lrwxrwxrwx   1 root     adm             25 Oct 04 18:01 ptecms.awk ->
/usr/sbin/acct/ptecms.awk
lrwxrwxrwx   1 root     adm             25 Oct 04 18:01 ptelus.awk ->
/usr/sbin/acct/ptelus.awk
lrwxrwxrwx   1 root     adm             21 Oct 04 18:01 remove -> /usr/sbin/acct/remove
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 runacct -> /usr/sbin/acct/runacct
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 shutacct -> /usr/sbin/acct/shutacct
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 startup -> /usr/sbin/acct/startup
lrwxrwxrwx   1 root     adm             23 Oct 04 18:01 turnacct -> /usr/sbin/acct/turnacct
lrwxrwxrwx   1 root     adm             22 Oct 04 18:01 wtmpfix -> /usr/sbin/acct/wtmpfix
[#][/usr/lib/acct]>
```

- The data accounting files pacct*i*, wtmp, dtmp, fee, and qacct reside in /var/adm.
- The report and summary files reside in /var/adm/acct.

Hereafter follows a short description of each file and directory.

3.7.1 Accounting commands

Some of these commands run automatically, called by the **cron** daemon, to generate summary and reports or to do maintenance tasks. Others are used to start and stop the accounting system. Some of these commands may also be launched from STDIN (by keyboard).

Commands usually called by the cron daemon

Table 3-2 Accounting commands launched from cron

Name	Description
runacct	The main accounting daily procedure. It calls several different other commands in order to produce the daily reports and summaries.
ckpacct	Handles the /var/adm/pacct file.
dodisk	Produces disk-usage records.
monacct	Produces the monthly summary from daily reports.
sa1	Collects and stores binary data in the /var/adm/sar*dd* file.
sa2	Writes a daily report in the /var/adm/sa/sa*dd* file. This command removes sa*dd* files that are more than a week old.

Commands called at system startup

Table 3-3 Accounting commands (system startup)

Name	Description
startup	Starts the accounting procedures at system startup, usually from /etc/rc.
shutacct	Properly turns off the accounting subsystem by calling acctwtmp, which writes a shutdown record into /var/adm/wtmp, then calls turnacct off to stop the accounting system.

Commands launched from the shell prompt (using keyboard)

Table 3-4 Accounting commands launched interactively

Name	Description
ac	Prints connect-time records.
acctcms	Generates command-usage summaries from accounting records.
acctcom	Displays process accounting summaries. Available to users.
acctcon1	Summarizes the login and logout records from the wtmp file into session records.
acctcon2	Converts the output of acctcon1 to tacct format.
acctdisk	Converts dtmp file generated by diskusg or acctdusg to tacct format.
acctdusg	Generates dtmp file - disk usage statistics for local filesystems or directories.
acctmerg	Merges tacct files into a single tacct file.
accton	Turns on and off the accounting system.
acctprc1	Generates ASCII records from acct file input, adding the corresponding user login name for the existing user ID.
acctprc2	Generates tacct files from the output of acctprc1.
acctwtmp	Writes a record to the wtmp file.
chargefee	Charges the user a predetermined fee for units of work performed. The charges are added to the daily report by the acctmerg command.
diskusg	Generates dtmp file - disk usage statistics for local filesystems.
fwtmp	Converts files between binary and ASCII formats.
last	Displays information about previous logins.
lastcomm	Displays information about the last commands that were executed.

Name	Description
lastlogin	Displays the time each user last logged in.
nulladm	Creates void files with user and group adm, read and write permissions for user and group, and read permission for the others.
pac	Prepares printer/plotter accounting records.
prctmp	Displays a record session.
prdaily	Creates an ASCII report of the previous day's accounting data.
prtacct	Prints total accounting files.
remove	Removes pacct*, wtmp*, and lock* files.
sa	Summarizes raw accounting information to help manage large volumes of accounting information.
sa1	Collects and stores binary data in the /var/adm/sa/sa*dd* file.
sa2	Writes a daily report in the /var/adm/sa/sar*dd* file.
sadc	Reports on various local system actions, such as buffer usage, disk and tape I/O activity, TTY device activity counters, and file access counters.
sar	Writes to standard output the contents of selected cumulative activity counters in the operating system. The sar command reports only on local activities.
turnacct	Starts or stops the accounting system. It can restart a new void pacct file if used with the switch parameter.
time	Prints real time, user time, and system time required to run a command.
timex	Reports in seconds the elapsed time, user time, and run time.
wtmpfix	Verifies and fixes inconsistencies in the wtmp file.

3.7.2 Accounting data files

The accounting data files are generated in the /var/adm directory.

Table 3-5 Accounting data files

Name	Type	Format	
diskdiag	ASCII	-	Diagnostic output during the running of disk accounting programs
dtmp	ASCII	-	Output from the **acctdusg** command

Name	Type	Format	
fee	ASCII	ASCII tacct	Output from the **chargefee** command
pacct	BIN	tacct	Process accounting file
qacct	BIN	accres	Printer accounting data file
wtmp	BIN	utmp	Connect-time accounting records
Spacct*i.mmdd*	BIN	tacct	Temporary files generated by the **runacct** command from pacct*i* files for the month *mm* and day *dd*. This are actually the pacct*i* files renamed in Spacct*i.mmdd* for day *dd* month *dd*.

3.7.3 Report and summary files

The report and summary files reside in the /var/adm/acct directory; in nite, sum, and fiscal directories; or in sumx, nitex, and fiscal directories if long login use names are used:

- The nite subdirectory contains files that the **runacct** command reuses daily (Example 3-44):

Example 3-44 The /var/adm/acct/nite directory

```
<adm>[/var/adm/acct/nite]> ls -la
total 152
drwxr-xr-x   2 adm      adm            4096 Oct 14 08:59 .
drwxrwxr-x   7 adm      adm             256 Oct 11 08:22 ..
-rw-r--r--   1 adm      adm               0 Oct 14 04:00 accterr
-rw-rw-r--   1 adm      adm            1928 Oct 14 04:00 active
-rw-rw-r--   1 adm      adm            5411 Oct 14 04:00 cms
-rw-rw-r--   1 adm      adm             990 Oct 14 04:00 ctmp
-rw-rw-r--   1 adm      adm            5411 Oct 14 04:00 daycms
-rw-rw-r--   1 adm      adm             432 Oct 14 04:00 daytacct
-rw-rw-r--   1 adm      adm               5 Oct 14 04:00 lastdate
-rw-rw-r--   1 adm      adm             296 Oct 14 04:00 lineuse
-rw-rw-r--   1 adm      adm               0 Oct 14 04:00 log
-rw-rw-r--   1 adm      adm           23328 Oct 14 04:00 owtmp
-rw-rw-r--   1 adm      adm             100 Oct 14 04:00 reboots
-rw-rw-r--   1 adm      adm               9 Oct 14 04:00 statefile
-rw-rw-r--   1 adm      adm               0 Oct 14 04:00 wtmperror
```

- The sum subdirectory contains a summary that the **runacct** command updates daily (Example 3-45).

Example 3-45 The /var/adm/acct/sum directory

```
<adm>[/var/adm/acct/sum]> ls -la
total 456
drwxr-xr-x   2 adm      adm            4096 Oct 14 08:58 .
drwxrwxr-x   7 adm      adm             256 Oct 11 08:22 ..
-rw-rw-r--   1 adm      adm           19620 Oct 14 04:00 cms
-rw-rw-r--   1 adm      adm           19320 Oct 14 04:00 cmsprev
-rw-rw-r--   1 adm      adm            8760 Oct 14 04:00 daycms
-rw-r--r--   1 adm      adm             340 Oct 14 08:58 loginlog
-rw-rw-r--   1 adm      adm           12967 Oct 06 04:00 rprt1006
-rw-rw-r--   1 adm      adm           13496 Oct 07 04:00 rprt1007
-rw-rw-r--   1 adm      adm           13732 Oct 08 04:00 rprt1008
-rw-rw-r--   1 adm      adm           14039 Oct 09 04:00 rprt1009
-rw-rw-r--   1 adm      adm           13729 Oct 11 04:00 rprt1011
-rw-rw-r--   1 adm      adm           14562 Oct 12 04:00 rprt1012
-rw-rw-r--   1 adm      adm           14589 Oct 13 04:00 rprt1013
-rw-rw-r--   1 adm      adm           13134 Oct 14 04:00 rprt1014
-rw-rw-r--   1 adm      adm            1224 Oct 14 04:00 tacct
-rw-rw-r--   1 adm      adm             720 Oct 06 04:00 tacct1006
-rw-rw-r--   1 adm      adm            1008 Oct 07 04:00 tacct1007
-rw-rw-r--   1 adm      adm            1080 Oct 08 04:00 tacct1008
-rw-rw-r--   1 adm      adm            1152 Oct 09 04:00 tacct1009
-rw-rw-r--   1 adm      adm            1080 Oct 11 04:00 tacct1011
-rw-rw-r--   1 adm      adm            1152 Oct 12 04:00 tacct1012
-rw-rw-r--   1 adm      adm            1152 Oct 13 04:00 tacct1013
-rw-rw-r--   1 adm      adm             432 Oct 14 04:00 tacct1014
-rw-rw-r--   1 adm      adm            1152 Oct 14 04:00 tacctprev
```

- The fiscal subdirectory contains monthly summary files created by the **monacct** command (Example 3-46).

Example 3-46 The /var/adm/acct/fiscal directory

```
total 96
drwxr-xr-x   2 adm      adm             512 Oct 08 17:54 .
drwxrwxr-x   5 adm      adm             512 Sep 29 19:12 ..
-rw-rw-r--   1 adm      adm           11100 Oct 08 04:00 cms10
-rw-r--r--   1 root     system        20926 Oct 08 17:54 fiscrpt10
-rw-rw-r--   1 adm      adm             432 Oct 08 04:00 tacct10
[#][/var/adm/acct/fiscal]>
```

Files in /var/adm/acct/nite directory

Table 3-6 Files in the /var/adm/acct/nite directory

Name	Type	Format	Description
active	ASCII		Used by the **runacct** command to record progress and print warning and error messages. The file active.mmdd is a copy of the active file made by the runacct program after it detects an error
accterr	ASCII		Diagnostic output produced during the execution of the **runacct** command
cms	ASCII		Total command summary used by the **prdaily** command
ctmp	ASCII	ctmp	Connect session records
dacct	BIN		Disk total accounting records, created by the **dodisk** command
daycms	ASCII		Daily command summary used by the **prdaily** command
daytacct	BIN	tacct	Total accounting records for one day
lastdate	ASCII	date +%m%d	Last day the **runacct** executed, in date +%m%d format.
lineuse	ASCII		tty line usage report used by the **prdaily** command.
log	ASCII		Diagnostic output from the acctcon1 command
lock1	ASCII		Used to control serial use of the **runacct** command
log*mmdd*	ASCII	-	Same as log after the **runacct** command detects an error.
reboots	ASCII	-	Contains beginning and ending dates from wtmp, and a listing of system restarts.
statefile	ASCII	-	Used to record the current state during execution of the **runacct** command.
tmpwtmp	BIN	utmp	wtmp file corrected by the wtmpfix command.
wtmperror	ASCII	-	Contains wtmpfix error messages.

Name	Type	Format	Description
wtmperr*mmdd*	ASCII	-	Same as wtmperror after the **runacct** command detects an error.
wtmp.*mmdd*	BIN	utmp	Contains previous day's wtmp file. Removed during the cleanup of **runacct** command.

Files in /var/adm/acct/sum directory

Table 3-7 The files in the sum directory

Name	Type	Format	Description
cms	BIN	tacct	Total command summary file for the current fiscal period.
cmsprev	BIN	tacct	Command summary file without the latest update.
daycms	BIN	tacct	Command summary file for the previous day, in binary format.
loginlog	ASCII		File created by the **lastlogin** command.
rprt*mmdd*	ASCII	-	Saved output of the **prdaily** command.
tacct	BIN	tacct	Cumulative total accounting file for the current fiscal period.
tacct*mmdd*	BIN	tacct	Total accounting file for *mmdd*.
tacctprev	BIN	tacct	Same as tacct without the latest update.

Files in /var/adm/acct/fiscal directory

Table 3-8 The fiscal directory

Name	Type	Format	Description
cms*i*	BIN	tacct	Total command summary file for the fiscal period *i*.
fiscrpt*i*	ASCII		A report similar to that of the **prdaily** command for fiscal period *i*.
tacct*mmdd*	BIN	tacct	Total accounting file for fiscal period

3.7.4 Accounting file formats

The following tables give short descriptions of accounting output binary files and formats.

Table 3-9 Accounting file formats

Name	Description
cms	Total accounting command summary used by the prdaily command, in binary format. The ASCII version is nite/cms.
ctacct	Connects total accounting records. The output of this file is defined in the tacct.h file.
ctmp	Connects session records files. The format is described in the ctmp.h file.
dayacct	Total accounting records for one day. The format of the file is defined in the tacct file format.
daycms	Daily command summary used by the prdaily command, in binary format. The ASCII version is nite/daycms.
pacct	Active process accounting file. The format of the file is defined in the acct.h file.
Spacct*i.mmdd*	Process accounting files for *mmdd* during the running of the runacct command. The format of these files is defined in the acct.h file.
sum/tacct	Total accounting records for one day. The format of the file is defined in the tacct file format.
wtmp	The cumulative connect accounting file. The format of the wtmp file is defined in the utmp.h file.
utmp	The active connect accounting file. The format of the utmp file is defined in the utmp.h file. Voided during the restart of the machine.

4

Accounting and the Workload Manager

This chapter describes the Workload Manager (WLM) feature of AIX and includes examples of how to use the WLM in conjunction with the AIX accounting subsystem. The following topics are discussed:

- Overview of AIX WLM
- Administering WLM on the AIX system
- WLM performance tools
- AIX accounting with WLM classes
- Resource management and workload control with WLM

This chapter intends to provide a framework for the users to set up WLM on AIX systems and illustrate the use of WLM performance tools and the AIX accounting subsystem in monitoring resource utilization.

4.1 Overview

It is imperative for businesses to understand the behavior of applications under heavy system workload and how these applications react to changes in system workload. This ensures optimal response times and the proper resource utilization, and guarantees server uptime in accordance with service level agreements by effectively gathering of statistics about resource usage.

It is becoming increasingly vital for system administrators to be able to determine and control resource usage by processes. There is a need to monitor how the resources on a system are being used, and to implement effective mechanisms to efficiently balance the allocation of resources among the processes.

The WLM feature on AIX provides a set of tools that assist in gleaning useful performance statistics and provide the system administrators with an efficient mechanism to control resource allocation to processes. WLM is primarily intended for use with large systems running multiple applications, databases, and transaction processing systems, where workloads are combined into a single large system (in a server consolidation environment).

WLM provides the flexibility for dividing system resources between jobs without having to partition the system (where reinstallation and reconfiguration are required). It also provides an effective means of isolation between jobs with very different system behaviors.

More and more organizations are charging user communities for computing services. WLM can be used effectively in conjunction with the AIX accounting subsystem to profile accounting information for WLM classes. These resource-usage statistics can be used for billing users for system resources.

4.2 WLM concepts

This section presents the concepts and terms used in AIX Workload Manager. The WLM uses object oriented technology based on classes, objects, and attributes.

4.2.1 Definitions

WLM functionality is based on entities called *classes*. System administrators can define classes with a set of attributes and resource limits and assign processes to a class based on assignment rules for the class. AIX WLM provides the ability to control allocation of resources (CPU, physical memory, and bandwidth) to these classes.

Processes are placed in these classes based on *users, groups, application paths, process types,* or *application tags*. These attributes form the assignment rules for classification of processes.

User ID The user name owning a process can be used to classify the process to a class. User IDs are available in the /etc/password file or from the NIS (Network Information Service). The **smitty lsuser** command lists the users on the system.

Group The group name of a process can be used to classify the process to a class. The group names are available in the /etc/group file or the NIS. The **smitty lsgroup** command will list the group name on the system.

Application path The complete path name of the binary running the application.

Process types Process type attributes specifying whether the process is 32-bit or 64-bit can be used to determine the class for a process.

Application tag An attribute set by the WLM API to enable classification for different instances of the same binary application.

Resource usage can be monitored and controlled at the class level. As the resource limits are set and resource utilization is regulated for each class, applications are prevented from interfering with each other when sharing a single server.

Web servers, databases, and batch programs executing low priority tasks in the background can be grouped into separate distinct classes.

4.2.2 Class hierarchy

A hierarchy of classes can be specified, with processes automatically assigned to these classes by their characteristics and manually placed in the classes based on simple rules.

The class hierarchy with two levels can be set up depending on the needs of the organization by defining *superclasses and subclasses*. A superclass has subclasses associated with it, and a subclass can belong only to one superclass.

The primary distinction between a superclass and a subclass is in the resource entitlement and control. At the superclass level, the resource shares and limits are based on the resources available on the system and managed by WLM. At the subclass level, the resource allocation is dependent on the resources that are available to the parent superclass of the subclass.

Resource entitlements are defined by the system administrator.

Notes:

- A class name can be a maximum of 16 characters.
- The names of subclasses belonging to a superclass must be different.
- Subclasses of different superclasses can have the same name.
- The fully qualified name of a subclass is *superclass_name.subclass name*.
- On AIX 5.1, WLM supports 32 superclasses (27 user-defined and 5 predefined). Each superclass in turn can have 12 subclasses (10 user-defined and 2 predefined).
- On AIX 5.2 and later, WLM supports 69 superclasses (64 user-defined) and 64 subclasses per superclass (61 user-defined).

The term “class” applies to both superclass and subclass throughout the discussions in this chapter.

The predefined superclasses are created automatically and are classified as:

Default All non-root processes that are not assigned automatically to a specific superclass are assigned to the default superclass.

System System superclass has all privileged (root) processes assigned to it if they are not assigned by rules to a specific class.

Shared Shared superclass receives all memory pages that are shared by processes in more than one superclass.

Unclassified Memory pages that cannot be directly tied to any processes (and thus, to any class) at the time of the initial classification are charged to the unclassified superclass.

Unmanaged A special superclass to which no processes are assigned. This class is used to accumulate the memory usage for all pinned pages that are not managed by WLM.

4.2.3 Class attributes

Class tiers Tiers define class importance relative to other classes. Tiers 0 through 9 can be defined to prioritize classes, with 0 being the most important and 9 the least important.

Inheritance Specifies whether the child process inherits the class assignment from its parent.

Localshm	Prevents memory segments belonging to one class from migrating to shared class.
Shares	Numbers for each class to determine the percentage share for allocation of CPU, memory, and disk I/O for the class.
Resource set	Limits the set of resources a given class has access to in terms of CPUs.

Figure 4-1 shows the WLM structure.

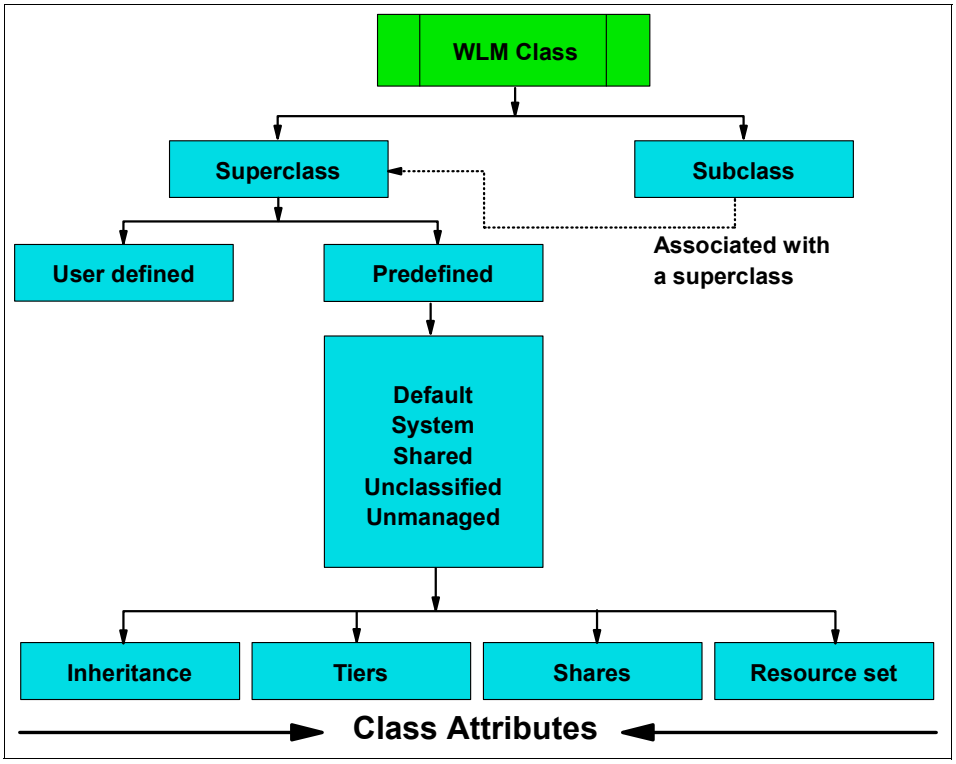

Figure 4-1 WLM class concept diagram

4.3 Administering WLM

This section describes how to configure and use the Workload Manager, and gives some practical examples.

4.3.1 WLM configuration: the six-step process

WLM can be set up on the system using the following six steps:

1. Determining the processes running on the system (the ones you intend to monitor)
2. Classification of the processes
3. Creation of WLM classes for these processes
4. Assigning the processes to pertinent classes by using assignment rules
5. Verifying the classes and assignment rules
6. Starting WLM in passive mode

The main idea is to classify the processes on the system based on certain parameters, such as the applications or workloads these processes belong to. Subsequently, these processes can be grouped into WLM classes and each class can be monitored and managed separately for its resource usage.

Read more about the steps to set up WLM in 4.3.3, “Setting up WLM” on page 131.

4.3.2 WLM administration tools

WLM can be administered in three different ways:

Command line WLM can be administered using simple commands and by editing a few configuration files.

SMIT (System Management Interface Tool) The classic ASCII-based AIX system administration tool provides a menu-based interface to WLM commands.

WebSM (Web-based System Manager) Java-based graphical tool for managing AIX systems.

We have used SMIT in the examples throughout this chapter.

Section 4.3.4, “Introduction to WLM commands and WebSM” on page 138 provides a brief introduction to WLM commands and the WebSM tool.

4.3.3 Setting up WLM

This section describes the steps needed to configure the WLM on AIX.

1. Determine the processes running on the system.

   The first step is to check for all processes that are running on the system, determine what the processes are doing and which application or workload they belong to, and decide how to classify the processes.

   The command in Example 4-1 can be used to check for the processes on the system.

Example 4-1 Sample output of ps -e -o pid,tag,user.group,comm,args

```
[p630n02][/]> ps -e -o pid,tag,user,group,comm,args

 4470 -                   root   system sshd     /usr/sbin/sshd
 5082 -                   root   system hostmibd /usr/sbin/hostmibd
 5168 -                   root   system shlap    /usr/ccs/bin/shlap
 5476 -                   root   system errdemon /usr/lib/errdemon
 6542 -                   root     cron cron     /usr/sbin/cron
 6722 -                   root   system getty    getty /dev/console console
 8010 -                   root   system dtlogin  /usr/dt/bin/dtlogin -daemon
13336 -                  user1    staff prog1    ./prog1 -c 1000
13592 -                   root   system telnetd  telnetd -a
19750 -                   root   system prog3    ./prog3 -m 2000
20056 -                   root   system ksh      -ksh
23016 -                   root   system sshd     sshd: root@pts/5
23882 -                   root   system ksh      -ksh
24428 -                  user2    staff prog2    ./prog2 -c 500
.................. Ommited lines ..................
```

The processes prog1, prog2, and prog3 (bold in the Example 4-1) will be used in this section for illustration.

prog1	CPU-intensive program executed by user1
prog2	CPU-intensive program executed by user2
prog3	Memory-intensive program

The programs simulate resource utilization and have been used for illustration purposes only.

2. Classify the processes.

   The next step is to classify the processes into pertinent WLM classes. The classes can be defined based on the computing needs of the users, the

nature of the applications, the resource requirements of the processes running on the system, and business priorities.

For example, the processes can be classified into different classes based on whether they are CPU-intensive or memory-intensive. By grouping the applications that have the same resource utilization patterns, system administrators can effectively use WLM to regulate resource utilization.

If you want to subsequently bill the users according to the resources utilized, you may group two CPU-intensive processes executed by different users in different classes (to be able to monitor the classes for resource utilization).

Notes: The programs prog1, prog2, and prog3, shown in Example 4-1 on page 131, can be grouped into different classes. In our tests, we have placed prog1 and prog2 in different classes to monitor CPU utilization by user1 and user2, respectively, and the data is collected to bill the users based on resources used.

The program prog3 is a memory-intensive process and will be placed in a different class to ensure effective control of resource use by the process.

Advanced classification may be required for cases where multiple instances of the same executable are being used for different applications. For example, the Java runtime executable may be used by different Java programs on the system.

3. Create the WLM classes.

   When the processes have been classified, it is time to create the WLM classes for these processes. We can use the **smitty wlm** fast path to create the WLM classes.

   a. Execute **smitty wlm** on the command line. This displays the main SMIT screen for the Workload Manager.

Example 4-2 Screen output for **smitty wlm**

```
                                Workload Manager

Move cursor to desired item and press Enter.

  Manage time-based configuration sets

  Work on alternate configurations
  Work on a set of Subclasses
  Show current focus (Configuration, Class Set)

  List all classes
  Add a class
```

```
  Change / Show Characteristics of a class
  Remove a class
  Class assignment rules

  Start/Stop/Update WLM
  Assign/Unassign processes to a class/subclass
```

b. Select `Add a Class`. A new screen opens showing fields to specify the attributes of the class.

c. Specify the attributes of the class. The *Inheritance* and the *Localshm* characteristics must be set to `Yes`. The Tab key may be used to change the values from the default No to Yes in the screen (Example 4-3).

Inheritance means that any child processes of a monitored process will belong to the same WLM class. This is useful for applications that start a lot of processes. *Localshm* means that any shared memory created by a process in a class also belongs to that class.

Example 4-3 Screen output for specifying WLM class characteristics

```
                        General characteristics of a class

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Class name                                         [app1]
  Description                                        [CPU Intensive]
  Tier                                               [0]                    +#
  Resource Set                                                              +
  Inheritance                                        [Yes]                  +
  User authorized to assign its processes to this cl []                     +
  ass
  Group authorized to assign its processes to this c []                     +
  lass
  User authorized to administrate this class         []                     +
  (Superclass only)
  Group authorized to administrate this class        []                     +
  (Superclass only)
  Localshm                                           [Yes]                  +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

We have created a WLM class named `app1` for the prog1 process in this example. We have also created the WLM classes `app2` and `app3` for the prog2 and prog3 programs, respectively, using the steps described in this section.

4. Assign the process to a class based on assignment rules.

   After the creation of WLM classes, the processes have to be assigned to these classes based on some assignment rules.

   a. Select `Class assignment rules` from the initial SMIT screen for the Workload Manager. This displays a SMIT screen with operations for the WLM class rules (Example 4-4).

Example 4-4 SMIT menu screen for class assignment rules

```
                              Class assignment rules

Move cursor to desired item and press Enter.

  List all Rules
  Create a new Rule
  Change / Show Characteristics of a Rule
  Delete a Rule
  Attribute value groupings


F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

   b. Select `Create a new Rule` from the SMIT screen. This displays a screen to specify the attributes for creating a rule for a WLM class (Example 4-5 and Example 4-6).

Example 4-5 Creating a new rule for a WLM class

```
                                  Create a new Rule

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Order of the rule                                  [1]                    #
* Class name                                          app1                  +
* User                                               [user1]                +
* Group                                              [-]                    +
  Application                                        [-]
  Type                                               [-]                    +
  Tag                                                [-]
```

   c. Add a new rule for app3 and prog3 (Example 4-6).

Example 4-6 Creating a new rule for a WLM class

```
                                Create a new Rule

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  [Entry Fields]
* Order of the rule                              [1]                    #
* Class name                                      app3                  +
* User                                           [-]                    +
* Group                                          [-]                    +
  Application                                    [/work/app3/prog3]
  Type                                           [-]                    +
  Tag                                            [-]
```

Note: The application being classified should be a binary. In case the application is a script (some interpreter is used), the binary being invoked in the script should be entered.

We have specified all of the processes of user1 to be assigned to WLM class app1 as a rule. Similarly, the complete path to the prog3 binary has been specified as a rule for class app3.

5. Verify WLM classes and assignment rules.

   After creating the WLM classes and assigning the processes to these classes based on assignment rules, it is worthwhile to list the classes and rules for verification.

   a. Select `List All Classes` from the initial SMIT screen for the Workload Manager. This displays the defined WLM classes (Example 4-7 and Example 4-8 on page 136).

Example 4-7 SMIT screen to list all classes

```
                                Workload Manager

Move cursor to desired item and press Enter.

  Manage time-based configuration sets

  Work on alternate configurations
  Work on a set of Subclasses
  Show current focus (Configuration, Class Set)

  List all classes
  Add a class
  Change / Show Characteristics of a class
```

```
  Remove a class
  Class assignment rules

  Start/Stop/Update WLM
  Assign/Unassign processes to a class/subclass
```

The default superclasses System, Default and Shared are listed along with the sample classes we have created (app1, app2, and app3).

Example 4-8 Screen output listing the WLM classes

```
                                COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

System
Default
Shared
app1
app2
app3
```

b. Select `List all Rules` from the initial SMIT screen for class assignment rules. This displays the assignment rules that are defined for WLM classes (Example 4-9 and Example 4-10).

Example 4-9 SMIT screen for class assignment rules

```
                                Class assignment rules

Move cursor to desired item and press Enter.

  List all Rules
  Create a new Rule
  Change / Show Characteristics of a Rule
  Delete a Rule
  Attribute value groupings
```

Example 4-10 Screen output listing class assignment rules

```
                                     COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

 #  Class     User      Group     Application             Type      Tag
```

```
001 app3      -         -         /work/app3/prog3        -         -
002 app2      user2     -         -                       -         -
003 app1      user1     -         -                       -         -
004 System    root      -         -                       -         -
005 Default   -         -         -                       -         -
```

6. Start WLM in passive mode.

   WLM can be run in either passive or active mode.

   Passive WLM places all processes in the defined classes and lets you monitor the classes without controlling anything.

   Active WLM proactively controls the classes based on the share, tier, rset, and limit attributes.

   a. Select `Start/Stop/Update WLM` from the initial Workload Manager screen. This displays the screen to start, stop, or update WLM (Example 4-11).

Example 4-11 Screen output for starting/stopping/updating WLM

```
                                  Start/Stop/Update WLM

Move cursor to desired item and press Enter.

  Start Workload Manager
  Update Workload Manager
  Stop Workload Manager
  Show WLM status
```

   b. Select `Start Workload Manager`. This displays a screen to select attributes for starting Workload Manager.

   c. Specify `Management mode` as `Passive` and select `No` for `Enforce Resource Set bindings` (see Example 4-12).

Example 4-12 SMIT screen output for starting WLM

```
                                Start Workload Manager

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Configuration, or for a set: set name/currently     current
      applicable configuration
  Management mode                                     Passive
  Enforce Resource Set bindings                       No
  Disable class total limits on resource usage        Yes
  Disable process total limits on resource usage      Yes
  Start now, at next boot, or both ?                  Now
```

d. Select `Show WLM status` from the Start/Stop/Update WLM screen. This displays information about WLM status (Example 4-13 on page 138).

Example 4-13 SMIT screen output for WLM class listing

```
                                COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

WLM is running in passive mode, Rset bindings not active.
Checking classes and rules for 'current' configuration...
System
Default
Shared
app1
app2
app3
```

4.3.4 Introduction to WLM commands and WebSM

This section introduces the basic WLM commands and how to use WebSM to set up the Workload Manager.

WLM commands

WLM configuration can also be done using simple command line options. Table 4-1 shows an overview of the WLM commands and their usage.

Table 4-1 WLM basic commands

Command	Description	Usage
mkclass	Creates a WLM class	mkclass <class name> mkclass -a inheritance=yes -a localshm =yes <class name>
wlmassign	Assigns a process to a WLM class	wlmassign <class name> <process id>
lsclass	Returns the list of superclasses	lsclass
wlmcheck	Checks WLM settings	wlmcheck
rmclass	Removes a WLM class	rmclass <class name>

Command	Description	Usage
wlmcntrl -p	Starts WLM in passive mode	wlmcntrl -p
wlmcntrl -a	Starts WLM in active mode	wlmcntrl -a
wlmcntrl -o	Stops WLM	wlmcntrl -o

The class assignment rules for a class can be added by editing the /etc/wlm/current/rules file. All of the user-defined classes must be added above the `System` and `Default` class lines, because the rules file is examined from top to bottom to decide the class of a process.

The /etc/wlm/current/classes file contains the definitions of WLM superclasses and subclasses for a given configuration.

Example 4-14 shows a sample /etc/wlm/current/rules file.

Example 4-14 Sample WLM rules file

```
[p630n02][/]> cat /etc/wlm/current/rules
* class resvd user group application type tag
app3    -       -       -       /work/app3/prog3        -       -
app2    -       user2   -       -       -       -
app1    -       user1   -       -       -       -
System   -    root   -    -           -   -
Default  -    -      -    -           -   -
[p630n02][/]>
```

Example 4-15 shows a sample /etc/wlm/current/classes file.

Example 4-15 Sample WLM classes file

```
[p630n02][/]> cat /etc/wlm/current/classes
System:

Default:

Shared:

app1:
        description = "CPU intensive app"
        inheritance = "yes"
        localshm = "yes"

app2:
        description = "CPU intensive app"
```

```
        inheritance = "yes"
        localshm = "yes"

app3:
        description = "Memory intensive app"
        inheritance = "yes"
        localshm = "yes"

[p630n02][/]>
```

WebSM

Web-based System Manager (WebSM) is a graphical system administration tool on AIX. It is initiated using the AIX **wsm** command. WLM can be configured through the Workload Manager section in the WebSM screen.

WebSM provides a convenient user interface to create and administer WLM classes. Configurations can be managed in WSM from the Configurations/Classes window (Figure 4-2).

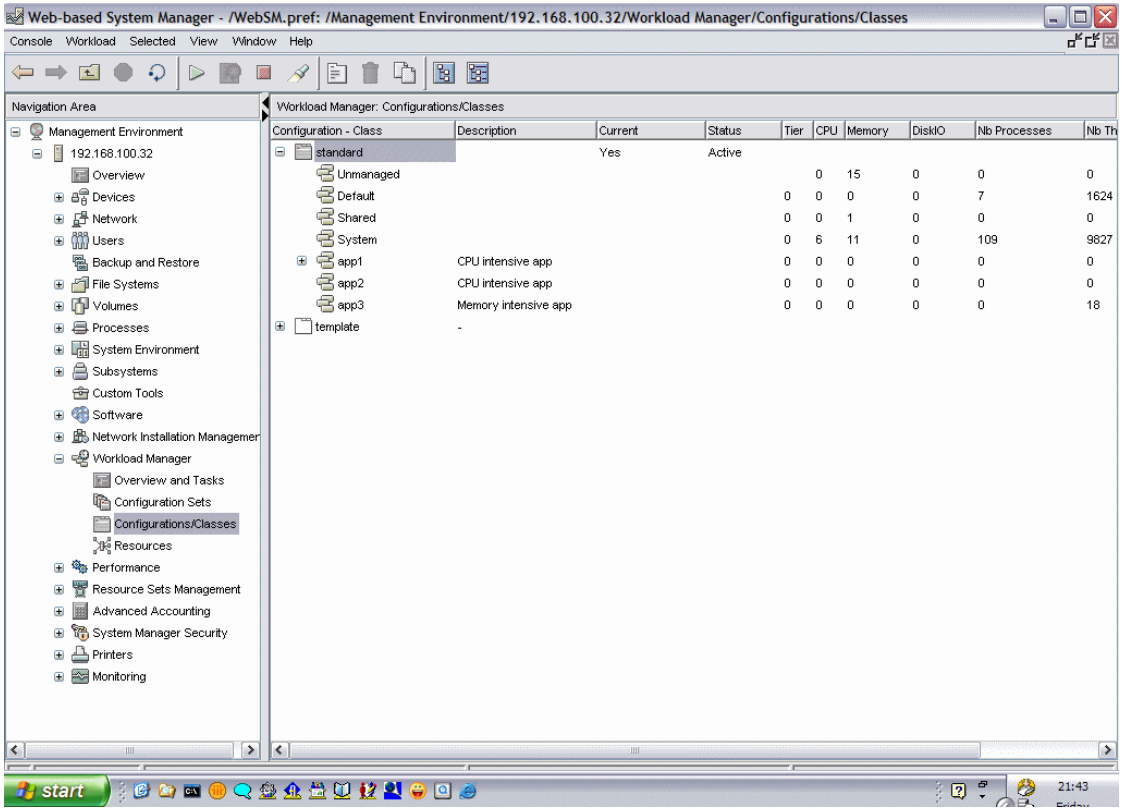

Figure 4-2 Initial WebSM screen

4.4 WLM performance monitoring tools

Various tools are available on AIX to monitor WLM class resource usage. These tools give an idea of how the processes are utilizing the resources (accounting) on the system and how they can be used by system administrators for resource monitoring and control. Some of these tools are available with the AIX operating system, and the others have to be installed separately.

This section provides a brief introduction to the following most commonly used tools for monitoring WLM classes:

- wlmstat (bos.rte.control package)
- topas (bos.perf.tools package)
- svmon (bos.perf.tools package)
- nmon (free, but unsupported by IBM)
- Performance Toolbox (PTX® - perfmgr.* packages)

You can read more about these in the redbook *AIX 5L Workload Manager (WLM)*, SG24-5977.

wlmstat

The **wlmstat** command reports WLM per-class resource utilization. If a count is specified, wlmstat loops *Count* times and sleeps *Interval* seconds after each block is displayed.

```
wlmstat -l [Class] -t [Tier] [Interval][Count]
```

wlmstat displays information about CPU, memory, and disk I/O utilization for all of the predefined and user-defined classes (Example 4-16).

Example 4-16 Sample output of wlmstat command

```
p630n02][/etc/wlm/current]> wlmstat
         CLASS CPU MEM DKIO
     Unmanaged   0  14   0
       Default   0   0   0
        Shared   0   1   0
        System   0   7   0
          app1  44   1   0
          app2  22   0   0
          app3   8  55   0
         TOTAL  74  64   0
```

The **wlmstat** command can be used to display individual detail information using specific flags: CPU (-Svc), memory (-Svm), or disk I/O (-Svi). Example 4-17 on page 142 shows sample WLM CPU information.

Example 4-17 Sample wlmstat output displaying detailed CPU usage statistics

```
[p630n02][/etc/wlm/current]> wlmstat -Svc
         CLASS tr i #pr CPU sha min smx hmx des  rap urap pri
     Unmanaged  0 0   0   0  -1   0 100 100 100  100    0  10
       Default  0 0   0   0  -1   0 100 100 100  100    0   0
        Shared  0 0   0   0  -1   0 100 100 100  100    0   0
        System  0 0  80   0  -1   0 100 100 100  100    0   0
          app1  0 1   2  43  -1   0 100 100 100  100    0   0
          app2  0 1   2  19  -1   0 100 100 100  100    0   0
          app3  0 1   1  06  -1   0 100 100 100  100    0   0
```

topas

The topas command displays performance statistics updated on the screen at regular intervals. When used with the -W flag, the command displays information about percentage of CPU, memory, and disk I/O utilization for the WLM classes (Example 4-18).

Example 4-18 Sample topas output displaying WLM class data

```
[p630n02][/work]> topas -W
Topas Monitor for host:    p630n02     Interval:   2    Mon Oct 25 15:13:04
2004

WLM-Class (Passive)             CPU%      Mem%     Disk-I/O%
app1                            47         1           0
app2                            23         0           0
app3                            12        55           0
System                           0         7           0
Shared                           0         1           0
Default                          0         0           0
Unmanaged                        0        14           0
Unclassified                     0         0           0
```

svmon

The **svmon** command captures and analyzes a snapshot of virtual memory. **svmon** provides the ability to report activity related to workload management with the following two types of report:

Class report Prints memory usage information pertinent to a class. Usage is with the -W flag.

Tier report Prints memory usage information with respect to a class tier. Usage is with the -T flag.

Example 4-19 Sample svmon output displaying WLM class data

```
[p630n02][/work]> svmon -W app3
WLM is running in passive mode

===============================================================================
Superclass                          Inuse      Pin     Pgsp  Virtual
app3                               512670        4        0   512077

    Vsid      Esid Type Description              LPage  Inuse   Pin Pgsp Virtual
   20564         - work                             -  65536     0    0 65536
   90572         - work                             -  65536     0    0 65536
   88551         - work                             -  65536     0    0 65536
   c8579         - work                             -  65536     0    0 65536
   485e9         - work                             -  65536     0    0 65536
   f855f         - work                             -  65536     0    0 65536
   c85b9         - work                             -  65483     0    0 65485
   18543         - work                             -  53319     0    0 53319
   18463         - clnt /dev/hd2:49716              -    304     0    -     -
   48589         - clnt /dev/hd9var:467             -     57     0    -     -
   98433         - clnt /dev/hd3:4103               -     57     0    -     -
     540         - clnt /dev/hd2:1098               -     42     0    -     -
```

```
80550         - work                          -     37     0     0    37
48449         - clnt /dev/hd2:289             -     32     0     -     -
20544         - clnt /dev/hd2:1025            -     19     0     -     -
104e2         - clnt /dev/hd2:45304           -     17     0     -     -
  560         - work                          -     15     0     0    15
```

nmon

nmon is a freely downloadable tool developed within (but not officially supported by) IBM, available at:

http://www-106.ibm.com/developerworks/eserver/articles/analyze_aix/agree_down.html

It can be used to monitor AIX performance statistics. The **nmon** command shows the percentage of CPU, memory, and disk I/O by each class; the desired target, share values, and number of processes; and the tier, inheritance, and local shared memory flags (Example 4-20).

Example 4-20 Initial nmon screen

```
[p630n02][/work]> nmon32
nmon32 v9a               Hostname=p630n02  Refresh=2.0secs  15:25.10

   ------------------------------       For help type H or ...
   #    #  #    #   ####   #    #        nmon -?  - hint
   ##   #  ##  ##  #    #  ##   #        nmon -h  - full
   # #  #  # ## #  #    #  # #  #
   #  # #  #    #  #    #  #  # #       To start the same way every time
   #   ##  #    #  #    #  #   ##        set the NMON ksh variable
   #    #  #    #   ####   #    #
   ------------------------------

   Use these keys to toggle statistics on/off:
      c = CPU             l = Long-term CPU         - = Faster screen updates
      m = Memory          k = Kernel Stats          + = Slower screen updates
      d = Disks           a = Adapters (disk only)  v = Verbose hints
      r = RS6000/pSeries  n = Network               U = command arguments
      e = ESS Disks       g = Disk Groups (see -g)  . = only busy disks/procs
      j = JFS             t = Top-proc's (1,2,3,4,5 - different data)

      h =more options
```

nmon can also display the WLM class data (Example 4-21).

Example 4-21 Sample nmon output displaying WLM class data

```
Work Load Manager CPU MEM BIO  CPU MEM IO  CPU   MEM   BIO     Tier Inheritance
Class Name       |---Used----||--Desired-||----Shares-----|Proc's T I Localshm
Unclassified       0%  0%  0% 100 100 100    -1    -1    -1     0 0 0 0
```

```
Unmanaged          0% 14%  0% 100  99 100    -1    -1    -1     0 0 0 0
Default            0%  0%  0% 100 100 100    -1    -1    -1     0 0 0 0
Shared             0%  1%  0% 100  98 100    -1    -1    -1     0 0 0 0
System             0%  7%  0% 100  99 100    -1    -1    -1    83 0 0 0
app1              44%  0%  0% 100 100 100    -1    -1    -1     3 0 1 1
app2              22%  0%  0% 100 100 100    -1    -1    -1     2 0 1 1
app3              10% 55%  0% 100  98 100    -1    -1    -1     2 0 1 1
```

Performance Toolbox (PTX)

The **wlmmon** and **wlmperf** commands provide graphical views of Workload Manager resource activities by class.

The **wlmmon** and **wlmperf** commands generate resource usage reports of system WLM activity. The **wlmperf** command, which is a part of PTX, can generate reports from trend recordings made by PTX daemons for periods covering minutes, hours, days, weeks, or months.

The **wlmmon** command generates three types of visual reports:

- Snapshot display
- Detailed display
- Tabulation display

While the **wlmstat** command provides a per-second view of WLM activity, it is not suited for long-term analysis. To supplement the **wlmstat** command, the **wlmmon** and **wlmperf** commands provide reports of WLM activity over much longer time periods, with minimal system impact.

4.5 WLM accounting

The AIX accounting subsystem provides utilities for system administrators to monitor resource utilization. Accounting enables users to collect and report on individual and group use of various system resources.

Features in accounting enable accurate determination of the utilization of computing resources and evaluate costs expended for the utilized resources.

To assist with billing, the accounting system provides the resource-usage totals defined by the members of the adm group, and if the **chargefee** command is included, the system factors in the billing fee.

Information from accounting data helps in assessing the adequacy of current resources on the system, setting resource limits, and forecasting future needs.

4.5.1 Process accounting using WLM classes

The accounting subsystem monitors resource utilization with the processes running on the system, and it works in conjunction with the kernel to gather pertinent information.

Process accounting for WLM classes encompasses the following steps:

- Processes are assigned to WLM classes to enable monitoring of resource utilization at class level.
- The AIX accounting utility is started on the system. (Read a detailed description in Chapter 3, “Accounting on AIX” on page 41.)
- The system collects data about resource usage for each process as it runs.
- Whenever the process exits, the kernel appends a record with the process’ accounting information on the process accounting file, /var/adm/pacct.
- AIX accounting commands can be used to process and display accounting information from the /var/adm/pacct file.

Accounting information collected by the system for each process includes:

- User and group numbers that the process runs
- First eight characters of the name of the command
- A 64-bit numeric key representing the Workload Manager class that the process belongs to
- Memory usage
- Number of characters transferred
- Number of disk blocks read or written on behalf of the process

The accounting subsystem uses a 64-bit key instead of the 34-bit full character class name in order to save space. Upon running the accounting command to extract the per-process data, the key is translated back into a class name.

This translation uses the class names that are currently in the WLM configuration files. Therefore, if a class has been deleted between the time that the accounting record was written and the time that the accounting report is run, the class name corresponding to the key will not be found, and the class will appear as Unknown.

To keep accurate records of the resource usage of classes deleted during an accounting phase, system administrators can keep the class name in the classes file and remove the class from the rules file (instead of just deleting the class) so that no process will be assigned to it.

4.5.2 Displaying WLM class accounting information

The AIX accounting command **acctcom** enables the display of process resource usage statistics per user, group, or WLM class from the /var/adm/pacct file.

These flags can be used with **acctcom** to display accounting information pertinent to WLM classes:

-w Displays class names to which the processes belong (Example 4-22)

-c *class* Selects processes belonging to the specified class (Example 4-23 on page 147)

Example 4-22 Screen output for acctcom -w command

```
[p630n02][/work]> acctcom -w | grep app1
prog1      user1    app1.Default     pts/2   12:02:48 12:05:22   154.12   144.55    10008.00
prog1      user1    app1.Default     pts/2   12:05:42 12:07:19    97.16   111.80     7705.00
prog1      user1    app1.Default     pts/2   12:09:27 12:10:04    37.20    32.22     4370.00
prog1      user1    app1.Default     pts/2   12:10:14 12:14:42   268.50   233.17    10256.00
prog1      user1    app1.Default     pts/2   12:17:51 12:20:45   174.62   250.88    13296.00
prog1      user1    app1.Default     pts/2   12:21:37 15:39:05 11848.00 11169.00    16064.00
prog1      user1    app1.Default     pts/7   15:39:30 15:41:07    97.16   238.81    14192.00

[p630n02][/work]> acctcom -w | grep app2
prog2      user2    app2.Default     pts/3   12:03:02 12:05:22   140.50   117.78     9400.00
prog2      user2    app2.Default     pts/3   12:05:57 12:07:19    82.77    62.42     7745.00
prog2      user2    app2.Default     pts/3   12:10:28 12:13:46   198.88   143.59     9408.00
prog2      user2    app2.Default     pts/3   12:14:53 12:14:53     0.00     0.00        0.00
prog2      user2    app2.Default     pts/3   12:15:04 12:18:16   192.88   214.44    11464.00
prog2      user2    app2.Default     pts/3   12:22:03 15:41:07 11944.00  9028.00     8664.00
```

Example 4-23 Screen output for acctcom -c command

```
[p630n02][/work]> acctcom -c app1

COMMAND                    START    END          REAL      CPU        MEAN
NAME       USER     TTYNAME  TIME     TIME       (SECS)    (SECS)     SIZE(K)
prog1      user1    pts/2    12:02:48 12:05:22   154.12   144.55    10008.00
prog1      user1    pts/2    12:05:42 12:07:19    97.16   111.80     7705.00
prog1      user1    pts/2    12:09:27 12:10:04    37.20    32.22     4370.00
prog1      user1    pts/2    12:10:14 12:14:42   268.50   233.17    10256.00
prog1      user1    pts/2    12:17:51 12:20:45   174.62   250.88    13296.00
```

The **acctcom -w** command displays class names to which processes belong. In Example 4-22 on page 147, processes prog1 and prog2 are displayed as belonging to classes app1 and app2 respectively.

The **acctcom -c** command displays accounting information for processes belonging to a specific class. In Example 4-23, accounting information pertaining to class app1 is displayed.

These commands detail the information recorded for the different time periods that the accounting data was collected for the classes.

Data is displayed about the amount of resource utilization for the intervals of the time the data was recorded.

In Example 4-22 on page 147 and Example 4-23, the displayed information includes user name, tty name, start time, end time, real seconds, CPU seconds, and mean memory size in kilobytes.

The **/usr/sbin/acct/chargefee *User Number*** command can be used by someone with administrative authority to charge the individual specified by the *User* parameter for the number of work units specified by the *Number* parameter.

The **chargfee** command writes a record to /var/adm/fee file.

4.5.3 WLM application programming interface (API)

A WLM API is a set of routines provided by the /usr/lib/libwlm.a library that can be used by applications to perform all the tasks that can be achieved using WLM commands.

The WLM API provides the accounting initialization routines per class shown in Table 4-2.

Table 4-2 WLM API description

WLM API routine	Purpose
wlm_initkey	Allocates and initializes the classes to keys transition table
wlm_class2key	Class name to key translation
wlm_key2class	Retrieves a class name from a key
wlm_endkey	Frees the classes to keys translation table

4.6 Resource control using WLM

This section provides a brief overview of the ways workloads can be controlled and managed using WLM. Refer to the WLM redbook for a detailed description.

WLM manages resource allocation and control using the following features:

Shares Shares are a means to assigning relative importance to each class with respect to the percentage of system resources that can be assigned to a class. Each class is assigned share values for CPU, memory, and disk I/O, which determines the percentage of these resources that will be available to each class.

Tiers Theclass tier value is the position of the class within the hierarchy of resource limitation desirability for all classes. Tiers 0 through 9 can be defined to prioritize classes, with 0 being the most important and 9 least important. If one tier's classes use all of the resources that are available on the system, the higher tiers cannot get resources for a period of time. The concept of tiers is useful in preventing low priority processes to use resources when the system is loaded.

Limits WLM can be used to set limits to the amount of a resource being used by a class to prevent the class from hogging system resources.

5

Advanced Accounting

This chapter provides information about Advanced Accounting, a new feature for AIX5L V5.3. This new feature is provided in addition to traditional accounting, and it is based on mainframe technology, such as interval accounting and transaction accounting.

Traditional accounting and Advanced Accounting are separate functions, and you can run both at the same time. Advanced Accounting also has an enhancement feature to support LPAR (Logical Partition), WLM (Work Load Management), and Micro-Partitioning.

Advanced Accounting provides usage-based information for a wide variety of system resources so that you can develop comprehensive charge-back strategies. You can collect accounting data on resources such as disks, network interfaces, virtual devices, file systems, processors, and memory. Interval accounting gives you the ability to view this data over system administrator-defined time intervals in order to develop chronological views. This has several potential applications, including capacity planning.

5.1 Managing Advanced Accounting

Advanced Accounting can be managed through the command line, SMIT (System Management Interface Tool), or WebSM (Web-based System Management tool). The menus that are provided with WebSM differ from the ones in SMIT. In this chapter, we use SMIT.

Figure 5-1 presents a diagram of the components, and how to set up advanced accounting.

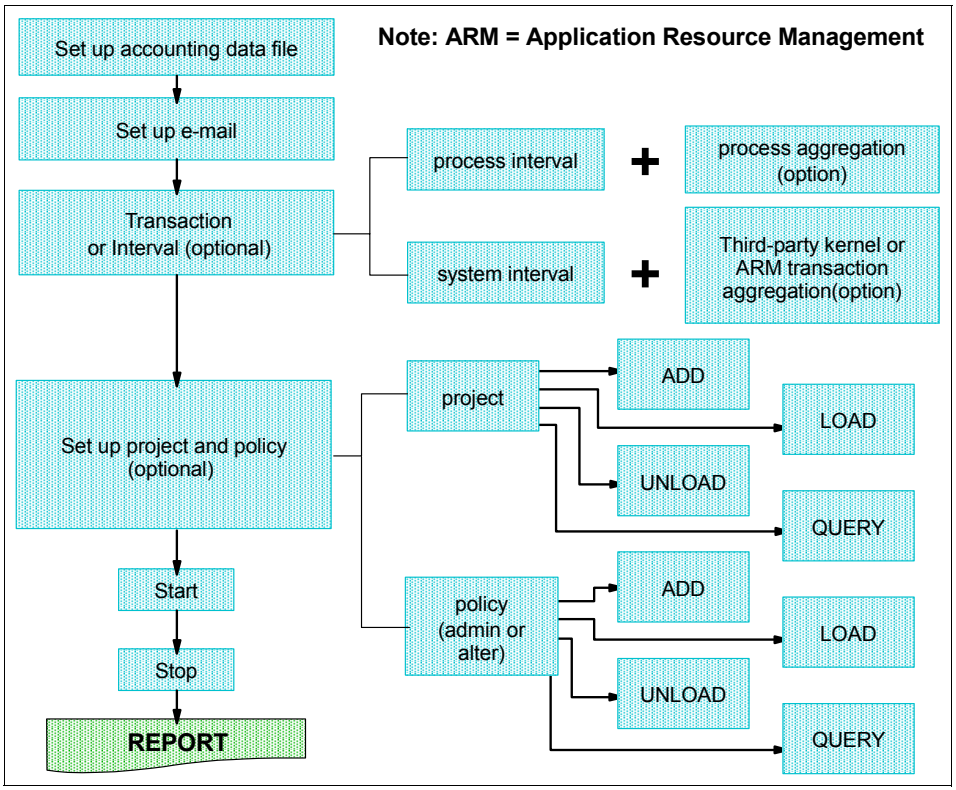

Figure 5-1 Advanced accounting setup overview

5.1.1 Controlling the advanced accounting

This section provides information about how to control (start, stop, query) the Advanced Accounting subsystem. The **`acctctl`** command is used for this purpose. It is also used for setting up account data files, e-mail notification, and interval accounting. We present several examples of using **`acctctl`**.

Checking the status

To query the status of the Advance Accounting subsystem, enter **acctctl** with no parameters (Example 5-1).

Example 5-1 Query advanced accounting

```
[p630n02][/]> acctctl
Advanced Accounting is not running.
Email notification is off.
The current email address to be used is not set.
Process Interval Accounting is off.
System Interval Accounting is off.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 0 defined, 0 available.
```

Starting Advanced Accounting

To start Advanced Accounting, use the **acctctl on** command:

```
[p630n02][/]> acctctl on
```

Verify it by using **acctctl** (Example 5-2).

Example 5-2 Advanced Accounting running

```
[p630n02][/]> acctctl
Advanced Accounting is running.
Email notification is off.
The current email address to be used is not set.
Process Interval Accounting is off.
System Interval Accounting is off.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 0 defined, 0 available.
```

Stopping Advance Accounting

To stop the Advance Accounting subsystem, use **acctctl off** (Example 5-3).

Example 5-3 Stopping Advanced Accounting

```
[p630n02][/]> acctctl off
[p630n02][/]> acctctl
Advanced Accounting is not running.
Email notification is off.
The current email address to be used is not set.
Process Interval Accounting is off.
```

```
System Interval Accounting is off.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 0 defined, 0 available.
```

5.1.2 Using SMIT to control Advanced Accounting

To start Advanced Accounting via SMIT use the following fastpath:

```
smitty aacct
```

You can also use **`smitty`** → **`Advanced Accounting`** to display the Advanced Accounting main menu (Example 5-4).

Example 5-4 SMIT menu for Advanced Accounting

```
                              Advanced Accounting

Move cursor to desired item and press Enter.

  Manage Accounting Data Files
  Manage Project Definitions and Assignments
  Manage Transactions
  Manage Advanced Accounting Subsystem


F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Automatic start of Advanced Accounting

By default, Advanced Accounting is not automatically started at system boot. To enable automatic start of Advanced Accounting, enter:

```
smitty aacct → Manage Advanced Accounting Subsystem → Start Advanced
Accounting
```

Specify one of the three modes to start (in this case, `Both` or `At next boot`):

Now Starts Advanced Accounting by /etc/rc.startacct script.

At next boot Adds an inittab entry to make Advanced Accounting start automatically on next reboot.

Both Adds an entry in /etc/inittab file to start the Advanced Accounting subsystem on next system boot. Also, you can use the /etc/rc.startacct script to start Advanced Accounting.

Additional Advanced Accounting commands

projctl Used for setting up project name, policy, manual load project, load and unload project or policy for Advanced Accounting.

readaacct A sample command used to get reports from the accounting data file. This command and its source code (in C language) reside in the /usr/samples/aacct directory.

5.2 Accounting data file and e-mail notification

To collect data and to notify about events, you have to set up Advanced Accounting for data files and e-mail notifications.

5.2.1 Accounting data file

The accounting data file collects usage records (interval, policy), depending on the setup of the Advanced Accounting. The default directory for the accounting data file is /var/acct, but you can specify another location. We can have more than one accounting data file, but only one data file will be active at a time. To define an accounting data file, enter the following on a command line:

```
acctctl fadd filename filesize
```

You can also use **smitty create_aacct_file**. The file size is in megabytes.

Example 5-5 shows how to add two accounting data files. If you have started the accounting and the data file was not available, you have to reset old data file or create a new data file.

Example 5-5 Set up accounting data file

```
[p630n02][/]> acctctl fadd /var/aacct/aacct1.dat 5
[p630n02][/]> acctctl fadd /var/aacct/aacct2.dat 5
[p630n02][/]> acctctl
Advanced Accounting is not running.
Email notification is off.
The current email address to be used is not set.
Process Interval Accounting is off.
System Interval Accounting is off.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 2 defined, 2 available.
[p630n02][/]> acctctl on
[p630n02][/]> acctctl
Advanced Accounting is running.
Email notification is on.
```

```
The current email address to be used not set.
Process Interval Accounting is off.
System Interval Accounting is off.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 2 defined, 1 available.
```

We recommend using at least two data files so that Advanced Accounting always remains active. When the active data file is full, the other file takes over.

After data collection we use post-processing tools (such as **readaacct)**, which extract the data from the data file that has been filled up. After extracting data, we can reset the data file, and the file can be used again. Both defined files are recyclable.

To reset the accounting data file, enter:

```
acctclt freset file
```

To switch accounting data file, enter:

```
acctclt fswitch [file]
```

To query status e.g.% use of accounting file, enter:

```
acctclt fquery [file]
```

Figure 5-2 shows possible status for Advanced Accounting data collection files.

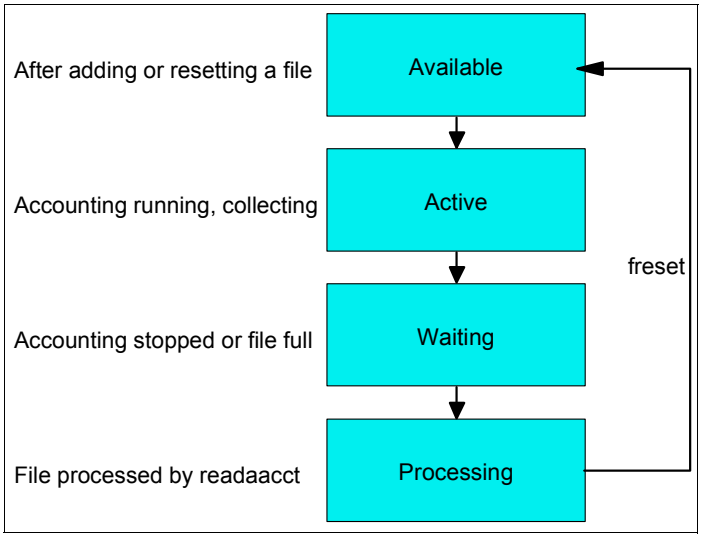

Figure 5-2 Status of accounting files

5.2.2 E-mail notification

We set e-mail notification to send messages to a mail ID about the status of the accounting data files, such as File error or File nearly full.

To set up e-mail notification, use:

```
acctctl email {on|off|addr}
```

Example 5-6 Set up e-mail notification

```
[p630n02][/]> acctctl email on
[p630n02][/]> acctctl email root
[p630n02][/]> acctctl
Advanced Accounting is not running.
Email notification is on.
The current email address to be used is root.
Process Interval Accounting is off.
System Interval Accounting is off.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 2 defined, 2 available.
```

There are five types of e-mail notification messages, as listed in Table 5-1.

Table 5-1 E-mail notification messages

Subject	Detail
AACCT: File nearly full	1400-330: The accounting file is 90% full.
AACCT: File ready	1400-331: The accounting file is ready for processing.
AACCT: Subsystem out of files	1400-332: The Advanced Accounting subsystem has run out of files for use.
AACCT: Subsystem out of kernel buffers	1400-333:The Advanced Accounting subsystem has run out of kernel buffers.
AACCT: File I/O error	1400-334: The accounting file encountered an I/O error while writing.

5.3 Interval accounting

Interval accounting provides a way to collect accounting data at specified intervals. You can configure it to produce intermediate process records for active processes, and to periodically capture accounting data for system resources

such as processors, memory, disks, network interfaces, and file systems. This information can be used to generate a bill that reflects the total use of a partition.

Interval accounting uses different records from traditional UNIX accounting. Advanced Accounting provides more information compared to traditional accounting (network adapter, remote file I/O utilization, and so on).

There are two types of interval accounting: system interval and process interval.

5.3.1 System interval

System interval accounting collects resource usage for both logical and physical resources such as processors, disks, file systems, and memory usage. Logical resources can be Logical Partitions (LPARs) or Workload Manager (WLM). This data may also be used for capacity planning and performance analysis tools.

Under normal circumstances, interval accounting of one hour is sufficient for collecting data. Note that system interval does not collect any process accounting data.

- To enable system interval accounting every 60 minutes, enter:

  ```
  acctctl isystem 60
  ```

- To disable System Interval accounting, enter:

  ```
  acctctl isystem off
  ```

- You can also use SMIT to enable, disable, and set up interval accounting:

  ```
  smitty system_interval
  ```

5.3.2 Process interval

Process interval is used for long-running jobs, such as data modeling applications that can run for months. Normally, process interval is set to capture data once a day (1,440 minutes). But if process records are being aggregated automatically by the system, the process interval should be set to one hour.

- To enable the process interval every 1440 minutes, enter:

  ```
  acctctl iprocess 1440
  ```

- To disable process interval, enter:

  ```
  acctctl iprocess off
  ```

- You can also use SMIT to enable, disable, and set up interval accounting:

  ```
  smitty process_interval
  ```

5.4 Transaction accounting

Transaction accounting uses ARM (Application Resource Measurement) interfaces to monitor the workload. To use this feature, you need to understand the programming model associated with ARM interfaces and the mechanism for passing information to Advanced Accounting.

The ARM standard describes a common method for integrating enterprise applications as manageable entities. This standard enables users to extend their enterprise management tools directly to applications, creating a comprehensive end-to-end management capability that includes measuring application availability, application performance, application usage, and end-to-end transaction response time. Learn more about ARM 4.0 at:

http://www.opengroup.org/tech/management/arm

For more information, refer to *Understanding the Advanced Accounting subsystem*, SC23-4882.

We also need to modify the application or the middleware to support transaction accounting. An example is a database application or a WebSphere application. In this case, the transaction IDs 13 to 15 will be recorded.

The following list shows the ARM interfaces:

- arm_register_application
- arm_start_application
- arm_register_transaction
- arm_start_transaction
- arm_block_transaction
- arm_unblock_transaction
- arm_bind_transaction
- arm_unbind_transaction
- arm_stop_transaction
- arm_stop_application
- arm_destroy_application

Figure 5-3 on page 160 shows ARM API calls in a three-tier application environment between the Web server, the application server, and the database server.

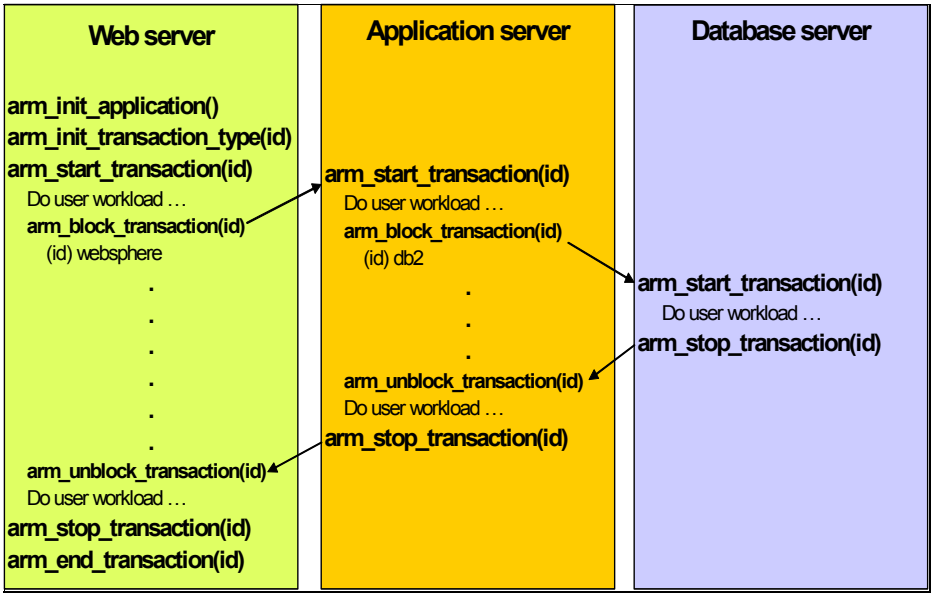

Figure 5-3 ARM API calls

5.5 Data aggregation

Data aggregation is used to collect data into an accounting record. This method reduces the number of records in an accounting data file. To make data aggregation available, interval accounting must be enabled at specified intervals. The recommended value for process and system intervals is 60 minutes. (See 5.3, “Interval accounting” on page 157.)

There are two types of data aggregation: *system-level* and *project-level*.

5.5.1 System-level data aggregation

You must enable interval accounting (as described in 5.3, “Interval accounting” on page 157) before you enable any system-level aggregation. Example 5-7 shows the error you receive if process interval is not enabled.

Example 5-7 Error message for process aggregation

```
[p630n02][/]> acctctl agproc on
[p630n02][/]> acctctl
Advanced Accounting is not running.
Email notification is on.
The current email address to be used is root.
```

```
Process Interval Accounting is off.
System Interval Accounting is off.
System-wide aggregation of process data will occur, when a process interval is
set.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 2 defined, 2 available.
```

- Process interval must enabled prior to enabling system-wide process aggregation.
- System interval must enabled prior to enabling system-wide aggregation for third-party kernel, or system-wide aggregation for ARM transaction.

System-wide process aggregation

The aggregated process record (type 2; see Appendix C, “Accounting records in Advanced Accounting” on page 229) is recorded in the accounting data file. To enable or disable system-wide process aggregation, use:

```
acctctl agproc {on|off}
```

System-wide aggregation for third-party kernel

The third-party kernel extension common aggregation record (type 12) is also recorded in an accounting data file. To enable or disable system-wide third-party aggregation, use:

```
acctctl agke {on|off}
```

System -wide aggregation for ARM transaction

The ARM aggregation transaction instance record (type 16) is recorded in the accounting data file. To enable or disable system-wide aggregation for ARM transaction, use:

```
acctctl agarm {on|off}
```

5.5.2 Project-level data aggregation

- To aggregate data at project level, use:

  ```
  projctl chattr agg projname {-s|-u} [-d projpath]
  ```

  - The -u flag is for disabling aggregation.
  - The -s flag is used to enable aggregation.

  You can also use the SMIT fastpath **smitty show_chg_proj.**

- To query the aggregation state for all projects, enter:

  ```
  projctl qprojs
  ```

5.6 Projects and policies

Projects represent billable entities such as users, departments, divisions, companies, or tasks. Each project has a project number, project attribute, and project name, which collectively represent a project definition. Project definitions are entered into the project definition database.

Policies automate project assignment. A policy is comprised of classification criteria and classification results. Project assignments take place during subroutines and kernel services, such as **exec**, **initp**, **setuid**, and **setgid**.

We set up the project and the policy that are associated to a specific user, group, and application name, for separating reports from the default project. By default, all users, groups, and applications belong to the System project. We recommend setting up both the project and the policy, so we can specify accounting records for users and groups.

Figure 5-4 shows the project overview and related components and operations.

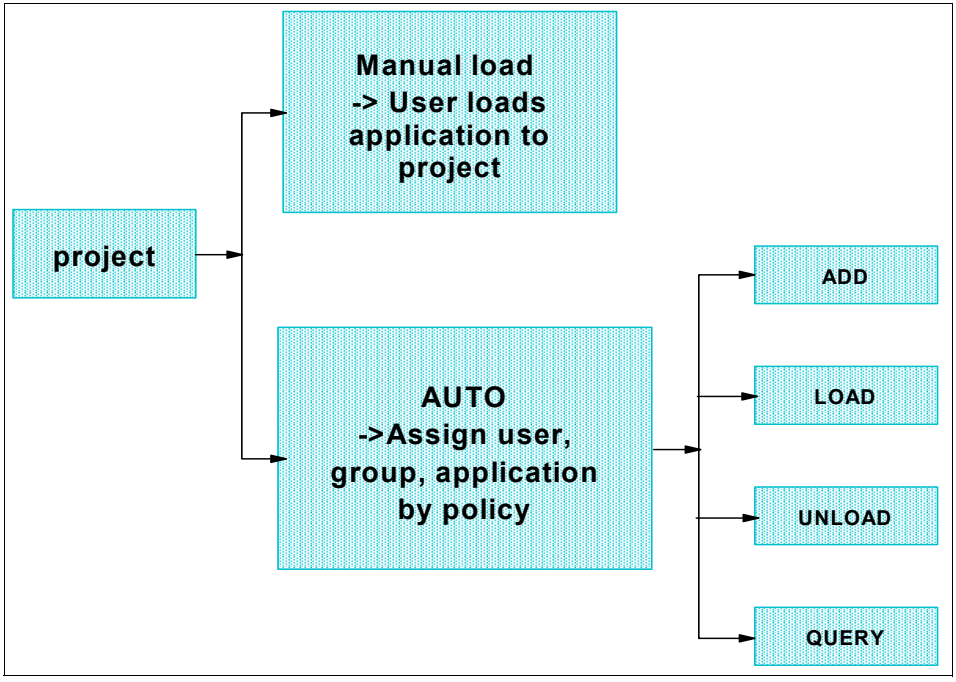

Figure 5-4 Project overview

5.6.1 Projects

Projects represent the billable entries for users, departments, divisions, companies, or tasks. Each project consists of a project name, project number, and project attributes. Projects are written to accounting records. Report and analysis commands convert the project number into the project name through the system project definition database. The default file for the Projects definition is /etc/project/projdef.

Example 5-8 presents a sample /etc/project/projdef file.

Example 5-8 Sample file /etc/project/projdef

```
[p630n02][/]> cat /etc/project/projdef
:: @(#)34 1.3 src/bos/etc/project/projdef, cmdaacct, bos530 12/8/03 08:23:44
........................... Ommited lines ...........................
System:0:n::Default System Project
itsotest:8833:n::ITSOREDBOOKSTEST
```

Adding a project

- To add a project, use the command syntax:

  ```
  projctl add projname projnumber [comment] [-d projpath]
  ```

- To add a project named `itsotest`, project number 8833, enter;

  ```
  projctl add itsotest 8833 ITSOREDBOOKSTEST
  ```

  You could also use the SMIT fastpath **`smitty add_proj.`**

Note: Project-level aggregation must be turned OFF when adding a project via SMIT. After adding the project, you can manually turn on aggregation.

Load, unload, and query projects

- To load a projects definition:

  ```
  smitty → Advanced Accounting → Manage Project Definitions and
  Assignments → Project Definitions → Load/Re-load Project
  Definitions → Turn on auto-loading, when starting accounting [yes]
  ```

  Or on a command line, use:

  ```
  projctl ldprojs -r -a
  ```

- To unload projects definitions:

  ```
  smitty → Advanced Accounting → Manage Project Definitions and
  Assignments → Project Definitions → Unload Active Project Definitions
   → Turn off auto-loading, when starting accounting [yes]
  ```

Or on a command line, use:

```
projctl unldprojs -a
```

Note: Unload all policies under a project before unloading the project, or force unloading all by using: **projctl unldall -a**

- To query project status:

  ```
  smitty → Advanced Accounting → Manage Project Definitions and
  Assignments → Project Definitions → List All Active Project
  Definitions
  ```

  Or on a command line (see Example 5-9), use:

  ```
  projctl qprojs
  ```

Example 5-9 load and unload project

```
[p630n02][/]> projctl ldprojs -r -a
[p630n02][/]> projctl qprojs
Project Name      Project Number          Aggregation
System                         0             DISABLED
itsotest                    8833             DISABLED
[p630n02][/]> projctl unldprojs -a
[p630n02][/]> projctl qprojs
Project definitions are not loaded.
```

5.6.2 Policies

Policies automate project assignment. Policy consist of classification criteria and classification results. An administrator can specify user name, group name, application name, and project list in a policy file. However, it is not necessary to have all four components defined. There are four types of policies (Figure 5-5 on page 165):

- admin policy
- alter admin policy
- user policy
- group policy

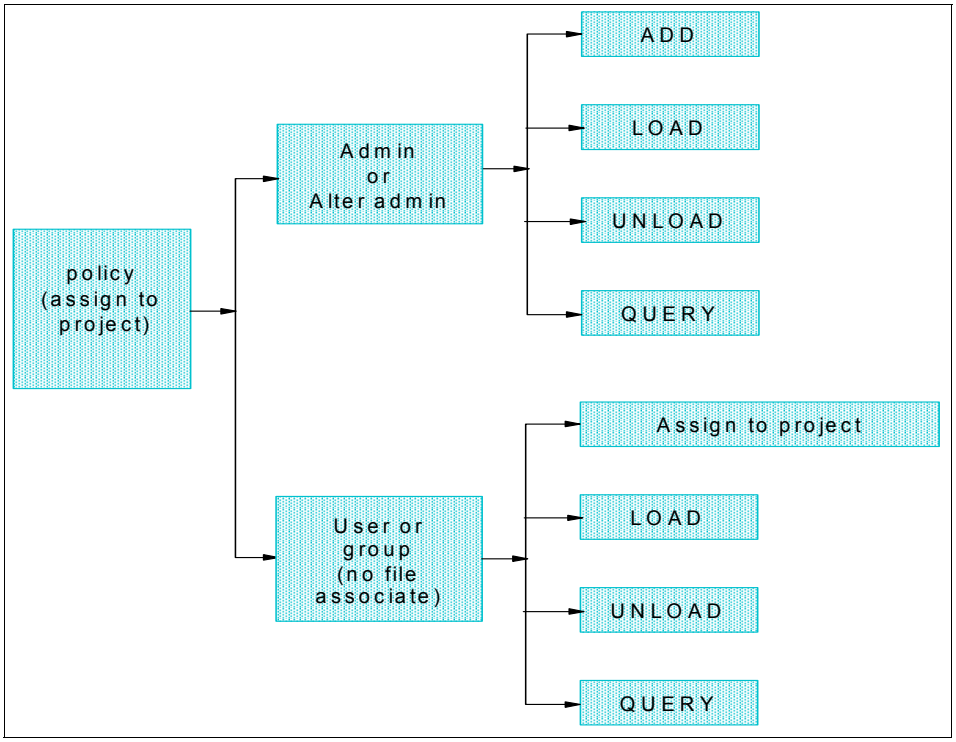

Figure 5-5 Policy overview

Note: When you change anything in the policy, you must reload the policy into the kernel to make it active. Use the **`projctl`** command or SMIT.

Admin policy and alternate admin policy

Admin policy uses user name, group name, application name, and process attributes to classify processes. This application-based policy provides the ability to collect account statistics at application level.

The *default* admin policy file is /etc/project/admin. An alternate admin policy is used at different times than the default; for example, we may have an admin policy running Monday through Friday, and an alternate admin policy on Saturday and Sunday. The default *alternate* admin policy file is /etc/project/alter/template/admin.

The syntax for both default and alternate admin policies is:

```
user name:group name:application name:Projects::comments(option)
```

Table 5-2 shows details about user name, group name, and application name used in the policies.

Table 5-2 User, group, and application rules

Type of rule	Description
User name	At least one valid user name as defined in /etc/passwd. An exclamation point (!) can be used before a user name to exclude it from the class. In a list, user names are separated by comma. If a hyphen (-) is used, Advanced Accounting skips to the next field.
Group name	At least one valid group name as defined in /etc./group. An exclamation point (!) can be used before a group name to exclude it from a class. In a list, group names are separated by comma. If a hyphen(-) is used, Advanced Accounting skips to the next field.
Application name	A list of application path names or command name of kernel process. An exclamation point (!) can be used before an application name to exclude it from a class. In a list, application names are separated by comma. If a hyphen (-) is used, Advanced Accounting skips to the next field.

To add admin policy, use:

```
smitty → Advanced Accounting → Manage Project Definitions and
Assignments → Automatic Project Assignment → Work with Admin Policies →
Work with Admin Policies → Add Rules
```

You could also use the SMIT fastpath **smitty add_admin_rule** (Example 5-10).

Note: There are no comments in the SMIT menu. To add comments, use an editor, such as vi.

Example 5-10 Add admin policy via SMIT

```
                                Add Rules

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Order                                              [1]                    +#
* User-ID/Alias                                      [tester1]              +
* Group-ID/Alias                                     [-]                     +
* Application Name                                   [-]
* Projects                                           [itsotest]             +
```

```
F1=Help           F2=Refresh        F3=Cancel         F4=List
F5=Reset          F6=Command        F7=Edit           F8=Image
F9=Shell          F10=Exit          Enter=Do
```

After adding the new admin policy, the content of the /etc/project/admin file will be updated with the new records. The user tester1 must be defined in the system (in the /etc/passwd file -- see Example 5-11).

Example 5-11 Sample file /etc/project/admin

```
[p630n02][/]> cat /etc/project/admin
:: This is an automatically generated prolog.
.............. Ommited lines ......................
:: Format of the entries -> UseID:GroupID:AppName:Projects::Comments
tester1:-:-:itsotest::
```

Set up the alternate admin policy

The alternate admin policy uses the same user name, group name, and application name rules (see Table 5-2 on page 166). The alternate policy file is located in another directory. To load the alternate admin policy, we have to change *focus* (in fact we have to change from the default admin directory /etc/project to the alternate admin directory /etc/project/alter). We can change focus by as follows:

smitty → Advanced Accounting → Manage Project Definitions and Assignments → Automatic Project Assignment → Work with Admin Policies → Change/Show Current Focus

or use **smitty chang_show_focus**

To load the alternate admin policy, we need to assign another accounting data file or to reset the old data file that has been used by the admin policy. Alternate admin policy is managed via SMIT or the **projctl** command. We can use **cron** to specify load time, reset the old accounting data, and reload back admin policy. Alternate admin policy can be loaded without unloading the admin policy.

Aliases in admin policy and alternate admin policy

Aliases are used to regroup the user name, group name, and application name. The alias file resides in the focus directory (/etc/project) or in the /etc/project/alter/template. To add an alias we can edit the file, or use SMIT.

smitty → Advanced Accounting → Manage Project Definitions and Assignments → Automatic Project Assignment → Work with Admin Policies → Work with Alias

You could also use the SMIT fastpath **smitty work_alias** (see Example 5-12).

Example 5-12 /etc/project/alias file

```
[p630n02][/etc/project]> cat alias
:: @(#)33 1.1  src/bos/etc/project/alias, cmdaacct, bos530  9/24/03  05:12:33
:: IBM_PROLOG_BEGIN_TAG
..............  Ommited lines ......................
:: IBM_PROLOG_END_TAG
:: The format for Alias entries ->  Alias Name:List of Users/Groups::Comments
Redteam:tirapat,rajeev,sorin::
```

Now you can use the alias name in the /etc/project/admin or /etc/project/alter /template/admin file. It is also possible to exclude aliases (use ! in front of the actual alias name in the configuration file).

```
$RedAcct:-:-::Aliastest
```

Disabling accounting for selected processes

We may disable accounting for a selected process by using the NoAccounting project name in the admin or alternate admin file. Accounting can only be disabled for some of the processes (see Table 5-3).

Table 5-3 Disable accounting for selected processes

User	Group	Application	Project
Oracle	Oracle	/usr/*/oracle	NoAccounting
Root	Root	kbiod	NoAccounting

Relative project classification

A relative project ID is used to associate the project with the user or group ID. For this project ID. we used the keyword+constant format. Keyword is either $UID or $GID, and constant is either a decimal or hexadecimal number (see Table 5-4).

Table 5-4 Relative project classification

User	Group	Application	Project
-	-	-	$UID+1000000

User and group policy

This policy-associated project is listed with user and group ID. There is no file associate with this policy. Table 5-5 on page 169 shows the structure of a user policy.

Table 5-5 Structure of a user policy

User name	Project list
tester1	itsotest, System
tester2	Sell, Service

To create user and group policy

- Create a user or group name:

```
mkuser tester1 or mkgroup testgp
```

- Create the project:

```
projctl add projname projnumber [comment]; we use:
projctl add itsotest 8833 ITSOREDBOOKSTEST
```

- Associate the project with a user or group:

```
chuser projects=itsotest tester1 or chgroup projects=itsotest testgp
```

5.6.3 Load, unload, and query policies

Loading policies

After setting up the policy, we can load it. The Advanced Accounting subsystem uses the loaded policy to monitor the specified user, group, or application.

- Loading the admin policy

  smitty → Advanced Accounting → Manage Project Definitions and Assignments → Automatic Project Assignment → Work with Admin Policies → Load/Reload Admin Policy

  You could also use the SMIT fastpath **smitty load_admin** (Example 5-13).

Example 5-13 Load admin policy via SMIT

```
                        Load/Re-load Admin Policy

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Current focus is on                                 /etc/project          +
  Turn on auto-loading, when starting accounting      [yes]                 +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

- Load user policy or group policy

  Prior to loading user policy or group policy, we need to add the user policy or group policy associated with the project (tester1, projects=itsotest)

  ```
  chuser projects='itsotest' tester1
  ```

Example 5-14 shows the associated user's project status.

Example 5-14 Sample user status

```
[p630n02][/]> lsuser -a projects ALL
root
daemon
bin
sys
adm
uucp
guest
nobody
lpd
lp
invscout
snapp
nuucp
ipsec
tester1 projects=itsotest
tester2
tester3
freeware
sshd
```

- To load the user policy, use:

  smitty → Advanced Accounting → Manage Project Definitions and Assignments → Automatic Project Assignment → Work with User Policies → Load/Re-load User Policy → Turn on auto-loading, when starting accounting [yes]

  You could also use the SMIT fastpath **smitty load_users**.

Unloading policies

- To unload the admin policy:

  smitty → Advanced Accounting → Manage Project Definitions and Assignments → Automatic Project Assignment → Work with Admin Policies → Unload Admin Policy → Turn off auto-loading, when starting accounting [yes]

  You could also use the SMIT fastpath **smitty unload_admin**.

- To unload the user policy or group policy:

  smitty → Advanced Accounting → Manage Project Definitions and Assignments → Automatic Project Assignment → Work with User Policies → Unload User Policy

  You could also use the SMIT fastpath **smitty unload_users**.

Query status

- To query policy status:

  smitty → Advanced Accounting → Manage Project Definitions and Assignments → Automatic Project Assignment → Query Policies → Show Loaded Policies

  You could also use the command **projctl qpolicy** (Example 5-15).

Example 5-15 Querying loaded project and associated policy

```
[p630n02][/]> projctl qpolicy
Project definitions are loaded.
Project definition file name:  /etc/project/projdef
Admin policies are loaded.
Admin policy file name:  /etc/project/admin
User policies are loaded.
```

5.6.4 Manual loading of a project

By default all applications or programs are under the System project and it may be best not to change to another project name until you have run that program. Manual loading can be used for changing the application to another project before or after program the program runs. The user who issues the command will become the project owner. No policy is required to manually load a project, though you have to add the project name before manually loading the project.

- To manually assign an application to a project, use:

  ```
  projctl exec projname Applicationname
  ```

- To check which application is assigned to which project, you must know the process ID of the application. To find this out, use:

  ```
  smitty manual_assign → Show project assignment for all processes
  ```

  Example 5-16 shows the manual loading of the application `hog` and assigning it to project `Test`.

Example 5-16 Manual loading of a project with the projctl command

```
[node6][/]> id
uid=0(root) gid=0(system)
groups=2(bin),3(sys),7(security),8(cron),10(audit),11(lp)
```

```
[node6][/]> projctl exec Test '/tmp/hog -m 1' -f
[node6][/]> ps -eP | grep hog
       0        0   831650  pts/8  0:00 *Test            0  hog
```

If the application is already running, you can change the project associated with the application later (Example 5-17).

Example 5-17 Manually changing the project

```
$ id
uid=202(tester1) gid=1(staff) groups=0(system)
$ /tmp/hog -m 1 1>/tmp/msg1.out 2>>/tmp/msg1.out &
[1]     21290
$ ps -ef | grep hog
 tester1 21290 24784   0 09:31:19  pts/0  0:00 /tmp/hog -m 1
 tester1 23244 24784   0 09:31:23  pts/0  0:00 grep hog
$ projctl qapp /tmp/hog
List of projects for /tmp/hog as below:
System


smitty manual_assign  → Change project assignment for a process

                   Change project assignment for a process

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Project Name                                       [testApp]               +
  Process-ID                                         [21290]                  #
* Override policy rules for the Project               yes                    +




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do

smitty manual_assign  → Show project assignments for a program

                          COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

List of projects for /tmp/hog as below:
```

```
testApp


F1=Help          F2=Refresh       F3=Cancel        F6=Command
F8=Image         F9=Shell         F10=Exit         /=Find
n=Find Next
```

5.7 Reporting and analysis

Advanced Accounting provides accounting data for a variety of resources that can be used in charge back. Actually, the tools for analyzing the report are not provided. The format for accounting data and individual records is described in the /usr/include/sys/aacct.h header file.

A sample program (C source code) is provided as a starting point for analyzing accounting data files. The source file is /usr/samples/aacct/readaacct.c.

There is also a compiled version of this file, and it has the following syntax:

```
/usr/samples/aacct/readaacct [-F file] [-t trid] [-b begin_time] [-e
end_time] [-c] [-h]
```

- The -c flag is used to display information in colon-separated format.
- The -h flag is used to display information about the file, such as the host name, machine model, and serial number of the server where the data was produced.
- The -b (begin) and -e (end) flags give a time-based view of the information. The format is MMDDHHmmYY (1025141504 represents Oct 25,2004 14:15).
- The -t flag gives a record-based view of the information.

Before we run the **readaacct** command, we need to check the data files that we set up previously (Example 5-18).

Example 5-18 Accounting data file usage

```
[p630n02][/]> acctctl fquery
FILENAME
  STATE      | FIRST RECORD TIME          | LAST RECORD TIME           | UTIL
/var/aacct/aacct1.dat
  Processing | Mon Oct 11 15:43:32 2004   | Mon Oct 11 15:43:43 2004   |   0%

/var/aacct/aacct2.dat
  Waiting    | Mon Oct 11 16:19:56 2004   | Mon Oct 11 16:31:48 2004   |  18%
```

Example 5-19 and Example 5-20 on page 175 show relevant parts of the report obtained from the **readaacct** command. Additional details about LPAR name and WLM class can be found in the Advanced Accounting record.

In Example 5-19, you can see the process record for transaction ID=1. This record shows who executed the command and related information: user ID (UID), group ID (GID), process ID (PID), terminal ID (major, minor number in /dev directory), command name, process start time (in seconds from the EPOCH), WLM class key, process time, memory usage, disk usage, and network adapter usage. The start and end time are in UNIX EPOCH format (the number of seconds starting 00:00:00 UTC, January 1,1970). You may use this Web site to convert EPOCH time to human-readable format.

http://www.forestasia.com/tools/epoch.asp

To display the system date in this format, use:

```
# date +%s
```

In Advanced Accounting, the records for CPU time are in microseconds (this is an enhancement compared to traditional accounting), and are used to support the new POWER5™ features such as micro-partitioning.

The report contains:

- Elapsed process time shows elapsed time in microseconds.
- Elapsed thread time shows thread time in microseconds.
- Process CPU time shows processor time (combined threads) in microseconds.
- Elapsed Page seconds of disk pages shows the real page on physical disk e.g hdisk0.
- Elapsed Page seconds of real pages (in real memory -RAM).
- Elapsed Page seconds of virtual memory.

Example 5-19 Excerpt from readacct command report (full system partition)

```
# /usr/samples/aacct/readaacct -F /var/aacct/aacctdata
File Name=/var/aacct/acctdata
Version=0
Flags=0
Offset=2580480
File Size=5242880
State=2
ID=1
First Time=1097010706
Last Time=1097075182
System ID=IBM,0110685BF
```

```
System Model=IBM,7028-6C4
Host Name=p630n02
Partition Name=NULL
Partition Number=1
.................. Omitted lines .....................
Transaction ID=1
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 16:21:40
UID=0
GID=0
PID=170230
eWLM Service Class=0
Flags=1
Command Name=dd
Controlling Terminal's Device Number=31,5
Process Start Time=1097529700
WLM Class key=4927656507296075472
Incrementing Statistics:
Elapsed process time=0.005951 seconds
Elapsed thread time=0.005951 seconds
Process CPU time=0.005626 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=0 seconds
Elapsed Page seconds of virtual memory=0 seconds
Bytes of local file I/O=24826
Bytes of other file I/O=73
Bytes of local sockets=0
Bytes of remote sockets=0
```

Example 5-20 presents the same command executed on an LPAR system.

Example 5-20 readaacct on a LPAR-ed system

```
# /usr/samples/aacct/readaacct -F /var/aacct/aacct.dat1
File Name=/var/aacct/aacct.dat1
Version=0
Flags=0
Offset=12288
File Size=5242880
State=2
ID=1
First Time=1098210137
Last Time=1098210191
System ID=IBM,01022BE2A
System Model=IBM,7040-681
Host Name=p690_LPAR1
Partition Name=p690_LPAR1
```

```
Partition Number=2
------------------------------------------------------------

Transaction ID=1
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-19-2004 13:22:17
UID=0
GID=0
PID=13322
eWLM Service Class=0
Flags=1
Command Name=acctctl
Controlling Terminal's Device Number=22,3
Process Start Time=1098210137
WLM Class key=7770295601810996315
Incrementing Statistics:
Elapsed process time=0.027034 seconds
Elapsed thread time=0.027034 seconds
Process CPU time=0.008704 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=2 seconds
Elapsed Page seconds of virtual memory=1 seconds
Bytes of local file I/O=0
Bytes of other file I/O=0
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
```

There are 17 transaction IDs (or records) in Advanced Accounting. The transaction ID that is used to record data in the accounting data file depends on the setup (policy, interval, aggregation). Not all transaction IDs will be recorded in the accounting data file. For more information about transaction IDs, refer to Appendix C, “Accounting records in Advanced Accounting” on page 229.

The following is a summary of records in Advanced Accounting:

- Pad record: type0
- Process record: type1
- Aggregated process record: type2
- Aggregated application record: type3
- Processor and memory use record: type4
- Policy record: type5
- File system activity record: type6
- Network interface I/O record: type7
- Disk I/O record: type8

- Lost data record: type9
- Server VIO record: type10
- Client VIO record: type11

  VIO (Virtual I/O) is a new feature in AIX 5L V5.3 that enables you to create a virtual I/O network (such as Ethernet, SCSI, FC) in a micropartitioned or LPAR environment. Learn more in the redbook *Advanced POWER Virtualization on IBM @server p5 Servers: Introduction and Basic Configuration",* SG24-7940.

- Third-party kernel extension common aggregation record: type12
- ARM application environment record: type13
- ARM transaction environment record: type14
- ARM transaction instance record: type15
- ARM aggregated transaction instance record: type16
- Project definition record: type17

You can disable some transaction IDs using SMIT:

```
smitty manage_transaction -> Enable/Disable transactions
```

Note: We cannot disable agg_proc, agg_app, agg_KE, agg_applenv, arm_trenv, agg_arm (Transaction ID 2,3,12,13,14,16).

5.8 Testing example

This section describes a sample test we have performed in our ITSO lab.

5.8.1 Using no interval, aggregation, project, and policy

This example describes the record types in an accounting data file when there is no interval, policy, or aggregation turned on. The record types 1, 4, 7, and 8 are recorded in the accounting file. Accounting is turned on for only one minute (Example 5-21).

Example 5-21 Setup example

```
[p630n02][/]> projctl qpolicy
Currently none of the policies are loaded.
[p630n02][/]> projctl qproj
Project definitions are not loaded.
[p630n02][/]> acctctl on; sleep 60; acctctl off
------------------------------------------------------
In another terminal, during the execution of the previous comand, verify that
Advanced Accounting subsystem is running:
------------------------------------------------------
[p630n02][/]> acctctl
```

```
Advanced Accounting is running.
Email notification is on.
The current email address to be used is root.
Process Interval Accounting is off.
System Interval Accounting is off.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 2 defined, 1 available.
[p630n02][/]>
```

Example 5-22 shows the report that was obtained from the **readacct** command.

Example 5-22 Accounting report

```
[p630n02][/]> /usr/samples/aacct/readaacct -h -F /var/aacct/aacct1.dat  | pg
File Name=/var/aacct/aacct1.dat
Version=0
Flags=0
Offset=12288
File Size=5242880
State=2
ID=1
First Time=1097521750
Last Time=1097521757
System ID=IBM,0110685BF
System Model=IBM,7028-6C4
Host Name=p630n02
Partition Name=NULL
Partition Number=1
---------------------------------------------------------------
Transaction ID=1
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 14:9:10
UID=0
GID=0
PID=135958
eWLM Service Class=0
Flags=1
Command Name=acctctl
Controlling Terminal's Device Number=31,1
Process Start Time=1097521750
WLM Class key=4927656507296075472
Incrementing Statistics:
Elapsed process time=0.011544 seconds
Elapsed thread time=0.011544 seconds
Process CPU time=0.003993 seconds
```

```
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=0 seconds
Elapsed Page seconds of virtual memory=0 seconds
Bytes of local file I/O=0
Bytes of other file I/O=0
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=1
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 14:9:17
UID=202
GID=1
PID=131786
eWLM Service Class=0
Flags=1
Command Name=sleep
Controlling Terminal's Device Number=31,7
Process Start Time=1097521742
WLM Class key=3152067346752840677
Incrementing Statistics:
Elapsed process time=15.002608 seconds
Elapsed thread time=15.002608 seconds
Process CPU time=0.002321 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=480 seconds
Elapsed Page seconds of virtual memory=466 seconds
Bytes of local file I/O=0
Bytes of other file I/O=0
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=1
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 14:9:17
UID=202
GID=1
PID=131788
eWLM Service Class=0
Flags=1
Command Name=w
Controlling Terminal's Device Number=31,7
Process Start Time=1097521757
WLM Class key=3152067346752840677
Incrementing Statistics:
```

```
Elapsed process time=0.007104 seconds
Elapsed thread time=0.007104 seconds
Process CPU time=0.006711 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=1 seconds
Elapsed Page seconds of virtual memory=1 seconds
Bytes of local file I/O=69477
Bytes of other file I/O=1059
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=1
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 14:9:17
UID=202
GID=1
PID=131790
eWLM Service Class=0
Flags=1
Command Name=ls
Controlling Terminal's Device Number=31,7
Process Start Time=1097521757
WLM Class key=3152067346752840677
Incrementing Statistics:
Elapsed process time=0.003102 seconds
Elapsed thread time=0.003102 seconds
Process CPU time=0.002796 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=0 seconds
Elapsed Page seconds of virtual memory=0 seconds
Bytes of local file I/O=10360
Bytes of other file I/O=62
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=4
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 14:9:10
Record Type=13
No. of CPUs=4
Entitled Capacity=400
Pad length=0
Idle Time=1
User Process Time=2
Interrupt Time=0
```

```
Memory Size=8192
Total Large Pages=0
Large Page In use=0
No. of page ins=0
No. of Page outs=0
No. of I/Os=0
No. of page steals=0
************************************************************
Transaction ID=4
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 14:9:17
Record Type=14
No. of CPUs=4
Entitled Capacity=400
Pad length=13512
Idle Time=28274
User Process Time=20
Interrupt Time=25
Memory Size=8192
Total Large Pages=0
Large Page In use=0
No. of page ins=0
No. of Page outs=0
No. of I/Os=0
No. of page steals=0
************************************************************
Transaction ID=7
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 14:9:17
Interface Name=en0
No. of I/Os=43
No. of bytes=2826
************************************************************
Transaction ID=7
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-11-2004 14:9:17
Interface Name=lo0
No. of I/Os=0
No. of bytes=0
************************************************************
Transaction ID=8
Flags=0
Transaction Project=0
```

```
Sub project ID=0
Transaction start time=10-11-2004 14:9:17
Total transfers=0
Total read blocks=0
Total write blocks=0
Block size=512
Disk Name=hdisk0
************************************************************
```

5.8.2 System interval on

In this example, only system interval is turned on. Compared to the previous test (in 5.8.1, “Using no interval, aggregation, project, and policy” on page 177), a new transaction record, transaction ID6, shows up. Accounting data is collected for only two minutes (Example 5-23 and Example 5-24).

Example 5-23 Advanced Accounting enabled with system interval turned on

```
[p630n02][/]> projctl qpolicy
Currently none of the policies are loaded.
[p630n02][/]> projctl qproj
Project definitions are not loaded.
[p630n02][/]> acctctl isystem 60
[p630n02][/]> acctctl on; sleep 120; acctctl off
------------------------------------------------------
In another terminal, during the execution of the previous comand, verify that
Advanced Accounting subsystem is running:
------------------------------------------------------
[p630n02][/]> acctctl
Advanced Accounting is running.
Email notification is on.
The current email address to be used is root.
Process Interval Accounting is off.
System Interval Accounting every 60 minutes.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 2 defined, 0 available.
[p630n02][/]>
```

The report shows the new transaction ID6 (Example 5-24).

Example 5-24 Sample report for system interval turned on

```
[p630n02][/]> /usr/samples/aacct/readaacct -h -F /var/aacct/aacct2.dat
File Name=/var/aacct/aacct2.dat
Version=0
Flags=0
```

```
Offset=12288
File Size=5242880
State=2
ID=2
First Time=1097617824
Last Time=1097617832
System ID=IBM,0110685BF
System Model=IBM,7028-6C4
Host Name=p630n02
Partition Name=NULL
Partition Number=1
------------------------------------------------------------
Transaction ID=1
.................. Ommited lines ..........................
************************************************************
Transaction ID=6
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-12-2004 16:50:32
No. of bytes transferred=413
No. of read/write=63
No. of Opens=0
No. of creates=0
No. of locks=0
File system type=0
Device name length=12
Mount point length=4
Device Name=/dev/hd4
Mount Point=/
************************************************************
Transaction ID=6
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-12-2004 16:50:32
No. of bytes transferred=29477
No. of read/write=15
No. of Opens=27
No. of creates=0
No. of locks=0
File system type=0
Device name length=12
Mount point length=8
Device Name=/dev/hd2
Mount Point=/usr
************************************************************
Transaction ID=6
Flags=5
```

```
Transaction Project=0
Sub project ID=0
Transaction start time=10-12-2004 16:50:32
No. of bytes transferred=2292
No. of read/write=6
No. of Opens=3
No. of creates=0
No. of locks=1
File system type=0
Device name length=12
Mount point length=8
Device Name=/dev/hd9var
Mount Point=/var
..................... Ommited lines .....................
```

5.8.3 Aggregation for each user ID

The previous two examples may seem to have a lot of transaction record for running Advanced Accounting for one or two minutes. To reduce the record and have more summary data record, use the aggregation method described in this example.

Example 5-25 shows how to accumulate usage by user ID (in our example, user ID 202) for total CPU usage time, and disk usage. We need to turn on both process interval and process aggregation. In this example, records (transaction IDs) of types 2, 4, 7, and 8 are recorded. Record type 2 shows the aggregate CPU, disk, memory, and file systems usage per user name (Example 5-25).

Example 5-25 Testing aggregation on each user

```
[p630n02][/]> projctl qprojs
Project definitions are not loaded.
[p630n02][/]> projctl qpolicy
Currently none of the policies are loaded.
[p630n02][/]> acctctl iprocess 60
[p630n02][/]> acctctl agproc on
[p630n02][/]> acctctl on; sleep 240; acctctl off
------------------------------------------------------
In another terminal, during the execution of the previous comand, verify that
Advanced Accounting subsystem is running:
------------------------------------------------------
[p630n02][/]> acctctl
Advanced Accounting is running.
Email notification is on.
The current email address to be used is root.
Process Interval Accounting every 60 minutes.
System Interval Accounting is off.
```

```
System-wide aggregation of process data is on.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 2 defined, 0 available.
```

After three minutes (240 seconds), Advanced Accounting is stopped. Alternately, you can wait until the active accounting data file is full.

Example 5-26 shows the report generated from the **readaacct** command. There is no transaction ID 1 recorded, as this is process aggregation. All transactions are in project ID0 (default). Transaction ID2 is not appropriate for WLM configuration, because it does not record any WLM class.

There is no information about the running program or command. You have to specify the project and the policy associates with the application.

Example 5-26 Sample aggregation on process

```
# /usr/samples/aacct/readaacct -h -F /var/aacct/aacct1.dat
File Name=/var/aacct/aacct1.dat
Version=0
Flags=0
Offset=12288
File Size=5242880
State=2
ID=1
First Time=1097590350
Last Time=1097591599
System ID=IBM,0110685BF
System Model=IBM,7028-6C4
Host Name=p630n02
Partition Name=NULL
Partition Number=1
----------------------------------------------------------------
Transaction ID=2
......................... Ommited lines ...................
**************************************************************
Transaction ID=2
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-12-2004 9:33:19
Start Time=1097590362222976
UID=202
No. of processes aggregated=188
Incrementing Statistics:
Elapsed process time=3218.729538 seconds
Elapsed thread time=3218.729538 seconds
```

```
Process CPU time=0.827582 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=405072 seconds
Elapsed Page seconds of virtual memory=384291 seconds
Bytes of local file I/O=4704374
Bytes of other file I/O=24780
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
.................. Ommited lines .......................
```

5.8.4 Loading both the project and the policy

This example shows account records when loading both the project and the policy. Unlike the previous example, you can see the project number associated with the user ID.

Note that any change in policy becomes active by reloading the policy during user login; thus the user must log in again before policy has an effect on the accounting record. In this example, we also perform project-level aggregation.

The steps to create the project name and admin policy are:

1. Add the project named `TesterPolicy`, project number `8833` (as in 5.6.1, “Projects” on page 163).
2. Add admin policy for user `tester1`.

The following list shows details about user ID and group ID in our testing environment. The files are /etc/project/projdef, /etc/project/admin, /etc/passwd, and /etc/group (Example 5-27).

Example 5-27 Listing the related accounting files

```
[p630n02][/]> cat /etc/project/projdef file;
System:0:y::Default System Project
TestPolicy:8833:y::TestLoadProjectPolicy
[p630n02][/]> cat /etc/project/admin file;
tester1:-:-:TestPolicy::
[p630n02][/]> cat /etc/passwd file ;
root:!:0:0::/:/usr/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/var/adm:
uucp:!:5:5::/usr/lib/uucp:
guest:!:100:100::/home/guest:
nobody:!:4294967294:4294967294::/:
```

```
lpd:!:9:4294967294::/:
lp:*:11:11::/var/spool/lp:/bin/false
invscout:*:6:12::/var/adm/invscout:/usr/bin/ksh
snapp:*:200:13:snapp login user:/usr/sbin/snapp:/usr/sbin/snappd
ipsec:*:201:1::/etc/ipsec:/usr/bin/ksh
nuucp:*:7:5:uucp login user:/var/spool/uucppublic:/usr/sbin/uucp/uucico
gdm:*:202:1::/home/gdm:/usr/bin/ksh
sshd:*:203:203::/var/empty:/usr/bin/ksh
tirapat:!:204:1::/home/tirapat:/usr/bin/ksh
rajeev:*:205:1::/home/rajeev:/usr/bin/ksh
sorin:*:206:1::/home/sorin:/usr/bin/ksh
kumiko:!:207:1::/home/kumiko:/usr/bin/ksh
tester1::208:1::/home/tester1:/usr/bin/ksh
tester2::209:1::/home/tester2:/usr/bin/ksh

[p630n02][/]> cat /etc/group file ;
system:!:0:root
staff:!:1:ipsec,gdm,sshd,tirapat,rajeev,sorin,kumiko,tester1,tester2
bin:!:2:root,bin
sys:!:3:root,bin,sys
adm:!:4:bin,adm
uucp:!:5:nuucp,uucp
mail:!:6:
security:!:7:root
cron:!:8:root
printq:!:9:lp
audit:!:10:root
ecs:!:28:
nobody:!:4294967294:nobody,lpd
usr:!:100:guest
perf:!:20:
shutdown:!:21:
lp:!:11:root,lp,printq
invscout:!:12:invscout
snapp:!:13:snapp
ipsec:!:200:
gdm:!:201:gdm
games:!:202:
sshd:!:203:sshd
```

Example 5-28 shows our sample test when loading both project and admin policy.

Example 5-28 Sample test when loading both project and policy

```
[node6][/]> projctl ldprojs -r -a
[node6][/]> projctl qprojs
Project Name      Project Number          Aggregation
System                         0              ENABLED
```

```
TestPolicy                  8833              ENABLED
[node6][/]> projctl ldadm -d /etc/project
[node6][/]> projctl qpolicy
Project definitions are loaded.
Project definition file name:  /etc/project/projdef
Admin policies are loaded.
Admin policy file name:  /etc/project/admin
[node6][/]> acctctl on
[node6][/]> acctctl
Advanced Accounting is running.
Email notification is on.
The current email address to be used is root.
Process Interval Accounting every 60 minutes.
System Interval Accounting is off.
System-wide aggregation of process data is on.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 2 defined, 1 available.
[p630n02][/]> acctctl off
```

Example 5-29 shows the report from the **readaacct** command, where the user tester1 (UID208) has assigned the project ID 8833, and user tester2 (UID209) is still assigned to the default project (project ID 0).

Example 5-29 Sample report when loading both the project and the policy

```
[node6][/]> /usr/samples/aacct/readaacct -h -F /var/aacct/aacct2.dat
File Name=/var/aacct/aacct2.dat
Version=0
Flags=0
Offset=12288
File Size=2097152
State=2
ID=4
First Time=1098740152
Last Time=1098740771
System ID=IBM,0110A395A
System Model=IBM,7028-6E4
Host Name=node6
Partition Name=NULL
Partition Number=1
----------------------------------------------------------------
Transaction ID=2
Flags=f1
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Start Time=1098740152141117
UID=0
```

```
No. of processes aggregated=129
Incrementing Statistics:
Elapsed process time=74965.534345 seconds
Elapsed thread time=239138.849592 seconds
Process CPU time=5.407039 seconds
Elapsed Page seconds of disk pages=15605752 seconds
Elapsed Page seconds of real pages=24894567 seconds
Elapsed Page seconds of virtual memory=36733009 seconds
Bytes of local file I/O=3808010
Bytes of other file I/O=203014
Bytes of local sockets=16376
Bytes of remote sockets=3071013
************************************************************
Transaction ID=2
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Start Time=1098740771161784
UID=1
No. of processes aggregated=1
Incrementing Statistics:
Elapsed process time=773.004355 seconds
Elapsed thread time=7730.043559 seconds
Process CPU time=0.000000 seconds
Elapsed Page seconds of disk pages=74208 seconds
Elapsed Page seconds of real pages=177790 seconds
Elapsed Page seconds of virtual memory=219532 seconds
Bytes of local file I/O=0
Bytes of other file I/O=0
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=2
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Start Time=1098740175025591
UID=203
No. of processes aggregated=2
Incrementing Statistics:
Elapsed process time=11.350287 seconds
Elapsed thread time=11.350287 seconds
Process CPU time=0.352280 seconds
Elapsed Page seconds of disk pages=225 seconds
Elapsed Page seconds of real pages=1958 seconds
Elapsed Page seconds of virtual memory=3223 seconds
Bytes of local file I/O=3276
```

```
Bytes of other file I/O=0
Bytes of local sockets=0
Bytes of remote sockets=123447
************************************************************
Transaction ID=2
Flags=0
Transaction Project=8833
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Start Time=1098740175048075
UID=208
No. of processes aggregated=282
Incrementing Statistics:
Elapsed process time=2223.842311 seconds
Elapsed thread time=2223.842311 seconds
Process CPU time=1.183154 seconds
Elapsed Page seconds of disk pages=11902 seconds
Elapsed Page seconds of real pages=254964 seconds
Elapsed Page seconds of virtual memory=410620 seconds
Bytes of local file I/O=7730384
Bytes of other file I/O=122251
Bytes of local sockets=0
Bytes of remote sockets=33881398
************************************************************
Transaction ID=2
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Start Time=1098740311680985
UID=209
No. of processes aggregated=121
Incrementing Statistics:
Elapsed process time=1520.682498 seconds
Elapsed thread time=1520.682498 seconds
Process CPU time=0.682171 seconds
Elapsed Page seconds of disk pages=9170 seconds
Elapsed Page seconds of real pages=207955 seconds
Elapsed Page seconds of virtual memory=307853 seconds
Bytes of local file I/O=3425270
Bytes of other file I/O=64784
Bytes of local sockets=0
Bytes of remote sockets=35258345
************************************************************
Transaction ID=2
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
```

```
Start Time=1098740771161802
UID=4294967294
No. of processes aggregated=1
Incrementing Statistics:
Elapsed process time=773.004362 seconds
Elapsed thread time=3092.017450 seconds
Process CPU time=0.002660 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=115177 seconds
Elapsed Page seconds of virtual memory=129091 seconds
Bytes of local file I/O=0
Bytes of other file I/O=0
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=4
Flags=f1
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:35:52
Record Type=13
No. of CPUs=2
Entitled Capacity=200
Pad length=227597
Idle Time=1
User Process Time=0
Interrupt Time=0
Memory Size=4096
Total Large Pages=0
Large Page In use=0
No. of page ins=0
No. of Page outs=0
No. of I/Os=0
No. of page steals=0
************************************************************
Transaction ID=4
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Record Type=14
No. of CPUs=2
Entitled Capacity=200
Pad length=227597
Idle Time=1229591
User Process Time=6775
Interrupt Time=932
Memory Size=4096
Total Large Pages=0
```

```
Large Page In use=0
No. of page ins=0
No. of Page outs=0
No. of I/Os=104
No. of page steals=0
************************************************************
Transaction ID=17
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:35:52
Project ID=0
Project Origin=0
Offset=System
************************************************************ Project ID=8833
Project Origin=0
Offset=TestPolicy
************************************************************
Transaction ID=5
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:35:52
Type=1
Event=1
************************************************************
Transaction ID=7
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Interface Name=en0
No. of I/Os=13674
No. of bytes=1483643
************************************************************
Transaction ID=7
Flags=4c
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Interface Name=lo0
No. of I/Os=290
No. of bytes=37872
************************************************************
Transaction ID=8
Flags=4c
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
```

```
Total transfers=0
Total read blocks=0
Total write blocks=0
Block size=2048
Disk Name=cd0
************************************************************
Transaction ID=8
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Total transfers=0
Total read blocks=0
Total write blocks=0
Block size=512
Disk Name=hdisk0:0
************************************************************
Transaction ID=8
Flags=30
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Total transfers=0
Total read blocks=0
Total write blocks=0
Block size=512
Disk Name=hdisk0
************************************************************
Transaction ID=8
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Total transfers=456
Total read blocks=0
Total write blocks=4072
Block size=512
Disk Name=hdisk1:0
************************************************************
Transaction ID=8
Flags=30
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Total transfers=456
Total read blocks=0
Total write blocks=4072
Block size=512
Disk Name=hdisk1
```

```
************************************************************
Transaction ID=8
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Total transfers=0
Total read blocks=0
Total write blocks=0
Block size=512
Disk Name=hdisk2:0
************************************************************
Transaction ID=8
Flags=30
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 16:46:11
Total transfers=0
Total read blocks=0
Total write blocks=0
Block size=512
Disk Name=hdisk2
************************************************************
```

5.8.5 Aggregation based on application record

In this example we use an application named `hog`. The same program is used for user tester1 and tester2 but on different directories.

The source code for hog can be found in the redbook *AIX 5L Workload Manager (WLM)*, SG24-5977.

Example 5-30 uses the files projdef, admin, passwd, and group.

Example 5-30 Aggregation based on application record

```
[p630n02][/]> cat /etc/projdef/projdef file;
System:0:y::Default System Project
TestApp:2004:y::TestHogApplication

[p630n02][/]> cat /etc/project/admin file;
tester1:-:/tmp/hog:TestApp::
tester2:-:/home/tester2/hog:TestApp::
```

The other files, /etc/passwd and /etc/group, are the same as in Example 5-27 on page 186.

Example 5-31 shows application accounting for the applications /tmp/hog and /home/tester2/hog for user IDs 208 and 209, respectively. The project number is 2004 and the project name is TestApp. There are four aggregation transaction records. (We log four Telnet sessions). Transaction ID3 is recorded.

Example 5-31 Application accounting

```
[node6][/]> projctl qprojs
Project Name      Project Number          Aggregation
System                         0              ENABLED
TestApp                     2004              ENABLED
[node6][/]> projctl qpolicy
Project definitions are loaded.
Project definition file name:  /etc/project/projdef
Admin policies are loaded.
Admin policy file name:  /etc/project/admin
[node6][/]> acctctl on
[node6][/]> acctctl
Advanced Accounting is running.
Email notification is on.
The current email address to be used is root.
Process Interval Accounting every 60 minutes.
System Interval Accounting is off.
System-wide aggregation of process data is on.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Files: 3 defined, 0 available.
[node6][/]> acctctl off

[node6][/]> /usr/samples/aacct/readaacct -F /var/aacct/aacct1.dat -h
File Name=/var/aacct/aacct1.dat
Version=0
Flags=0
Offset=12288
..................... Ommited lines .........................
************************************************************
Transaction ID=3
Flags=0
Transaction Project=2004
Sub project ID=0
Transaction start time=10-25-2004 14:58:6
Start Time=1098731497913218
UID=208
Aggregated Invocations=2
Device No. for command=9223372079804448775
Inode for command=122
Incrementing Statistics:
Elapsed process time=4064.968285 seconds
Elapsed thread time=5251.674063 seconds
```

```
Process CPU time=0.159828 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=253295 seconds
Elapsed Page seconds of virtual memory=232971 seconds
Bytes of local file I/O=581882
Bytes of other file I/O=114942
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=3
Flags=0
Transaction Project=2004
Sub project ID=0
Transaction start time=10-25-2004 14:58:6
Start Time=1098731487375628
UID=209
Aggregated Invocations=2
Device No. for command=9223372079804448776
Inode for command=27
Incrementing Statistics:
Elapsed process time=3980.405918 seconds
Elapsed thread time=5607.501343 seconds
Process CPU time=0.252653 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=249846 seconds
Elapsed Page seconds of virtual memory=229944 seconds
Bytes of local file I/O=581882
Bytes of other file I/O=101371
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=3
Flags=0
Transaction Project=2004
Sub project ID=0
Transaction start time=10-25-2004 15:8:1
Start Time=1098734844944865
UID=208
Aggregated Invocations=3
Device No. for command=9223372079804448775
Inode for command=122
Incrementing Statistics:
Elapsed process time=569.631784 seconds
Elapsed thread time=588.234780 seconds
Process CPU time=0.036740 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=34829 seconds
Elapsed Page seconds of virtual memory=31981 seconds
Bytes of local file I/O=581882
```

```
Bytes of other file I/O=41012
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
Transaction ID=3
Flags=0
Transaction Project=2004
Sub project ID=0
Transaction start time=10-25-2004 15:8:1
Start Time=1098734830715385
UID=209
Aggregated Invocations=2
Device No. for command=9223372079804448776
Inode for command=27
Incrementing Statistics:
Elapsed process time=578.413553 seconds
Elapsed thread time=679.898337 seconds
Process CPU time=0.089616 seconds
Elapsed Page seconds of disk pages=0 seconds
Elapsed Page seconds of real pages=35688 seconds
Elapsed Page seconds of virtual memory=32795 seconds
Bytes of local file I/O=290941
Bytes of other file I/O=45562
Bytes of local sockets=0
Bytes of remote sockets=0
************************************************************
.................. Ommited lines ..................
************************************************************
Transaction ID=17
Flags=0
Transaction Project=0
Sub project ID=0
Transaction start time=10-25-2004 13:58:6
Project ID=0
Project Origin=0
Offset=System
************************************************************
Project ID=2004
Project Origin=0
Offset=TestApp
************************************************************
.................. Ommited lines ..........................
```

To conclude, we review the steps for setting up Advanced Accounting.

1. Set up accounting data file.
2. Set up e-mail notification.

3. Define the interval and aggregation (optional):
   - Process interval
   - System interval
4. Define the project and the policy, then load project and policy (option).
5. Start Advanced Accounting subsystem (**`acctctl on`**).
6. After the desired time, stop it, or wait until the accounting data file is full.
7. Run the **`readaacct`** command or use a customer-defined billing application.

Charging by process and CPU time is the most practical approach because the CPU is the most expensive resource. To chargeback by CPU usage time, we use a simple calculation based on this method:

```
Charge per user = (CPU usage-per-user/ Total CPU usage-all user) * Cost
```

Cost per month may be the machine price divided by the number of months the machine is expected to be used, plus hardware and software maintenance costs.

This information may also be used for capacity planning, to allocate busy resources (the busiest disks, file systems, network adapters, applications, users, groups, LPAR, WLM).

Furthermore, we can check how the application utilizes the resources (that is, whether it is CPU-intensive, memory-intensive, I/O-intensive, or a combination). The accounting data is useful because you can predict resource utilization based on historical usage.

This may help with resource constraint and future performance problems by adding new resources at the right time or balance existing ones for better throughput.

The Advanced Accounting subsystem is bundled in the kernel. It has virtually no influence on system performance. The accounting data file is not considered as consuming too many resources, especially when running in aggregation mode.

A

Security audit events in AIX 5.3

The /etc/security/audit/events file contains all known events in AIX 5.3.

```
auditpr[3]:

*  shmget()
   SHM_Create = printf “key: %d size: %ld flags: %o shmid: %d”

*  adjtime()
   PROC_Adjtime = printf “old time: %T, delta: %d:%d”

*  settimer()
   PROC_Settimer = printf “old time: %T, new time: %T”

*  mknod()
   FILE_Mknod = printf “mode: %o dev: %D filename %s”

*  mkdev
   DEV_Create = printf “mode: %o dev: %D filename %s”
   DEV_Start = printf “ %s “

auditpr:

* kernel proc events
```

```
*  fork()
   PROC_Create = printf “forked child process %d”

*  exit()
   PROC_Delete = printf “exited child process %d”

*  exec()
   PROC_Execute = printf “euid: %d egid: %d epriv: %x:%x name %s”

*  exec() of a large page data process
   PROC_LPExecute = printf “euid: %d egid: %d epriv: %x:%x name %s”

*  setuidx()
   PROC_RealUID = printf “real uid: %d”
   PROC_AuditID = printf “login uid: %d”
   PROC_SetUserIDs = printf “effect: %d, real: %d, saved: %d, login: %d”

*  setgidx()
   PROC_RealGID = printf “old rgid: %d, new gid: %d, which: %s”

*  accessx()
   FILE_Accessx = printf “mode: %o, who: %d, path: %s”

*  statacl()
   FILE_StatAcl = printf “cmd: %o, path: %s”

*  chxacl()
   FILE_WriteXacl = printf “path: %s, ACL: %C”

*  fchxacl()
   FILE_FWriteXacl = printf “fd: %d, ACL: %C”

*  statxacl()
   FILE_ReadXacl = printf “path: %s”

*  statxacl()
   FILE_FReadXacl = printf “fd: %d”

*  aclxcntl()
   SEC_aclxcntl = printf “path: %s, cmd: %d”

*  statpriv()
   FILE_StatPriv = printf “cmd: %o, path: %s”

*  revoke()
   FILE_Revoke = printf “path: %s”
```

```
*  frevoke()
   FILE_Frevoke = printf “fd: %d”

*  usrinfo()
   PROC_Environ = printf “buf: %s”

*  sigaction()
   PROC_SetSignal = printf ““

*  setrlimit()
   PROC_Limits = printf ““

*  nice()
   PROC_SetPri = printf “new priority: %d”

*  setpri()
   PROC_Setpri = printf “new priority: %d”

*  setpriv()
   PROC_Privilege = printf “cmd: %x privset: %x:%x”

*  settimer()
   PROC_Settimer = printf “old time: %x:%x, new time: %x:%x”

*  adjtime()
   PROC_Adjtime = printf “old time: %x:%x, delta: %x:%x”

*  ptrace()
   PROC_Debug = printf “pid: %d command: %d”

*  kill()
   PROC_Kill = printf “pid: %d, sig: %d”

*  setpgid()
   PROC_Setpgid = printf “pid: %d, pgrp: %d”

*  ld_loadmodule()
   PROC_Load = printf “file: %s”
   PROC_LoadMember = printf “file: %s, member: %s”
   PROC_LoadError = printf “flags: %x, libpath: %s, file: %s”

*  setgroups()
   PROC_SetGroups = printf “group set: %G”

*  sysconfig()
   PROC_Sysconfig = printf “%x”

*  audit
```

```
*  audit()
   AUD_It = printf “cmd: %d arg: %d”

*  auditbin()
   AUD_Bin_Def = printf “cmd: %d cur_fd: %d next_fd: %d, threshold: %d”

*  auditevents()
   AUD_Events = printf “cmd: %d”

*  auditobj()
   AUD_Objects = printf “cmd: %d”

*  auditproc()
   AUD_Proc = printf “pid: %d cmd: %d”

*  acct()
   ACCT_Disable = printf ““
   ACCT_Enable = printf “file: %s”

* file system events

*  open() and creat()
   FILE_Open = printf “flags: %d mode: %o fd: %d filename %s”
   TCB_Leak = printf ““
   TCB_Mod = printf ““
   TCB_Exec = printf “filename: %s”

*  read()
   FILE_Read = printf “file descriptor = %d”

*  write()
   FILE_Write = printf “file descriptor = %d”

*  close()
   FILE_Close = printf “file descriptor = %d”

*  link()
   FILE_Link = printf “linkname %s filename %s”

*  unlink()
   FILE_Unlink = printf “filename %s”

*  rename()
   FILE_Rename = printf “frompath: %s topath: %s”

*  chown()
   FILE_Owner = printf “owner: %d group: %d filename %s”
```

```
*  chmod()
   FILE_Mode = printf “mode: %o filename %s”

*  fchmod()
   FILE_Fchmod = printf “mode: %o file descriptor %d”

*  fchown()
   FILE_Fchown = printf “owner: %d group: %d file descriptor %d”

*  truncate()
   FILE_Truncate = printf “filename = %s”

*  symlink()
   FILE_Symlink = printf “link = %s, target = %s”

*  pipe()
   FILE_Pipe = printf “read: %d write: %d”

*  mknod()
   FILE_Mknod = printf “mode: %o dev: %d filename %s”

*  fcntl(F_DUPFD)
   FILE_Dupfd = printf “original fd: %d new fd: %d”

*  fscntl()
   FS_Extend = printf “vfs: %d, cmd: %d”

*  mount()
   FS_Mount = printf “mount: object %s stub %s”

*  umount()
   FS_Umount = printf “umount: object %s stub %s”

*  chacl()
   FILE_Acl = printf “filename: %s, ACL: %A”
   FILE_Facl = printf “fd: %d, ACL: %A”

*  chpriv()
   FILE_Privilege = printf “pcl: %d”
   FILE_Chpriv = printf “file: %s, pcl: %P”
   FILE_Fchpriv = printf “fd: %d, pcl: %P”

*  chdir()
   FS_Chdir = printf “change current directory to: %s”

*  fchdir()
   FS_Fchdir = printf “fd = %d”
```

```
*  chroot()
   FS_Chroot = printf “change root directory to: %s”

*  rmdir()
   FS_Rmdir = printf “remove of directory: %s”

*  mkdir()
   FS_Mkdir = printf “mode: %o dir: %s”

*  utimes()
   FILE_Utimes = printf “filename: %s”

*  stat()
   FILE_Stat = printf “cmd: %x filename: %s”

* SVIPC system events

*  msgget()
   MSG_Create = printf “key: %d flags: %o msqid: %d”

*  msgrcv()
   MSG_Read = printf “msqid: %d muid: %d mpid: %d”

*  msgsnd()
   MSG_Write = printf “msqid: %d”

*  msgctl()
   MSG_Delete = printf “msqid: %d”
   MSG_Owner = printf “msqid: %d owner: %d group: %d mode: %o”
   MSG_Mode = printf “msqid: %d mode: %o”

*  semget()
   SEM_Create = printf “key: %d nsems: %d flags: %o semid: %d”

*  semop()
   SEM_Op = printf “semid: %d”

*  semctl()
   SEM_Delete = printf “semid: %d”
   SEM_Owner = printf “semid: %d owner: %d group: %d mode: %o”
   SEM_Mode = printf “semid: %d mode: %o”

*  shmget()
   SHM_Create = printf “key: %d size: %d flags: %o shmid: %d”

*  shmget(SHM_LGPAGE)
   SHM_LPCreate = printf “key: %d size: %ld flags: %o shmid: %d”
```

```
*  shmat()
   SHM_Open = printf “shmid: %d”

*  shmat()
   SHM_Detach = printf “shmid: %d”

*  shmctl()
   SHM_Close = printf “shmid: %d”
   SHM_Owner = printf “shmid: %d owner: %d group: %d”
   SHM_Mode = printf “shmid: %d mode: %o”

* TCPIP user level

   TCPIP_config = printf “%s %s %s %s %s”
   TCPIP_host_id = printf “%s %s %s %s”
   TCPIP_route = printf “%s %s %s %s %s”
   TCPIP_connect = printf “%s %s %s %s %s”
   TCPIP_data_out = printf “%s %s %s %s %s”
   TCPIP_data_in = printf “%s %s %s %s %s”
   TCPIP_access = printf “%s %s %s %s %s”
   TCPIP_set_time = printf “%s %s %s %s”

* TCPIP kernel level

   TCP_ksocket = printf “fd%d %s, %s, Protocol %d”
   TCP_ksocketpair = printf “fd%d fd%d %s, %s, Protocol %d”
   TCP_kclose = printf “fd%d”
   TCP_ksetopt = printf “fd%d Port %s, Level %d, Option %d, Value %d”
   TCP_kbind = printf “fd%d %S”
   TCP_klisten = printf “fd%d qlimit %d”
   TCP_kconnect = printf “fd%d %L”
   TCP_kaccept = printf “fd%d Port %S  %L”
   TCP_kshutdown = printf “fd%d %s”
   TCP_ksend = printf “fd%d %s”
   TCP_kreceive = printf “fd%d %s”

* commands

*  tsm
   USER_Login = printf “user: %s tty: %s”
   PORT_Locked = printf “Port %s locked due to invalid login attempts”
   TERM_Logout = printf “%s”

*  rlogind/telnetd
   USER_Exit = printf “tty: %s”

*  sysck
   SYSCK_Check = printf “%s”
   SYSCK_Update = printf “%s”
```

```
    SYSCK_Install = printf "%s"
    SYSCK_Delete = printf "%s %s"

*   tcbck
    TCBCK_Check = printf "%s"
    TCBCK_Update = printf "%s"
    TCBCK_Delete = printf "%s"

*   usrck
    USER_Check = printf "%s %s %s"
    USRCK_Error = printf "%s %s"

*   logout
    USER_Logout = printf "%s"

*   chsec
    PORT_Change = printf "Changed attributes of port %s; new values: %s"

*   chuser
    USER_Change = printf "%s %s"

*   rmuser
    USER_Remove = printf "%s"

*   mkuser
    USER_Create = printf "%s %s"

*   setgroups
    USER_SetGroups = printf "%s %s"

*   setsenv
    USER_SetEnv = printf "environment %s"

*   su
    USER_SU = printf "%s"

*   grpck
    GROUP_User = printf "grpck: removed user %s from %s in /etc/group"

*   grpck
    GROUP_Adms = printf "grpck: removed admin user %s from %s in
/etc/security/group"

*   chgroup
    GROUP_Change = printf "%s %s"

*   mkgroup
    GROUP_Create = printf "%s %s"
```

```
*  rmgroup
   GROUP_Remove = printf “%s”

*  passwd
   PASSWORD_Change = printf “%s”

*  pwdadm
   PASSWORD_Flags = printf “%s %s”

*  pwdck
   PASSWORD_Check = printf “User = %s Error/Fix = %s Status = %s”
   PASSWORD_Ckerr = printf “User/File = %s Error = %s”

*  pagdel
   USER_PagDelete = printf “username: %s”

*  paginit
   USER_Paginit = printf “tty: %s”


*  startsrc
   SRC_Start = printf “%s”

*  stopsrc
   SRC_Stop = printf “%s”

*  addssys
   SRC_Addssys = printf “%s”

*  chssys
   SRC_Chssys = printf “%s”

*  addserver
   SRC_Addserver = printf “%s”

*  chserver
   SRC_Chserver = printf “%s”

*  rmssys
   SRC_Delssys = printf “%s”

*  rmserver
   SRC_Delserver = printf “%s”

*  enq
   ENQUE_admin = printf “queue = %s device = %s request = %s to: %s op = %s”
```

```
*  qdaemon
   ENQUE_exec = printf “queue = %s request = %s host = %s file = %s to: %s op
= %s”

*  sendmail
   SENDMAIL_Config = printf “%s”
   SENDMAIL_ToFile = printf “Mail from user %s to file %s”
   MAIL_ToUser = printf “sender: %s  recipient: %s”

*  at
   AT_JobAdd = printf “file name = %s User = %s time = %s”
   AT_JobRemove = printf “file name = %s User = %s”

*  cron
   CRON_JobRemove = printf “file name = %s User = %s time = %s”
   CRON_JobAdd = printf “file name = %s User = %s time = %s”
   CRON_Start = printf “event = %s cmd = %s time = %s”
   CRON_Finish = printf “user = %s pid = %s time = %s”

*  nvload
   NVRAM_Config = printf “ %s”

*  cfgmgr
   DEV_Configure = printf “ device %s”

*  chdev and mkdev
   DEV_Change = printf “ params = %s”

*  mkdev
   DEV_Create = printf “mode: %o dev: %d filename %s”
   DEV_Start = printf “ %s “

*  installp
   INSTALLP_Inst = printf “Option Name: %s Level: %s  Installation %s”
   INSTALLP_Exec = printf “Option Name: %s Level: %s  Executed Program %s”

*  rmdev
   DEV_Stop = printf “ device %s”
   DEV_UnConfigure = printf “ device %s”
   DEV_Remove = printf “ device %s”

*  DSMIT
   DSMIT_start = printf “%s”
   DSMIT_end = printf “%s”

* LVM events

*  lchangelv, lextendlv, lreducelv
   LVM_ChangeLV= printf “ %s “
```

```
*  lchangepv, ldeletepv, linstallpv
   LVM_ChangeVG = printf " %s "

*  lcreatelv
   LVM_CreateLV = printf " %s "

*  lcreatevg
   LVM_CreateVG = printf " %s "

*  ldeletepv
   LVM_DeleteVG = printf " %s "

*  rmlv
   LVM_DeleteLV = printf " %s "

*  lvaryoffvg
   LVM_VaryoffVG = printf " %s "

*  lvaryonvg
   LVM_VaryonVG = printf " %s "

*  Logical volume operations
   LVM_AddLV = printf "Logical Volume ID: %08x%08x%08x%08x.%d"
   LVM_KDeleteLV = printf "Logical Volume ID: %08x%08x%08x%08x.%d"
   LVM_ExtendLV = printf "Logical Volume ID: %08x%08x%08x%08x.%d %s"
   LVM_ReduceLV = printf "Logical Volume ID: %08x%08x%08x%08x.%d %s"
   LVM_KChangeLV = printf "Logical Volume ID: %08x%08x%08x%08x.%d %s"
   LVM_AvoidLV = printf "Logical Volume ID: %08x%08x%08x%08x.%d %s"

*  Physical volume operations
   LVM_MissingPV = printf "Volume Group ID: %08x%08x%08x%08x Physical Volume
Index: %d"
   LVM_AddPV = printf "Volume Group ID: %08x%08x%08x%08x Physical Volume
device(major,minor): %X"
   LVM_AddMissPV = printf "Volume Group ID: %08x%08x%08x%08x Physical Volume
Index: %d"
   LVM_DeletePV = printf "Volume Group ID: %08x%08x%08x%08x Physical Volume
Index: %d"
   LVM_RemovePV = printf "Volume Group ID: %08x%08x%08x%08x Physical Volume
Index: %d"
   LVM_AddVGSA = printf "Volume Group ID: %08x%08x%08x%08x Physical Volume
Index: %d"
   LVM_DeleteVGSA = printf "Volume Group ID: %08x%08x%08x%08x Physical Volume
Index: %d"

*  Volume group operations
   LVM_SetupVG = printf "Volume Group ID: %08x%08x%08x%08x"
   LVM_DefineVG = printf "Volume Group ID: %08x%08x%08x%08x"
```

```
   LVM_KDeleteVG = printf “Volume Group ID: %08x%08x%08x%08x”
   LVM_ChgQuorum = printf “Volume Group ID: %08x%08x%08x%08x %s”
   LVM_Chg1016 = printf “Volume Group ID: %08x%08x%08x%08x New Factor value:
%d”
   LVM_UnlockDisk = printf “Volume Group ID: %08x%08x%08x%08x”
   LVM_LockDisk = printf “Volume Group ID: %08x%08x%08x%08x”

*  backup, restore
   BACKUP_Export = printf “ %s “
   BACKUP_Priv = printf “ %s “
   RESTORE_Import = printf “ %s “

*  shell
   USER_Shell = printf “tty: %s “

*  reboot
   USER_Reboot = printf “ %s “
   PROC_Reboot = printf “cmd: %d  time: %T”

* objects (files)

*  /etc/security/environ
   S_ENVIRON_WRITE = printf “%s”

*  /etc/group
   S_GROUP_WRITE = printf “%s”

*  /etc/security/limits
   S_LIMITS_WRITE = printf “%s”

*  /etc/security/login.cfg
   S_LOGIN_WRITE = printf “%s”

*  /etc/security/passwd
   S_PASSWD_READ = printf “%s”

*  /etc/security/passwd
   S_PASSWD_WRITE = printf “%s”

*  /etc/security/user
   S_USER_WRITE = printf “%s”

*  /etc/security/audit/config
   AUD_CONFIG_WR = printf “%s”

*  /etc/security/roles
   ROLE_Create = printf “%s %s”
```

```
*  /etc/security/roles
   ROLE_Change = printf “Role: %s  Attribute: %s”

*  /etc/security/roles
   ROLE_Remove = printf “%s”

*  /etc/init
   INIT_Start = printf “pid: %d, command: %s”
   INIT_End = printf “pid: %d, status: %x”

* miscellaneous

*  count of audit recs authat didn’t get written
   AUD_Lost_Recs = printf “Recs Lost: %d”

* SecureWay Directory Server

*  LDAP_Bind
   LDAP_Bind = printf “ConnectID: %d Host: %s Port: %d BindDN: %s”

*  LDAP_Unbind
   LDAP_Unbind = printf “ConnectID: %d”

*  LDAP_Add
   LDAP_Add = printf “ConnectID: %d Entry: %s”

*  LDAP_Delete
   LDAP_Delete = printf “ConnectID: %d Entry: %s”

*  LDAP_Modify
   LDAP_Modify = printf “ConnectID: %d Entry: %s”

*  LDAP_Modifydn
   LDAP_Modifydn = printf “ConnectID: %d NewEntry: %s OldEntry: %s”

*  LDAP_Search
   LDAP_Search = printf “ConnectID: %d Search: %s”

*  LDAP_Compare
   LDAP_Compare = printf “ConnectID: %d Compare: %s”

* Certificate Authentication Services

*  certcreate
   CERT_Create = printf “User: %s %s tty: %s”

*  certdelete
   CERT_Delete = printf “User: %s %s”
```

```
*  certget
   CERT_Get = printf “User: %s %s”

*  certlist
   CERT_List = printf “User: %s %s”

*  certadd
   CERT_Add = printf “User: %s %s”

*  certlink
   CERT_Link = printf “User: %s %s”

*  certverify
   CERT_Verify = printf “User: %s %s tty: %s”

*  certrevoke
   CERT_Revoke = printf “User: %s %s tty: %s”

*  keypasswd
   KEY_PasswordChange = printf “User: %s %s tty: %s”

*  keydelete
   KEY_Delete =  printf “User: %s %s tty: %s”

*  keylist
   KEY_List = printf “User: %s %s tty: %s”

*  keyadd
   KEY_Add = printf “User: %s %s tty: %s”

* RTIPC system events

*  rtsem_open()
   RTSEM_Open = printf “flags: %o mode: %o value: %d name: %s usemid: %08x%08x
semx: %d”

*  rtsem_close()
   RTSEM_Close = printf “semx: %d”

*  rtsem_unlink()
   RTSEM_Unlink = printf “name: %s”

*  rtsem_init()
   RTSEM_Init = printf “pshared: %d value: %d semx: %d”

*  rtsem_destroy()
   RTSEM_Destroy = printf “semx: %d”
```

```
*  rtsem_getvalue()
   RTSEM_Getvalue = printf “semx: %d value: %d”

*  rtsem_post()
   RTSEM_Post = printf “semx: %d”

*  rtsem_wait()
   RTSEM_Wait = printf “semx: %d”

*  rtsem_trywait()
   RTSEM_TryWait = printf “semx: %d”

*  rtmq_open()
   RTMQ_Open = printf “flags: %o mode: %o name: %s mqd: %x mqx: %d”

*  rtmq_close()
   RTMQ_Close = printf “mqd: %x”

*  rtmq_unlink()
   RTMQ_Unlink = printf “name: %s”

*  rtmq_getattr()
   RTMQ_Getattr = printf “mqd: %x flags: %o maxmsg: %d msgsize: %d curmsg: %d”

*  rtmq_setattr()
   RTMQ_Setattr = printf “mqd: %x flags: %o oflags: %o omaxmsg: %d omsgsize:
%d ocurmsg: %d”

*  rtmq_notify()
   RTMQ_Notify = printf “mqd: %x type: %d signo: %d tid: %d”
   RTMQ_NotifyCanc = printf “mqd: %x”

*  rtmq_receive()
   RTMQ_Receive = printf “mqd: %x size: %d prio: %d”

*  rtmq_send()
   RTMQ_Send = printf “mqd: %x size: %d prio: %d”

*  rtshm_open()
   RTSHM_Open = printf “flags: %o mode: %o fd: %d name: %s”

*  rtshm_unlink()
   RTSHM_Unlink = printf “name: %s”

* Advanced Accounting system events

*  _acctctl(ACC_START, ...)
   AACCT_On = printf “AACCT: startup”
```

```
*  _acctctl(ACC_STOP, ...)
   AACCT_Off = printf “AACCT: shutdown”

*  _acctctl(ACC_ADD_FILE, ...)
   AACCT_AddFile = printf “AACCT: File added: ID %d, size %ld, filename %s”

*  _acctctl(ACC_POST_FILE, ...)
   AACCT_ResetFile = printf “AACCT: File reset: ID %d, filename %s”

*  _acctctl(ACC_RM_FILE, ...)
   AACCT_RmFile = printf “AACCT: File removed: ID %d, filename %s”

*  _acctctl(ACC_SWITCH_FILE, ...)
   AACCT_SwtchFile = printf “AACCT: Active file switched: ID %d”

*  set_trid(..., ON, ...)
   AACCT_TridOn = printf “AACCT: Trid enabled: TRID %d”

*  set_trid(..., OFF, ...)
   AACCT_TridOff = printf “AACCT: Trid disabled: TRID %d”

*  set_interval(System Queue, 0)
   AACCT_SysIntOff = printf “AACCT: System Interval turned off”

*  set_interval(System Queue, time)
   AACCT_SysIntSet = printf “AACCT: System Interval set to %d”

*  set_interval(Process Queue, 0)
   AACCT_PrIntOff = printf “AACCT: Process Interval turned off”

*  set_interval(Process Queue, time)
   AACCT_PrIntSet = printf “AACCT: Process Interval set to %d”

*  projctl_proc(PROJ_SET, ...)
   AACCT_SwtchProj = printf “AACCT: PID %d switched to project %d”

*  proj_regop(PROJ_ADD*, ...)
   AACCT_AddProj = printf “AACCT: Project added: ID %d, name %s”

*  proj_regop(PROJ_REMOVE, ...)
   AACCT_RmProj = printf “AACCT: Project removed: ID %d, name %s”

*  add_policy()
   AACCT_PolLoad = printf “AACCT: Policy loaded: Type %d”

*  remove_policy()
   AACCT_PolUnload = printf “AACCT: Policy unloaded: Type %d”
```

```
*  set_email(EMAIL_ON)
   AACCT_NotChange = printf “AACCT: E-mail notification set to %s”

*  set_email(EMAIL_OFF)
   AACCT_NotifyOff = printf “AACCT: E-mail notification disabled”

* IPSEC user level

   IPSEC_chtun = printf “%s %s %s”
   IPSEC_exptun = printf “%s %s %s”
   IPSEC_gentun = printf “%s %s %s %s %s”
   IPSEC_imptun = printf “%s %s %s %s %s”
   IPSEC_lstun = printf “%s %s %s”
   IPSEC_mktun = printf “%s %s %s %s %s”
   IPSEC_rmtun = printf “%s %s %s”
   IPSEC_chfilt = printf “%s %s %s”
   IPSEC_expfilt = printf “%s %s %s”
   IPSEC_genfilt = printf “%s %s %s %s”
   IPSEC_trcbuf = printf “%s %s”
   IPSEC_impfilt = printf “%s %s %s %s”
   IPSEC_lsfilt = printf “%s %s %s %s”
   IPSEC_mkfilt = printf “%s %s %s %s %s %s %s %s”
   IPSEC_mvfilt = printf “%s %s %s”
   IPSEC_rmfilt = printf “%s %s %s %s”
   IPSEC_unload = printf “%s %s %s”
   IPSEC_stat = printf “%s %s %s”
   IKE_tnl_creat = printf “%s %s %s”
   IKE_tnl_delet = printf “%s %s %s”
   IPSEC_p1_nego = printf “%s %s %s”
   IPSEC_p2_nego = printf “%s %s %s”
   IKE_activat_cmd = printf “%s %s %s”
   IKE_remove_cmd = printf “%s %s %s”
```

B

The accounting files

This appendix describes the header files that are used for defining the format (structure) of the binary accounting files, and the contents of the bos.acct software package. These headers are located in your system's /usr/include directory.

The acct file format

The acct file format is defined in the /usr/include/sys/acct.h header file.

```
/* IBM_PROLOG_BEGIN_TAG                                                   */
/* This is an automatically generated prolog.                             */
/*                                                                        */
/* bos530 src/bos/kernel/sys/acct.h 1.17                                  */
/*                                                                        */
/* Licensed Materials - Property of IBM                                   */
/*                                                                        */
/* (C) COPYRIGHT International Business Machines Corp. 1988,1993          */
/* All Rights Reserved                                                    */
/*                                                                        */
/* US Government Users Restricted Rights - Use, duplication or            */
/* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.      */
/*                                                                        */
/* IBM_PROLOG_END_TAG                                                     */
/* @(#)39       1.17  src/bos/kernel/sys/acct.h, sysproc, bos530 4/11/04
16:55:57 */
/*
 *   COMPONENT_NAME: SYSPROC
 *
 *   FUNCTIONS:
 *
 *   ORIGINS: 27,3
 *
 *
 *   (C) COPYRIGHT International Business Machines Corp. 1988,1993
 *   All Rights Reserved
 *   Licensed Materials - Property of IBM
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
 */


#ifndef _H_ACCT
#define _H_ACCT

#include <sys/types.h>
#include <sys/param.h>

/*
 * Accounting structures
 * these use a comp_t type which is a 3 bit base 8
 * exponent, 13 bit fraction "floating point" number.
 * Units are 1/AHZ seconds.
 */
```

```
typedef ushort comp_t;          /* "floating point" */
                /* 13-bit fraction, 3-bit exponent  */

struct  acct
{
        char    ac_flag;                /* Accounting flag */
        char    ac_stat;                /* Exit status */
        char    ac_version;             /* File version */
        char    ac_len;                 /* Length of structure */
        uid_t   ac_uid;                 /* Accounting user ID */
        gid_t   ac_gid;                 /* Accounting group ID */
#ifdef __64BIT__
        dev32_t ac_tty;                 /* control typewriter */
```

The tacct file format

The tacct structure, which is not part of the acct.h header file, represents the total accounting format used by the various accounting commands.

```
struct tacct {
       uid_t ta_uid;          /* user-ID */
       char ta_name[8];       /* login name */
       float ta_cpu[2];       /* cum. CPU time, p/np (mins) */
       float ta_kcore[2];     /* cum. kcore-mins, p/np */
       float ta_io[2];        /* cum. chars xferred (512s) */
       float ta_rw[2];        /* cum. blocks read/written */
       float ta_con[2];       /* cum. connect time, p/np, mins */
       float ta_du;           /* cum. disk usage */
       long ta_qsys;          /* queuing sys charges (pgs) */
       float ta_fee;          /* fee for special services */
       long ta_pc;            /* count of processes */
       unsigned short ta_sc;  /* count of login sessions */
       unsigned short ta_dc;  /* count of disk samples */
};
```

The utmp file format

The utmp file format is defined in the /usr/include/utmp.h include header file.

```
/* IBM_PROLOG_BEGIN_TAG                                                 */
/* This is an automatically generated prolog.                           */
/*                                                                      */
/* bos530 src/bos/usr/include/utmp.h 1.11.1.19                          */
/*                                                                      */
/* Licensed Materials - Property of IBM                                 */
/*                                                                      */
```

```
/* (C) COPYRIGHT International Business Machines Corp. 1989,1994          */
/* All Rights Reserved                                                    */
/*                                                                        */
/* US Government Users Restricted Rights - Use, duplication or            */
/* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.      */
/*                                                                        */
/* IBM_PROLOG_END_TAG                                                     */
/* @(#)87  1.11.1.19  src/bos/usr/include/utmp.h, libcadm, bos530 2/27/04
15:37:47 */
/*
 *   COMPONENT_NAME: CMDOPER
 *
 *   FUNCTIONS: UTMP_DATA_INIT
 *
 *   ORIGINS: 27,71
 *
 *   (C) COPYRIGHT International Business Machines Corp. 1989,1994
 *   All Rights Reserved
 *   Licensed Materials - Property of IBM
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
 */

/*
 * (c) Copyright 1990, 1991, 1992 OPEN SOFTWARE FOUNDATION, INC.
 * ALL RIGHTS RESERVED
 */


#ifndef _H_UTMP
#define _H_UTMP

/* <sys/types.h> must be included.*/

#ifndef _H_UTMPX
#defineUTMP_FILE“/etc/utmp”
#define WTMP_FILE       “/var/adm/wtmp”
#define ILOG_FILE       “/etc/security/failedlogin”
#defineut_nameut_user
#endif /* _H_UTMPX */

struct utmp
  {
   char ut_user[256] ;/* User login name */
   char ut_id[14] ;/* /etc/inittab id */
   char ut_line[64] ;/* device name (console, lnxx) */
   pid_t ut_pid ;/* process id */
   short ut_type ; /* type of entry */
#if !defined(__64BIT__) && !defined(__ia64)
```

```
        int __time_t_space;             /* for 32vs64-bit time_t PPC */
#endif
   time_t ut_time ;/* time entry was made */
#if !defined(__64BIT__) && defined(__ia64)
        int __time_t_space;             /* for 32vs64-bit time_t IA64 */
#endif
   struct exit_status
     {
       short e_termination ;/* Process termination status */
       short e_exit ;/* Process exit status */
     }
   ut_exit ; /* The exit status of a process
                * marked as DEAD_PROCESS.
                */
   char ut_host[256] ;/* host name */
   int __dbl_word_pad;/* for double word alignment */
   int __reservedA[2];
   int __reservedV[6];
  } ;

/* Definitions for ut_type*/

#defineEMPTY0
#defineRUN_LVL1
#defineBOOT_TIME2
#defineOLD_TIME3
#defineNEW_TIME4
#defineINIT_PROCESS5/* Process spawned by “init” */
#defineLOGIN_PROCESS6/* A “getty” process waiting for login */
#defineUSER_PROCESS7/* A user process */
#defineDEAD_PROCESS8
#defineACCOUNTING9

#defineUTMAXTYPEACCOUNTING/* Largest legal value of ut_type */

/* Special strings or formats used in the “ut_line” field when*/
/* accounting for something other than a process.*/
/* No string for the ut_line field can be more than 63 chars +*/
/* a NULL in length.  */

#define RUNLVL_MSG      “run-level %c”
#defineBOOT_MSG“system boot”
#defineOTIME_MSG“old time”
#defineNTIME_MSG“new time”

#ifdef _THREAD_SAFE
#ifndef _UTMPX_H
#define utmp_data _utmp_data
#endif /*_UTMPX_H*/
```

```
struct _utmp_data {
   int   ut_fd;
   long  loc_utmp;
   struct utmp ubuf;
   char  *name;
};
#define UTMP_DATA_INIT(__s) (__s.ut_fd=-1, __s.name=UTMP_FILE)
#endif /* _THREAD_SAFE */

#ifdef_NO_PROTO
extern void endutent();
extern struct utmp *getutent();
extern struct utmp *getutid();
extern struct utmp *getutline();
extern struct utmp *pututline();
extern void setutent();
extern int utmpname();
extern void updwtmp();

#ifdef _THREAD_SAFE
extern  void    endutent_r();
externintgetutent_r();
externintgetutid_r();
externintgetutline_r();
externintpututline_r();
extern  void    setutent_r();
/* See comments in stdlib.h on _AIX32_THREADS */
#if _AIX32_THREADS
extern  void    utmpname_r();
#else/* POSIX 1003.4a Draft 7 prototype */
extern  int     utmpname_r();
#endif /* _AIX32_THREADS */
#endif /* _THREAD_SAFE */

#else/* _NO_PROTO */
extern void endutent(void);
extern struct utmp *getutent(void);
extern struct utmp *getutid(const struct utmp *);
extern struct utmp *getutline(const struct utmp *);
extern struct utmp *pututline(const struct utmp *);
extern void setutent(void);
extern int  utmpname(char *);
extern void updwtmp(const char *, const struct utmp *);

#ifdef _THREAD_SAFE
externintgetutent_r(struct utmp **utmp, struct _utmp_data *utmp_data);
externintgetutid_r(const struct utmp *utent, struct utmp  **utmp,
            struct _utmp_data *utmp_data);
```

```
externintgetutline_r(const struct utmp *utent, struct utmp **utmp,
             struct _utmp_data *utmp_data);
externintpututline_r(const struct utmp *utent,
              struct _utmp_data *utmp_data);
extern  void    setutent_r(struct _utmp_data *utmp_data);
extern  void    endutent_r(struct _utmp_data *utmp_data);
/* See comments in stdlib.h on _AIX32_THREADS */
#if _AIX32_THREADS
extern  void    utmpname_r(char *newfile, struct _utmp_data *utmp_data);
#else/* POSIX 1003.4a Draft 7 prototype */
extern  int     utmpname_r(char *newfile, struct _utmp_data *utmp_data);
#endif /* _AIX32_THREADS */
#endif /* _THREAD_SAFE */

#endif/* _NO_PROTO */

#endif /* _H_UTMP */
```

The ctmp.h header file format

The ctmp file format

The ctmp file format is defined in the /usr/include/sys/ctmp.h include header file.

```
/* IBM_PROLOG_BEGIN_TAG                                                   */
/* This is an automatically generated prolog.                             */
/*                                                                        */
/* bos530 src/bos/usr/sbin/acct/ctmp.h 1.2.1.2                            */
/*                                                                        */
/* Licensed Materials - Property of IBM                                   */
/*                                                                        */
/* (C) COPYRIGHT International Business Machines Corp. 1985,1993          */
/* All Rights Reserved                                                    */
/*                                                                        */
/* US Government Users Restricted Rights - Use, duplication or            */
/* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.      */
/*                                                                        */
/* IBM_PROLOG_END_TAG                                                     */
/* @(#)20 1.2.1.2  src/bos/usr/sbin/acct/ctmp.h, cmdacct, bos530 11/12/03
18:22:10 */
/*
 * COMPONENT_NAME: (CMDACCT) Command Accounting
 *
 * FUNCTIONS: none
 *
 * ORIGINS: 3,9,27
 *
 * (C) COPYRIGHT International Business Machines Corp. 1985, 1993
```

```
 * All Rights Reserved
 * Licensed Material - Property of IBM
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
 */

/*
 * connect time record (various intermediate files)
 */
#include “userpw.h”

struct ctmp {
   dev_tct_tty; /* major minor */
   uid_tct_uid; /* userid */
   charct_name[8];/* login name */
   longct_con[2];/* connect time (p/np) secs */
   time_tct_start;/* session start time */
};

struct ctmpx {
   dev_tct_tty; /* major minor */
   uid_tct_uid; /* userid */
   charct_name[MAXIMPL_LOGIN_NAME_MAX];/* login name */
   longct_con[2];/* connect time (p/np) secs */
   time_tct_start;/* session start time */
};
```

The accrec file format

The acctrec file format is defined in the /usr/include/sys/accrec.h include header file.

```
/* IBM_PROLOG_BEGIN_TAG                                                 */
/* This is an automatically generated prolog.                           */
/*                                                                      */
/* bos530 src/bos/usr/bin/que/accrec.h 1.2                              */
/*                                                                      */
/* Licensed Materials - Property of IBM                                 */
/*                                                                      */
/* (C) COPYRIGHT International Business Machines Corp. 1989,1993        */
/* All Rights Reserved                                                  */
/*                                                                      */
/* US Government Users Restricted Rights - Use, duplication or          */
/* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.    */
/*                                                                      */
```

```
/* IBM_PROLOG_END_TAG                                                    */
/* @(#)22       1.2  src/bos/usr/bin/que/accrec.h, cmdque, bos530 6/17/93
15:17:17 */
/*
 *   COMPONENT_NAME: CMDQUE
 *
 *   FUNCTIONS: none
 *
 *   ORIGINS: 27
 *
 *
 *   (C) COPYRIGHT International Business Machines Corp. 1989,1993
 *   All Rights Reserved
 *   Licensed Materials - Property of IBM
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
    */

   /* This is the accounting record used for keeping track of
    * how many pages are charged to each user.
    */
   struct acctrec
           {   char from[255];     /* User's name.... e.g. dean@jlkey */
               char acctchar;      /* not used                        */
            long acctdate;      /* date last job made...not used   */
            int  pages;         /* number of pages charged.        */
            int numjobs;        /* number of jobs charged.         */
       } acctrec;
```

The files in the bos.acct package

```
[#][/usr/sbin/acct]> lslpp -f bos.acct
  Fileset               File
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.acct 5.3.0.0      /usr/lib/acct/monacct -> /usr/sbin/acct/monacct
  bos.acct 5.3.0.0      /usr/bin/vmstat
                        /usr/bin/acctras
                        /usr/lib/sa
                        /usr/lib/acct/acctdisk -> /usr/sbin/acct/acctdisk
                        /usr/lib/acct/chargefee -> /usr/sbin/acct/chargefee
                        /usr/sbin/acct
                        /usr/lib/acct/ptelus.awk -> /usr/sbin/acct/ptelus.awk
                        /usr/sbin/acct/acctdusg
                        /usr/sbin/acct/accton
                        /usr/sbin/sar
                        /usr/sbin/acct/turnacct
```

```
                                /usr/sbin/diskusg
                                /usr/bin/sysline
                                /usr/sbin/acct/prdaily
                                /usr/lib/acct/acctprc1 -> /usr/sbin/acct/acctprc1
                                /usr/lib/acct/acctprc2 -> /usr/sbin/acct/acctprc2
                                /usr/lib/acct/holidays -> /etc/acct/holidays
                                /usr/sbin/acct/prctmp
                                /usr/lib/sa/sadc
                                /usr/sbin/acct/startup
                                /usr/lib/acct/prdaily -> /usr/sbin/acct/prdaily
                                /usr/sbin/acct/acctcms
                                /usr/sbin/acct/acctmerg
                                /usr/sbin/acct/remove
                                /usr/sbin/acct/runacct
                                /usr/sbin/acct/acctcom
                                /usr/lib/acct/shutacct -> /usr/sbin/acct/shutacct
                                /usr/lib/acct/nulladm -> /usr/sbin/acct/nulladm
                                /usr/bin/acctcom -> /usr/sbin/acct/acctcom
                                /usr/lib/acct/acctcms -> /usr/sbin/acct/acctcms
                                /usr/lib/acct/fwtmp -> /usr/sbin/acct/fwtmp
                                /usr/bin/vmstat64
                                /usr/sbin/acct/ptecms.awk
                                /usr/sbin/acct/diskusg -> /usr/sbin/diskusg
                                /usr/lib/acct/dodisk -> /usr/sbin/acct/dodisk
                                /usr/sbin/acct/acctwtmp
                                /usr/bin/mpstat
                                /usr/lib/acct/acctcon1 -> /usr/sbin/acct/acctcon1
                                /usr/lib/acct/acctcon2 -> /usr/sbin/acct/acctcon2
                                /usr/samples/aacct/readaacct.c
                                /usr/sbin/projdata
                                /usr/bin/projctl
                                /usr/bin/iostat
                                /usr/sbin/acct/chargefee
                                /usr/bin/acctctl
                                /usr/lib/acct/lastlogin -> /usr/sbin/acct/lastlogin
/usr/bin/timex
                                /usr/lib/acct/ac -> /usr/sbin/acct/ac
                                /usr/sbin/aacct_stat
                                /usr/sbin/acct/acctdisk
                                /usr/sbin/acct/ckpacct
                                /usr/lib/acct/wtmpfix -> /usr/sbin/acct/wtmpfix
                                /usr/sbin/acct/ptelus.awk
                                /usr/sbin/acct/ac
                                /usr/lib/acct/ckpacct -> /usr/sbin/acct/ckpacct
                                /usr/sbin/acct/lastlogin
                                /usr/samples/aacct/Makefile
                                /usr/sbin/acct/acctprc1
                                /usr/sbin/acct/acctprc2
                                /usr/sbin/acct/prtacct
```

```
                          /usr/sbin/acct/shutacct
                          /usr/lib/acct/accton -> /usr/sbin/acct/accton
                          /usr/lib/acct/prtacct -> /usr/sbin/acct/prtacct
                          /usr/lib/acct/startup -> /usr/sbin/acct/startup
                          /usr/sbin/acct/acctcon1
                          /usr/sbin/acct/acctcon2
                          /usr/lib/acct/acctdusg -> /usr/sbin/acct/acctdusg
                          /usr/sbin/acct/dodisk
                          /usr/lib/acct/turnacct -> /usr/sbin/acct/turnacct
                          /usr/sbin/acct/nulladm
                          /usr/samples/aacct
                          /usr/lib/acct/prctmp -> /usr/sbin/acct/prctmp
                          /usr/lib/acct/runacct -> /usr/sbin/acct/runacct
                          /usr/samples/aacct/readaacct
                          /usr/lib/acct/remove -> /usr/sbin/acct/remove
                          /usr/lib/acct/diskusg -> /usr/sbin/diskusg
                          /usr/lib/acct/acctmerg -> /usr/sbin/acct/acctmerg
                          /usr/lib/acct/ptecms.awk -> /usr/sbin/acct/ptecms.awk
                          /usr/lib/sa/sa1
                          /usr/lib/sa/sa2
                          /usr/lib/acct/acctwtmp -> /usr/sbin/acct/acctwtmp
                          /usr/sbin/sa
                          /usr/sbin/admindata
                          /usr/lib/drivers/aacctdd
                          /usr/sbin/acct/fwtmp
                          /usr/bin/lparstat
                          /usr/sbin/acct/monacct
                          /usr/lib/acct
                          /usr/sbin/acct/wtmpfix

Path: /etc/objrepos
bos.acct 5.3.0.0      /var/adm/acct
                          /etc/project/alter/template/alias
                          /etc/project/alter/template/projdef
                          /etc/project/alter/template
                          /etc/project/.active
                          /etc/project/README
                          /etc/project/admin
                          /etc/project/alias
                          /var/adm/sa
                          /etc/acct
                          /etc/acct/holidays
                          /var/aacct
                          /etc/project/alter
                          /etc/project/.config
                          /etc/project/alter/.current -> /etc/project
                          /etc/rc.stopaacct
                          /etc/project/alter/template/admin
                          /etc/rc.startaacct
```

C

Accounting records in Advanced Accounting

Advanced Accounting produces 17 types of accounting records, which are defined in the sys/acct.h file. The following table describes these records.

Accounting record	Description
Pad record (type 0)	This record does not provide any meaningful accounting data. Report and analysis tools should skip this record because it is generated for alignment purposes only.

Accounting record	Description
Process record (type 1)	This record is written when a process exits, when a process is reclassified (setUser ID(), chproj(), exec()), and when the system is reclassified. This record is written by the process interval and contains the following information: ► User ID ► Group ID ► Process ID ► Process flags (exited, core, killed by signal, killed by checkpoint) ► Base command name ► WLM class ► Controlling terminal ► Process start time (in seconds from the EPOCH) ► Process elapsed time in micro seconds ► Combined thread elapsed time in micro seconds ► Process (combined threads) processor time in micro- seconds ► Elapsed page seconds of disk pages ► Elapsed page seconds of real pages ► Elapsed page seconds of virtual memory ► Local logical file I/O (JFS, J2) in bytes ► Other logical file I/O (NFS, DFS™) in bytes ► Local socket I/O (UNIX domain and loopback) in bytes ► Remote socket I/O in bytes The process start time and Process ID can be used to correlate interval records for a particular process. The exit flag can be used to distinguish between interval and exit records.

Accounting record	Description
Aggregated process record (type2)	This record is derived from the process record. A different record is produced for each user by project. This record is produced by the process interval and contains the following information: ► User ID ► Time of first record aggregated (in seconds from the EPOCH) ► Number of processes aggregated ► Aggregate process elapsed time in microseconds ► Aggregate thread elapsed time in microseconds ► Aggregate process (combined threads) processor time in microseconds ► Aggregate elapsed page seconds of disk pages ► Aggregate elapsed page seconds of real pages ► Aggregate elapsed page seconds of virtual memory ► Aggregate local logical file I/O (JFS, J2) in bytes ► Aggregate other logical file I/O (NFS, DFS) in bytes ► Aggregate local socket I/O (UNIX domain and loopback) in bytes ► Aggregate remote socket I/O in bytes

Accounting record	Description
Aggregated application record (type3)	This record is derived from the process record. Records are produced at the user, project, and application level. This record is similar to the aggregated process record, except that the application is named. This record is produced when the process is classified with an application-specific rule, which is supported only through the Admin policy. This record is produced by the process interval and contains the following information: ► User ID ► Time of first record aggregated (in seconds from the EPOCH) ► Inode of the command that generated the project classification ► Device number of the command that generated the project classification ► Number of applications aggregated ► Aggregate process elapsed time in microseconds ► Aggregate thread elapsed time in microseconds ► Aggregate process (combined threads) processor time in microseconds ► Aggregate elapsed page seconds of disk pages ► Aggregate elapsed page seconds of real pages ► Aggregate elapsed page seconds of virtual memory ► Aggregate local logical file I/O (JFS, J2) in bytes ► Aggregate other logical file I/O (NFS, DFS) in bytes ► Aggregate local socket I/O (UNIX-domain and loopback) in bytes ► Aggregate remote socket I/O in bytes

Accounting record	Description
Processor and memory use record (type4)	This record provides information about the use of processors and memory at the system level. It is generated before and after Dynamic Logical Partitioning operations and when the size of the large page pool changes. This record is also generated by the system interval and contains the following information: ► Reason the record was generated ► Number of online logical processors ► Entitled physical processor capacity of the partition ► Total idle time, in milliseconds ► Total I/O wait time, in milliseconds ► Total kernel process time, in milliseconds ► Total user process time, in milliseconds ► Total interrupt time, in milliseconds ► Size of physical memory, in megabytes ► Size of the large page pool, in megabytes ► Large pages in use, in megabytes ► Number of page ins from paging space ► Number of page outs to paging space ► Number of start I/Os ► Number of page steals
Policy record (type 5)	This record is written when a policy file is loaded or unloaded. It is provided for informational purposes only. It contains the following information: ► Type of policy: Admin, User, or Group ► Load or unload
File system activity record (type 6)	This record describes the use of file systems at the system level. A separate record is generated for each mounted file system. This record is produced by the system interval and has the following information: ► Total bytes transferred through read and write ► Total number of read and write requests ► Total number opens ► Total number of creates ► Total number of locks ► File system type ► Device name ► Mount point

Accounting record	Description
Network interface I/O record (type 7)	This record provides information about the use of network interfaces at the system level. It is produced by the system interval and contains the following information: ► Logical name of the network interface ► Number of I/Os ► Total bytes transferred
Disk I/O record (type 8)	This record provides information about the use of disks at the system level. A separate record is written for each logical disk device. This record is produced by the system interval and contains the following information: ► Logical name of the disk ► Total disk transfers ► Total reads ► Total writes ► Block size of the disk transfer
Lost data record (type 9)	This record provides information about accounting records that were deleted because Advanced Accounting did not have the ability to record them. This occurs when all of the accounting data files are full. When the ability to write new accounting records is restored, Advanced Accounting produces the lost data record describing the outage. This record contains the following information: ► Number of lost records ► Number of microseconds of lost processor time associated with process records ► The time that data loss began (in microseconds from EPOCH) ► Number of microseconds of lost processor time associated with third-party kernel extension records
Server VIO record (type 10)	This record is produced in hosting partitions. A separate record is produced for each logical device that is shared with a client partition. The system interval may be used to periodically produce this record, which contains the following information: ► Client partition number ► Server unit ID ► Device logical unit ID (LUN) ► Number of bytes in ► Number of bytes out

Accounting record	Description
Client VIO record (type 11)	This record is produced in client partitions. It describes the use of virtual devices in client partitions. A separate record is recorded for each instance of a virtual device. The system interval may be used to periodically produce this record, which contains the following information: ► Server partition number ► Server unit ID ► Device logical unit ID ► Number of bytes in ► Number of bytes out
Third-party kernel extension common aggregation record (type 12)	This record provides accounting information for the named accounting record. It is derived from aggregated accounting records that are produced by third-party kernel extensions. This record is written to the Advanced Accounting subsystem by the system interval. This record contains the following information: ► Command name of the kernel extension (from u-block) ► Third-party kernel extension transaction ID, in the range of 129 to 256 ► Number of accounting records that have been aggregated ► Resource use, or accumulated processor time, for this class of transactions ► Time of first record aggregated (in seconds from the EPOCH)

Accounting record	Description
ARM application environment record (type 13)	This record describes an application environment instance. It is created from data that is passed to the operating system through the arm_register_application() system call and the arm_start_application() system call. The record is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information: ► Character set in which the data in this record is recorded ► Application environment identifier ► Offset to application name ► Offset to application group ► Offset to application identity properties ► Offset to application context properties The operating system attempts to record the content of the application environment in each accounting data file so that each accounting data file can be post-processed as a stand-alone item. This is designed to eliminate dependency among accounting data files.
ARM transaction environment record (type 14)	This record describes a transaction environment instance. It is created from data that is passed to the operating system through the arm_register_transaction() system call. The record is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information: ► Character set in which the data in this record in recorded ► Transaction environment identifier ► Offset to transaction name ► Offset to application identity properties ► Offset to application context properties (names only) The operating system attempts to record the content of the transaction environment in each accounting data file (not guaranteed), so that each accounting data file can be post-processed as a stand-alone item. This is designed to eliminate dependency among accounting data files.

Accounting record	Description
ARM transaction instance record (type 15)	This record describes an ARM transaction instance. It is created from data that is passed to the operating system through the arm_start_transaction() and the arm_stop_transaction() system calls. It is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information: ► Completion status of the transaction ► Application environment identifier ► Transaction environment identifier ► Offset to user identifier (not User ID) ► Offset to user name (not uname) ► Offset to accounting code ► Response time, in milliseconds ► Queued time, in milliseconds ► Resource use The application and transaction environment identifiers are defined respectively in the application and transaction environment records. These records must be used to associate application names, application groups, transaction names, and properties with the transaction instance.
ARM aggregated transaction instance record (type 16)	This record is produced instead of the ARM transaction instance record (type 15) when aggregation is enabled for ARM transactions. It contains the following information: ► Completion status of the transaction ► Time of first record aggregated (in seconds from EPOCH) ► Application environment identifier ► Transaction environment identifier ► Offset to user identifier (not User ID) ► Offset to user name (not uname) ► Offset to accounting code ► Aggregate response time, in milliseconds ► Aggregate queued time, in milliseconds ► Aggregate resource use

Accounting record	Description
Project definition record (type 17)	This record provides a list of project definitions. It is written when the project definition file is loaded. Multiple records may be needed to record all project definitions. This record is used to provide the full set of project information in each data file, so that data files may be treated as stand-alone entities. This may not be required by the billing application, depending on the nature of the billing application. This feature may be disabled by disabling the project definition accounting record. This record is variable in length and contains the following information: ► Number of projects ► Number of bytes in the project definition area ► Project definition area

Abbreviations and acronyms

ACL	access control list
AIX	Advanced Interactive Executive (advanced IBM UNIX)
API	application programming interface
ARM	Application Response Measurement
ASCII	American Standard Code for Information Interchange
CPU	central processing unit
DAC	discretionary access control
FC	Fibre Channel
FTP	File Transfer Protocol
GID	group identifier
GL	graphical language
IBM	International Business Machines Corporation
ID	identifier
ITSO	International Technical Support Organization
LPAR	logical partition
LVM	Logical Volume Manager
MRTG	Multi Router Traffic Grapher
NFS	Network File System
NIM	Network Installation Management
NIS	Network Information Service
PCL	Printer Control Language
PID	Process Identifier
SCSI	Small Computer System Interface
SMIT	System Management Interface Tool
SQL	Structured Query Language
SRC	System Resource Controller
TCP/IP	Transmission Control Protocol/ Internet Protocol
UID	user identifier
URL	Universal Resource Locator
VIO	Virtual I/O
WebSM	Web-based System Manager
WLM	Workload Manager

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 242. Note that some of the documents referenced here may be available in softcopy only.

- *Advanced POWER Virtualization on IBM @server p5 Servers: Introduction and Basic Configuration*, SG24-7940
- *AIX 5L Workload Manager (WLM),* SG24-5977
- *Auditing and Accounting on AIX*, SG24-6020

Other publications

These publications are also relevant as further information sources:

- *AIX 5L Version 5.3, Commands Reference*, Volume 1, a - c , SC23-4888
- *AIX 5L Version 5.3, Commands Reference*, Volume 3, i - m , SC23-4890
- *AIX 5L Version 5.3, Commands Reference*, Volume 4, n - r , SC23-4891
- *AIX 5L Version 5.3, Commands Reference*, Volume 5, s - u , SC23-4892
- *AIX 5L Version 5.3, Commands Reference*, Volume 6, v -z, SC23-4893
- *AIX 5L Version 5.3, Files reference*, SC23-4895
- *AIX 5L Version 5.3, Installation Guide and Reference*, SC23-4887
- *AIX 5L Version 5.3, Security Guide*, SC23-4907
- *AIX 5L Version 5.3, System Management Guide: Operating System and Devices*, SC23-4910
- *Understanding the Advanced Accounting subsystem*, SC23-4882

Online resources

These Web sites are also relevant as further information sources:

- Transaction accounting, ARM API

  http://www.opengroup.org/tech/management/arm

- University of Arizona Computer Science Department: “Administering the System Accounting Services”

  http://www.cs.arizona.edu/computer.help/policy/DIGITAL_unix/AA-PS2RD-TET1_html/maint13.html

- UCLA Public Domain Software Library for AIX Web site

  http://aixpdslib.seas.ucla.edu

- MRTG Web site

  http://www.mrtg.org

- MRTG PME Web site, maintained at sourceforge.net

  http://mrtg-pme.sourceforge.net/

- Setting up AIX Workload Manager in 30 minutes

  http://www-106.ibm.com/developerworks/eserver/library/es-Practical_WLM.html

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications, and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

- IBM Support and downloads

  ibm.com/support

- IBM Global Services

  ibm.com/services

Index

Symbols

$bin 17
$trail 17
/audit/auditb 20
/audit/bin1 9, 20
/audit/bin2 9, 20
/audit/pick 29
/audit/stream.out 14, 20, 32
/audit/trail 20, 32
/dev/audit 18
/etc/project/projdef 163
/etc/security/audit/config 16
/etc/security/audit/streamcmds 32
/etc/wlm/current/classes 139
/etc/wlm/current/rules 139
/usr/lib/acct 43
/usr/sbin/acct 44, 50
/var/adm/acct 43
/var/adm/acct/fiscal files 123
/var/adm/acct/nite files 122
/var/adm/acct/sum files 123
/var/adm/pacct 147

A

ac command 80, 103, 118
Access Control List (ACL) 34
accounting 2, 41, 125
  commands 117–119
  errors 110
accounting data file 155
accounting package 43
accounting record 160
accounting subsystem 126, 145–148
accprc1 command 105
accprc2 command 105
accrec.h header file 64
acctcms command 82, 84, 118
acctcom command 51, 82–83, 118, 147–148
  fields 84
acctcon1 command 79–80, 118
acctcon2 command 79, 81, 118
acctctl command 152–153
acctdisk command 88, 118
acctdusg command 61, 63, 88, 108–109, 118–119
accterr file 76, 122
acctmerg command 79, 81, 87–88, 118
accton command 54, 118
acctprc1 command 81, 83, 118
acctprc2 command 81, 83, 118
acctwtmp command 56, 59, 81, 118
ACL 34
active file 75, 122
active mode 137
active process 83
admin policy 165
Advanced Accounting ix, 151
aggregation for ARM transaction 161
aggregation for third-party kernel 161
aliases 167
ALL event class 12, 21
alternate admin policy 165, 167
API 148
application tags 127
ARM 159
assignment rules 127, 134–135
attributes 128
audit bin records 19
audit class 7
audit command 17
audit header 6–7
audit limitations 30
audit logger 6, 23
audit off command 25, 27
audit on (panic) command 24
audit query command 25
audit record 6, 14
audit shutdown command 24, 27
audit start command 13, 24
audit tail 6–7
audit trail 2, 8–9, 16
auditable event 2, 6, 32
auditbin command 17, 21
auditcat command 9, 17
auditconv command 17
auditing 2, 6
auditmerge command 17
auditpr command 10, 14, 16, 19, 29, 32, 36–37

auditpr stanza 13
auditselect command 10, 18, 29–30, 32
auditstream command 10, 16, 18, 32
Autostart Advanced Accounting 154

B

backend 12, 32
big numbers 66
billing 2, 126
BIN 8–9, 11, 16, 32
bin 11, 37
BIN collection method 21
BIN mode 8, 33
bin1 attribute 23
bin1 file 17
bin2 attribute 23
bin2 file 17
binary files
  displaying 51
bincmds file 16
binsize attribute 12, 23
bos.acct 43
bottlenecks 42
bytethreshold attribute 23

C

cat command 35
chargefee command 42, 65, 88–89, 118, 120, 145
charging 126
chdev 49
ckpacct command 44, 48, 51, 90, 117
class tier 128, 149
classes 12, 126
CLEANUP state 74
cmds attribute 12, 23
cms file 76, 122–124
CMS state 74
cmsi file 123
cmsn file 77
cmsprev file 76, 123
commands
  accounting 117–119
  called by cron 117
  interactive (shell) 118
  system startup 118
config file classes 12
configuration examples 31
configuring and setting up auditing 7
connect time accounting 54
CONNECT1 state 73
CONNECT2 state 73
Connect-Time Accounting 52
CPU usage 105
CPU utilization 92
cron class 12
cron daemon 89
ctacct file 124
ctmp file 76, 122, 124

D

DAC 34
dacct disk accounting data file 61
dacct file 61, 76, 87, 122
daily accounting procedures 48
daily command summary 86
  fields 86
daily reports 42, 66
data aggregation 160
data collection 21, 37
data files 119
date change 59
date change errors 114
date command 54
dayacct file 124
daycms file 76, 122–124
daytacct file 76, 122
Default superclass 128
digest command 47
disk accounting data 60
disk space consideration 29
DISK state 73
disk utilization report 92
disk-accounting data 60
diskdiag file 119
Disk-Usage Accounting 52
disk-usage report 87
diskusg command 61, 63, 87, 106, 108, 118
display connect time 103
dodisk command 48, 60, 63, 87, 109, 117, 122
dtmp file 61, 87, 119

E

e-mail notification 157
error message 39
event auditing 7
event file 13

event specific information 7
eventthreshold attribute 23

F
FAIL 34
FAIL_ACCESS 34
FAIL_AUTH 34
FAIL_DAC 34
FAIL_PRIV 34
failedlogin file 55, 80
Fee Accounting 53
fee file 89, 120
fee report 88
FEES state 73
file formats 124
files class 12
filesystems file 47
fiscal directory files 123
fiscal period 77
fiscrpti file 123
fiscrptn file 77
freespace attribute 24
fwtmp command 51, 81, 112, 118

G
general class 12, 39
group policy 168

H
headers 99
hierarchy 127
hog factor 82, 87
holidays file 45

I
incorrect file permissions 112
Inheritance 133
Inheritence 128
init command 56
init level changes 59
init program 58
Interval accounting 157
interval accounting ix, 151–158
iostat command 91
isvmon
  tier report 143

K
kernel audit trail 6
kernel class 12

L
last command 81, 118
last login report 87
lastcomm command 83, 85, 118
lastdate file 76, 122
lastlog file 55, 80
lastlogin command 81, 119, 123
lineuse file 76, 122
Localshm 129, 133
lock files 71
lock1 file 122
log file 76, 122
login command 55–56
  detecting unsuccessful 6
loginlog file 123
logmmdd file 122
long login user names 48–49
LPAR 158
ls command 28
lsattr command 49
lslpp 43
lvm class 12

M
mail class 12
MAILCOM 44
manually load a project 171
MERGE state 73
MERGEACCT state 74
mistakes and errors 39
monacct command 48, 77–79, 117
monthly accounting summary 48
monthly reports 77

N
NIS 108
nite directory files 75–76
nitex 48
nmon 141, 144
non-prime time 42
nulladm command 45, 47, 54, 112, 119

O

object auditing 7–8
objects class 12
objects file 15
oconfig file 13
OK 34
out-of-date holidays file 113
output binary files 124
owtmp file 76

P

pac command 51–52, 88, 109, 119
pacct file 45, 82–83, 120, 124
paccti file 82
passive mode 137
PASSWORD_Change 35
PATH 44
performance 30
Performance Toolbox 141, 145
pioformat program 64
policies 164–171
  loading 169–170
  querying status 171
  unloading 170
prctmp command 81, 119
prdaily command 119, 122–123
predefined superclasses 128
prime time 42, 45
printer accounting records 63
Printer-Usage Accounting 52
printer-usage report 88
printf 14
process 131
  CPU-intensive 131
  memory-intensive 131–132
process aggregation 161
Process and Command Accounting 52
process interval 158, 160
process records 157
PROCESS state 73
prog1 131–132
prog2 131–132
prog3 131–132
projctl 155
project
  adding 163
  load 163
  manual loading 171–173
  query 163
  unload 163
project-level data aggregation 161
projects 163–164
prtacct command 51, 83, 89, 105–106, 119
ps command 83, 100
PTX 145

Q

qacct access file error 113
qacct file 88, 120
qconfig file 47
qdaemon 47
QUEUEACCT state 73
quick security check 27

R

read the system audit trail file 19
readaacct 155
real-time alert 6
real-time modification monitor 31
reboot command 54
reboots file 76, 122
recommendations for auditing 26
Redbooks Web site 242
  Contact us xi
remove command 49, 53, 119
reporting and analysis 173–177
resource allocation 126
resource control 148
resource entitlement 127
resource limits 127
Resource set 129
resource usage 126
resource utilization 125
rprmmdd file 76
rprtmmdd file 123
rule
  creating for WLM class 134
runacct command 44, 48, 71, 75, 86, 117, 120, 122
runacct errors 113

S

sa command 119
sa1 command 117, 119
sa1 data collector 95
sa2 command 117, 119

sa2 data collector 96
sadc command 119
sadc data collector 95
sar command 93, 119
scientific notation 66
security 2
server consolidation 126
setting up auditing 11
Shared superclass 128
Shares 129
shut down the audit 26
shutacct command 53–54, 118
shutdown command 51, 54, 56, 59
SMIT 130
Spaccti file 82, 120, 124
SRC class 12
start stanza 11
start the audit process 25
starting and stopping auditing 24
startup command 46, 48, 53–54, 118
statefile file 71, 76, 122
stopping the accounting 49
STREAM collection method 22
STREAM mode 8–10, 14, 16, 32, 35–37
stream stanza 12, 16, 33
stream.out 30, 40
streamcmds file 16, 36
su command
  detecting unsuccessful 6
subclasses 127
sum directory files 76, 123
sum/tacct file 124
summary files 66, 120
sumx 48
superclass 127, 139
svipc class 12
svmon 141, 143
  class report 143
system activity 89
system administrators 2
system interval 160
system interval accounting 158
System superclass 128
system1 17
system2 17
system-level data aggregation 160

T

tacct errors 110
tacct file 76, 123
tacctmmdd file 76, 123
tacctn file 77
tacctprev file 76, 123
tail 14–15, 32, 35
tcpip class 12
telnet command 56
terminated process 83
time command 100, 119
timex command 101, 119
tmpwtmp file 122
topas 141–142
traditional accounting 151
trail attribute 23
trail file 30
transaction accounting ix, 151, 159
trend recordings 145
ttymon 56
turnacct command 54, 119

U

Unclassified superclass 128
unknown class 146
Unmanaged superclass 128
uptime 126
user and group policy 168
user policy 168
user stanza 12
USER_SU 35
USEREXIT state 74
usr/lib/acct directory 50
utmp file 55, 80, 124
utmp.h header file 55

V

vmstat command 97
vmstat headers 99

W

watch command 27
WebSM 130
weekly reports 42
who command 81, 103
WLM 3, 125–126, 130, 132, 158
  API 127, 148

passive mode 130
setting limits 149
shares 149
structure 129
tiers 149
WLM accounting 145–148
WLM class 125, 130, 132–133
process accounting for 146
WLM class rule 134
WLM configuration 130
wlmmon 145
detailed report 145
snapshot report 145
tabulation report 145
wlmperf 145
wlmstat 141–142
workload 126
Workload Manager
See WLM
wtmp errors 111
wtmp file 45, 55, 80, 120, 123–124
wtmperrmmdd file 123
wtmperror file 76, 122
wtmpfix command 76, 81, 119
WTMPFIX state 72

Accounting and Auditing on AIX 5L

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Accounting and Auditing on AIX 5L

A comprehensive guide to setting up accounting and auditing on AIX 5L

Advanced Accounting and Workload Manager integration

Resource planning and billing techniques

This IBM Redbook is a comprehensive guide to setting up, maintaining, and troubleshooting the advanced auditing and accounting features on AIX 5L systems. It goes through the steps to develop, monitor, troubleshoot, and optimize best practices for auditing and accounting in your environment.

We provide an overview of what auditing and accounting can do for you, how to set up an auditing system, procedures for creating the right accounting system for your environment, and a summary of available third-party accounting systems that can plug into the AIX suite. Learn to decide how much accounting and auditing you need to do on your system and how to size subsystems to handle your requirements, and find a list of rules to help prevent and fix common mistakes.

We also present the new AIX 5L V5.3 feature Advanced Accounting, which is based on mainframe technology such as interval accounting and transaction accounting. Advanced Accounting supports LPAR (logical partitioning), WLM (Workload Management), and Micro-Partitioning.

This book is useful for system administrators, system security officers, companies needing to bill clients for system resource use, and any others looking for a flexible system to monitor system resources.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6396-00 ISBN 0738491705