Auditing and Accounting on AIX

Comprehensive guide to auditing and accounting your AIX system

Step-by-step instructions on auditing your system

Find the most effective way to use accounting to track system resources

Laurent Vanel,
Rosabelle Zapata-Balingit,
Gonzalo R. Archondo-Callao

ibm.com/redbooks

Redbooks

SG24-6020-00

International Technical Support Organization

Auditing and Accounting on AIX

October 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 157.

First Edition (October 2000**)**

This edition applies to AIX Version 4.3 (5765-C34) and subsequent releases running on an RS/6000 server.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures

Tables

Preface

Auditing and Accounting on AIX is your comprehensive guide to setting up, maintaining, and troubleshooting the advanced auditing and accounting features on your AIX systems. Generously illustrated instructions will guide 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 will plug into the AIX suite. A chapter specific to SP solutions is provided.

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 a list of rules of thumb 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.

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.

Laurent Vanel is an AIX and RS/6000 specialist at the International Technical Support Organization, Austin Center. Before joining the ITSO three years ago, Laurent Vanel was working in the French RS/6000 Technical Center in Paris, where he conducted benchmarks and presentations for AIX and RS/6000 solutions.

Rosabelle Zapata-Balingit is an AIX IT specialist in the Philippines. She holds a Bachelor of Science degree in Computer Engineering from Adamson University, Manila. She joined IBM in 1996 as an RS/6000 Systems Service Representative. She has seven years of experience in AIX. Her areas of expertise include AIX, HACMP, and SP.

Gonzalo R. Archondo-Callao is a systems administrator and manager of the High-Performance Computing Group at the Computing Center of the Federal University of Rio de Janeiro (NCE-UFRJ) in Brazil. He also teaches Operating

Systems classes at UFRJ. He has 15 years of experience with UNIX systems and has been working with the RS/6000 SP and AIX since 1996. His areas of expertise include UNIX systems, Windows NT, TCP/IP, and network security. He holds an M.Sc. degree in computer science from the University of California, Los Angeles.

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

Troy Bollinger
IBM Austin

Vani Ramagiri
IBM Austin

Scott Vetter
IBM Austin

Wade Wallace
International Technical Support Organization, Austin Center

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 181 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introduction

This first chapter introduces the definitions of accounting and auditing. It also gives a brief refresher on some elementary commands that you might want to run before setting up either accounting or auditing.

This book is not about performance troubleshooting. If you are interested in this subject, we recommend you read *Understanding IBM RS/6000 Performance and Sizing,* SG24-4810.

1.1 Definitions

Let's start with the definitions of the accounting and auditing utilities.

1.1.1 Auditing

The auditing subsystem provides the means to record security-related information and to alert system administrators of potential and actual violations of the system security policy. The information collected by auditing includes: the name of the auditable event, the status (success or failure) of the event, and any additional event-specific information related to security auditing.

1.1.2 Accounting

The accounting system utility allows you to collect and report on individual and group use of various system resources.

This accounting information can be used to bill users for the system resources they utilize, and to monitor selected aspects of the system's operation. To assist with billing, the accounting system provides the resource-usage totals defined by members of the adm group, and, if the `chargefee` command is included, factors in the billing fee.

The accounting system also provides data to assess the adequacy of current resource assignments, set resource limits and quotas, forecast future needs, and order supplies for printers and other devices.

The following information 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

1.2 Do you really need the full report?

If your problem is not permanent, and you just want to know at one point what is going on your system, you do not need to set up and start the auditing or accounting subsystems. You might want to instead consider running some elementary commands first, such as `ps`, `sar`, or `tprof`.

1.2.1 The ps command

The `ps` command writes the current status of active processes and (if the -m flag is given) associated kernel threads to standard output. Note that while the -m flag displays threads associated with processes using extra lines, you must use the -o flag with the THREAD field specifier to display extra thread-related columns.

Without flags, the `ps` command displays information about the current workstation. The -f, -o, l, -l, s, u, and v flags only determine how much information is provided about a process; they do not determine which processes are listed. The l, s, u, and v flags are mutually exclusive.

With the -o flag, the `ps` command examines memory or the paging area and determines what the command name and parameters were when the process was created. If the `ps` command cannot find this information, the command name stored in the kernel is displayed in square brackets.

1.2.2 sar command

The `sar` command writes to standard output the contents of selected cumulative activity counters in the operating system. The accounting system, based on the values in the Number and Interval parameters, writes information the specified number of times spaced at the specified intervals in seconds. The default sampling interval for the Number parameter is 1 second. The collected data can also be saved in the file specified by the -o File flag.

The `sar` command also extracts and writes to standard output records previously saved in a file. This file can be either the one specified by the -f flag or, by default, the standard system activity daily data file (the /var/adm/sa/sadd file), where the dd parameter indicates the current day.

Without the -P flag, the `sar` command reports system-wide (global among all processors) statistics, which are calculated as averages for values expressed as percentages, and as sums otherwise. If the -P flag is given, the `sar` command reports activity which relates to the specified processor or processors. If -P ALL is given, the `sar` command reports statistics for each individual processor, followed by system-wide statistics.

You can select information about specific system activities using flags. Not specifying any flags selects only system unit activity. Specifying the -A flag selects all activities.

The default version of the `sar` command (CPU utilization report) might be one of the first facilities the user runs to begin system activity investigation, because it monitors major system resources. If CPU utilization is near 100 percent (user + system), the workload sampled is CPU-bound. If a considerable percentage of time is spent in I/O wait, it implies that CPU execution is blocked waiting for disk I/O. The I/O may be required file accesses or it may be I/O associated with paging due to a lack of sufficient memory.

1.2.3 tprof command

The `tprof` command reports CPU usage for individual programs and the system as a whole. This command is a useful tool for anyone with a C or FORTRAN program that might be CPU-bound, and who wants to know which sections of this program are using the CPU the most. The `tprof` command also reports the fraction of time the CPU is idle. These reports can be useful in determining CPU usage (in a global sense).

The `tprof` command specifies the user program to be profiled, executes the user program, and then produces a set of files containing reports. The user specifies the name of the program to be profiled, or alternatively, the name of the program to be profiled and a command line to be executed. Both the Program and Command variables must be executable.

In the AIX operating system, an interrupt occurs periodically to allow a "housekeeping" kernel routine to run. This housekeeping occurs 100 times per second. When the `tprof` command is invoked, the housekeeping kernel routine records the process ID and the address of the instruction executing when the interrupt occurred. With both the instruction address and process ID, the tprof analysis routines can charge CPU time to processes and threads, to subprograms, and even to source lines of programs. Charging CPU time to source program lines is called microprofiling.

More information on these commands are available from the AIX base documentation.

Chapter 2. Auditing on AIX

An audit is defined as an examination of a group, individual account, or activity. Thus, the auditing subsystem provides a means of tracing and recording what is happening on your system.

By default, auditing is not activated in AIX. When you start the audit subsystem, it gathers information depending on your configuration file. It may be unnecessary for you to start auditing if you just let the files sit in your busy system. What is important is for you to be able to interpret an auditing record. Depending on your environment, it may or may not be necessary for auditing to run every time. It is a decision you have to make.

2.1 Auditing concepts

This section will briefly describe how auditing works, from reading the configuration file to recording audit information.

2.1.1 General

When you start the auditing process, a configuration file is read. This file contains information, such as mode, classes, events, objects, and users.

Mode: This message tells you the type of data collection you want to use. The type can be binary mode, which we will cover in Section 2.1.2.1, “BIN mode” on page 7, and/or stream mode, which we will cover in detail in Section 2.1.2.2, “STREAM mode” on page 9.

Binary mode is useful when you plan to store records on a long term basis.

Stream mode is useful when you want to do immediate processing that reads data as it is processed.

You can choose BIN mode, STREAM mode, or you can choose both at the same time.

Events: Events are system-defined activity. Here are two examples:

- The USER_SU event gives you information about whether a user tries to su to another user, and the PASSWORD_Change event will give you information if a password has been changed. Both of these events can be grouped in a class called general.
- The CRON_Start event gives you information about whether a cron job has started, and the CRON_Finish event will give you

information about whether a cron job has just finished running. Both of these events can be grouped in a class called cron.

Classes: Classes define groups of events. You can have one or more events in a class. For example, consider an event called USER_SU, which checks if a user does an su to another user. There is also an event called PASSWORD_Change, which checks if there is a process that changes the password of a user. Since both events are usually done in the system, both events can be grouped in a class called general. Class names are arbitrary, and you can define any class name for certain group of events.

Objects: When one speaks of auditing objects, this means files; so, auditing objects means auditing files. Read, write, and execute of a file can be audited though audit objects.

Users: User enables you to define what class you want to audit for a specific user. You can audit one or more classes per user. For example, you can audit user joe for every general and cron group of events while you only audit the general class for user bob.

After every event or objects are triggered, an audit record is generated. This is the most exciting part of the story. After gathering a handful of information, you now have a chance to interpret and make use of what audit record you have. The name of the file to which audit records are written depends on the audit selection mode. Figure 1 on page 7 gives you an overall overview of how auditing works.

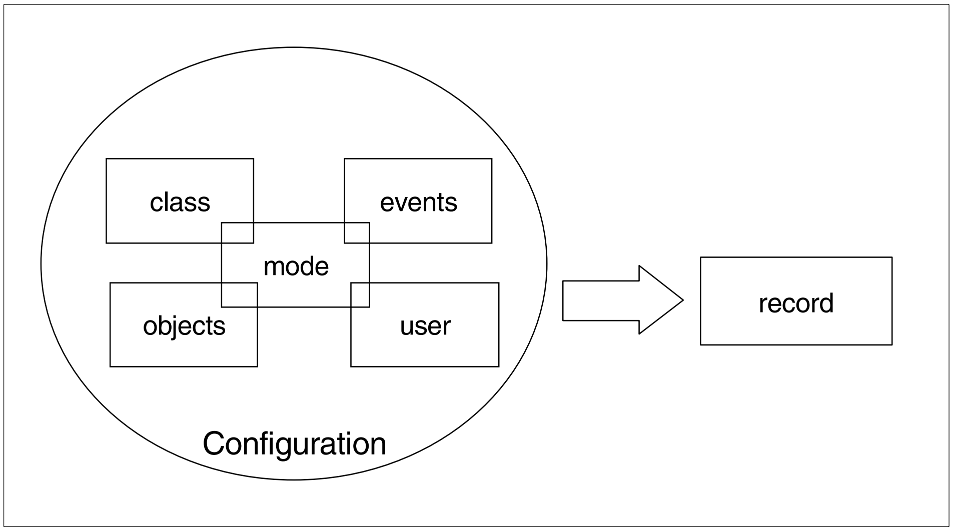


Figure 1. General overview

2.1.2 Data collection method

There are two modes of operation for auditing: BIN and STREAM. The type of data collection method depends on how you will use the data. If you plan to store them on a long-term basis, select BIN mode. If you want to read the data as it is collected, choose STREAM mode. If you want long-term storage and immediate processing, select both.

2.1.2.1 BIN mode

BIN mode is for binary data collection. Figure 2 on page 8 shows bin mode operation.

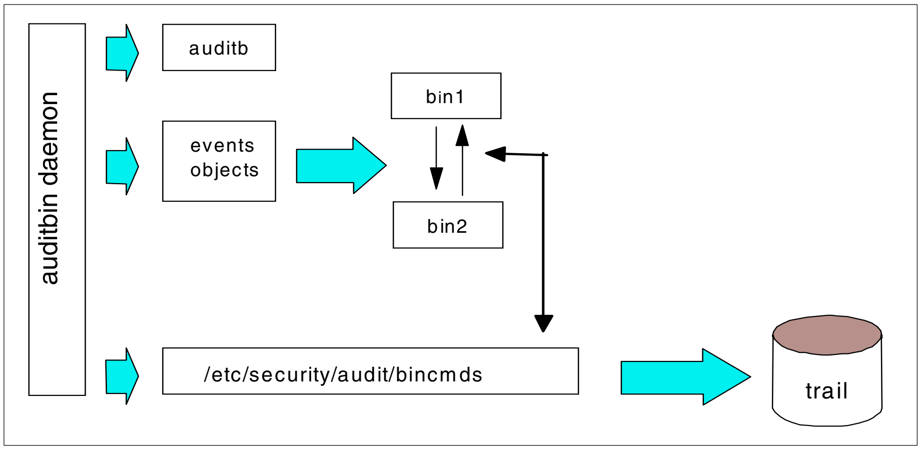


Figure 2. Data collection in BIN mode

Once you start the audit process in binary mode, it executes the file /usr/sbin/auditbin. This creates the auditbin daemon, which manages binary audit information, and creates an active indicator that BIN auditing is running, which is an auditb file of zero length. The auditbin daemon also manages bin1 and bin2, temporary bin files that alternately collect audit event data.

As audit events and objects occurs, the kernel writes a record to a bin file. First it writes to /audit/bin1; if bin1 gets full, the kernel goes to /audit/bin2. When /audit/bin2 gets full, the kernel goes back to /audit/bin1. The size of the bin file is determined by the binsize parameter in /etc/security/audit/config (in bytes). When a bin file is full, the auditbin daemon reads the /etc/security/audit/bincmds file. Each line of this file contains one or more commands with input and output that can be piped together or redirected. The auditbin daemon searches each command for the $bin string and the $trail string, and substitutes the path names of the current bin file and the system trail file.

The auditbin daemon ensures that each command encounters each bin at least once, but does not synchronize access to the bins. When all commands have run, the bin file is ready to collect more audit records.

You can also suspend BIN auditing at a given time and resume it afterwards. Once you resume auditing, the auditbin daemon continues writing to the bin file used before suspending it.

The accumulated data written into /audit/trail must be processed by the auditpr command to make it readable.

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

2.1.2.2 STREAM mode

The STREAM mode of auditing allows you to read the audit record as it is processed. Unlike BIN mode, which is used to keep records on a long-term period, this mode zeroes out the stream.out file as the audit is started by the `audit start` command. Figure 3 on page 9 shows what happens from the time the audit command is started to the time data is recorded.

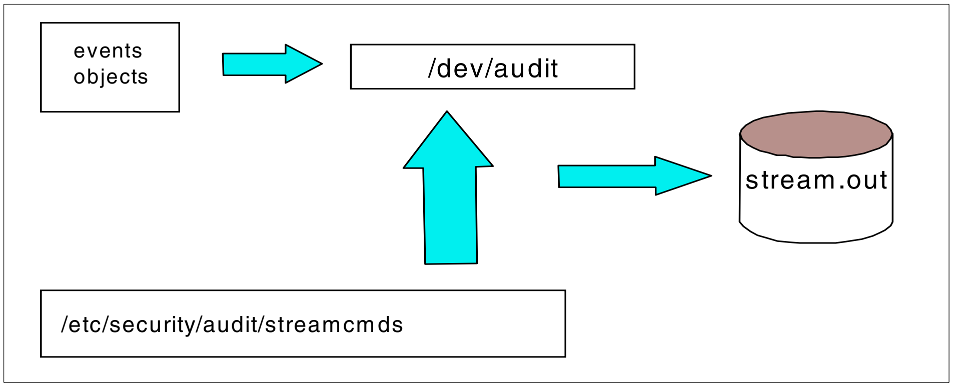


Figure 3. Data collection in STREAM mode

As audit events and objects occurs, data is written to /dev/audit, which is the audit device. The auditstream command in the /etc/security/audit/streamcmds file reads audit records from the audit device, and writes the record to the standard output in binary format. There is also an auditpr command in the same file that is used to format the output and writes to the file /audit/stream.out. In this mode, data is being processed as it is collected.

The STREAM mode writes audit records in a circular buffer in memory and zeroes out the audit record (which is stream.out) as you start auditing.

You can continuously view the record from stream.out with the following command:

```
#tail -f /audit/stream.out
```

You can also temporarily suspend STREAM auditing and resume it afterwards.

Once you start auditing, an audit directory is automatically created for you. If, by any chance, this directory gets deleted, it will be created after the `audit start` command. If there is an ordinary file called audit, you must delete or rename it, since no two files can exist in the same location; otherwise, audit start will fail. Since audit records can produce large amounts of data, and since the audit directory is created in the root (/) filesystem, it is a good idea for you to create a separate file system for audit. There is a good reason to have a separate file system; if you do not monitor the audit record file while it is in the root system, it will consume all the resources of the root file system. Note that the size of the audit file system depends on the amount of data you have.

To create an audit file system you can use this command:

```
#crfs -v jfs -g {volume group name} -m /audit -A yes -a size=8192
```

2.1.3 Events and objects

Auditing events are generally defined at a system call level. A single operation of a command, such as `ls`, will record a log similar to Table 1.

Table 1. Audit record generated by the ls command using event auditing

Event	Login	Status	Date/Time	Command
PROC_Create	root	OK	Fri Jun 09 11:02:41 2000	ksh
FILE_Close	root	OK	Fri Jun 09 11:02:41 2000	ksh
FILE_Open	root	OK	Fri Jun 09 11:02:41 2000	ksh
FILE_Read	root	OK	Fri Jun 09 11:02:41 2000	ksh
FILE_Close	root	OK	Fri Jun 09 11:02:41 2000	ksh
PROC_Execute	root	OK	Fri Jun 09 11:02:41 2000	ls
FILE_Open	root	OK	Fri Jun 09 11:02:41 2000	ls
FILE_Close	root	OK	Fri Jun 09 11:02:41 2000	ls
FILE_Write	root	OK	Fri Jun 09 11:02:41 2000	ls
FILE_Close	root	OK	Fri Jun 09 11:02:41 2000	ls
PROC_Delete	root	OK	Fri Jun 09 11:02:41 2000	ls

You can also use the `watch` command to observe a program. This command observes all the processes that are created while the program runs, including any child process. The `watch` command continues until all processes exit, including the process it created, in order to observe all the events that occur.

The `watch ls` command will give you an output similar to the next two displays:

```
#watch ls - display 1 of 2

filea
fileb
filec
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
AUD_Proc        root     OK          Wed Jun 21 18:09:05 2000 watch
   pid: 0 cmd: 4
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
PROC_SetUserIDs root     OK          Wed Jun 21 18:09:05 2000 watch
   effect: 0, real: 0, saved: -1, login: -1
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
TCB_Exec        root     OK          Wed Jun 21 18:09:05 2000 watch
   filename: /usr/bin/ls
***** WATCH *****
event           login    status      time                     command

--------------- -------- ----------- ------------------------ ------------------
PROC_Execute    root     OK          Wed Jun 21 18:09:05 2000 ls
   euid: 0 egid: 0 epriv: ffffffff:ffffffff name /usr/bin/ls
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
PROC_Load       root     OK          Wed Jun 21 18:09:05 2000 ls
   file: /usr/lib/nls/loc/en_US
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
PROC_LoadMember root     OK          Wed Jun 21 18:09:05 2000 ls
   file: /usr/lib/libi18n.a, member: shr.o
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
FILE_Accessx    root     OK          Wed Jun 21 18:09:05 2000 ls
   mode: 0, who: 1, path: /usr/lib/nls/msg/en_US/ls.cat
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
FILE_Stat       root     OK          Wed Jun 21 18:09:05 2000 ls
   cmd: 9 filename: .
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
FILE_Stat       root     OK          Wed Jun 21 18:09:05 2000 ls
   cmd: 0 filename: .
```

```
#watch ls - display 2 of 2

**** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
FILE_Open       root     OK          Wed Jun 21 18:09:05 2000 ls
    flags: 0 mode: 0 fd: 4 filename .
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
FILE_Close      root     OK          Wed Jun 21 18:09:05 2000 ls
    file descriptor = 4
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
FILE_Write      root     OK          Wed Jun 21 18:09:05 2000 ls
    file descriptor = 1
***** WATCH *****
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
    FILE_Close      root     OK          Wed Jun 21 18:09:05 2000 ls
file descriptor = 1
** child process exiting:  18578
** all processes have exited
```

For the `watch` command to work, the auditing subsystem must not be enabled.

Auditing all possible events can produce a large amount of data. Imagine if you audited everything; you would have tons of information with you! By using audit control, you can select the events to be recorded by customizing the configuration file.

Event auditing is ALWAYS associated with a user ID. For example, you can audit user joe for general class, and you can audit user bob for both general and cron classes. It may not be necessary for you to audit all users; not all user names have audit events.

Auditing objects refers to individual files that will be monitored. Objects are NOT associated with user IDs. Audit records are generated whenever an audit object is referenced by ANY user, including root. You do not need to define any user for object auditing.

To customize audit objects, refer to the file /etc/security/audit/objects. This contains files that record information when there is a read, write, or execute operation.

2.1.4 Audit commands

The `audit` command controls system auditing. It can be invoked to start, shutdown, suspend, resume, and query auditing. There are five parameters for the audit command:

`audit start` This command is used to activate system auditing. This creates a process called auditbin, and an auditb file in the audit directory, which is used for BIN mode. It also creates a process called auditstream, and sets the stream.out file to zero length, which is used for STREAM mode.

`audit shutdown` This command resets the audit subsystem, processes final BIN records (appends whatever is in the temporary bin file to the record trail file), and removes the /audit/auditb file. This is used as an active indicator by the audit BIN module.

`audit off` This command temporarily suspends auditing.

`audit on` This command resumes auditing after the `audit off`. command. This is NOT a substitute for the `audit start` command.

`audit query` This command displays the status of the audit subsystem. The Parent ID (PID) of the BIN process refers to auditbin.

To start auditing, you have to use the `audit start` command, NOT the `audit on` command . The `audit on` command, without the `audit start` command, gives you a zero return value. This means that the command ran successfully but will not record audit information. This is true for both data collection methods. If the subsystem is confused, or you got confused, you can do either of the following:

- If you used BIN mode, run the `audit shutdown` command to write whatever is in the temporary bin file to the trail file; then, issue the `audit start` command.
- If you used STREAM mode, use the `audit shutdown` command, and copy stream.out to your directory. The reason for this action is because when you start auditing, it will set stream.out to zero, and you might want to save the data for your own record. After this, you can issue the `audit start` command.
- Issue the `audit shutdown` command, and reset everything by deleting all the files in the audit directory (/audit). Do not forget to save all files that you might use later, and then issue the `audit start` command.

Auditing can be started automatically at system startup. You can add the following line in the /etc/rc file, before the line `dspmsg rc.cat 5 'Multi-user initialization completed'`

```
/usr/sbin/audit start
```

or you can add this line in /etc/inittab:

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

To stop auditing properly, add the following line to/usr/sbin/shutdown:

```
/usr/sbin/audit shutdown
```

> **Note**
>
> If you do not stop auditing properly and you reboot the system, the auditb file will not be deleted. In this case, after the reboot, the auditb file can become a false indicator that BIN auditing is running.

2.2 Configuration files

All auditing configuration files are located in the /etc/security/audit directory. This is part of the base operating system run time environment security fileset(bos.rte.security). By default, auditing is not activated in AIX.

There are six ASCII files in this directory: config, oconfig, events, objects, bincmds, and streamcmds.

2.2.1 The config file

The config file contains audit system configuration information. It contains five major stanzas. A description of each stanza follows.

- Start - This tells you the type of data collection method you want to use: BIN or STREAM. To turn on BIN auditing, specify `on` after the line binmode; otherwise, specify `off` after the same line. For stream mode, use the streammode parameter, and do the same as in binmode. You can turn on both methods at the same time.

  The next display shows an example of the config file. The start stanza is highlighted.

```
config file display 1 of 2
#more /etc/security/audit/config

start:
    binmode = on
    streammode = off

bin:
    trail = /audit/trail
    bin1 = /audit/bin1
    bin2 = /audit/bin2
    binsize = 10240
    cmds = /etc/security/audit/bincmds

stream:
    cmds = /etc/security/audit/streamcmds
```

- BIN - This defines the binary mode files. This gives the location of the temporary files, such as bin1 and bin2. This also gives the location of your audit record trail file and the pathname of the `backend` program command. It also includes the binsize parameter value, which indicates the size of the temporary bin file in bytes, before it switches to the other bin file. The cmds parameter gives the full pathname of the audit backend program, which is called by the auditbin process.

  The next display shows an example of the config file. The binary stanza is highlighted.

```
config file display 1 of 2
#more /etc/security/audit/config

start:
    binmode = on
    streammode = off

bin:
    trail = /audit/trail
    bin1 = /audit/bin1
    bin2 = /audit/bin2
    binsize = 10240
    cmds = /etc/security/audit/bincmds

stream:
    cmds = /etc/security/audit/streamcmds
```

- STREAM - This stanza contains attributes that the `audit start` command uses to set up initial stream mode auditing. The cmds parameter gives the full pathname of the file, which contains commands executed during initialization of the audit system.

The next display shows an example of the config file. The stream stanza is highlighted.

```
config file display 1 of 2
#more /etc/security/audit/config

start:
    binmode = on
    streammode = off

bin:
    trail = /audit/trail
    bin1 = /audit/bin1
    bin2 = /audit/bin2
    binsize = 10240
    cmds = /etc/security/audit/bincmds

stream:
    cmds = /etc/security/audit/streamcmds
```

- Classes - This stanza defines sets of audit events. By default, the following classes are defined:

  general Refers to general commands, such as user su and password change.

  objects Refers to files. Read or write from /etc/security/passwd. Writes to other security files, such as environ, group, limits, login.cfg, user, and config file.

  SRC Refers to the system resource controller(SRC) activity, such as the start and stop of SRC. This also includes adding, changing, and deleting a subsystem or subserver.

  kernel Refers to kernel related activities.

  files Filesystem related events.This includes system calls, such as: read, write, open, close, link, unlink, rename, change ownership, change mode, and so forth.

  svipc System V Inter Process Communication related events. This includes shared memory, semaphores, system message exchange, and so forth.

  mail Refers to mail exchange. This includes mail-related activities, such as receive, write, and send.

  cron Refers to cron related activities, such as start, stop, add, and delete.

  tcpip Refers to tcpip user level, such as config, route, connect, access, data in, data out, and so forth. It also includes tcpip kernel level, such as tcp socket, socketpair, close, listen bind, connect, send, receive, and so forth.

lvm Refers to the logical volume manager, such as add, delete, extend, reduce, setup, quorum, create volume group, delete volume group, varyoffvg, varyonvg, and so forth.

Only 32 audit classes are supported. One class is implicitly defined by the system to include all audit events (ALL). You should not attempt to define more than 31 audit classes.

The next display shows an example of the config file. The class stanza is highlighted.

```
config file display 2 of 2
#more /etc/security/audit/config

classes:
    general = USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,......
    objects = S_ENVIRON_WRITE,S_GROUP_WRITE,S_LILITS_WRITE,S_LOGIN_WRITE,......
    SRC = SRC_Start,SRC_Stop,SRC_Addssys,SRC_Chssys,SRC_Delssys,SRC_Addserver,......
    kernel = PROC_Create,PROC_Delete,PROC_Execute,PROC_RealUID,PROC_AuditID,......
    files = FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_Link,FILE_Unlink,......
    svipc = MSG_Create,MSG_Read,MSG_Write,MSG_Delete,MSG_Owner,MSG_Mode,......
    mail = SENDMAIL_Config,SENDMAIL_ToFile
    cron = AT_JobAdd,AT_JobRemove,CRON_JobAdd,CRON_JobRemove,CRON_Start,CRON_Finish
    tcpip = TCPIP_config,TCPIP_host_id,TCPIP_route,TCPIP_connect,......
    lvm = LVM_AddLV,LVM_KDeleteLV,LVM_ExtendLV<LVM_ReduceLV,......

users:
    root = general
    joe = general,files
    bob = files,cron
```

- Users - This stanza defines, for a given user, the audit class to be audited. Each user name should be defined in the system, and each audit class should be defined in the config file classes stanza. By default, only the root user with general class is defined.

  The next display shows an example of the config file. The users stanza is highlighted.

```
config file display 2 of 2

classes:
    general = USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,......
    objects = S_ENVIRON_WRITE,S_GROUP_WRITE,S_LILITS_WRITE,S_LOGIN_WRITE,......
    SRC = SRC_Start,SRC_Stop,SRC_Addssys,SRC_Chssys,SRC_Delssys,SRC_Addserver,......
    kernel = PROC_Create,PROC_Delete,PROC_Execute,PROC_RealUID,PROC_AuditID,......
    files = FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_Link,FILE_Unlink,......
    svipc = MSG_Create,MSG_Read,MSG_Write,MSG_Delete,MSG_Owner,MSG_Mode,......
    mail = SENDMAIL_Config,SENDMAIL_ToFile
    cron = AT_JobAdd,AT_JobRemove,CRON_JobAdd,CRON_JobRemove,CRON_Start,CRON_Finish
    tcpip = TCPIP_config,TCPIP_host_id,TCPIP_route,TCPIP_connect,......
    lvm = LVM_AddLV,LVM_KDeleteLV,LVM_ExtendLV<LVM_ReduceLV,......

users:
    root = general
    joe = general,files
    bob = files,cron
```

In this example, each user can have one or more defined classes. Classes are defined only to users that you want to audit. It is not necessary for all users to be defined in the stanza.

2.2.2 The oconfig file

The oconfig file is a backup copy of the config file. This file is automatically created everytime you start auditing. If there are any changes in your config file, it automatically creates another copy in the form of the oconfig file, to reflect any changes made.

2.2.3 The events file

The events file contains audit event information and has only one stanza: the auditpr. This also contains formatting information, which is that the `auditpr` command needs to write an audit tail of each event.

An audit event name can be up to 15 bytes long. Longer names are rejected.

The next display shows an example of the events file.

```
#more /etc/security/audit/events

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”
```

Notice the events PROC_Create, PROC_Delete, and PROC_Execute. All have printf information beside them. This is the information that will be recorded to your trail file as audit events are logged.

Refer to Section 2.6, “More on the events file” on page 42, for detailed information about event formatting, and Appendix A, “Audit events” on page 143, for more information about audit events.

2.2.4 The objects file

This is an ASCII file that contains information about audited objects or files. Each stanza defines exactly one audited file. Each stanza defines one or more access modes:

- r for read
- w for write
- x for execute

The next display shows an example of the objects file.

```
#more /etc/security/audit/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”
```

Take a look at the highlighted entry from the previous display.

The first highlighted stanza tells you that an event called S_ENVIRON_WRITE will be recorded everytime there is a write (denoted by the letter w), to the file /etc/security/environ.

The second highlighted stanza tells you that everytime there is a read to the /etc/security/passwd file, an event called S_PASSWD_READ will be recorded, and an event called S_PASSWD_WRITE will be recorded if there is a write to the file.

You can edit this file and add objects that you want to audit together with the mode of operation. Also, do not forget to include the event name for each mode.

In AIX, a file can be an ordinary file or a directory. That means you can audit every write or read attempt to a directory.

2.2.5 The bincmds file

This file contains commands that process audit bin data. The default content of this file is given in the next display.

```
#more /etc/security/audit/bincmds

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

This command compresses audit bin records and appends them to the audit trail. The name of the current bin file (such as bin1 or bin2), and the system audit trail file, are substituted for $bin and $trail parameter respectively. The value of the temporary files are defined in the configuration file, that is, the binsize parameter of the config file. The -p option tells you to compress the bin file, because it does not compress bin files by default. The -o option shows the output file where the `auditcat` command writes records. The output file in this example is $trail.

There is also one other command that you can use: the `auditselect` command. This selects audit records that match identified criteria and writes the records to standard output. With the `auditselect` command, you can filter the audit trail to obtain specific records for analysis or select specific records for long-term storage. If the bin files are compressed, the `auditselect` command unpacks them prior to processing. You can add commands to the file depending on your requirement. For example:

- To select audit events indicating unsuccessful authentications or use of privilege, and append the events to the /audit/trail.violations file, you must include the following line in the /etc/security/audit/bincmds file:

  ```
  /usr/sbin/auditselect -e "result == FAIL_AUTH || \
  result == FAIL_PRIV" $bin >> /audit/trail.violations
  ```

- To create a hard copy audit log of all user authentication audit events, include the following line in the /etc/security/audit/bincmds file:

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

  Customize the name of the printer to adjust to your definition.

In the first example, you will need to use the `auditpr` command to read the data in /audit/trail.violations file. In the second example, after selecting the event, we added the `auditpr` command, and redirect the output to the printer.

2.2.6 The streamcmds file

Contains the audit stream command invoked when the audit system is started.

```
#more /etc/security/audit/streamcmds

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

The `auditstream` command reads audit records from the audit device, that is, the /dev/audit file, and copies the record to standard output, that is, the

/audit/stream.out file, in binary format. The `auditpr` command formats the record for viewing or printing.

For stream data, configure both the auditstream command and the auditselect command in the /etc/security/audit/streamcmds file, or enter both commands from the command line.

Like in bincmds, you can add commands to this file depending on your requirement. For example:

- To format all record of events in the general class, and write them on the system console, you can add this line:

  ```
  /usr/sbin/auditstream -c general | /usr/sbin/auditpr -v > /dev/console &
  ```

- To format all records that resulted in access denial, and prints them on printer /dev/lp0:

  ```
  /usr/sbin/auditstream |/usr/sbin/auditselect -e \
  "result == FAIL_ACCESS" | /usr/sbin/auditpr -v > /dev/lp0 &
  ```

- To format and write all user login and su events to the line printer /dev/lp0, you can add this line:

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

Note

The auditstream command should run in the background with an ampersand (&) at the end. This is true only for stream mode auditing.

2.3 How to set up auditing

Assuming that you want to audit all su and password change for user joe, what will you do? What is the first step that you can think of? When should you start auditing? In this section, we will show you how to set up both BIN and STREAM mode data collection. We will also show you how to set up event and object auditing.

In the preceeding example, we will audit events USER_SU and PASSWORD_Change for user joe. For object auditing, we will audit all default objects.

First, you have to decide on the following:

- The type of data collection you need: BIN or STREAM. If you want to use BIN mode, then proceed to Section 2.3.1, “BIN mode auditing” on page 23. If you want to use STREAM mode, proceed to Section 2.1.2.2, “STREAM mode” on page 9. If you want to activate BIN and STREAM mode, then please reference both pages.
- Which events you want to audit. Refer to Section 2.3.3, “Events” on page 24.
- Which objects you want to audit. Refer to Section 2.3.4, “Objects” on page 29.

2.3.1 BIN mode auditing

To set up BIN mode auditing, you have to do the following:

- Check the config file, and modify if necessary. Binmode should be turned on, for example binmode = on.
- Check the bin stanza. Check the path, trail, and bin files. Decide if you want to change the binsize value in bytes. Take a look at the path of cmds.

You can do both of the preceding actions by using the vi editor. Take a look at the following item:

```
#vi /etc/security/audit/config
```

```
start:
   binmode = on
   streammode = off

bin:
   trail = /audit/trail
   bin1 = /audit/bin1
   bin2 = /audit/bin2
   binsize = 10240
   cmds = /etc/security/audit/bincmds

stream:
   cmds = /etc/security/audit/streamcmds
```

- Check the bincmds file. By default, this is inside the /etc/security/audit directory. Modify if necessary.

```
#vi /etc/security/audit/bincmds
```

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

2.3.2 STREAM mode auditing

Follow these procedure for setting STREAM mode auditing.

- Check the config file and modify if necessary. Streammode should be turned on, for example streammode = on.
- Check the stream stanza, and look for the cmds line.

You can do both of the preceding actions by using the vi editor. Focus on the following item.

```
#vi /etc/security/audit/config
```

```
start:
    binmode = off
    streammode = on

bin:
    trail = /audit/trail
    bin1 = /audit/bin1
    bin2 = /audit/bin2
    binsize = 10240
    cmds = /etc/security/audit/bincmds

stream:
    cmds = /etc/security/audit/streamcmds
```

- Check the streamcmds file. By default, this is in /etc/security/audit directory. Modify if necessary.

```
#vi /etc/security/audit/streamcmds
```

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

2.3.3 Events

The following steps should be taken in order to choose which events you wish to audit:

- Decide on the events you want to audit. Take a look at the events file and check the lines USER_SU and PASSWORD_Change.

```
#vi /etc/security/audit/events
```

```
auditpr:
* kernel proc events

*   fork ()
    PROC_Create = printf “forked child process %d”
    .
    .
    .
    .
    .
* commands
    .
    .
*   su
    USER_SU = printf “%s”

*   passwd
    PASSWORD_Change = printf “%s”
```

Note

Refer to Appendix A, “Audit events” on page 143 for a complete list of all events and their description.

- Group each event in a class. Check if events USER_SU and PASSWORD_Change are part of the general class. We will use the default class general.

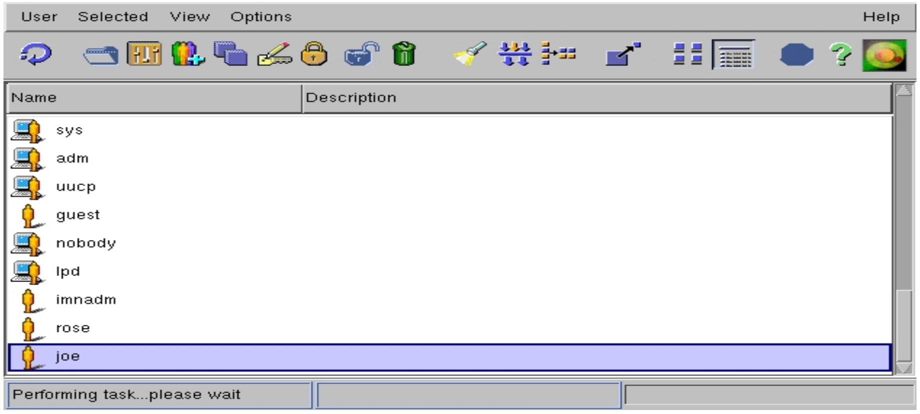
```
#vi /etc/security/config
```

```
classes:
    general =
USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,......
    objects = S_ENVIRON_WRITE,S_GROUP_WRITE,S_LILITS_WRITE,S_LOGIN_WRITE,......
    SRC = SRC_Start,SRC_Stop,SRC_Addssys,SRC_Chssys,SRC_Delssys,SRC_Addserver,......
    kernel = PROC_Create,PROC_Delete,PROC_Execute,PROC_RealUID,PROC_AuditID,......
    files = FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_Link,FILE_Unlink,......
    svipc = MSG_Create,MSG_Read,MSG_Write,MSG_Delete,MSG_Owner,MSG_Mode,......
    mail = SENDMAIL_Config,SENDMAIL_ToFile
    cron = AT_JobAdd,AT_JobRemove,CRON_JobAdd,CRON_JobRemove,CRON_Start,CRON_Finish
    tcpip = TCPIP_config,TCPIP_host_id,TCPIP_route,TCPIP_connect,......
    lvm = LVM_AddLV,LVM_KDeleteLV,LVM_ExtendLV<LVM_ReduceLV,......

users:
    root = general
    joe = general
    bob = files,cron
```

- Decide which class, for a particular user, you want to audit. You can do this by using wsm, smit or vi. Remember that event auditing is ALWAYS associated by a user ID.

```
#/usr/websm/bin/wsmuser
```

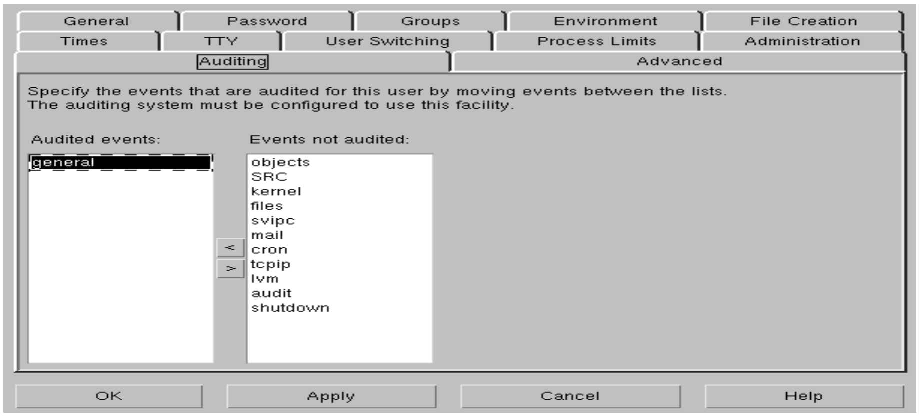


Performing task...please wait

Figure 4. WSM user interface - Select a user

Using Figure 4 on page 26 as an example, select the user name you want to audit (in this case joe) and press **Enter**. Look for the status at the lower left hand portion of the display. The message `performing task...please wait` indicates that the request is being processed. After this task ends, you will have a display similar to Figure 5 on page 27. Select a class for auditing.

Go to the auditing folder, and select the class you want for user joe (in this case, general). The class name should appear under the column Audited Objects. After selecting the class, click **Apply** then **OK**. This will save changes in your configuration.



Specify the events that are audited for this user by moving events between the lists.
The auditing system must be configured to use this facility.

Audited events:
general

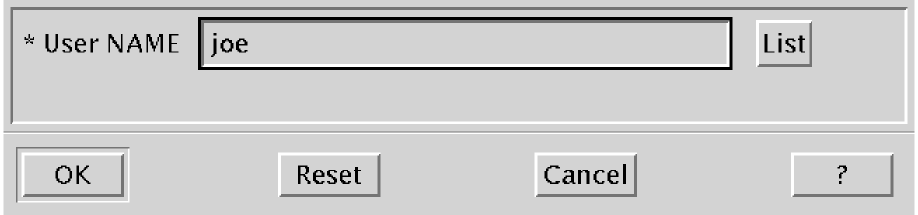
Events not audited:
objects
SRC
kernel
files
svipc
mail
cron
tcpip
lvm
audit
shutdown

Figure 5. WSM user interface - Select a class for auditing

You can also use `smit`.

```
#smit user
```

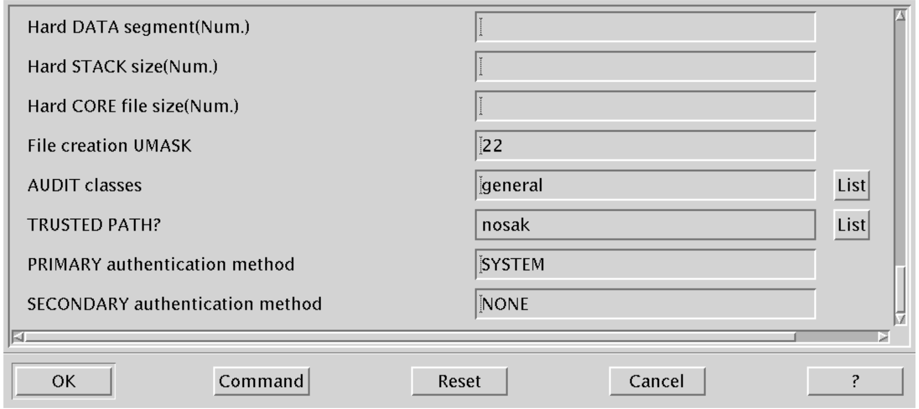
Select the entry to change/show characteristics of a user. The assumption is that there is an existing user name joe. You will now see the next display.



* User NAME joe

Figure 6. SMIT user interface - Select a user name

Using Figure 6 on page 27 as an example, click the list button. This button will give you a list of existing users. Select user name joe and click **OK**.



Hard DATA segment(Num.)

Hard STACK size(Num.)

Hard CORE file size(Num.)

File creation UMASK: 22

AUDIT classes: general

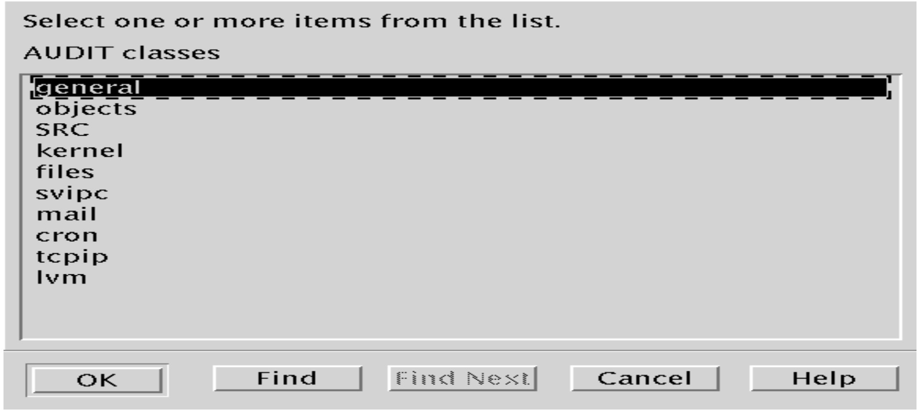
TRUSTED PATH?: nosak

PRIMARY authentication method: SYSTEM

SECONDARY authentication method: NONE

Figure 7. SMIT user interface - AUDIT class

Using Figure 7 on page 28 as an example, go to the audit classes entry, and click the list button. It will then give you the preceding display



Select one or more items from the list.

AUDIT classes

```
general
objects
SRC
kernel
files
svipc
mail
cron
tcpip
lvm
```

Figure 8. SMIT user interface - Select the class you want for a user

Using Figure 8 on page 28 as an example, select the class you want to audit for user joe, and click the **OK** button. You have now selected audit class general for user joe.

You can also use the vi editor to select an audit class for user joe. Include the line joe = general in the users stanza.

```
#vi /etc/security/audit/config
```

```
config file display 2 of 2

classes:
    general =
USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,......
    objects = S_ENVIRON_WRITE,S_GROUP_WRITE,S_LILITS_WRITE,S_LOGIN_WRITE,......
    SRC = SRC_Start,SRC_Stop,SRC_Addssys,SRC_Chssys,SRC_Delssys,SRC_Addserver,......
    kernel = PROC_Create,PROC_Delete,PROC_Execute,PROC_RealUID,PROC_AuditID,......
    files = FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_Link,FILE_Unlink,......
    svipc = MSG_Create,MSG_Read,MSG_Write,MSG_Delete,MSG_Owner,MSG_Mode,......
    mail = SENDMAIL_Config,SENDMAIL_ToFile
    cron = AT_JobAdd,AT_JobRemove,CRON_JobAdd,CRON_JobRemove,CRON_Start,CRON_Finish
    tcpip = TCPIP_config,TCPIP_host_id,TCPIP_route,TCPIP_connect,......
    lvm = LVM_AddLV,LVM_KDeleteLV,LVM_ExtendLV<LVM_ReduceLV,......

users:
    root = general
    joe = general
    bob = files,cron
```

Be careful in using the vi editor; there is a probability that you might type in extra characters.

2.3.4 Objects

The following steps should be taken in order to choose which objects you wish to audit:

- Decide on the objects you want to audit:

```
#vi /etc/security/audit/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"

/etc/security/audit/config:
    w = "AUD_CONFIG_WR"
```

Add the file you want to audit. It is normally added at the end of the objects file.

If you do not want the objects audit records when auditing a user ID, comment out the objects defined in the /etc/security/audit/objects file or rename the file.

At this point, you can now start auditing:

```
#/usr/sbin/audit start
```

2.4 Advanced auditing setup

Now that you know how to set up auditing, let us try some advanced auditing configuration. This section will discuss how to add an event and a class to the config file. Some auditpr techniques will also be discussed.

Let us now perform a little exercise that will help us understand how to exploit the information provided by the auditing subsystem.

You are the head of your IT services group. You have two other people working with you in maintaining your servers. As part of their role as system administrators, they need to know the root password of all your RS/6000 servers.

One day, you were working on a project when suddenly your session was disconnected. Thinking that this could just be a time out problem, you did not bother to check it. The next day, it happened again twice. You decided to check your server and network to find out the probable cause of your session being disconnected. While checking on the files, you notice that all history logs and other information that can help you determine the cause are all gone. You decided to change the root password but after a month it happened again. It appears that someone knows the root password and is trying to kill the process!

What auditing techniques could be used to find out the real cause of the problem?

For this example, we will use BIN data collection method.

First, you have to know what events are involved in a kill process. Use the vi editor to check the events file:

```
#vi /etc/security/audit/events
```

```
#more /etc/security/audit/events

auditpr:

    ......more......
*   kill()
    PROC_Kill = printf "pid: %d, sig: %d"

    ......more......
```

You now have the PROC_Kill event.

Check the config file to see if it contains the PROC_Kill event. Add the event if you do not have it on file. You may also want to create a separate class for this event.

We will define a new class called *kill*, and include the event PROC_Kill under the new class. After this, we will assign this class to all users, including root.

Your config file should look like this:

```
start:
    binmode = off
    streammode = on

bin:
    trail = /audit/trail
    bin1 = /audit/bin1
    bin2 = /audit/bin2
    binsize = 10240
    cmds = /etc/security/audit/bincmds

stream:
    cmds = /etc/security/audit/streamcmds
classes:
    .....more......
    kill = PROC_Kill
users:
    root = general,kill
    joe = general,kill
```

Now you can start auditing by using the command:

```
#audit start
```

Note

The newly assigned audit class will take effect at the next login for that user.

Refer to Section 2.5.4, “Output for advance auditing setup” on page 40 to read an example of the output.

2.5 Understanding the output

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 also contain too much information. You have to 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 want to set up auditing in order to know what is happening on your system. This can be for one user, for all users, for several classes, or for all system activity. Now that you have the audit record, what do you plan to do with it?

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 user joe did the following:

- Connected to the server through tcp/ip using the command `telnet lv9510c`, where lv9510c is the hostname.
- Login as user joe from the login prompt.
- Change joe's password, using the command `passwd`; the change was not successful.
- Change joe's password again, using the command `passwd`. Now joe was able to change his password successfully.
- Try to su to user root, but gave a wrong password. He used the command `su`.
- Try to su to user root again using the `su` command. Now he was able to su to user root successfully.

We are assuming that:

- The data collection mode is for both BIN and STREAM.
- All the default BIN and STREAM stanza are used.
- We will audit the class general for user joe.

2.5.1 Event auditing - BIN mode

To read the output, you can use this command:

```
auditpr -v < /audit/trail
```

The next display gives you a sample output for BIN mode auditing:

```
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
S_PASSWD_READ   root     OK          Thu Jun 15 12:33:12 2000 telnetd
       sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   root     OK          Thu Jun 15 12:33:12 2000 telnetd
       sample audit object read event detected /etc/security/passwd
TCPIP_connect   root     OK          Thu Jun 15 12:33:12 2000 telnetd
       TCP/IP ::ffff:9.3.1.130 telnet/tcp open
FS_Chdir        joe      OK          Thu Jun 15 12:33:22 2000 tsm
       change current directory to: /home/joe
PASSWORD_Change joe      FAIL        Thu Jun 15 12:33:30 2000 passwd
       joe
PASSWORD_Change joe      OK          Thu Jun 15 12:33:40 2000 passwd
       joe
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:43 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:44 2000 su
        sample audit object read event detected /etc/security/passwd
USER_SU         joe      FAIL        Thu Jun 15 12:33:44 2000 su
        root
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:46 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
        sample audit object read event detected /etc/security/passwd
USER_SU         joe      OK          Thu Jun 15 12:33:48 2000 su
        root
FS_Chdir        joe      OK          Thu Jun 15 12:33:48 2000 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 if the execution of an event is successful or not. 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; a description of each follows:

- The OK value means that there was a successful execution of an event.
- The FAIL value means that there was an unsuccessful execution of an event. This is the default FAIL value.
- The FAIL_AUTH value indicates that authentication was denied. The user may have tried to login and failed authentication by giving an incorrect password, or tried to login to a console where they do not have permission to do so.
- The FAIL_PRIV value indicates lack of privilege.
- The FAIL_ACCESS value indicates lack of access.
- The FAIL_DAC value indicates 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(permission) to shared resources/objects. ACLs are categorized on DAC mechanism.

Look at the four highlighted entries of this record. Notice the PASSWORD_Change and USER_SU events. Notice also the status of each event. The second line of each event is what we call the tail portion. This came from the printf information of the events file. The tail portion can be modified (depending on what you want to be recorded) if the `auditpr` command used the events file.

When joe tried changing his password and entered a wrong password, the PASSWORD_Change event was triggered with a FAIL status. The second line gives you the user whose password joe wanted to change. Joe tried changing his password again, and now he was able to successfully change his password. This time the PASSWORD_Change event was again triggered, but with the status equivalent to OK.

The same is true when he tried to su to user root. The only difference is, now event USER_SU was triggered, and the second line gives you the ID to which user joe wants to su.

Let us now take a look at the output of stream mode.

2.5.2 Event auditing - STREAM mode

To read the output, you can use the vi editor.

```
#vi /audit/stream.out
```

The next display gives you a sample output for STREAM mode data collection method:

```
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
S_PASSWD_READ   root     OK          Thu Jun 15 12:33:12 2000 telnetd
S_PASSWD_READ   root     OK          Thu Jun 15 12:33:12 2000 telnetd
TCPIP_connect   root     OK          Thu Jun 15 12:33:12 2000 telnetd
FS_Chdir        joe      OK          Thu Jun 15 12:33:22 2000 tsm
PASSWORD_Change joe      FAIL        Thu Jun 15 12:33:30 2000 passwd
PASSWORD_Change joe      OK          Thu Jun 15 12:33:40 2000 passwd
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:43 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:44 2000 su
USER_SU         joe      FAIL        Thu Jun 15 12:33:44 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:46 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
S_PASSWD_READ   joe      OK          Thu Jun 15 12:33:48 2000 su
USER_SU         joe      OK          Thu Jun 15 12:33:48 2000 su
FS_Chdir        joe      OK          Thu Jun 15 12:33:48 2000 su
```

Like in BIN mode, this mode has five columns.

Look at the four highlighted entries of the file stream.out. Take a look at the events PASSWORD_Change and USER_SU. Notice also the status of each event.

You interpret this record the same way you interpreted the BIN mode record.

However, do you notice the difference in the records? Why is the output of BIN mode different from the output of the STREAM mode?

Note

Look closely at how the data was formatted.

Do you remember the `auditpr` command?

If you remember the `auditpr` command, you will see the difference. The reason for this is that in BIN mode we used the -v option. The -v option displays the tail of each audit record using the format specification in the /etc/security/audit/events file. Remember the `auditpr` command formats the record to make it readable. This will be discussed further in Section 2.6, “More on the events file” on page 42.

Aside from the formatting difference, do not forget that BIN mode can store records for a long term period, while STREAM mode records get overwritten whenever you start auditing.

2.5.3 Object auditing - STREAM mode

This is the scenario for object auditing. This record was generated when user joe did the following:

- Connected to the server through tcp/ip using the command `telnet lv9510c`, where lv9510c is the hostname.
- Logged in as user joe from the login prompt. Joe belongs to staff group.
- Tried to su to root with the correct password using the `su` command.
- Viewed the /etc/security/passwd file using the command `vi /etc/security/passwd`.
- Viewed the /etc/security/passwd file using the command `vi /etc/security/passwd`, then save and exited from vi editor.

We assume that:

- STREAM mode data collection is on.
- All the default BIN and STREAM stanza are used.
- We have edited the file /etc/security/audit/streamcmds to include the -v option. The file should look like this:

  ```
  #vi /etc/security/audit/streamcmds
  ```

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

The next two displays give you a sample output for the STREAM mode data collection method that used object auditing.

```
/audit/stream.out display 1 of 2

event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
_PASSWD_READ   root     OK          Mon Jun 19 15:44:28 2000 telnetd
       sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   root     OK          Mon Jun 19 15:44:28 2000 telnetd
       sample audit object read event detected /etc/security/passwd
TCPIP_connect   root     OK          Mon Jun 19 15:44:28 2000 telnetd
       TCP/IP ::ffff:9.3.1.130 telnet/tcp open
FS_Chdir        joe      OK          Mon Jun 19 15:44:31 2000 tsm
        change current directory to: /home/joe
S_PASSWD_READ   joe      OK          Mon Jun 19 15:44:35 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Mon Jun 19 15:44:36 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Mon Jun 19 15:44:36 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Mon Jun 19 15:44:36 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Mon Jun 19 15:44:36 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Mon Jun 19 15:44:36 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Mon Jun 19 15:44:36 2000 su
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK          Mon Jun 19 15:44:36 2000 su
        sample audit object read event detected /etc/security/passwd
USER_SU         joe      OK          Mon Jun 19 15:44:36 2000 su
        root
FS_Chdir        joe      OK          Mon Jun 19 15:44:36 2000 su
        change current directory to: /
```

```
/audit/stream.out display 2 of 2

FILE_Unlink     joe      OK         Mon Jun 19 15:44:43 2000 vi
        filename /var/tmp/Ex17820
S_PASSWD_READ   joe      OK         Mon Jun 19 15:44:43 2000 vi
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK         Mon Jun 19 15:44:43 2000 vi
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK         Mon Jun 19 15:44:43 2000 vi
        sample audit object read event detected /etc/security/passwd
FILE_Unlink     joe      OK         Mon Jun 19 15:44:45 2000 vi
        filename /var/tmp/Ex17820
FILE_Unlink     joe      OK         Mon Jun 19 15:44:47 2000 vi
        filename /var/tmp/Ex17822
S_PASSWD_READ   joe      OK         Mon Jun 19 15:44:47 2000 vi
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK         Mon Jun 19 15:44:47 2000 vi
        sample audit object read event detected /etc/security/passwd
S_PASSWD_READ   joe      OK         Mon Jun 19 15:44:47 2000 vi
        sample audit object read event detected /etc/security/passwd
S_PASSWD_WRITE  joe      OK         Mon Jun 19 15:44:48 2000 vi
        audit object write event detected /etc/security/passwd
FILE_Unlink     joe      OK         Mon Jun 19 15:44:48 2000 vi
        filename /var/tmp/Ex17822
```

Look at the /audit/stream.out file display 2 of 2. Notice the first highlighted S_PASSWD_READ event. This event was triggered when user joe tried reading the /etc/security/passwd file using the vi editor. Notice the second attempt of joe. He issued another `vi` command, and the same event, which is S_PASSWD_READ, was triggered. Take a look at the third highlighted line. Previously, joe read the file, and now he wrote something on that file. At this point, we do not know exactly if something was appended on the /etc/security/passwd file. What we know is that joe used the exit and save option of vi.

Why did the system allowed joe to view and save the /etc/security/passwd file? Remember one of the assumptions? Joe is a member of the staff group.

Note

Examine closely the output and use your knowledge of event auditing.

Yes, you are right! The first display shows that joe actually did an su to user root; that is why the USER_SU event was logged. He was able to login successfully; that is why the FS_Chdir was logged. Note that if a user logs in the system, the user goes to the home directory. This is the reason why he was able to go to the root (/) directory. Notice the last two event entries in display one.This is a combination of user and object auditing.

2.5.4 Output for advance auditing setup

Let us now read the output from the scenario in Section 2.4, “Advanced auditing setup” on page 30.

We will use the `auditpr` command to read the output and save it in a file called trail.out.

```
#auditpr -h e,l,r,t,R,c < /audit/trail > trail.out
```

The -h option of the `auditpr` command allows you to select what field 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 discussion of the `auditpr` command, refer to *AIX Version 4.3 Commands Reference,* SBOF-1877.

The next display shows you a sample output of the `auditpr` command.

```
#auditpr -h e,l,r,t,R,c < trail > trail.out

event           login    real     time                     status      command

--------------- -------- -------- ------------------------ ----------- ---------
S_PASSWD_READ   root     root     Tue Jun 20 16:27:37 2000 OK          telnetd
S_PASSWD_READ   root     root     Tue Jun 20 16:27:37 2000 OK          telnetd
TCPIP_connect   root     root     Tue Jun 20 16:27:37 2000 OK          telnetd
FS_Chdir        joe      joe      Tue Jun 20 16:27:41 2000 OK          tsm
S_PASSWD_READ   joe      joe      Tue Jun 20 16:27:42 2000 OK          su
S_PASSWD_READ   joe      joe      Tue Jun 20 16:27:43 2000 OK          su
S_PASSWD_READ   joe      root     Tue Jun 20 16:27:43 2000 OK          su
S_PASSWD_READ   joe      root     Tue Jun 20 16:27:43 2000 OK          su
S_PASSWD_READ   joe      root     Tue Jun 20 16:27:43 2000 OK          su
S_PASSWD_READ   joe      root     Tue Jun 20 16:27:43 2000 OK          su
S_PASSWD_READ   joe      joe      Tue Jun 20 16:27:43 2000 OK          su
S_PASSWD_READ   joe      joe      Tue Jun 20 16:27:43 2000 OK          su
USER_SU         joe      root     Tue Jun 20 16:27:43 2000 OK          su
FILE_Rename     root     root     Tue Jun 20 16:27:45 2000 OK          xntpd
FS_Chdir        joe      root     Tue Jun 20 16:27:51 2000 OK          ps
FS_Chdir        joe      root     Tue Jun 20 16:27:51 2000 OK          ps
FS_Chdir        joe      root     Tue Jun 20 16:27:51 2000 OK          ps
FS_Chdir        joe      root     Tue Jun 20 16:27:51 2000 OK          ps
FS_Chdir        joe      root     Tue Jun 20 16:27:51 2000 OK          ps
FS_Chdir        joe      root     Tue Jun 20 16:27:51 2000 OK          ps
FS_Chdir        joe      root     Tue Jun 20 16:27:51 2000 OK          ps
PROC_Kill       joe      root     Tue Jun 20 16:28:09 2000 OK          ksh
FILE_Unlink     joe      root     Tue Jun 20 16:28:23 2000 OK          rm
FILE_Unlink     joe      root     Tue Jun 20 16:28:23 2000 OK          rm
FILE_Unlink     joe      root     Tue Jun 20 16:28:23 2000 OK          rm
FILE_Unlink     root     root     Tue Jun 20 16:28:27 2000 OK          cdsclerk
FILE_Unlink     root     root     Tue Jun 20 16:28:27 2000 OK          cdsclerk
FILE_Rename     root     root     Tue Jun 20 16:28:27 2000 OK          cdsclerk
```

With this output, what could have happened is:

- There was a TCPIP_connect attempt to the server. Note the `telnetd` command and the event TCPIP_connect.
- User joe logged into the system. Notice the event FS_Chdir after the TCPIP_connect event.
- Joe issued an `su` command to root user, that is, through the event USER_SU. Look at the login and real name column. The login ID is Joe, but the real ID is root.
- Look at the commands column. Notice the `ps` command. The real user, which is root, with joe as the login ID, was checking some process.
- Now, notice the event PROC_Kill. This time a signal was sent to a process that could have killed one or more process.
- After the PROC_Kill event, there were multiple FILE_Unlink events. Several `rm` commands were issued. There could be several files deleted

from the system. Note that it is still user joe (using the root ID) doing these commands.

With the right combination, and the proper timing, you can make good use of the things you have learned. This time you were able to capture several events that occurred on a given time, and you were able to interpret the audit record.

2.6 More on the events file

The events file contains information about system audit events. This file contains the auditpr stanza, which lists all audit events in the system, and provides formatting information. In the previous example, Section 2.5, “Understanding the output” on page 32, you saw the difference of the two outputs, and what the -v option can do. This section will explain further formatting information.

The `auditpr` command reads audit records, in Bin or Stream format, from standard input and sends formatted records to standard output.

The next display shows you an example of what the events file contain.

```
auditpr:
......more......
*   su
    USER_SU = printf "%s"
*   chdir()
    FS_Chdir = printf "change current directory to: %s"
......more......
```

A sample output using the auditpr stanza of the events file follows:

```
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------
USER_SU         joe      OK          Mon Jun 19 15:44:36 2000 su
    ...... tail portion ......
FS_Chdir        joe      OK          Mon Jun 19 15:44:36 2000 su
    ...... tail portion ......
```

Each attribute in the auditpr stanza, which is from /etc/security/audit/events file, has the following format:

auditevent = FormatCommand

Audit events are generally defined at a system call level. FormatCommand have the following formatting information:

Table 2. Audit event formatting information

Format	Description
%A	Formatted output similar to the aclget command
%d	Formatted as a 32-bit signed decimal integer
$G	Formatted as a comma-separated list of group names or numerical identifiers
%o	Formatted as 32-bit octal integer
%P	Formatted output is similar to the pclget command
%s	Formatted as a text string
%T	Formatted as a text string giving include date and time with 6 significant digits for the seconds DD Mmm YYYY HH:MM:SS:mmmuuu).
%u	Formated as a 32-bit unsigned integer
%x	Formatted as a 32-bit hexadecimal integer
%X	Formatted as a 32-bit hexadecimal integer with upper case letters

To format the tail of an audit record, add format specifications in the auditpr stanza of /etc/security/audit/events file. An example of how the output tail looks is shown in Table 3. This includes event names, format information, and tail portion. The tail portion’s format came from the events file.

Table 3. Sample formatting output

Event Name	Formatted Output	Tail
USER_SU	printf “%s”	root
FS_Chdir	printf “change current directory to: %s”	change current directory to: /home/joe
PROC_Create	printf “forked child process %d”	forked child process 3534
FILE_Mode	printf “mode:%o filename %s”	mode: 777 filename sample
PROC_Privilege	printf "cmd: %x privset: %x:%x"	cmd: 30005 privset: ffffffff:ffffffff

Event Name	Formatted Output	Tail
FILE_Open	printf "flags: %d mode: %o fd: %d filename %s"	flags: 0 mode: 0 fd: 3 filename /usr/lib/nls/msg/en_US/ksh.cat
FILE_Read	printf "file descriptor = %d"	file descriptor = 3

For example:

- Using the event USER_SU, the formatted output gives you %s. The auditpr command then substitutes the value as a text string, thus giving the tail output.
- Using the event FILE_Mode, the formatted output gives you %o, which is for 32-bit octal integer, and %s as a text string, thus giving you the tail portion.

You may or may not use the events file. In case you do not want to use the formatting information from the events file, then make sure you specify the fields that will help you build a scenario of what happened.

For a detailed discussion of the `auditpr` command refer to *AIX Version 4.3 Commands Reference,* SBOF-1877.

2.7 Exceptions

You might have noticed from the trail or stream.out file that there are some events, not actually set to be logged, that are being logged. Given the way that tcpip and cron codes are written, they have their own set of audit events. These events will be written to the audit trail regardless of how you set up the config file. To eliminate tcpip and cron codes from being logged you can use the `auditselect` command. The `audiselect` command is the best way to filter those codes out. For example:

```
/usr/sbin/auditselect -e "event != AT_JobAdd && \

event != ATJobRemove" $bin >> /audit/train.exceptions
```

This will exclude events AT_JobAdd and ATJobRemove.

```
/usr/sbin/auditselect -e "command != cron && \

command != at" $bin >> /audit/trail.exceptions
```

This excludes recording related to commands `at` and `cron`.

The example refers to the use of BIN mode auditing. If you are using STREAM mode auditing, the `auditselect` can be included in the streamcmds file. For example:

```
/usr/sbin/auditstream |/usr/sbin/auditselect -e "command != cron \

&& command != at" | /usr/sbin/auditpr -v > /audit/stream.exceptions &
```

This command excludes recording related to commands `cron` and `at`.

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 Version 4.3 Commands Reference,* SBOF-1877.

2.8 Common problems with auditing

Identifying the source of a problem can sometimes be too simple or too complicated. It is often a circular process. Once you have the problem description, initial investigation is performed. A hypothesis is drawn based on the facts gathered. Action is taken in an attempt to fix the problem and to test the validity of the hypothesis. If this is unsuccessful, another investigation is performed.

Once the cause is determined, you can draw a plan in an attempt to solve the problem. You have to consider a lot of factors, such as the effect on your system, downtime, the risk factors and so forth. It is important for you to understand that merely fixing a problem may address the symptom, but problem determination is not yet complete. Identifying the cause of a problem is different from identifying the source of the problem. Problem determination is not complete until the cause has been determined.

It is not the intent of this section to cover all problem determination techniques. We will just present to you some commonly encountered problems and error messages.

Symptom When I run the `audit start` command, it gives me an error.

Error Message ** /audit is not a directory

Action The `audit start` command automatically creates the audit directory. There could have been an existing ordinary file called audit. You can either delete or rename this file.

Symptom When I run the `audit start` command, it gives me an error.

Error Message ** auditing enabled already

A system call received a parameter that is not valid.

Action None. Auditing was already started, and you are trying to start it again.

Symptom I would like to set up event auditing, but when I run the `audit start` command, it gives me an error.

Error Message ** cannot find "streammode" keyword in "start" stanza in "/etc/security/audit/config"

A system call received a parameter that is not valid.

Action Check the config file. Look for the streammode line in the start stanza. If it was deleted, then add the line with the corresponding value, such as off or on. If the streammode line is there, check for extra characters. You might have typed in extra characters before the start stanza.

Symptom When I run the `audit start` command, it gives me an error.

Error Message ** cannot find "users" stanza in "/etc/security/audit/config"

** failed setting kernel audit objects

Action Check the config file, and look for the users stanza. It could either be missing, or extra characters could have been typed in the users stanza.

Symptom I configured my streamcmds file using the -v option for the `auditpr` command. When I ran the `audit start` command, it gives me this error.

Error Message /etc/security/audit/events: A file or directory in the path name does not exist.

** warning - no tails will be printed type here

Action You can disregard this error, but the tail portion will not be printed. Also, check the events file; it could have been renamed, or deleted.

Symptom the `audit start` command fails with this message

Error Message ** failed setting kernel audit objects

Action Check the objects file. There could be a syntax error in this file.

2.9 Sizing considerations

Whenever you want to run something new on your system, you often raise the following concerns:

- What are the benefits?
- How much disk space does it consume?
- Does it have any performance impact on my system?

After going through this chapter, you now know the benefits that you can get if you turn auditing on. But what about disk space and performance issue? This section will discuss these concerns.

2.9.1 Disk space

Depending upon the amount of activity on a system, the mode of data collection used and the tail information, the audit trail size may vary. It is safe to assume though that the audit trail grows in size.

Audit information is recorded as auditable events occur. Although most information is not required to be immediately available for real-time analysis, you should have the capability to retrieve audit information within minutes of its recording. The audit trail should be reviewed at least once a week. It is very possible, however, that once a week may be too long to wait to review the audit trail. Depending on the amount of audit data and the need to review these information, this parameter should be adjusted accordingly.

The audit trail should be sufficient enough to give you a complete sequence of what happened on your system at a given time. To do this, the audit trail should contain, at least, the following information: date and time of the event, user, type of event, success or failure of the event, and the command used.

The size of each event varies depending on the tail information from the events file. It also depends on the fields you want to include in your report. If you have decided that all activities should be audited, there are overhead factors to be considered. Table 4 gives you an idea of the size of the trail file. In this example, the trail file contains information about the following: event, login, status, time, and command.

Table 4. Sample size of each event with header information

Events	Trail in bytes with header	Stream.out in bytes w/ header
PASSWORD_Change	127	285
TCPIP_connect	145	285
USER_SU and FS_Chdir	132	380
USER_SU	102	285
FS_Chdir	108	285

Each audit record has a header. This header gives you a description of the content of each column. For BIN mode, the header part is approximately 88 bytes; thus, each event can range form 20 bytes or more. For STREAM mode, the header part is approximately 190 bytes; thus, each event can range from 100 bytes or more. Note that the less information you have, the smaller the size of your record is.

2.9.2 Performance

Each event does a preliminary check to see if 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 finish audit logging, the difference is still so small that is almost negligible.

Overall, regardless if auditing is enabled or not, the system has a slight overhead utilization hit based on auditing. The reason is that there are some codes written that have their own set of audit events. There is no overhead in having auditing on if the events that are being audited are not occurring.

Chapter 3. Accounting on AIX

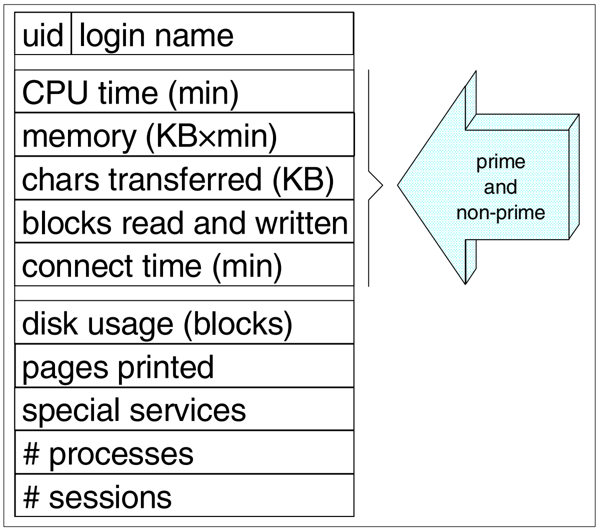
The AIX accounting system provides you with information about the use of system resources. These data can be used to:

- Bill the users for the resources used.
- Control the use of some resources (such as disk space).
- Substantiate the need for system expansion.
- Perform performance monitoring.
- Maintain an audit trail for security purposes.

AIX accounting implementation is fully compatible with AT&T System V Release 4. With AIX, you can also use 4.3 BSD accounting utilities to inspect some of the accounting files. Because of this two-fold interface, you will occasionally find two different ways to obtain the same accounting data.

3.1 Inside accounting

The main goal of the accounting system is to generate the data shown in Figure 9.



uid | login name
CPU time (min)
memory (KB×min)
chars transferred (KB)
blocks read and written
connect time (min)
disk usage (blocks)
pages printed
special services
# processes
# sessions

Figure 9. The total accounting record (tacct)

3.1.1 Accounting resources

The total accounting record (tacct) contains the following information for each user:

- The user’s login name and uid
- The total CPU time (system time plus user time) utilized by the user’s processes in minutes
- The integral of core memory allocated by the user’s processes while they were holding the CPU, in KB times minutes
- The number of KB read or written by the user’s processes
- The number of disk blocks read or written by the user’s processes
- The time in minutes the user was connected to the system
- The number of 512 byte disk blocks allocated by the user’s files
- The number of pages printed by the user
- Any special services charged by the system administrator
- The number of processes ran by the user
- The number of sessions started by the user

The exact structure of the total accounting files can be found in Appendix B.1, “The tacct file” on page 153.

3.1.2 Billing periods

The accounting system breaks up resource usage into two different billing periods: prime-time and non-prime time. Prime-time hours are defined by you and are, by default, from 8:00 a.m. to 5:00 p.m., with the exception of weekends and holidays. Thus, as depicted in Figure 9, the total accounting record contains, for some resources, such as CPU time, their usage during each of the two billing periods.

3.1.3 Accounting processes

The process of gathering the above information is quite complex, involving several daemons and commands, as well as some kernel functions. There are four distinct modules used by the accounting system to generate the usage data as depicted in Figure 10. These modules generate intermediate files that are processed by the `/usr/sbin/acct/runacct` command, generating the total accounting files. This command is run daily as a cron job.

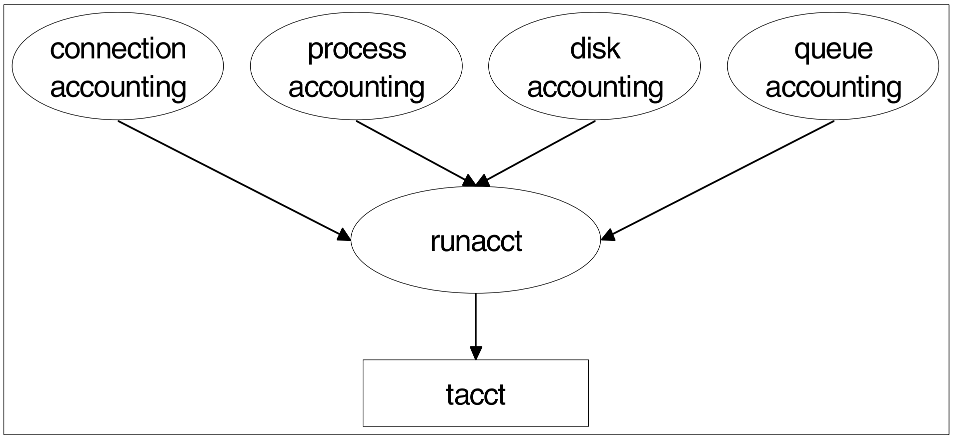


Figure 10. Overall view of the usage gathering process

In the following four sections, we describe how AIX generates the accounting information for each type of usage data. In Section 3.1.8, “Consolidation of the accounting data” on page 57, we discuss how the `runacct` command consolidates the accounting data into tacct files.

3.1.4 Connection accounting

The accounting system collects the time users were connected to the system. It gathers such information from the /var/adm/wtmp and /etc/utmp files. The structure of both files can be found in Appendix B.2, “The wtmp file” on page 153.

The /etc/utmp file contains a record of the users *currently* logged into the system plus some information about the init processes. But the information in utmp is transient, being lost after a system shutdown, and it is not accumulative. Thus, the accounting system requires a second file, /var/adm/wtmp, which accumulates practically the same data as utmp.

3.1.4.1 The init daemon

The init daemon is the main maintainer of the utmp and wtmp files. During startup, init re-initializes the utmp file and writes boot time information to both files. It also writes information about the processes it starts (defined in /etc/inittab), including the getty processes. The getty process initializes a terminal connected to the system and waits for a user to log in, writing a corresponding record in the utmp file.

After a user logs in successfully, the getty process, already transformed into a login process, writes a *user process* record to the utmp and wtmp files to inform that a user connection has begun. This record contains the user name, the device name, and the login time. When the user's shell exits, init, being its parent, receives the corresponding signal and consequently writes a *dead process* record to both files informing that the user connection has ended. This record contains the device name and the logout time.

Notice that init overwrites the corresponding user process record with the new dead process record in the utmp file. On the other hand, init appends that dead process record to the end of the wtmp file. In this manner, the accounting system, as described in a subsequent section, can extract from the wtmp file the relevant connection information: user name, device name, connection starting time, and connection ending time.

3.1.4.2 Remote and X11 sessions

The procedure above is repeated by all other daemons and processes that start a new user connection, such as `telnetd`, `rlogind`, `dtlogin`, `aixterm`, and `xdm`. The X11 processes behave slightly different since they do not invoke the login program to log the user in, thus being responsible for writing both user process and dead process records to the utmp and wtmp files.

3.1.4.3 Other actions

Other system components also generate records that are used by AIX accounting. The `date` command, for example, when changing the system's date and time, writes two records to the utmp and wtmp files, informing the *old time* and the *new time*. In addition, when the system is shut down, both the `halt` and `reboot` commands write a record to the wtmp file with the appropriate information.

The accounting related actions involving the utmp and wtmp files are summarized in Figure 11 on page 53.

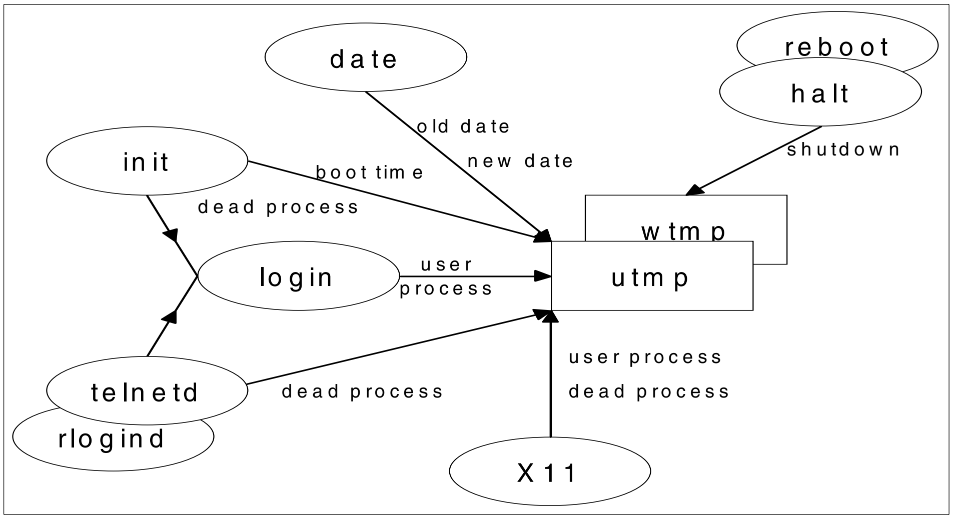


Figure 11. Gathering of connection accounting data

3.1.5 Process accounting

The accounting system needs to collect the resources used by the users' processes: CPU time, memory, and I/O counts. To gather these resources, the accounting system needs the assistance of the kernel. The kernel, whenever a process exits, appends a record with the process' accounting information to the process accounting file, normally /var/adm/pacct. The structure of this file can be found in Appendix B.3, "The pacct file" on page 154.

3.1.5.1 Starting accounting

The AIX kernel does not generates process accounting data by default. To start the data gathering process, the kernel's acct subroutine should be invoked. This subroutine has one parameter that is the name of an existing file that contains the process accounting records. The acct subroutine opens the file, after which the kernel starts appending the accounting information to it. The accounting system provides a command that calls the acct subroutine, `/usr/sbin/acct/accton`, whose parameter is the name of the process accounting file.

3.1.5.2 Stopping accounting

To turn process accounting off, the acct subroutine (and therefore the `accton` command) has to be invoked with a null parameter. When called with a null parameter, acct closes the accounting file, informing the kernel’s process exit subroutines that accounting should not take place.

Figure 12 depicts process accounting:

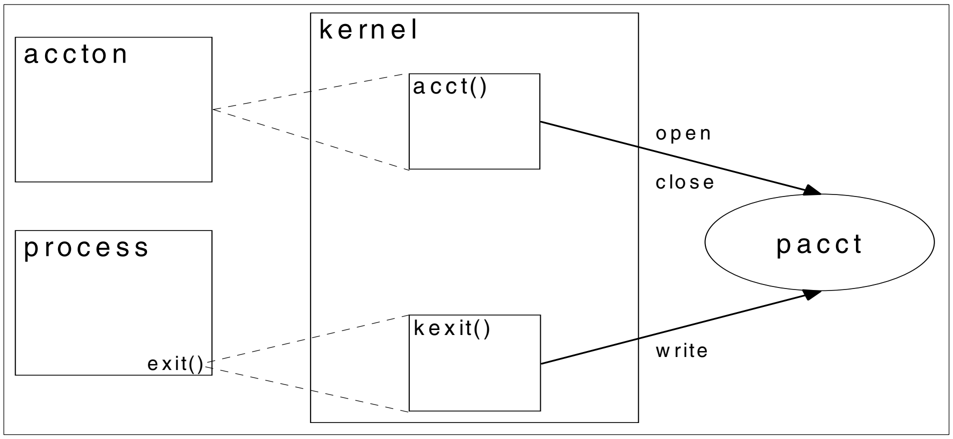


Figure 12. Gathering of process accounting data

Process accounting is automatically shut off by the `shutdown` command. It does so by calling the `/usr/sbin/acct/shutacct` command while it is stopping the system’s processes. The `shutacct` command, besides writing a record to the /var/adm/wtmp file, calls the `/usr/sbin/acct/turnacct` command to turn process accounting off. The `turnacct` command, in turn, is an interface to the `accton` command.

> **Note**
>
> If you use the `halt` or `reboot` commands to shut down your system, process accounting (as well as other system activities) will not be rightfully terminated, causing a possible loss of information. If you want to make sure no process accounting data is lost, you should run the `shutacct` command before the `halt` or `reboot` commands.

3.1.5.3 Managing the log size

One final aspect of the implementation of process accounting is managing the size of /var/adm/pacct. The kernel does not control the file’s size; it just appends accounting records until the file system fills up, at which point accounting is automatically turned off. The accounting system, though, provides the `/usr/sbin/acct/ckpacct` command, a script that, if ran periodically by cron, checks the disk space remaining in the file system where

/var/adm/pacct resides. If the free space is less than 1000 blocks, the script turns process accounting off. The script also checks the size of the pacct file. If the file gets larger than 500 blocks (a default that can be changed), the script renames the file to /var/adm/pacct*i*, where *i* is an integer starting at 1, and switches accounting to a new and empty /var/adm/pacct file. This script makes use of the `turnacct` command to make the switch.

3.1.6 Disk accounting

The accounting system collects the number of disk blocks allocated by the users' files as an offline procedure. This is implemented by the `/usr/sbin/acct/dodisk` command, a script that is executed by cron. This script accesses the file systems you specify, gathers the disk-usage information, and stores it into the /var/adm/acct/nite/dacct file. This file is in the tacct format.

3.1.6.1 Fast mode

The dodisk script provides two ways to gather the disk-usage data. The default is the fastest and most preferred implementation; it is shown in Figure 13.

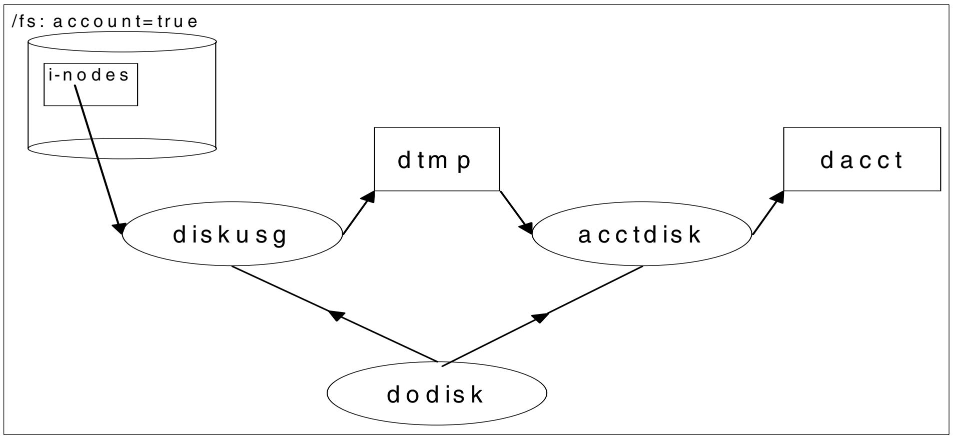


Figure 13. Gathering of disk accounting data (fast mode)

In fast mode, the dodisk script searches for all local file systems that have the attribute account set to true in the /etc/filesystems file. The script then calls the `/usr/sbin/diskusg` command with those file systems as the parameters. The diskusg program reads each file system's i-node table and generates disk-usage data per user. This data is written into the /var/adm/dtmp, a text file containing the user's uid, login name, and number of disk blocks in use.

After the diskusg program exits, dodisk executes the `/usr/sbin/acct/acctdisk` command that transforms the dtmp data into tacct format, generating the /var/adm/acct/nite/dacct file.

3.1.6.2 Slow mode

The default mode of operation of dodisk works fine if your disk accounting needs are restricted to whole, local file systems. But if you need to gather disk-usage information across remote NFS or DFS file systems, or maybe collect information from a subset of a local file system, you need to set up the slow mode operation of the dodisk script (the -o flag).

In slow mode, you specify the directories to be searched as parameters to the dodisk script. It enumerates all files in those directories (using the `find` command) and feeds those file names to the `/usr/sbin/acct/acctdusg` command, which generates the /var/adm/dtmp file. Then the acctdisk program is invoked to generate the /var/adm/acct/nite/dacct file. This whole process is shown in Figure 14.

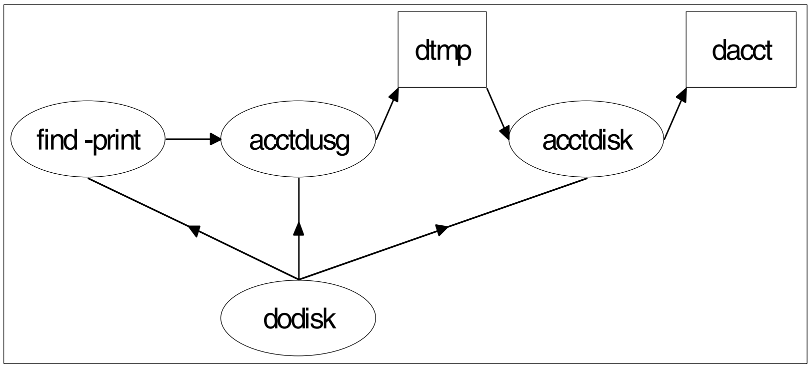


Figure 14. Gathering of disk accounting data (slow mode)

3.1.7 Queue accounting

The print-usage information (number of printed pages) is gathered by the queueing daemon (`qdaemon`) at the end of the print job. Queue accounting, such as process accounting, is not collected by default. For usage information to be collected for printing, you have to specify in the /etc/qconfig file the name of the printer's accounting file. The accounting file contains the user's name, the job's date, and the number of pages printed. The structure of the qacct file can be found in Appendix B.4, "The qacct file" on page 155.

Requirement

When gathering data for the AIX accounting system, you should have a *single* queue accounting file for *all* printers. This file should be named /var/adm/qacct.

3.1.7.1 Queue accounting limitations

You should know that printer accounting does not work in all printing environments. Therefore, a more in-depth explanation of how the queueing system calculates the number of pages printed by a job is in order. You will then be able to verify whether the system will or will not collect 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 queueing system. It is called by /usr/lib/lpd/piobe and 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). Now, 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.

So, in which situations does printer accounting function? When 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, since the transformation from ASCII to PostScript is done by the `/usr/bin/enscript` command, not by pioformat.

You should be aware of one further detail. 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.

3.1.8 Consolidation of the accounting data

The `/usr/sbin/acct/runacct` command is the central component of the accounting system. The runacct script gathers usage information from the files described in the preceding sections and generates several accounting files. This script should be run daily by cron. During normal processing, the runacct script is run without parameters. The script's parameters are used to restart the script after an unexpected error, as described in Section Section 3.4.5, “Restarting runacct” on page 104.

We will summarize the main actions of the runacct script in the following sections. Note that most files used by the command are in the /var/adm/acct directory, while a few can be found in the /var/adm directory. The /var/adm/acct directory is composed of three subdirectories:

- The nite subdirectory contains temporary files (files created and removed during the execution of runacct) and transient files (files that are removed during the script’s next execution).
- The sum subdirectory contains the permanent output files of the runacct script.
- The fiscal subdirectory contains the files generated by the `/usr/sbin/acct/monacct` command, described in Section Section 3.1.9, “Monthly accounting” on page 63.

In the following discussion, all references to the accounting files are relative to the /var/adm/acct directory, unless otherwise indicated.

3.1.8.1 The execution states

The execution of the runacct script is divided into precisely defined sequential steps. Each execution step (state) is named, and the script writes onto the nite/statefile file the step that it is currently executing. In this way, troubleshooting and recovery can be more easily achieved; the contents of nite/statefile always tell you at which point in its execution of `runacct` that it stopped, and you can set the script at which point (state) you want to restart its execution. Troubleshooting is discussed in Section Section 3.4.5, “Restarting runacct” on page 104.

The runacct script starts execution by making some initial checks. For example: it checks whether it is already running or whether it has already been run that day. Then it sets the state to SETUP and enters its first execution step.

3.1.8.2 The SETUP state

In this step, the runacct script freezes the dynamically generated connection and process data.

Using the turnacct program, the script renames the /var/adm/pacct file, creating a new and empty process accounting file. Then all current files in the form /var/adm/pacct*i* are renamed to /var/adm/Spacct*i.date*, where *date* is the current date in the *mmdd* format. In this manner, process accounting files from different days can be differentiated in case of an unexpected problem.

The script also copies the /var/adm/wtmp file as nite/wtmp.*date* and creates a new and empty wtmp file. Then, using the `/usr/sbin/acct/acctwtmp` command, it writes marker records to both nite/wtmp.*date* and /var/adm/wtmp to account for the open sessions at the time the wtmp file was frozen. The information about the open sessions is taken from the /etc/utmp file.

3.1.8.3 The WTMPFIX state

The next step of the runacct script adjusts the connection data in nite/wtmp.*date*. To do so, it calls the `/usr/sbin/acct/wtmpfix` command. This program checks whether the login names are printable or not. If a non-printable character is found, the corresponding login name is changed to INVALID. It also searches the input file for date change records. Then, on a second pass through the file, it adjusts the time of the file records, so those time changes are taken into account. The output of the command is the nite/tmpwtmp file.

3.1.8.4 The CONNECT1 state

In this step, runacct transforms the login and logoff records in the nite/tmpwtmp file into session records, calculating the time each user was connected to the system. All the work in this step is done by the `/usr/sbin/acct/acctcon1` command. This program, takes the nite/tmpwtmp file as input and generates three files:

- The nite/ctmp file is a text file with the user sessions for the day. Each session record contains the session time (in seconds) during prime-time hours and during non-prime-time hours.

> **Note**
>
> The acctcon1 program, as well as the acctprc1 and acctcms programs, read the /etc/acct/holidays file to distinguish between prime and non-prime hours. The usage of this file is discussed in Section Section 3.2.9, “Defining the billing periods” on page 76.

- The nite/lineuse file is a text file with a summary of terminal usage. For each terminal used during that day, the file contains usage statistics, such as the number of minutes the terminal was in use.
- The nite/reboots file is a text file with a summary of the exception records found in the nite/tmpwtmp file. This file contains the number of occurrences of date change, boot, and accounting records. Those accounting records are markers written by the runacct script, and the acctcon1 program, to correctly account for the sessions open at the time the /var/adm/wtmp file was frozen.

3.1.8.5 The CONNECT2 state

Now the runacct script calls the `/usr/sbin/acct/acctcon2` command to transform the information in the nite/ctmp file into the tacct format. The acctcon2 program stores the user's uid and name, as well as the session's connection times (prime and non-prime) in minutes, generating the nite/ctacct.*date* file.

3.1.8.6 The PROCESS state

This next step transforms the process accounting files into tacct format. For each process accounting file (/var/adm/Spacct*i.date*) the script calls the `/usr/sbin/acct/acctprc1` command. This program reads the process accounting files and generates a text output with the user's uid and login name, CPU times (prime and non-prime) in seconds, average memory used by the program in KB, number of characters transferred, and number of disk blocks read or written. The acctprc1 command gets the user's login name from the nite/ctmp file. The program's output is then fed into the `/usr/sbin/acct/acctprc2` command, which transforms that data into tacct format, generating the file nite/ptacct*i.date*.

3.1.8.7 The MERGE state

The runacct script now merges the connection time data (nite/ctacct.*date*) with the process usage data (ptacct*i.date*) into a single tacct file, nite/daytacct. The merge is done by calling the `/usr/sbin/acct/acctmerge` command.

3.1.8.8 The FEES state

In this step, extra service charges are merged into the nite/daytacct file. The extra services are kept in the /var/adm/fee file. The /var/adm/fee file is a text tacct file, which the acctmerge program merges with the nite/daytacct file.

3.1.8.9 The DISK state

Now the disk accounting file, nite/dacct, is merged into the nite/daytacct file. After the merge, the runacct script removes the nite/dacct file.

3.1.8.10 The QUEUEACCT state

In this step, the queue accounting file, /var/adm/qacct, is merged into the nite/daytacct file. Then the /var/adm/qacct is removed and a new, empty file is created.

At this point, the total accounting file for the day is completed. Figure 15 on page 61 shows a summary of the generation of that file and other important output files in the /var/adm/acct/nite directory.

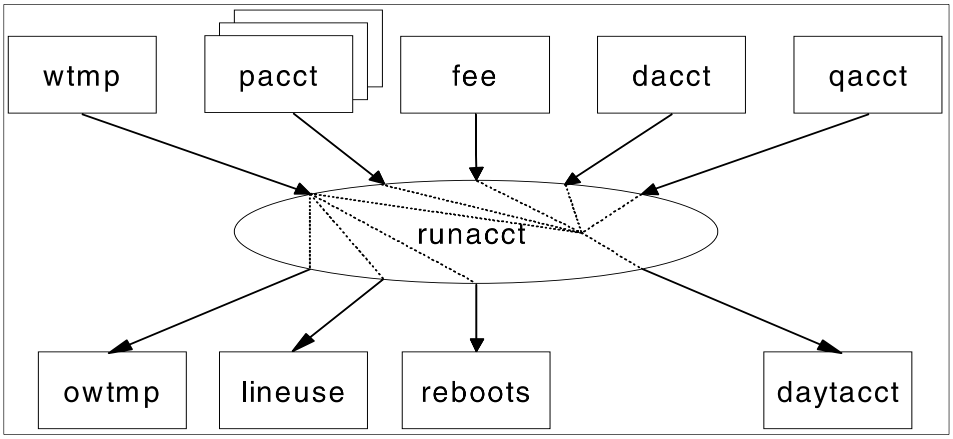


Figure 15. Generation of the /var/adm/acct/nite/daytacct file

3.1.8.11 The MERGETACCT state

In this state, the runacct script updates the sum subdirectory with the day's accounting information. First the nite/daytacct file is copied to sum/tacct*date*. Then nite/daytacct is merged into sum/tacct, the file containing the accumulated usage data. The script saves the previous version of sum/tacct in the file sum/tacctprev.

3.1.8.12 The CMS state

At this point the script generates two different kinds of accounting files: The command summaries and the last login information.

The command summary files

The total accounting files generated by the accounting system gather together the usage information per user. On the other hand, the command summary files contain total usage information per command. This total usage information includes:

- The name of the command
- The number of times the command was executed
- Total prime and non-prime CPU time in minutes
- Total prime and non-prime real (elapsed) time in minutes
- Total prime and non-prime memory usage in KB times minutes
- Total prime and non-prime characters transferred
- Total prime and non-prime disk blocks read and written

The exact structure of the command summary files can be found in Appendix B.5, “The cms file” on page 155.

The runacct scripts uses the `/usr/sbin/acct/acctcms` command to create the command summary files. First it creates the sum/daycms file from the Spaccti*.date* files. Then it merges the day's command summary into sum/cms, the file containing the accumulated command summary. The previous contents of sum/cms are saved in the sum/cmsprev file.

Finally, the scripts generates two command summary reports in the nite directory: the nite/daycms text file with the 50 commands that consumed the most memory that day, and the nite/cms text file, which contains the 50 commands that consumed the most memory overall.

The last login information

The runacct script also generates a file with the date of the last login for all system users. The accounting system needs to generate this file; otherwise, the last login information AIX normally maintains through the use of the /var/adm/wtmp file would be lost. (The `last` command uses the /var/adm/wtmp to get the last login information, and this file is removed by runacct.)

The work is done by the `/usr/sbin/acct/lastlogin` command, that updates the sum/loginlog file with the connection information in the nite/ctacct.*date* file. The sum/loginlog is a text file containing the date of last login (00-00-00 if the user never logged in) and the user's login name. It is sorted by login date.

Figure 16 describes (ignoring some of the intermediate files utilized) the generation of the files in the sum directory by the previous two steps.

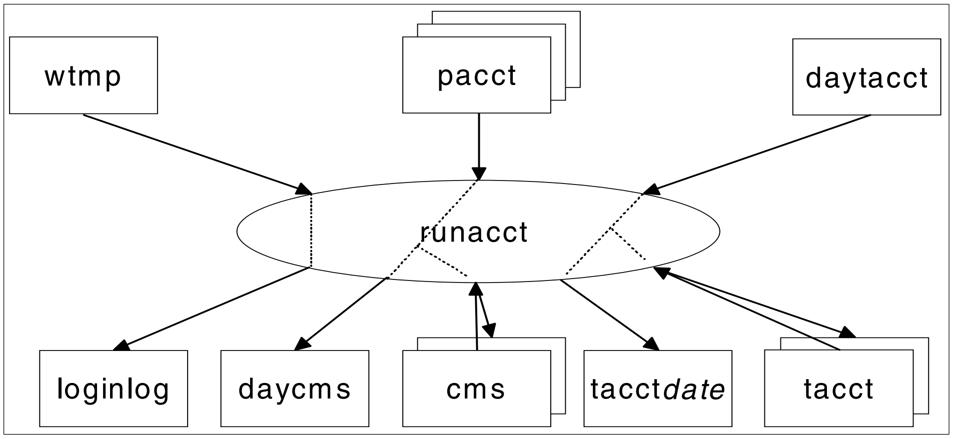


Figure 16. Generation of the sum directory files

3.1.8.13 The USEREXIT state

This step executes the `/var/adm/siteacct` command (if it exists). By creating the resultant file, you can add your own accounting procedures to the daily script.

3.1.8.14 The CLEANUP state

In this final step, the script removes most of the intermediate files created during its execution. A significant exception is the nite/tmpwtmp file, which is renamed to nite/owtmp. Thus, the nite/owtmp file contains the previous day's connection data.

The scripts also removes the /var/adm/fee file, creating a new and empty one.

At this point, the script executes the `/usr/sbin/acct/prdaily` command, which creates a report named sum/rprt*date*. This file contains all the relevant data gathered by runacct:

- The contents of the nite/reboots file
- The contents of the nite/lineuse file
- The contents of the nite/daytacct file as generated by the `prtacct` command
- The contents of the nite/daycms file
- The contents of the nite/cms file
- The contents of the sum/loginlog file

> **Note**
>
> You should note that, with the exception of the contents of the nite/daytacct file (saved as sum/tacct*date*), the sum/rprt*date* file saves the contents of files that would otherwise be lost, that is, overwritten during the next execution of the `runacct` command.

3.1.8.15 End of runacct

The script terminates, indicating that it has completed execution by writing the string COMPLETE in nite/statefile.

3.1.9 Monthly accounting

With the `runacct` command, you gather daily usage data. But accounting data is usually more useful when collected for longer periods, either for billing purposes or just to aggregate the usage data. The accounting system allows you to periodically gather total accounting data and command summaries. Usually this procedure is done once a month through cron, but you could program it to run with a different period.

The fiscal periodic accounting is implemented by the `/usr/sbin/acct/monacct` command. This script copies the sum/tacct and sum/cms files to the fiscal subdirectory. The files are copied, respectively, with the name fiscal/tacct*mm* and fiscal/cms*mm*, where *mm* is the value of the current month. Then the script removes the original files, creating new and empty ones. The script also removes all daily files from the sum directory, that is, all files with name starting with sum/tacct and sum/rprt.

The monacct script's last action is to generate a report for the fiscal accounting period, fiscal/fiscrpt*mm*. This file contains:

- The total accounting report for the period
- The command summary for the period
- The last login information

The generation of the files in the fiscal subdirectory is shown in Figure 17.

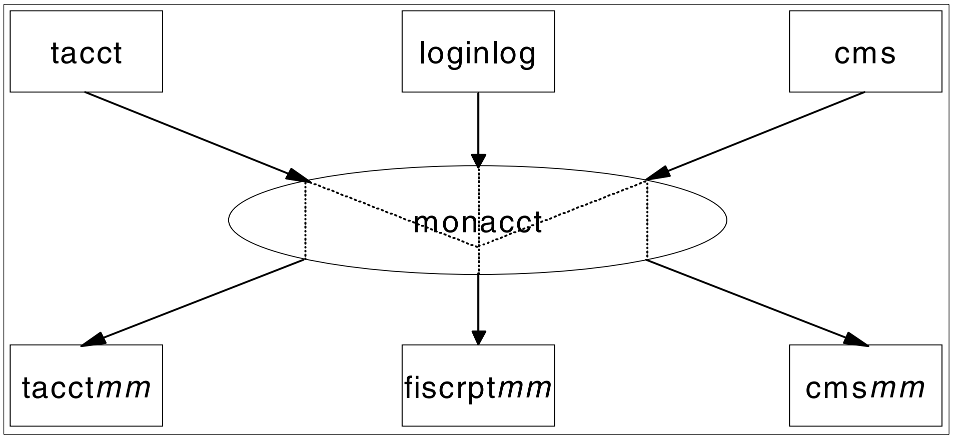


Figure 17. Generation of the fiscal subdirectory files

3.2 Setting up accounting

In this section, we will describe the steps required to start the accounting system. For in-depth justifications for the required setup procedures, please refer to Section Section 3.1, “Inside accounting” on page 49.

To execute the steps below, you should log in as root. Note, however, that the accounting procedures run under the adm user.

3.2.1 Installing the fileset

You should initially verify if the appropriate fileset is already installed. Use the command:

```
lslpp -L bos.acct
```

If it is not installed, you may install it through the WebSM interface. Run the `wsm` command to start the Launch Pad. Double click the **Software** icon. In the Software window, click **Software -> New Software (Install/Update) -> Additional Software (Custom)**, as shown in Figure 18.

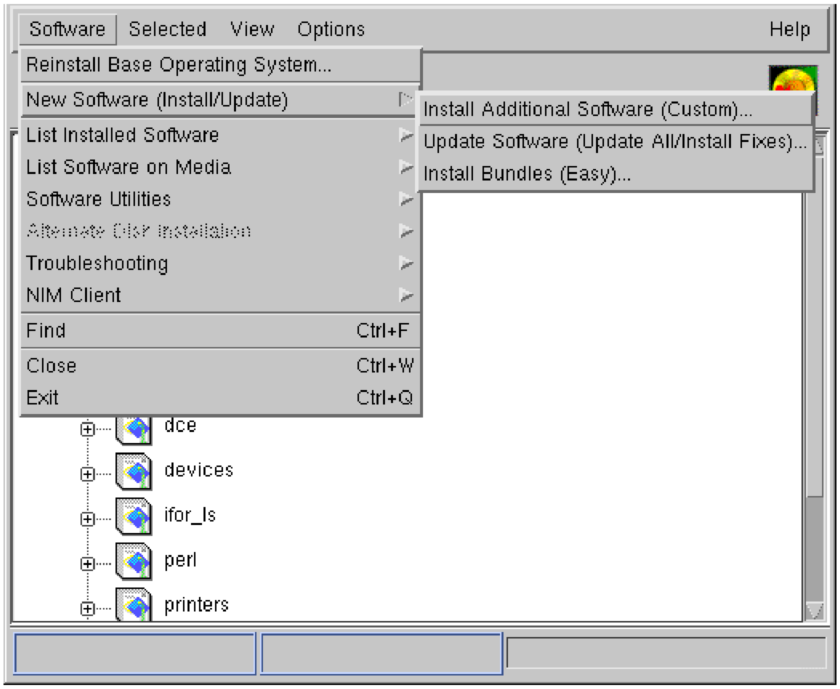
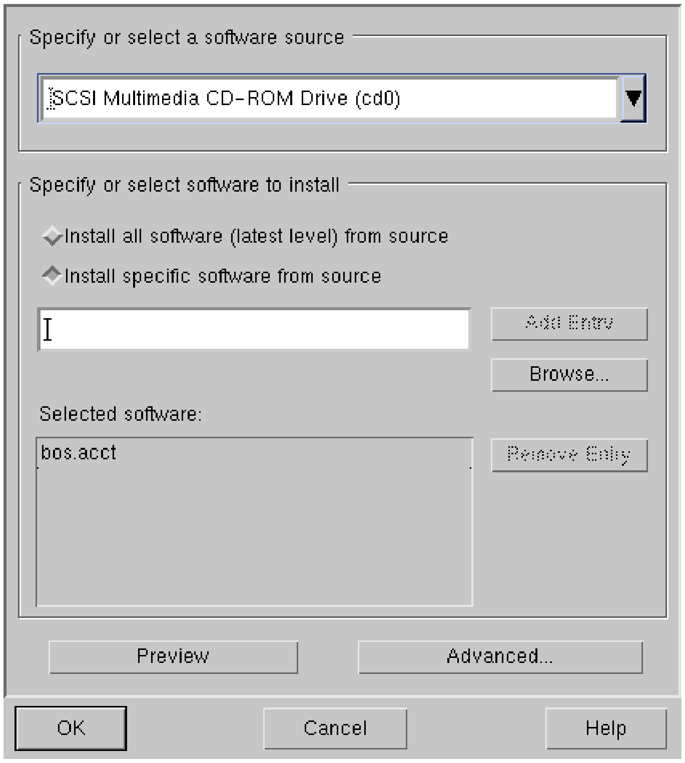

Software
Selected
View
Options
Help
Reinstall Base Operating System...
New Software (Install/Update)
List Installed Software
List Software on Media
Software Utilities
Alternate Disk Installation
Troubleshooting
NIM Client
Find Ctrl+F
Close Ctrl+W
Exit Ctrl+Q
Install Additional Software (Custom)...
Update Software (Update All/Install Fixes)...
Install Bundles (Easy)...
dce
devices
ifor_ls
perl
printers

Figure 18. Selecting to install additional software through WebSM

In the ensuing dialog, select the installation media. Type the fileset to be installed and click **Add Entry** as shown in Figure 19. Click **OK** to start the installation.


Specify or select a software source

SCSI Multimedia CD-ROM Drive (cd0)

Specify or select software to install

◇ Install all software (latest level) from source

◆ Install specific software from source

Selected software:

bos.acct

Figure 19. Selecting the software to be installed

3.2.2 Setting up the environment

Two of the accounting commands, `runacct` and `ckpacct`, can notify you by e-mail when an exception occurs. For this to happen, you have to set the MAILCOM variable in those scripts. That variable has to be set to the command that would send you mail. For example, you could use:

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

An easier way to set that variable is through the /etc/environment file. For example, you could use the following command to set the variable:

```
echo 'MAILCOM=mail root' >> /etc/environment
```

In this manner, you only need to set the variable once, without worrying about which script is using the variable, or about PTFs re-installing the accounting commands.

3.2.3 Creating the working directories

The fileset installation process does not create all the directories the accounting system requires. You should create these directories by hand using the following commands:

```
su - adm
cd acct
mkdir nite fiscal sum
exit
```

3.2.4 Updating crontab entries

All accounting jobs should run under user adm. You can modify user adm’s crontab file with the command:

```
su - adm -c crontab -e
```

If you have not removed or modified the default crontab file for user adm, the entries suggested in the following sections are already in the file, but commented out, as shown below:

```
# su - adm -c crontab -l | tail -12
#=================================================================
#      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 2>/usr/adm/acct/nite/accterr > /dev/null
#0 23 * * 0-6 /usr/lib/acct/dodisk > /dev/null 2>&1
#0 * * * * /usr/lib/acct/ckpacct > /dev/null 2>&1
#15 4 1 * * /usr/lib/acct/monacct > /dev/null 2>&1
#=================================================================
```

You might (when adding an entry) just edit the existing line, removing the comment character and making any modifications you deem necessary.

Note that the sample crontab entries make reference to commands in the /usr/lib/acct directory (the original System V accounting directory), while throughout this redbook we refer to commands in the /usr/sbin/acct directory (the AIX accounting directory). You can easily verify that the files in the /usr/lib/acct directory are just links to the commands in the /usr/sbin/acct directory.

3.2.5 Setting up connection accounting

By default, AIX creates the /var/adm/wtmp file when it is first installed. But when setting up accounting, it can be a good idea to create a new and empty wtmp file. By starting with a fresh wtmp file, you avoid having your connection accounting data being skewed with old session information (assuming your system has been running for some time now).

To create a new wtmp file, you should use the `/usr/sbin/acct/nulladm` command, which removes the file and creates a new one.

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

If you decide not to remove the wtmp file, you should verify whether the file is accessible by the adm user or not. For example:

```
# ls -l /var/adm/wtmp
-rw-rw-r--   1 adm      adm        10240 Jun 14 10:47 /var/adm/wtmp
```

If the output to the `ls` command is not similar to the display above you should change the ownership and permissions appropriately:

```
chown adm.adm /var/adm/wtmp
chmod ugo+r /var/adm/wtmp
chmod ug+w /var/adm/wtmp
```

Note

If you remove the /var/adm/wtmp file, the last login information stored in that file will be lost. To keep that information, you can copy the contents of /var/adm/wtmp to another file, and only then run the `nulladm` command to create a new and pristine /var/adm/wtmp file. Furthermore, if you want to add the existing login data to the loginlog file (described in Section Section 3.1.8.12, “The CMS state” on page 61), you can use the following perl script to do so:

```
#!/usr/bin/perl
die 'file name expected' unless $#ARGV != 1;
open WTMP,$ARGV[0] or die 'could not open wtmp file';
while (sysread(WTMP,$rec,64) == 64) {
    ($user,$id,$line,$type,$pid,$termination,$exit,$time,$host) =
        unpack 'A8A14A12sis2IA16',$rec;
    next unless $type == 7;
    $time{$user} = $time if $time > $time{$user};
}
open SORT,'| sort' or die 'could not pipe sort';
select SORT;
foreach $user (keys %time) {
    $time = $time{$user};
    ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
        localtime $time;
    $year = sprintf '%02d',substr($year + 1900,-1,2);
    $mon = sprintf '%02d',$mon+1;
    print "$year-$mon-$mday  $user\n";
}
close SORT;
```

This perl script accepts the name of the wtmp file as its only parameter and produces the loginlog file on standard output. You can use the script (named script) as follows:

```
perl script /var/adm/wtmp > /var/adm/acct/sum/loginlog
```

3.2.6 Setting up process accounting

You can start process accounting by issuing the `/usr/sbin/acct/startup` command. The startup script creates the /var/adm/pacct file (if it does not exist) and calls the `turnacct` command to turn on process accounting. It also writes a record to the /var/adm/wtmp file.

This procedure has to be executed on every system boot; therefore, the command should be added to one of the system initialization scripts. Note that the earlier the `startup` command is run, the better the accounting of resources used by the system initialization processes.

Thus, you should add the following line to the /etc/rc file.

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

An appropriate location for the new line is shown in the following display (for AIX 4.3.3):

```
# tail -15 /etc/rc
#MSG=`dspmsg rc.cat 7 'Write system start up record to /usr/adm/sa/sa'``date +%d
`
#echo $MSG
#/usr/bin/su - adm -c /usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`

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

# Manufacturing post install process.
# This must be at the end of this file, /etc/rc.
if [ -x /etc/mfg/rc.preload ]
then
        /etc/mfg/rc.preload
fi

dspmsg rc.cat 5 'Multi-user initialization completed\n'
exit 0
```

Note

If you want to start process accounting without rebooting your system, you should run the `startup` command from the command line.

Adding the crontab entry

You should also add an entry like the following to user adm's crontab file:

```
0 * * * * /usr/sbin/acct/ckpacct > /dev/null 2>&1
```

The entry makes the `ckpacct` command run every hour on the hour. The `ckpacct` command manages the space used by the process accounting file as described in Section Section 3.1.5.3, "Managing the log size" on page 54.

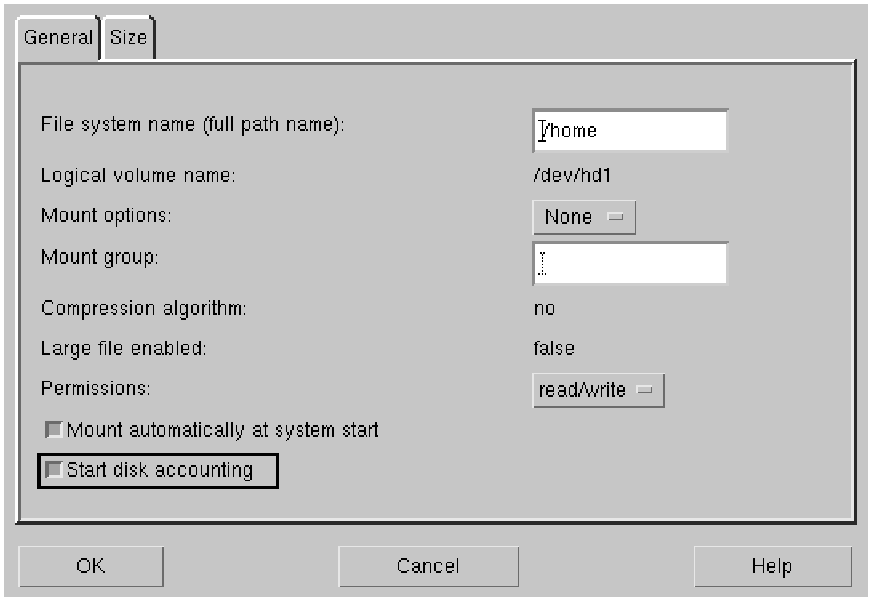
3.2.7 Setting up disk accounting

Disk accounting can be set up in one of two ways, depending on whether the directories you want to account for correspond exactly to one or more local file systems or not. If you want to gather disk space usage of local file systems only, you should start fast mode disk accounting. Otherwise, you must start slow mode disk accounting.

3.2.7.1 Setting up fast mode disk accounting

To start fast mode disk accounting, you should set the account attribute to true for all file systems to be accounted.

To set that attribute through the WebSM interface, double click the **File Systems** icon in Web-based System Management's launch pad. Double click the desired file system. Select the **Start disk accounting** option as shown in Figure 20. Click **OK** to apply your changes.



File system name (full path name): /home

Logical volume name: /dev/hd1

Mount options: None

Mount group:

Compression algorithm: no

Large file enabled: false

Permissions: read/write

□ Mount automatically at system start

□ Start disk accounting

Figure 20. Configuring disk accounting through WebSM

The above actions create a new entry in the /etc/filesystems stanza for the file system, as shown in the following display:

```
# grep -p home /etc/filesystems
/home:
        dev             = /dev/hd1
        vfs             = jfs
        log             = /dev/hd8
        mount           = true
        check           = true
        vol             = /home
        free            = false
        account         = true
```

Adding the crontab entry

Now that all file systems are correctly configured, you should add the following cron job that gathers disk accounting every evening:

```
0 23 * * 0-6 /usr/sbin/acct/dodisk > /dev/null 2>&1
```

That entry tells cron to start the `dodisk` command every night at 11:00 PM, discarding any possible output.

3.2.7.2 Setting up slow mode disk accounting

If accounting for only the local file systems does not attend to your needs, you must add an entry like the following to the user adm's crontab file:

```
0 23 * * 0-6 /usr/sbin/acct/dodisk -o /home > /dev/null 2>&1
```

With such an entry, the `dodisk` command gathers disk-usage data in the /home directory, with no consideration about where or how that directory is implemented. You can also pass more the one directory parameter to the `dodisk` command.

Restriction

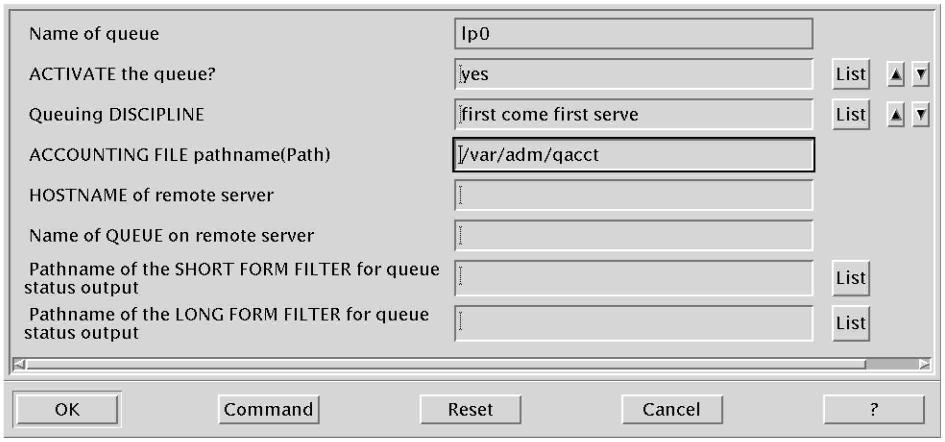
In AIX's current implementation, you cannot mix both type of disk accounting methods. For example, if you need to account the contents of a local file system and the contents of a remote file system, you must use slow mode accounting, passing the mount points of both file systems as parameters to the `dodisk` command.

3.2.8 Setting up queue accounting

To start accounting for a printer, you should specify that the name of the printer queue accounting file is /var/adm/qacct. You may use the `smit` command to do so:

```
smit pq_chque
```

After entering the name of the queue, you are presented with the window in Figure 21. Enter the name of the accounting file as shown and click **OK** to apply your changes.



Name of queue	lp0
ACTIVATE the queue?	yes
Queuing DISCIPLINE	first come first serve
ACCOUNTING FILE pathname(Path)	/var/adm/qacct
HOSTNAME of remote server	
Name of QUEUE on remote server	
Pathname of the SHORT FORM FILTER for queue status output	
Pathname of the LONG FORM FILTER for queue status output	

Figure 21. Specifying the queue accounting file

The specification of the accounting file is saved in the queue's /etc/qconfig stanza as shown below:

```
# tail -9 /etc/qconfig
lp0:
        acctfile = /var/adm/qacct
        device = lp0
lp0:
        file = /dev/lp0
        header = never
        trailer = never
        access = both
        backend = /usr/lib/lpd/piobe
```

You can also edit the /etc/qconfig file instead of using `smit`, but you should not forget to use the following command to re-digest the qconfig file after you are done modifying it:

```
enq -d
```

3.2.8.1 Creating the queue accounting file

You should create the qacct file with the following command:

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

The `nulladm` command creates a new qacct file with the appropriate permissions.

If you have not created the qacct by hand as previously discussed, the queueing system creates the file automatically but with the *wrong* permissions, as shown below:

```
# ls -l /var/adm/qacct
--w-r-xr--   1 root     printq       268 Jun 15 12:02 /var/adm/qacct*
```

Those default permissions allow the accounting system to read the printer-usage data, but do not allow the `runacct` command to create a new and empty qacct file.

3.2.8.2 Does queue accounting work?

As described in Section Section 3.1.7.1, “Queue accounting limitations” on page 57, accounting only functions with PCL or ASCII printers, and furthermore, only if those printers have the appropriate backend processor.

In the specific case of the printer shown in the preceding example, printer-usage information is collected. It is a local printer and uses the standard backend processor, /usr/lib/lpd/piobe. Piobe uses the pioformat program to format the output.

Now, if you have a standard remote printer like the one defined in the display below, accounting does not work:

```
# tail -9 /etc/qconfig
rm0:
        device = @prt3825
        up = TRUE
        host = prt3825
        s_statfilter = /usr/lib/lpd/aixshort
        l_statfilter = /usr/lib/lpd/aixlong
        rq = draft1
@prt3825:
        backend = /usr/lib/lpd/rembak
```

Notice that the printer uses the /usr/lib/lpd/rembak backend processor, which does not use the pioformat program. The same configuration would have been obtained if the remote printer was defined with NFS access to the server’s print queue attributes.

You should be aware that if you define a remote (PCL or ASCII) printer with local filtering before sending to the print server, accounting works, since the formatting is done locally (by pioformat). The configuration of such a printer looks something like the following display:

```
# tail -12 /etc/qconfig
text1:
        acctfile = /var/adm/qacct
        device = @prt3825
        host = prt3825
        rq = draft1
        s_statfilter = /usr/lib/lpd/aixshort
        l_statfilter = /usr/lib/lpd/aixlong
@prt3825:
        header = never
        trailer = never
        access = both
        backend = /usr/lib/lpd/pio/etc/piorlfb -f +
```

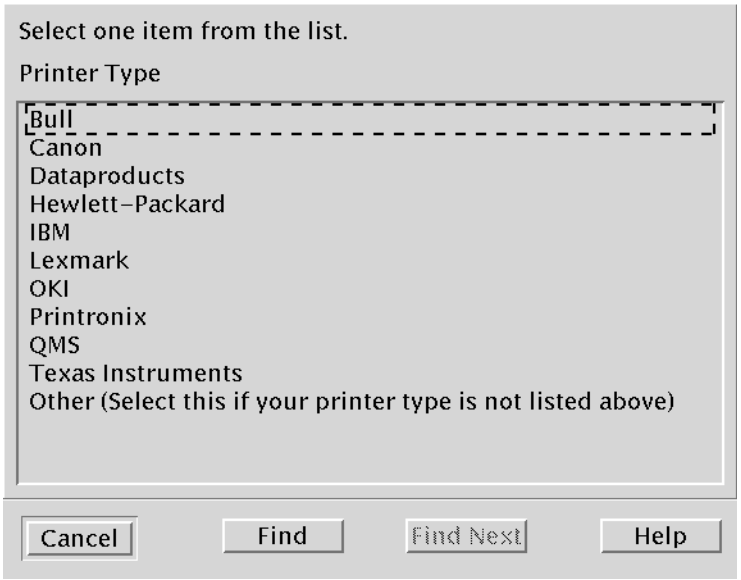
Note that this kind of printer uses a different backend processor, which calls both the piobe and the rembak programs.

An equivalent implementation can be found in IBM Network Printers, such as the IBM Network Printer 24. The configuration of such a printer is shown in the following display:

```
# tail -9 /etc/qconfig
np24pcl:
        acctfile = /var/adm/qacct
        device = ibm@np24
ibm@np24:
        file = /var/spool/lpd/pio/@local/dev/ibm@np24#ibmNetPrinter
        header = never
        trailer = never
        access = both
        backend = /usr/lib/lpd/pio/etc/pioibmnpm np24
```

As before, IBM Network Printers’ backend processor calls piobe before sending the document to the network.

Is there an easy way to figure out if accounting functions, besides experimentation or checking the backend code? Our answer is a tentative yes, if during printer installation, the system asks you for the printer type, then local formatting takes place and accounting should work (for a PCL or ASCII printer, of course). For example, if you are using `smit` to install a printer, the appearance of the window in Figure 22 on page 76 should indicate that printer-usage data could be collected for that printer.



Select one item from the list.

Printer Type

Bull
Canon
Dataproducts
Hewlett-Packard
IBM
Lexmark
OKI
Printronix
QMS
Texas Instruments
Other (Select this if your printer type is not listed above)

Figure 22. Selecting printer type through SMIT

3.2.9 Defining the billing periods

You should modify the /etc/acct/holidays file to indicate which hours are defined as prime-time hours and which are not. The file should also indicate when your company holidays are. The accounting system considers all hours during weekends and holidays non-prime hours.

The following display shows the default contents of the /etc/acct/holidays. The asterisk character at the start of a line denotes a comment.

```
# tail -20 /etc/acct/holidays
* Prime/Nonprime Table for AIX Accounting System
*
* Curr  Prime   Non-Prime
* Year  Start   Start
*
  1990  0800    1700
*
* Day of        Calendar        Company
* Year          Date            Holiday
*
    1           Jan 1           New Year's Day
   50           Feb 19          Washington's Birthday (Obsvd.)
  148           May 28          Memorial Day (Obsvd.)
  185           Jul 4           Independence Day
  246           Sep 3           Labor Day
  326           Nov 22          Thanksgiving Day
  327           Nov 23          Day after Thanksgiving
  359           Dec 25          Christmas Day
  365           Dec 31          New Years Eve
```

The first data line in the file contains three fields:

- The current year
- Beginning of prime time
- End of prime time

Note that times are specified using a 24-hour clock.

The next data lines describe the company holidays for the current year. Each line contains four fields:

- The day of the year
- The month
- The day of the month
- Description of the holiday

You should know that the *only* relevant field in the specification of the holidays is the first field. This field contains the number of the day of the year in which the holiday fails, and must be a number between 1 and 365 (366 if the year is a leap year). The other fields are treated as comments.

A U.S. company sample holidays file for the year 2000 follows:

```
# cat /etc/acct/holidays
  2000  0900    1700
*
    1          Jan 1          New Year's Day
   51          Feb 21         Washington's Birthday (Obsvd.)
  150          May 29         Memorial Day (Obsvd.)
  186          Jul 4          Independence Day
  248          Sep 4          Labor Day
  328          Nov 23         Thanksgiving Day
  329          Nov 24         Day after Thanksgiving
  360          Dec 25         Christmas Day
  366          Dec 31         New Years Eve
```

You should update this file every year; otherwise, chances are that the yearly holidays would be accounted for incorrectly. If you have set the MAILCOM variable, the accounting system sends you a daily e-mail when the holidays file is out-of-date. It starts doing so right after Christmas Day.

3.2.10 Setting up daily accounting

To start the daily accounting process you should add the following line to user adm's crontab file:

```
10 23 * * 0-6 /usr/sbin/acct/runacct 2>/var/adm/acct/nite/accterr > /dev/null
```

This line starts the `runacct` command every day at 11:10 PM. Note that the command must be run after the `dodisk` command has completed. In case you are gathering disk usage from a very large directory over the network, you would need to start the `runacct` command more than ten minutes after the `dodisk` command.

3.2.11 Setting up monthly accounting

To start the monthly accounting job, you should add the following line to user adm's crontab file:

```
15 4 1 * * /usr/sbin/acct/monacct > /dev/null 2>&1
```

This line starts the `monacct` command at 4:15 AM on the first day of every month.

3.3 Reading the accounting files

In this section, we describe the files managed by the accounting system, as well as some of the commands that can be used to examine those files. Please refer to Section Section 3.1, “Inside accounting” on page 49, for details of the inner workings of the accounting system.

For your reference, Table 5 lists the AIX accounting commands that are defined by the System V Interface Definition (SVID) Issue 3.

Table 5. System V accounting commands

Command	Function
acctcms	Produces command usage summaries from the pacct files.
acctcom	Displays selected process accounting records.
acctcon1	Converts login/logoff records to session records.
acctcon2	Converts session records to total accounting records.
acctdisk	Generates total accounting records from output of diskusg/acctdusg.
acctdusg	Generate disk accounting information.
acctmerg	Merges total accounting files into an intermediary file.
accton	Turns on accounting.
acctprc1	Processes accounting information from the pacct files.
acctprc2	Processes the output of acctprc1 into total accounting records.
acctwtmp	Writes accounting records to wtmp.
chargefee	Assigns charges to login name.
ckpacct	Checks size of the pacct file.
diskusg	Generates disk accounting information.
dodisk	Performs disk accounting.
fwtmp	Convert wtmp records to formatted ascii.
lastlogin	Updates the loginlog file with the last date on which each user logged in.
monacct	Creates monthly summary files.
nulladm	Creates an empty accounting file.
prctmp	Print session records as produced by the acctcon1 command.
prdaily	Formats a report of the daily accounting information.
prtacct	Formats and prints a total accounting file.
remove	Removes some temporary accounting files.
runacct	Runs daily accounting.
shutacct	Stops process accounting.

Command	Function
startup	Starts process accounting.
turnacct	Turns process accounting on or off.
wtmpfix	Corrects the time stamps in a file using the wtmp format.

The 4.3 BSD accounting commands provided by AIX are shown in Table 6.

Table 6. BSD accounting commands

Command	Function
ac	Connection time accounting.
accton	Turns process accounting on or off.
last	Indicates the last logons of users and terminals.
lastcomm	Shows, in reverse order, the last commands executed.
pac	Printer usage accounting.
sa	Maintains process accounting files.

We suggest, to be able to easily access the accounting commands, that you add the accounting executable directory, /usr/sbin/acct, to the shell's environment variable PATH. For example, we used the following value in subsequent sections:

```
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin:/usr/sbin/acct
```

3.3.1 The /var/adm directory

A typical listing of the /var/adm directory is shown in the following display, highlighting the relevant accounting files:

```
# ls -lF /var/adm
total 248
-rw-------   1 adm      adm         1052 Jun 05 13:13 .sh_history
drw-r-----   2 root     system       512 Jun 20 10:37 SRC/
drwxrwxr-x   5 adm      adm          512 Jun 01 19:06 acct/
dr-xr-x---   2 bin      cron         512 Jan 27 15:05 cron/
-rw-rw-r--   1 root     system       578 Jan 27 14:55 dev_pkg.fail
-rw-r--r--   1 adm      adm         1400 Jun 19 23:00 dtmp
-rw-rw-r--   1 adm      adm            0 Jun 19 23:10 fee
drwxrwxr-x   2 root     system       512 Jun 20 10:27 nim/
-rw-rw-r--   1 adm      adm       107920 Jun 20 14:37 pacct
-rw-rw-r--   1 adm      adm            0 Jun 09 23:10 qacct
drwxrwxrwt   2 root     system       512 Feb 23 15:58 ras/
drwxrwxr-x   2 adm      adm          512 Jun 01 09:10 sa/
drwxrwxr-x   2 adm      adm          512 Jan 27 14:39 streams/
-rw-------   1 root     system      4694 Jun 20 10:35 sulog
drwxr-xr-x   2 root     system       512 Jan 27 15:24 sw/
-rw-rw-r--   1 adm      adm         3008 Jun 20 11:41 wtmp
```

The acct subdirectory contains most of the accounting files, and its contents are discussed in Section Section 3.3.2, “The nite subdirectory” on page 91, Section Section 3.3.3, “The sum subdirectory” on page 99, and Section Section 3.3.4, “The fiscal subdirectory” on page 101.

3.3.1.1 The dtmp file

The /var/adm/dtmp file is a temporary file used by the dodisk script. The script does not remove the file after being done with it. The dtmp file contains disk accounting information (uid, login name, and number of disk blocks) as shown in the following display:

```
# cat dtmp
   0    root    1
   2    bin     1
 100    guest   1
 201    gonzalo 7729
 202    rose    777
 203    vanel   724
```

3.3.1.2 The fee file

The fee file contains the extra service charges. You should charge extra services through the `/usr/sbin/acct/chargefee` command, which appends the charge to the /var/adm/fee file. The following display shows the use of the `chargefee` command and the resulting fee file:

```
# chargefee gonzalo 100
# cat fee
201 gonzalo 0 0 0 0 0 0 0 0 0 0 0 0 100 0 0 0
```

3.3.1.3 The pacct files

The pacct file contains the current process accounting data. Because the `ckpacct` command keeps this file from growing too large, a busy system may have several pacct files. All but the current file have the name pacct*i*, where *i* represents an integer.

The acctcom command

Most of the contents of the pacct file can be displayed with the `/usr/sbin/acct/acctcom` command. This command has several flags to filter the input and modify the output. Please refer to the *AIX Version 4.3 Commands Reference, Volume 1,* SBOF-1877 for further information. The default output of the `acctcom` command is shown in the following display:

```
# acctcom | head -20

COMMAND                      START    END          REAL      CPU        MEAN
NAME       USER     TTYNAME  TIME     TIME       (SECS)   (SECS)     SIZE(K)
accton     adm      ?       23:10:01 23:10:01     0.02     0.02      252.00
bsh        adm      ?       23:10:01 23:10:01     0.19     0.02      146.00
mv         adm      ?       23:10:01 23:10:01     0.03     0.02      218.00
mv         adm      ?       23:10:01 23:10:01     0.02     0.02       60.00
cp         adm      ?       23:10:01 23:10:01     0.02     0.02      194.00
acctwtmp   adm      ?       23:10:01 23:10:01     0.00     0.00        0.00
fwtmp      adm      ?       23:10:01 23:10:01     0.06     0.02       86.00
awk        adm      ?       23:10:01 23:10:01     0.06     0.03      261.00
sed        adm      ?       23:10:01 23:10:01     0.05     0.00        0.00
fwtmp      adm      ?       23:10:01 23:10:01     0.11     0.02      164.00
cp         adm      ?       23:10:01 23:10:01     0.02     0.00        0.00
chmod      adm      ?       23:10:01 23:10:01     0.02     0.02      192.00
chown      adm      ?       23:10:01 23:10:01     0.03     0.02      153.00
bsh        adm      ?       23:10:01 23:10:01     0.11     0.02      132.00
acctwtmp   adm      ?       23:10:01 23:10:01     0.02     0.02      222.00
fwtmp      adm      ?       23:10:01 23:10:01     0.06     0.02      142.00
awk        adm      ?       23:10:01 23:10:01     0.05     0.02      262.00
```

For each record in the file, the `acctcom` command displays the following fields:

- The name of the command (preceded by the # character if running with super-user privileges).
- The login name of the user who ran the command.
- The controlling terminal of the command, if any.
- The time the command started.
- The time the command ended.

- The number of seconds the command stayed in the system.
- The number of seconds of CPU the command used.
- The average memory size of the command in KB.

The lastcomm command

You can also use the `/usr/bin/acct/lastcomm` command to browse the contents of the pacct file. Note in the following display that, besides showing the records in reverse order, the records are formatted differently:

```
# lastcomm | head -17
dtexec   S     root     __            0.20 secs Tue Jun 20 17:08
dtscreen S     root     __            0.20 secs Tue Jun 20 17:08
dtexec   S     root     __            0.17 secs Tue Jun 20 17:05
dtscreen S     root     __            0.17 secs Tue Jun 20 17:05
dtexec   S     root     __            0.19 secs Tue Jun 20 17:02
dtscreen S     root     __            0.17 secs Tue Jun 20 17:02
dtexec   S     root     __            0.17 secs Tue Jun 20 16:59
dtscreen S     root     __            0.19 secs Tue Jun 20 16:59
bsh            adm      __            0.09 secs Tue Jun 20 17:00
rm             adm      __            0.01 secs Tue Jun 20 17:00
sed            adm      __            0.01 secs Tue Jun 20 17:00
du             adm      __            0.02 secs Tue Jun 20 17:00
awk            adm      __            0.05 secs Tue Jun 20 17:00
df             adm      __            0.01 secs Tue Jun 20 17:00
ln             adm      __            0.01 secs Tue Jun 20 17:00
chmod          adm      __            0.01 secs Tue Jun 20 17:00
cp             adm      __            0.01 secs Tue Jun 20 17:00
```

The `lastcomm` command displays the following values:

- The name of the command.
- Flags giving additional information about the command’s execution:
  S: executed with super-user privileges
  F: ran after a fork, but without an ensuing exec
  C: ran in PDP-11 compatibility mode
  D: terminated with the generation of a core file
  X: was terminated with a signal
- The login name of the user who ran the command.
- The controlling terminal of the command, if any.
- The number of CPU seconds the command used.
- The command’s start time.

Please refer to the *AIX Version 4.3 Commands Reference, Volume 3,* SBOF-1877, for further information about the `lastcomm` command.

The acctcms command

You can also obtain command summaries from the pacct files with the -a flag of the `/usr/sbin/acct/acctcms` command. The `acctcms` command generates a summary of the resources used by each command. By default, the command displays the summaries in decreasing order of total memory usage. A sample output follows:

```
# acctcms -a pacct | 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           872     212.36       1.43   1360.27    148.86      0.00      0.10 9.412e+07      0.00

dtscreen         226      99.25       0.68    678.97    145.40      0.00      0.10 4.861e+07      0.00
dtexec           226      94.68       0.63    679.88    150.92      0.00      0.09 4.216e+07      0.00
bsh               43       4.59       0.03      0.23    161.82      0.00     12.47 529699.00      0.00
awk               14       1.89       0.01      0.01    302.29      0.00     55.81  66769.00      0.00
acctprc1           2       1.34       0.00      0.01    271.63      0.00     57.58 284480.00      0.00
ksh               12       1.09       0.00      0.17    280.00      0.00      2.30 107913.00      0.00
sa                 9       1.08       0.01      0.01    180.83      0.00     76.67 418824.00      0.00
acctcms            8       0.81       0.00      0.01    163.37      0.00     73.08 377568.00      0.00
cut               33       0.65       0.01      0.02     99.32      0.00     31.65 205454.00      0.00
odmget            14       0.56       0.00      0.01    112.74      0.00     42.22 169064.00      0.00
chown             12       0.55       0.00      0.01    151.57      0.00     58.33  39168.00      0.00
rm                21       0.49       0.00      0.01    144.23      0.00     39.39  18592.00      0.00
acctprc2           2       0.46       0.00      0.01    146.67      0.00     34.29 163496.00      0.00
cp                30       0.42       0.00      0.01    107.33      0.00     48.39  16680.00      0.00
```

The output of the `acctcms` command is composed of the following fields:

- `COMMAND NAME`: The name of the command.
- `NUMBER CMDS`: The number of times the command executed.
- `TOTAL KCOREMIN`: The accumulated memory used by the command's executions in KB times minutes.
- `TOTAL CPU-MIN`: The accumulated CPU time for the command's executions in minutes.
- `TOTAL REAL-MIN`: The accumulated time (in minutes) the command stayed in the system.
- `MEAN SIZE-K`: The average memory used by the command in KB.
- `MEAN CPU-MIN`: The average CPU time the command used in minutes.
- `HOG FACTOR`: An indication of how intensively the CPU was used by the command, given by the equation:

  $$\frac{\texttt{TOTAL CPU-MIN}}{\texttt{TOTAL REAL-MIN}} \times 100$$
- `CHARS TRNSFD`: The total number of characters transferred by the command.
- `BLOCKS READ`: The total number of disk blocks read or written by the command.

You may obtain additional information about the `acctcms` command in the *AIX Version 4.3 Commands Reference, Volume 1,* SBOF-1877.

The sa command

In summarizing the process accounting data in the pacct file, you have another alternative: the `/usr/sbin/sa` command. The `sa` command has several flags to specify the information desired in the report, as described in *AIX Version 4.3 Commands Reference Manual, Volume 5,* SBOF-1877. The `sa` command can display the following fields:

- The number of times the command executed.
- The accumulated time (in minutes) the command stayed in the system (`re`).
- The accumulated CPU time used by all the command's executions (in minutes) (`cpu`).
- The accumulated user CPU time used by all the command's executions (in minutes) (`u`).
- The accumulated system CPU time used by all the command's executions (in minutes) (`s`).
- The accumulated memory usage by all the program's executions in KB times seconds (`k*sec`).
- According to the manual, the average KB of CPU-time execution (`k`).
- The total number of characters transferred by all the program's executions (`tio`).
- The average number of characters transferred by a program execution (`avio`).

In the following display we show a sample output of the `sa` command taken immediately after the execution of the `acctcms` command displayed in the previous display. The summary information is practically the same, so you can compare the output of both commands.

```
# sa -K | head -15
     874    1360.28re       1.43cpu    107774avio     12752k*sec
     226     678.97re       0.68cpu    215104avio      5955k*sec    dtscreen
     226     679.88re       0.63cpu    186560avio      5681k*sec    dtexec
      28       0.22re       0.03cpu     18864avio       276k*sec    bsh
      14       0.01re       0.01cpu      4769avio       113k*sec    awk
       2       0.01re       0.00cpu    142240avio        81k*sec    acctprc1
       2       0.17re       0.00cpu     28300avio        66k*sec    ksh
       9       0.01re       0.01cpu     46536avio        65k*sec    sa
       9       0.01re       0.01cpu     48820avio        58k*sec    acctcms
      33       0.02re       0.01cpu      6226avio        39k*sec    cut
      14       0.01re       0.00cpu     12076avio        33k*sec    odmget
      12       0.01re       0.00cpu      3264avio        33k*sec    chown
      21       0.01re       0.00cpu       885avio        29k*sec    rm
       2       0.01re       0.00cpu     81748avio        28k*sec    acctprc2
      30       0.01re       0.00cpu       556avio        25k*sec    cp
```

We used the -K flag to sort the output by the accumulated memory usage, like the `acctcms` command. Note that the first line contains the totals.

You should be careful with the `k` field output by `sa` (not appearing in the previous display). This field is *not* the average memory used by the command. In reality, it is the average memory allocated divided by 100 (the number of clock interrupts per second in the RS/6000).

When using the `sa` command, we advise against using the -s flag, since it will cause the pacct file to be removed afterwards. This flag should only be used if you are not using the standard AIX accounting facilities and you want to keep a command summary file managed by the `sa` command.

The `sa` command can produce other type of reports besides command summaries. For example, the -m flag creates a summary per user, while the -u flag dumps the contents of the pacct file (much like the `acctcom` command). The following two displays show examples of those two kind of reports:

```
# sa -m
root       1767      1.76cpu  106864561tio        14822k*sec
adm         304      0.08cpu    1565259tio          804k*sec
gonzalo     991      0.44cpu    5163845tio         2016k*sec
rose        193      0.06cpu    1165125tio          339k*sec
```

```
# sa -u | head -20
     4   0.02 cpu      252k mem      0 io    accton
     4   0.02 cpu      146k mem   1721 io    bsh
     4   0.02 cpu      218k mem      0 io    mv
     4   0.02 cpu       60k mem      0 io    mv
     4   0.02 cpu      194k mem   1536 io    cp
     4   0.00 cpu        0k mem     64 io    acctwtmp
     4   0.02 cpu       86k mem   5133 io    fwtmp
     4   0.03 cpu      261k mem   7938 io    awk
     4   0.00 cpu        0k mem   1130 io    sed
     4   0.02 cpu      164k mem    885 io    fwtmp
     4   0.00 cpu        0k mem      0 io    cp
     4   0.02 cpu      192k mem      0 io    chmod
     4   0.02 cpu      153k mem   3264 io    chown
     4   0.02 cpu      132k mem   1277 io    bsh
     4   0.02 cpu      222k mem     64 io    acctwtmp
     4   0.02 cpu      142k mem   5133 io    fwtmp
     4   0.02 cpu      262k mem   7938 io    awk
     4   0.00 cpu        0k mem    885 io    fwtmp
     4   0.00 cpu        0k mem   4118 io    dspmsg
     4   0.02 cpu      222k mem     16 io    cat
```

The first column in the report by `sa -u` is the user’s uid. Observe that this report correctly displays the average memory used by each command.

3.3.1.4 The qacct file

The qacct file contains the queue accounting records. The `/usr/sbin/pac` command can be used to extract summary information from that file as shown in the following display. The `pac` command displays the user's login name, the number of pages printed, the number of print jobs, and the total cost of the print jobs.

```
# pac
  Login               pages/feet   runs            price
root                     12.00     6              USD .24
rose                      4.00     2              USD .08

total                    16.00     8              USD .32
```

Note that the `pac` command, by default, reads the accounting file of the default printer (which we assume you set to /var/adm/qacct). Observe also that the price column assumes a price per page which is arbitrarily set by the `pac` command. For further details about the command, please refer to *AIX Version 4.3 Commands Reference, Volume 4,* SBOF-1877.

3.3.1.5 The wtmp file

The wtmp file stores information about the init processes and user sessions, as described in Section Section 3.1.4, "Connection accounting" on page 51.

The fwtmp command

You can examine the contents of the wtmp file (or any file in the utmp format for that matter) with the `/usr/sbin/acct/fwtmp` command. The output of `fwtmp` contains the following fields:

1. Login name for a new session, init process identification for a new process, or empty for a dead session or dead process
2. Init process identification (same as the next column for pseudo-terminal sessions)
3. Terminal device name or additional information
4. Record type:
   a. Init run level is 1
   b. Boot time is 2
   c. Old date is 3
   d. New date is 4
   e. Init process is 5
   f. Login process is 6
   g. User process (logon record) is 7
   h. Dead process (usually logoff record) is 8
   i. Accounting information is 9

5. Process ID
6. Process termination status (for dead process records)
7. Process exit status (for dead process records)
8. Entry time in seconds since 00:00:00 Coordinated Universal Time (UTC), January 1, 1970 (time_t format)
9. Host name (for remote logons)
10. Entry time in display format

A sample output from the wtmp is shown in the following display:

```
# fwtmp < wtmp | head -35
                         openacct       9     0 0000 0000  961474201                  Mon Jun 19 23:10:01 CDT 2000   ①
root     pts/1           pts/1          7 18480 0000 0000  961442686 9.3.240.109      Mon Jun 19 14:24:46 CDT 2000
         pts/0           pts/0          6 17284 0000 0000  961514550 9.3.1.130        Tue Jun 20 10:22:30 CDT 2000
root     pts/0           pts/0          7 17284 0000 0000  961514554 9.3.1.130        Tue Jun 20 10:22:34 CDT 2000
         pts/1           pts/1          8 18480 0000 0000  961514738 9.3.240.109      Tue Jun 20 10:25:38 CDT 2000
                         accting off    9     0 0000 0000  961514873                  Tue Jun 20 10:27:53 CDT 2000   ②
shutdown                 lft0           0     0 0000 0000  961514952                  Tue Jun 20 10:29:12 CDT 2000
LOGIN    dt              lft0           6  5426 0000 0000  961515315                  Tue Jun 20 10:35:15 CDT 2000
                         system boot    2     0 0000 0000  961515327                  Tue Jun 20 10:35:27 CDT 2000   ③
                         run-level 2    1     0 0062 0123  961515327                  Tue Jun 20 10:35:27 CDT 2000
rc       rc                             5  6220 0000 0000  961515327                  Tue Jun 20 10:35:27 CDT 2000
                         AIX, acctg     9     0 0000 0000  961515331                  Tue Jun 20 10:35:31 CDT 2000   ④
         rc                             8  6220 0000 0000  961515331                  Tue Jun 20 10:35:31 CDT 2000
fbcheck  fbcheck                        5  6222 0000 0000  961515331                  Tue Jun 20 10:35:31 CDT 2000
         fbcheck                        8  6222 0000 0000  961515331                  Tue Jun 20 10:35:31 CDT 2000
srcmstr  srcmstr                        5  6224 0000 0000  961515331                  Tue Jun 20 10:35:31 CDT 2000
rctcpip  rctcpip                        5  6458 0000 0000  961515331                  Tue Jun 20 10:35:31 CDT 2000   ⑤
         rctcpip                        8  6458 0000 0000  961515345                  Tue Jun 20 10:35:45 CDT 2000
rcdceclu rcdceclup                      5  6460 0000 0000  961515345                  Tue Jun 20 10:35:45 CDT 2000
         rcdceclup                      8  6460 0000 0000  961515361                  Tue Jun 20 10:36:01 CDT 2000
rcdce    rcdce                          5  6462 0000 0000  961515361                  Tue Jun 20 10:36:01 CDT 2000
         rcdce                          8  6462 0000 0000  961515449                  Tue Jun 20 10:37:29 CDT 2000
cron     cron                           5  8832 0000 0000  961515449                  Tue Jun 20 10:37:29 CDT 2000
qdaemon  qdaemon                        5  8560 0000 0000  961515449                  Tue Jun 20 10:37:29 CDT 2000
         qdaemon                        8  8560 0000 0000  961515453                  Tue Jun 20 10:37:33 CDT 2000
dt       dt                             5  8562 0000 0000  961515453                  Tue Jun 20 10:37:33 CDT 2000
         dt                             8  8562 0000 0000  961515453                  Tue Jun 20 10:37:33 CDT 2000
cons     cons                           5  8564 0000 0000  961515453                  Tue Jun 20 10:37:33 CDT 2000
         pts/0           pts/0          6  5988 0000 0000  961515457 9.3.1.130        Tue Jun 20 10:37:37 CDT 2000
         pts/1           pts/1          6 12222 0000 0000  961515471 ausres01         Tue Jun 20 10:37:51 CDT 2000
root     pts/1           pts/1          7 12222 0000 0000  961515480 ausres01         Tue Jun 20 10:38:00 CDT 2000   ⑥
root     pts/2           pts/2          7  5710 0000 0000  961515482 9.3.240.109      Tue Jun 20 10:38:02 CDT 2000
         pts/1           pts/1          8 12222 0001 0000  961515491                  Tue Jun 20 10:38:11 CDT 2000   ⑦
root     pts/0           pts/0          7  5988 0000 0000  961515515 9.3.1.130        Tue Jun 20 10:38:35 CDT 2000
root     cons            lft0           7  8564 0000 0000  961516113                  Tue Jun 20 10:48:33 CDT 2000
```

To illustrate the contents of the wtmp file, we have numbered and commented some of its records:

1. An accounting marker record, indicating the time the daily accounting period has begun

2. An accounting record, indicating that process accounting has been turned off (because of the system shutdown indicated by the following record)

3. The boot records created by init

4. An accounting record, indicating that process accounting has been turned on (by the rc script)

5. Records indicating the start and end of the rctcpip script (spawned by init)
6. A logon record for user root at pts/1
7. The logoff record for the same session

The who command

You can also use the `/usr/bin/who` command to partially display the contents of the wtmp file. The `who` command uses the /etc/utmp file by default, displaying information about the currently logged users, as described in the *AIX Version 4.3 Commands Reference Manual, Volume 6,* SBOF-1877. To use a different file, you should specify the file's name as a parameter to the command.

The `who` command has several flags that indicate the record types to be output. With the -a flag, you can obtain all possible records the command recognizes. Note, though, that unlike the `fwtmp` command, `who -a` does not show all records in wtmp nor all its fields. A sample output follows:

```
# who -a wtmp
root      - lft0       Jun 20 10:47    old
root      + pts/0      Jun 20 10:47    old
root      + pts/1      Jun 12 18:45     .
root      + pts/3      Jun 20 16:26     .
root      + pts/5      Jun 21 14:05    2:05
LOGIN     + pts/2      Jun 22 13:16     .
root      + pts/2      Jun 22 13:16     .
root      + pts/4      Jun 22 13:16     .
   .        pts/4      Jun 22 13:16     .
   .        pts/2      Jun 22 13:16     .
LOGIN     + pts/2      Jun 22 13:17     .
rose      + pts/2      Jun 22 13:17     .
```

The `who -a` command displays the following fields:

1. Login name for a new session, `LOGIN` for a login process, init process identification for a new process, or empty for a dead process or session
2. Whether the terminal is writable (`+`) or not (`-`)
3. Terminal device name or additional information
4. Entry time
5. Hours and minutes since last activity in the terminal (a dot indicates activity within the last minute, while `old` indicates that the terminal has been idle for more than 24 hours)

Note that the terminal state and terminal activity fields are not obtained from the wtmp file but from the special file associated with the terminal. This means that those fields do not make sense, except for currently opened sessions. But the original purpose of the who command is to report current sessions.

The last command

The last command displays, in reverse chronological order, session information in the wtmp file. You can restrict the output to sessions by a specific user and/or from a specific terminal. You can also specify the file to be read with the -f flag. A sample output follows:

```
# last
rose      pts/2       lv9510a.itsc.aus       Jun 22 13:17 - 13:44  (00:27)
root      pts/4       9.3.240.109            Jun 22 13:16 - 13:16  (00:00)
root      pts/2       ausres01               Jun 22 13:16 - 13:16  (00:00)
root      pts/5       9.3.240.109            Jun 21 14:05   still logged in.
root      pts/3       9.3.240.109            Jun 20 16:26   still logged in.
root      pts/1       9.3.240.109            Jun 12 18:45   still logged in.
root      pts/0       :0.0                   Jun 20 10:47   still logged in.
root      lft0                               Jun 20 10:47   still logged in.

wtmp begins     Jun 21 23:10
```

The last command reports:

- The user’s login name
- The terminal
- The originating host
- The session’s start time
- The session’s end time (if applicable)
- The session’s total time (if applicable)

Refer to the *AIX Version 4.3 Commands Reference, Volume 3,* SBOF-1877, for further information.

The ac command

You can use the ac command to produce connection summary reports from the contents of /var/adm/wtmp. You can produce a report broken down by day (-d flag) or by user (-p flag). In the following display, we show a sample output of the ac command using both flags. The command shows the connection times in hours.

```
# ac -p -d
        root    204.80
Jun 20  total   204.80
        root    379.06
        gonzalo   0.15
        rose      0.10
Jun 26  total   379.31
```

3.3.2 The nite subdirectory

The usual contents of the /var/adm/acct/nite directory are:

```
# ls -l /var/adm/acct/nite
total 40
-rw-r--r--   1 adm      adm            0 Jun 18 23:10 accterr
-rw-rw-r--   1 adm      adm          708 Jun 18 23:10 active
-rw-rw-r--   1 adm      adm         5411 Jun 18 23:10 cms
-rw-rw-r--   1 adm      adm            0 Jun 18 23:10 ctmp
-rw-rw-r--   1 adm      adm         5108 Jun 18 23:10 daycms
-rw-rw-r--   1 adm      adm         6120 Jun 18 23:10 daytacct
-rw-rw-r--   1 adm      adm            5 Jun 18 23:10 lastdate
-rw-rw-r--   1 adm      adm           87 Jun 18 23:10 lineuse
-rw-rw-r--   1 adm      adm            0 Jun 18 23:10 log
-rw-rw-r--   1 adm      adm          128 Jun 18 23:10 owtmp
-rw-rw-r--   1 adm      adm          100 Jun 18 23:10 reboots
-rw-rw-r--   1 adm      adm            9 Jun 18 23:10 statefile
-rw-rw-r--   1 adm      adm            0 Jun 18 23:10 wtmperror
```

3.3.2.1 Execution state of runacct

Four of the files in the nite subdirectory contain information about the execution of the runacct script. They are the accterr, active, lastdate, and statefile files. The accterr file contains the standard error output of the command and is usually empty.

The lastdate file contains the date the command last executed, while the statefile file contains the last state of runacct's last execution:

```
# cat lastdate
0618
# cat statefile
COMPLETE
```

Finally, the active file contains the execution trace of runacct:

```
# cat active
Sun Jun 18 23:10:01 CDT 2000
-rw-r--r--   1 adm      adm         6048 Jun 18 23:00 /var/adm/acct/nite/dacct
-rw-rw-r--   1 adm      adm           64 Jun 17 23:10 /var/adm/wtmp
-rw-rw-r--   1 adm      adm            0 Jun 17 23:10 fee
-rw-rw-r--   1 adm      adm        34760 Jun 18 23:10 pacct
-rw-rw-r--   1 adm      adm            0 Jun 09 23:10 qacct
files setups complete
wtmp processing complete
connect acctg complete
process acctg complete for /var/adm/Spacct1.0618
all process actg complete for 0618
tacct merge to create daytacct complete
no fees
merged disk records
no queueing system records
updated sum/tacct
command summaries complete
system accounting completed at Sun Jun 18 23:10:05 CDT 2000
```

Note that the active file contains a listing of the input files with the raw usage data. It also contains summary line for each execution step.

3.3.2.2 Connection accounting

Six files in the nite subdirectory contain data generated for connection accounting. They are the ctmp, lineuse, log, owtmp, reboots, and wtmperror files. The log file is the standard error output of the `acctcon1` and `acctcon2` commands. The wtmperror file is the standard error output of the `wtmpfix` command. Both files are normally empty.

The owtmp file

The owtmp contains the original contents of the /var/adm/wtmp file run through the `wtmpfix` command, plus some marker records added by `runacct`. The owtmp file is the input file to the `acctcon1` command, which calculates the session times based on the logons and logoffs records in the input file.

A sample output of `fwtmp` is shown in the following display, with the logon and logoff records paired:

```
# fwtmp < owtmp
                         openacct      9     0 0000 0000  961387801                       Sun Jun 18 23:10:01 CDT 2000
root     pts/1           pts/1         7 18082 0000 0000  960853539 9.3.240.109           Mon Jun 12 18:45:39 CDT 2000
         pts/0           pts/0         6 18796 0000 0000  961444981 ausres01              Mon Jun 19 15:03:01 CDT 2000
root     pts/0           pts/0         7 18796 0000 0000  961444982 ausres01              Mon Jun 19 15:03:02 CDT 2000
root     pts/2           pts/2         7 20714 0000 0000  961444983 9.3.240.109           Mon Jun 19 15:03:03 CDT 2000
         pts/0           pts/0         8 18796 0001 0000  961444992                       Mon Jun 19 15:03:12 CDT 2000
         pts/0           pts/0         6 11486 0000 0000  961450473 lv9510a.itsc.aus      Mon Jun 19 16:34:33 CDT 2000
gonzalo  pts/0           pts/0         7 11486 0000 0000  961450476 lv9510a.itsc.aus      Mon Jun 19 16:34:36 CDT 2000
         pts/0           pts/0         8 11486 0000 0000  961457581                       Mon Jun 19 18:33:01 CDT 2000
         pts/0           pts/0         6 11488 0000 0000  961457593 ausres01              Mon Jun 19 18:33:13 CDT 2000
rose     pts/0           pts/0         7 11488 0000 0000  961457597 ausres01              Mon Jun 19 18:33:17 CDT 2000
         pts/0           pts/0         8 11488 0000 0000  961457598                       Mon Jun 19 18:33:18 CDT 2000
                         runacct       9     0 0000 0000  961457613                       Mon Jun 19 18:33:33 CDT 2000
root     pts/1           pts/1         8 18082 0000 0000  960853539 9.3.240.109           Mon Jun 12 18:45:39 CDT 2000
root     pts/2           pts/2         8 20714 0000 0000  961444983 9.3.240.109           Mon Jun 19 15:03:03 CDT 2000
```

Notice two interesting elements of the output. Note that one session (root at pts/1) was already opened at the start of the daily accounting period (indicated by the openacct record). Also note that two sessions (root at pts/1 and root at pts/2) were still opened by the end of the daily accounting period (indicated by the runacct record).

The ctmp file

The ctmp file is the output of the `acctcon1` command, containing the session records for the day. Each session record is composed of:

1. The terminal's device id in dev_t format (for 32-bit machines the device's major number is in the 16 more significant bits, while the device's minor number is in the 16 less significant bits)
2. The user's uid
3. The login name
4. The session time (in seconds) during prime time hours
5. The session time (in seconds) during non-prime time hours
6. The time the session started in time_t format
7. The time the session started in text format

The ctmp file is sorted by uid and login name. The contents of the ctmp file can be displayed directly or through the `/usr/sbin/acct/prctmp` command, which adds a header and pipes the file through the `pr` command. The following display shows the ctmp file generated from the owtmp file previously shown:

```
# cat ctmp
1900544 0       root    10      0       961444982       Mon Jun 19 15:03:02 CDT 2000
1900545 0       root    28800   41012   961387801       Sun Jun 18 23:10:01 CDT 2000
1900546 0       root    7017    5613    961444983       Mon Jun 19 15:03:03 CDT 2000
1900544 201     gonzalo 1524    5581    961450476       Mon Jun 19 16:34:36 CDT 2000
1900544 202     rose    0       1       961457597       Mon Jun 19 18:33:17 CDT 2000
```

The reboots file

The reboots file contains a summary of the exception records found in the owtmp file. The file shows the starting and ending times of the daily accounting period. It also contains the number of occurrences for date change, boot, and accounting records.

```
# cat reboots
from Tue Jun 20 23:10:01 CDT 2000
to   Wed Jun 21 23:10:01 CDT 2000
1       openacct
1       runacct
1       acctcon1
```

The lineuse file

The lineuse file contains a summary of terminal usage. The file initially shows the number of minutes in the daily accounting period. For each terminal used during that day, the file contains the name of the terminal, the total number of minutes the terminal was in use, the percentage of the total time the terminal was in use, the number of user sessions, the number of logons, and the number of logoffs.

An example of the lineuse file is shown in the following display:

```
# cat lineuse
TOTAL DURATION: 1440 MINUTES

LINE    MINUTES PERCENT # SESS  # ON    # OFF
lft0    1440    100     1       1       1
pts/0   1440    100     1       1       1
pts/1   1440    100     1       1       1
pts/2   392     27      1       1       1
pts/3   1440    100     1       1       1
pts/4   0       0       1       1       1
pts/5   544     38      1       1       1
TOTALS  6697    --      7       7       7
```

Note that the number of sessions, logons and logoffs in the example are the same, which is usually the case. To understand the meaning of the different values for those three statistics, you should know how they are generated:

- The number of sessions correspond to the number of user process records associated with the terminal, that is, the number of times a user logged on that terminal.
- The number of logons correspond to the number of times a user logged on an idle terminal.
- The number of logoffs correspond to the number of dead process records associated with the terminal.

Thus, if the number of sessions is larger than the number of logons, this indicates that a user initiated a new logon on that terminal without logging off first. In addition, a number of logoffs larger than the number of sessions suggests that the getty process associated with that terminal was killed before a user managed to log on. This usually happens when there are physical line problems.

3.3.2.3 The daily tacct file

The daytacct file contains all accounting data for the day. This file is in the tacct format. The accounting system provides the `/usr/sbin/acct/acctmerg` command to manipulate the total accounting files. Please refer to the *AIX Version 4.3 Commands Reference, Volume 1,* SBOF-1877 for more information about the `acctmerg` command.

To obtain a report with the contents of tacct files, you should use the `/usr/sbin/acct/prtacct` command, which offers a friendly interface to the `acctmerg` command. A sample output of the `prtacct` command (split in two parts and with all blank lines removed for readability) is shown in the following display:

```
# prtacct -f 1-9 daytacct | grep -v ^$
Wed Jun 21 16:08:40 CDT 2000  Page 1
            LOGIN    CPU       CPU       KCORE     KCORE     BLKIO     BLKIO     RD/WR
UID         NAME     PRIME     NPRIME    PRIME     NPRIME    PRIME     NPRIME    PRIME
0           TOTAL    2         5         1146      3236      74196     56712     68
0           root     2         5         1078      3204      56226     54018     68
2           bin      0         0         0         0         0         0         0
4           adm      0         0         8         27        816       2508      0
100         guest    0         0         0         0         0         0         0
201         gonzalo  0         0         60        3         17154     82        0
202         rose     0         0         0         1         0         104       0
203         vanel    0         0         0         0         0         0         0
# prtacct -f 10-18 daytacct | grep -v ^$
Wed Jun 21 16:09:46 CDT 2000  Page 1
RD/WR     CONNECT   CONNECT   DISK      PRINT       FEES      # OF        # OF   # DISK
NPRIME    PRIME     NPRIME    BLOCKS                          PROCS       SESS   SAMPLES
68        623       870       18468     1           100       2019        5      12
0         597       777       2         1           0         1288        3      2
0         0         0         2         0           0         0           0      2
68        0         0         0         0           0         691         0      0
0         0         0         2         0           0         0           0      2
0         25        93        15460     0           100       38          1      2
0         0         0         1554      0           0         2           1      2
0         0         0         1448      0           0         0           0      2
```

To split the output we used -f flag of the `prtacct` command. You may use this flag to specify the tacct fields to be displayed. The possible fields and their associated numbers are:

1. UID: The user’s uid
2. LOGIN NAME: The user’s login name
3. CPU PRIME: The accumulated CPU time for all the user’s processes in minutes during prime time
4. CPU NPRIME: The accumulated CPU time for all the user’s processes in minutes during non-prime time
5. KCORE PRIME: The accumulated memory used by all the user’s processes in KB times minutes during prime time
6. KCORE NPRIME: The accumulated memory used by all the user’s processes in KB times minutes during non-prime time
7. BLKIO PRIME: The total number of 1024 bytes blocks transferred during prime time
8. BLKIO NPRIME: The total number of 1024 bytes blocks transferred during non-prime time
9. RD/WR PRIME: The total number of disk blocks read or written during prime time
10. RD/WR NPRIME: The total number of disk blocks read or written during non-prime time
11. CONNECT PRIME: The total session times (in minutes) during prime time

12.CONNECT NPRIME: The total session times (in minutes) during non-prime time
13.DISK BLOCKS: The accumulated disk usage in 512 bytes disk blocks
14.PRINT: The total number of printed pages
15.FEES: The total extra service charges
16.# OF PROCS: The number of processes ran by the user
17.# OF SESS: The number of sessions initiated by the user
18.# DISK SAMPLES: The number of times disk usage information was collected for this user

To get a better feeling of what the KCORE fields express, note that the *average* memory usage of the user's processes can be calculated as follows:

$$\frac{\text{KCORE PRIME} + \text{KCORE NPRIME}}{\text{CPU PRIME} + \text{CPU NPRIME}}$$

For example, taking the values in the preceding sample file, the average size of root's processes is 612 KB.

Moreover, to work with disk-usage data, you should use the DISK BLOCKS and # DISK SAMPLES fields. To calculate the average number of disk blocks a user is allocating, you should use the following formula:

$$\frac{\text{DISK BLOCKS}}{\text{\# DISK SAMPLES}}$$

For further information on the `prtacct` command, please refer to the *AIX Version 4.3 Commands Reference, Volume 4,* SBOF-1877.

3.3.2.4 The command summaries

The runacct script creates two command summary reports in the /var/adm/acct/nite directory: The daycms and the cms files. These files are generated by the `acctcms` command.

The daycms file contains a report with the 50 largest programs that ran the previous day, while the cms file contains a report with the 50 largest programs running in the current fiscal accounting period (month). A sample of both files is shown in the following display:

```
# head -20 daycms
                                     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         3215  12922.95     6.60  12374.73   1958.56     0.00      0.05 5.087e+09 573007.00

cpu               5  11850.60     3.01      7.17   3936.19     0.60     41.99  30757.00      0.00
dcecp             2    287.43     0.14      1.74   2106.34     0.07      7.86 1.743e+06      0.00
smitty           16    262.72     0.11      8.30   2491.01     0.01      1.27 4.521e+06      0.00
xlcentry         49    115.35     0.06      0.28   1822.84     0.00     22.92 7.162e+06      0.00
aixterm           2     71.45     0.14   1520.67    494.33     0.07      0.01  4.62e+06      0.00
lsvg             21     49.58     0.03      0.16   1779.28     0.00     17.89 7.062e+07      0.00
vi               77     39.13     0.10    183.68    390.29     0.00      0.05 7.885e+06      0.00
dtgreet           1     33.70     0.04     10.84    905.00     0.04      0.34 348608.00      0.00
ksh             248     32.64     0.16   1761.28    206.48     0.00      0.01 8.471e+06      0.00
X                 1     28.19     0.04     13.19    717.00     0.04      0.30 177600.00      0.00
ps               54     19.19     0.05      0.06    379.94     0.00     91.51 610414.00      0.00
lqueryvg         58     15.05     0.03      0.23    540.09     0.00     11.92 1.213e+08      0.00
io               25     11.66     1.24      2.66      9.37     0.05     46.85 4.492e+09 548416.00
ls               50     10.85     0.04      0.26    291.24     0.00     14.44 645061.00      0.00
# head -20 cms
                                     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         4164  12957.39     6.79  12535.18   1909.45     0.00      0.05 5.172e+09 588607.00

cpu               5  11850.60     3.01      7.17   3936.19     0.60     41.99  30757.00      0.00
dcecp             2    287.43     0.14      1.74   2106.34     0.07      7.86 1.743e+06      0.00
smitty           16    262.72     0.11      8.30   2491.01     0.01      1.27 4.521e+06      0.00
xlcentry         49    115.35     0.06      0.28   1822.84     0.00     22.92 7.162e+06      0.00
aixterm           2     71.45     0.14   1520.67    494.33     0.07      0.01  4.62e+06      0.00
lsvg             21     49.58     0.03      0.16   1779.28     0.00     17.89 7.062e+07      0.00
vi               80     42.28     0.11    211.10    395.01     0.00      0.05 8.077e+06      0.00
ksh             300     40.65     0.19   1761.56    212.95     0.00      0.01 1.339e+07      0.00
dtgreet           1     33.70     0.04     10.84    905.00     0.04      0.34 348608.00      0.00
X                 1     28.19     0.04     13.19    717.00     0.04      0.30 177600.00      0.00
ps               55     19.31     0.05      0.06    378.22     0.00     91.59 632142.00      0.00
lqueryvg         58     15.05     0.03      0.23    540.09     0.00     11.92 1.213e+08      0.00
bsh             346     11.77     0.06      2.46    189.96     0.00      2.52 4.912e+06      0.00
io               25     11.66     1.24      2.66      9.37     0.05     46.85 4.492e+09 548416.00
```

Note that the file cms must have greater values in the TOTALS line than those values in the corresponding line in the daycms file. The cms file is a report of the accumulated command summaries, including the daily command summaries detailed in the daycms file. But in the specific example shown, the accumulated usage is only slightly greater than the day's usage. This is so because we have been running accounting for just a few days, and the day accounted in the daycms file was a particularly busy day.

3.3.3 The sum subdirectory

The typical contents of the /var/adm/acct/sum directory are shown in the following display:

```
# ls -l /var/adm/acct/sum
total 207
-rw-rw-r--   1 adm      adm          6660 Jun 21 23:10 cms
-rw-rw-r--   1 adm      adm          6120 Jun 21 23:10 cmsprev
-rw-rw-r--   1 adm      adm          4020 Jun 21 23:10 daycms
-rw-rw-r--   1 adm      adm           202 Jun 21 23:10 loginlog
-rw-rw-r--   1 adm      adm          7971 Jun 15 23:10 rprt0615
-rw-rw-r--   1 adm      adm         11512 Jun 16 23:10 rprt0616
-rw-rw-r--   1 adm      adm         11512 Jun 17 23:10 rprt0617
-rw-rw-r--   1 adm      adm         11512 Jun 18 23:10 rprt0618
-rw-rw-r--   1 adm      adm         12950 Jun 19 18:33 rprt0619
-rw-rw-r--   1 adm      adm         12984 Jun 20 23:10 rprt0620
-rw-rw-r--   1 adm      adm         13024 Jun 21 23:10 rprt0621
-rw-rw-r--   1 adm      adm           504 Jun 21 23:10 tacct
-rw-rw-r--   1 adm      adm           144 Jun 15 23:10 tacct0615
-rw-rw-r--   1 adm      adm           144 Jun 16 23:10 tacct0616
-rw-rw-r--   1 adm      adm           144 Jun 17 23:10 tacct0617
-rw-rw-r--   1 adm      adm           144 Jun 18 23:10 tacct0618
-rw-rw-r--   1 adm      adm           504 Jun 19 18:33 tacct0619
-rw-rw-r--   1 adm      adm           504 Jun 20 23:10 tacct0620
-rw-rw-r--   1 adm      adm           504 Jun 21 23:10 tacct0621
-rw-rw-r--   1 adm      adm           504 Jun 21 23:10 tacctprev
```

3.3.3.1 The tacct files

The sum subdirectory stores the total accounting files for the current month. The tacct*date* files contain the total accounting records for the corresponding day. The tacct file contains the accumulated total accounting records for the current month, that is, the aggregate of all the tacct*date* files. And finally, the tacctprev file is a copy of the previous contents of the tacct file.

As discussed in Section Section 3.3.2.3, “The daily tacct file” on page 95, you can use the `prtacct` and the `acctmerg` commands to display the tacct files.

3.3.3.2 The command summaries

The sum subdirectory stores the month’s command summary file, cms. The last daily command summary is kept in the daycms file. The cmsprev file is a copy of the previous cms file.

To obtain a report from a cms file, you should use the `acctcms` command with the -a -s flags. A sample output of the command is shown in the following display:

```
# acctcms -a -s cms | 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         4164  12957.39      6.79  12535.18    1909.45      0.00      0.05 5.172e+09 588607.00

cpu               5  11850.60      3.01      7.17    3936.19      0.60     41.99  30757.00      0.00
dcecp             2    287.43      0.14      1.74    2106.34      0.07      7.86 1.743e+06      0.00
smitty           16    262.72      0.11      8.30    2491.01      0.01      1.27 4.521e+06      0.00
xlcentry         49    115.35      0.06      0.28    1822.84      0.00     22.92 7.162e+06      0.00
aixterm           2     71.45      0.14   1520.67     494.33      0.07      0.01  4.62e+06      0.00
lsvg             21     49.58      0.03      0.16    1779.28      0.00     17.89 7.062e+07      0.00
vi               80     42.28      0.11    211.10     395.01      0.00      0.05 8.077e+06      0.00
ksh             300     40.65      0.19   1761.56     212.95      0.00      0.01 1.339e+07      0.00
dtgreet           1     33.70      0.04     10.84     905.00      0.04      0.34 348608.00      0.00
X                 1     28.19      0.04     13.19     717.00      0.04      0.30 177600.00      0.00
ps               55     19.31      0.05      0.06     378.22      0.00     91.59 632142.00      0.00
lqueryvg         58     15.05      0.03      0.23     540.09      0.00     11.92 1.213e+08      0.00
bsh             346     11.77      0.06      2.46     189.96      0.00      2.52 4.912e+06      0.00
io               25     11.66      1.24      2.66       9.37      0.05     46.85 4.492e+09 548416.00
```

Note the output of the `acctcms` command for the /var/adm/acct/sum/cms file is equal to the contents of the /var/adm/acct/nite/cms file (for the first 56 lines, that is).

3.3.3.3 The loginlog file

The loginlog file contains the date of last login (00-00-00, if the user never logged in) and the user’s login name for all the users in the system. It is sorted by login date. The contents of the file are as follow:

```
# cat loginlog
00-00-00  adm
00-00-00  bin
00-00-00  daemon
00-00-00  guest
00-00-00  imnadm
00-00-00  lpd
00-00-00  nobody
00-00-00  sys
00-00-00  uucp
00-00-00  vanel
00-06-19  rose
00-06-20  gonzalo
00-06-21  root
```

3.3.3.4 The report files

The rprt*date* files contain a report for every corresponding date. These reports are generated by the `prdaily` command. Each report contains:

- The contents of the nite/reboots file
- The contents of the nite/lineuse file
- The contents of the nite/daytacct file as generated by the `prtacct` command
- The contents of the nite/daycms file
- The contents of the nite/cms file
- The contents of the sum/loginlog file

3.3.4 The fiscal subdirectory

The /var/adm/acct/fiscal directory contains the command summary file (cms*mm*), the total accounting file (tacct*mm*), and a usage report for every month already passed (fiscrpt*mm*), as exemplified in the following display:

```
# ls -l /var/adm/acct/fiscal
total 2391
-rw-rw-r--   1 adm      adm        15660 Jun 18 23:10 cms06
-rw-rw-r--   1 adm      adm        11820 Jun 21 23:10 cms07
-rw-rw-r--   1 adm      adm       729541 Jun 19 16:25 fiscrpt06
-rw-rw-r--   1 adm      adm        38080 Jun 22 21:40 fiscrpt07
-rw-rw-r--   1 adm      adm       409968 Jun 18 23:10 tacct06
-rw-rw-r--   1 adm      adm         8136 Jun 21 23:10 tacct07
```

These files are identified by the month (the `monacct` command created them). Since `monacct` usually runs the first of each month, the identifications are one number off. This means, for example, that the file cms06 would contain the command summaries for the month of May.

A fiscal report file contains the total accounting report, the command summary report, and the contents of the loginlog file for the corresponding month.

3.4 Troubleshooting

Most of the problems you will encounter with accounting originate from an incorrect setup. The first few days after you have started collecting accounting data, you should carefully check all accounting files. You should also read Section Section 3.1, “Inside accounting” on page 49, paying particular attention to all the steps that generate the accounting files; this should help you solve any problems you encounter.

3.4.1 Detecting errors

The main accounting command, `runacct`, will send you an e-mail for most error situations it encounters (if you have set the MAILCOM variable as explained in Section Section 3.2.2, “Setting up the environment” on page 66). An example of the message sent by the command is shown in the following display:

```
# mail
Mail [5.2 UCB] [AIX 4.1]  Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N  1 adm              Mon Jun 19 23:10  12/366
? 1
Message  1:
From adm Mon Jun 19 23:10:05 2000
Date: Mon, 19 Jun 2000 23:10:05 -0500
From: adm
To: root


************ ACCT ERRORS : see /var/adm/acct/nite/active0619********

?
```

Most of the time, an absence of e-mail messages indicates that accounting is running error-free. But, there is one error situation where `runacct` fails to send out an e-mail message: when the `acctmerg` fails to merge the connection and process accounting files. Also, there are some error conditions that are not checked by `runacct`. For example: if there are permission problems with the qacct file. In conclusion, you should not rely completely on the mail facility of `runacct`.

You should periodically check the contents of the following files in the /var/adm/acct/nite directory:

- The active file, which contains the execution trace of `runacct`
- The accterr file, which is the standard error file for `runacct` and most of the commands it calls
- The log file, which is the standard error file for the `acctcon1` and `acctcon2` commands
- The wtmperror file, which is the standard error file for the `wtmpfix` command (which is renamed to wtmperror*date* if the `wtmpfix` command returns with an error)

Note that the last three files are usually empty. The simple fact that one of those files is not empty should indicate that something is wrong with the accounting process.

On the other hand, if `runacct` detects an error, it will rename the active file (in most cases, at least) to active*date*, where *date* is the current date in *mmdd* format. By renaming the active file, the command tries to guarantee that the error information in the file is not lost, in case the command is run again (the following day) without the error being fixed.

After detecting the error, you should fix the problem and then restart the `runacct` command.

3.4.2 Fixing file permissions

A possible cause for execution errors is incorrect file permissions. You should remember that the accounting procedures run under the adm user (while you are usually using the root account). You have to be careful to ensure that all accounting files allow full access to the adm user.

To create a new accounting file, such as /var/adm/qacct, you should use the `nulladm` command, which creates the file with the correct ownership and permissions. This is shown in the following display:

```
# nulladm qacct
# ls -l qacct
-rw-rw-r--   1 adm      adm            0 Jun 26 17:53 qacct
```

If you have an existing file with incorrect ownership or permissions, you should not run the `nulladm` command, since it always creates a new file. Instead you should manually adjust the file attributes using the following commands:

```
chown adm.adm qacct
(umask 002; chmod 0664 qacct)
```

3.4.3 Fixing the wtmp files

The accounting process terminates in error if the `wtmpfix` command detects a corrupt wtmp file. You can try to fix such problems with the help of the `fwtmp` command.

The input to the `wtmpfix` command is the /var/adm/acct/nite/wtmp.*date* file, where *date* is the date `runacct` ran in *mmdd* format. The first recovery step is to transform the wtmp file into ASCII format:

```
fwtmp < /var/adm/acct/nite/wtmp.date > /tmp/wtmp
```

Then you should edit the /tmp/wtmp file, reviewing its contents and deleting any strange records. After the file is fixed, you should once more use the `fwtmp` command to transform it back to binary wtmp format:

```
fwtmp -ic < /tmp/wtmp > /var/adm/acct/nite/wtmp.date
```

3.4.4 Fixing the tacct files

The accounting system also provides you with a method to fix tacct files. Let us assume that you found a strange looking record on the last daily usage report (file /var/adm/acct/sum/rprt*date*). The previous day report is fine, so you take a look at the last daily total accounting file (using the `prtacct` command on the /var/adm/acct/sum/tacct*date* file) and find out that you have a problem there. For example: a negative CPU usage for one of your users.

Since the problem occurred on the current daily tacct file, you can solve it by first fixing the tacct*date* file and then merging that file with the tacctprev file, generating a new and correct version of the tacct file. To do so, you should take the following steps:

1. Change your current directory:

   ```
   cd /var/adm/acct/sum
   ```

2. Transform the tacct*date* file into ASCII format:

   ```
   acctmerg -v < tacctdate > /tmp/dtacct
   ```

3. Edit the /tmp/dtacct file and fix the suspicious record.
4. Create a new daily tacct file:

   ```
   acctmerg -i < /tmp/dtacct > tacctdate
   ```

5. Generate the new tacct file:

   ```
   acctmerg tacctprev < tacctdate > tacct
   ```

3.4.5 Restarting runacct

After fixing the problem that caused `runacct` to fail, you should restart it.

3.4.5.1 The lock files

Before restarting `runacct`, you should to remove the lock files that the command uses to guarantee its serial execution. The command generates two lock files in the /var/adm/acct/nite directory, lock and lock1. Usually these files are removed when `runacct` exits, even on error conditions. But since there are a few situations where the files are not removed, you should always check for those files and remove them. To remove the files you can use the `/usr/sbin/acct/remove` command, which removes the lock files (as well as some files in the /var/adm/acct/sum directory that are not created nor used by the AIX accounting system).

3.4.5.2 The restart parameters

To restart the `runacct` command, you have to pass the command one or two parameters. The first parameter is the date (in *mmdd* format) you are reprocessing. The second parameter is in which state you want the re-execution to begin.

Arguably, the most common scenario is when you only need to pass a single parameter to runacct, like in the following command:

```
runacct 0629
```

With such a command, you are requesting for the processing of June 29th to be redone. The command will start processing the appropriate files at the state specified in the /var/adm/acct/nite/statefile file. This file contains the value of the last state runacct attempted to execute.

There are two situations where the content of the statefile file is not the correct state to start re-execution. The first one is when, in order to fix the problem, you need to backtrack some states. For example: runacct failed in the CONNECT1 state because of a corrupt wtmp file. To solve the problem you need to fix the wtmp.*date* file and run the wtmpfix program again. After correcting wtmp.*date*, you could use the following command:

```
runacct 0629 WTMPFIX
```

With that command, the statefile file will be ignored and re-execution will start at the state specified.

Another situation where you need to specify two parameters is when some days have passed since the problem first occurred. In such case, cron must have started the command for normal processing one or more times. If normal processing has managed to finish without encountering any errors, the statefile will contain the string COMPLETE. If normal processing terminated with errors, chances are that the last state executed by runacct was not the same state where the original error happened. In any case, you will probably need to specify the state to restart runacct.

3.4.5.3 Sequence of reexecutions

If the execution of runacct has been failing for several days, the accounting system generally allows you to rerun the runacct command for each incomplete day, correctly generating the daily data (except for queue and disk accounting information). This feature is possible because the SETUP state freezes the connection and process accounting files, changing their name to wtmp.*date* or Spacct*i.date*, respectively.

But if you are facing an unlikely problem in or before the SETUP state (usually caused because the previous execution of runacct did not remove the lock files), you will not be able to generate the daily data. Note that you will not necessarily lose any accounting information; the wtmp and pacct files will have that information stored. But since the accounting system will not be able to correctly divide that information by day of usage, all those days' of accounting data will be accounted to a single calendar day.

3.4.5.4 The lastdate file

When you are restarting the `runacct` command, it ignores the contents of the /var/adm/acct/nite/lastdate file. This file contains the date of the last normal execution of the command. You should only worry about this file when, for any reason, you run `runacct` without parameters. This will cause the lastdate file to contain the current date, which in turn will cause the next daily execution of `runacct` to fail, if it happens to be in the same day. (We think that this behavior is correct, that is, we do not see any obvious reason to have the normal execution of `runacct` twice in a day.)

3.5 Sizing considerations

AIX accounting, mainly process accounting, is considered by some systems administrators to impact system performance. That may have been true when UNIX ran on considerably less powerful machines than current RS/6000s. We have run several performance tests, saturating our test machine, and we could not find any significant impact caused by accounting.

On one test, we simultaneously spawned hundreds of very small processes on an F50. The idea was to see if there was a significant delay due to the serialized writing of the process accounting records to the /var/adm/pacct file. We measured the time needed for all processes to finish, with and without accounting. We ran the test several times and observed no significant differences.

The daily accounting processing should not impact your system either. Even if you have a busy system with thousands of users, you can always adjust cron to run the accounting job at the most convenient time (like early in the morning, instead of before midnight).

Disk space could be a problem if you have a very small system with serious disk constraints. But apart from such extreme conditions, we consider that accounting does not demand excessive disk space:

- The accounting directory has limited growth (remember that most files are removed daily, some daily files are removed monthly, and the monthly files are removed yearly).

- Most of the files have a size which is a function of the number of users on your system. But even if you have thousands of users, you would not be needing more than some tens of MB to keep those files.
- On a busy system, the pacct files could be a reason for concern. But each process generates only a 40 byte record, which means that you would have to be processing hundreds of thousands processes per day to fill up the standard /var file system with pacct files.

Chapter 4. Accounting on the SP

If you wanted to have accounting running on your SP system, you could follow the procedures described in Chapter 3, “Accounting on AIX” on page 49, in each and every node on your system. But, even for medium-sized systems, such a procedure would demand too much work. Fortunately, you have two non-mutually exclusive options to quickly implement accounting on your SP, either through PSSP or through LoadLeveler.

4.1 Accounting with PSSP

PSSP provides an automatic procedure to configure and run accounting in all SP nodes. You will realize that it takes less steps to configure and run accounting with PSSP than with standard AIX.

As we will see in the following sections, PSSP accounting does not equal AIX accounting replicated over tens or hundreds of nodes. Accounting on the SP has several distinct characteristics:

- PSSP accounting consolidates the accounting data from all the SP nodes into a single node: the accounting master node (usually the control workstation).
- With PSSP accounting, you can divide your nodes into accounting classes. The accounting master will collect and merge the data for each accounting class individually.
- PSSP allows you charge your users for the exclusive use of a node. A user can ask for exclusive use of a node by indicating in his LoadLeveler job that the node usage is not_shared. PSSP considers the time the node was allocated as extra service charges. PSSP also does not take into account the system resources used by such job steps; otherwise, it would constitute double charging.
- PSSP offers its own versions of the daily and monthly accounting commands, which use a slightly different file and directory structure.

In the following sections, we will describe how to install PSSP accounting and how to analyze the accounting data. Please refer to the *PSSP Version 3.1.1 Administration Guide*, SA22-7348, for further information.

4.1.1 Setting up PSSP accounting

To contrast the installation steps of the AIX and the PSSP implementations, we will use, in this section, the same order of the installation steps as in Section 3.2, “Setting up accounting” on page 64.

To install PSSP accounting, you should log on as root at the control workstation.

4.1.1.1 Installing the filesets with PSSP

You should install the bos.acct and the ssp.sysman filesets, both on the control workstation and on all the nodes. Do not forget to copy those filesets to the appropriate lppsource and pssplpp directories, so future node installations or re-installations will automatically install those filesets.

After installing the filesets, you should specify how accounting will be done on your SP system.

Enabling accounting

You should use the `spsitenv` command (or `smit enter_data`) to enable accounting on the SP system. The relevant site environment variables (with their default values) are highlighted in the following display:

```
# splstdata -e
             List Site Environment Database Information

attribute                value
-----------------------------------------------------------------------------
control_workstation      cws1
cw_ipaddrs               10.3.187.243:9.3.187.245:9.3.187.243:
install_image            bos.obj.ssp.433
remove_image             false
primary_node             1
ntp_config               consensus
ntp_server               ""
ntp_version              3
amd_config               true
print_config             false
print_id                 ""
usermgmt_config          true
passwd_file              /etc/passwd
passwd_file_loc          cws1
homedir_server           cws1
homedir_path             /home/cws1
filecoll_config          true
supman_uid               102
supfilesrv_port          8431
spacct_enable            false
spacct_actnode_thresh    80
spacct_excluse_enable    false
acct_master              0
cw_has_usr_clients       false
code_version             PSSP-3.1.1
layout_dir               ""
authent_server           ssp
backup_cw                ""
ipaddrs_bucw             ""
active_cw                ""
sec_master               ""
cds_server               ""
cell_name                ""
cw_lppsource_name        aix433
cw_dcehostname           ""
```

The function of the accounting site environment attributes are:

- SP Accounting Enabled (spacct_enable) enables (true value) or disables (false value) overall accounting. When you enable overall accounting, accounting will not necessarily take place on all nodes, as explained in the following subsection.
- SP Accounting Active Node Threshold (spacct_actnode_thresh) indicates the minimum percentage of nodes with valid accounting data that is necessary for the daily consolidation to take place. The default value, 80, indicates that if less than 80% of the nodes have collected their daily accounting data, then the accounting consolidation on the accounting master will not take place.

- SP Exclusive Use Accounting Enabled (spacct_excluse_enable) specifies an overall decision on whether to charge (true value) or not (false value) an user for the exclusive use of a node. You can control on a node by node basis whether the exclusive use of the individual node will be charged or not, as explained in the following subsection.
- Accounting Master (acct_master) contains the node number of the node that will gather all accounting data.

To start accounting, you should at least modify the spacct_enable attribute, as shown in the following display:

```
# spsitenv spacct_enable=true spacct_excluse_enable=true
rc.ntp: Starting ntp daemon(xntpd)
0513-029 The supfilesrv Subsystem is already active.
Multiple instances are not supported.
/etc/auto/startauto: The automount daemon is already running on this system.
f01n01: rc.ntp: Starting ntp daemon(xntpd)
f01n01: Updating collection sup.admin from server cwst1.
f01n01: File Changes:  1 updated, 0 removed, 0 errors.
f01n01: Updating collection user.admin from server cwst1.
f01n01: File Changes:  6 updated, 0 removed, 0 errors.
f01n01: Updating collection node.root from server cwst1.
f01n01: File Changes:  0 updated, 0 removed, 0 errors.
f01n01: /etc/auto/startauto: The automount daemon is already running on this system.
f01n01: /var/adm/acct
root=CWS1:cwst1:f02n01t.itsc.austin.ibm.com:cwst1t.itsc.austin.ibm.com,access=CWS1:cw
st1:f02n01t.itsc.austin.ibm.com:cwst1t.itsc.austin.ibm.com
f01n01: Exported /var/adm/acct
...
```

Note from the output that the SP services are reconfigured both in the control workstation and in the nodes. The main actions taken to configure accounting follow:

On the accounting master:

- The accounting directories are created.
- The appropriate crontab entries are created.

On the control workstation:

- The /etc/acct/holidays file is added to the user.admin file collection.

On the nodes:

- The accounting directories are created.
- The wtmp and pacct files are created if necessary.
- The appropriate crontab entries are created.

- Process accounting is started and the appropriate command is added to /etc/rc.
- The /var/adm/acct directory is exported (so that the node’s usage data can be collected from the accounting master).

As you can see, many of the actions you would need to manually execute on standard AIX are automatically done by PSSP.

Now accounting is configured and running on your SP system. But you may want to do some further customizing, as explained in subsequent subsections.

Customizing the nodes

After accounting is enabled on your system, you may want to disable it on some nodes or do some further node customizing. The SDR maintains the following accounting information for each node:

```
# splstdata -A
                  List Node Accounting Information

node#          hostname    acct_class_id  acct_enable  acct_excluse_enable acct_job_charge
------------------------------------------------------------------------------------------
    1            f01n01          default      default                false          1.0000
    3            f01n03          default      default                false          1.0000
    5            f01n05          default      default                false          1.0000
    6            f01n06          default      default                false          1.0000
    7            f01n07          default      default                false          1.0000
```

- The Accounting Class Identifier (acct_class_id) attribute indicates the class to which the node belongs. A class is identified by a string. You do not need any additional steps to create a class. You just need to specify, for all nodes that will belong to that class, the same string value as the node’s class identifier. By default, there is a single accounting class named *default*.
- The Accounting Enable (acct_enable) attribute is used to individually enable or disable accounting. If the attribute has the value true, accounting is enabled on that node, independently of the spacct_enable attribute’s value. If acct_enable is false, accounting is unconditionally disabled on that node. If acct_enable equals default, accounting on that node is enabled or disabled, depending on whether spacct_enable is true or false. As you can see, since the initial value for acct_enable is default, it is enough to set spacct_enable to true to enable accounting on all nodes.

- The Exclusive Use Accounting Enabled (acct_excluse_enable) attribute indicates whether exclusive use of that node will be charged to the user or not. The possible values are true and false, with false being the initial one.
- The Accounting Job Charge Value (acct_job_charge) attribute specifies the number of seconds of exclusive use per charge fee unit. The initial value is 1.0. For example, a value of 60.0 would mean that for every *minute* a non-shared LoadLeveler job step stays in the system, one fee unit would be charged to the user.

Note

To enable exclusive use accounting, you need to do *all* the following:

- Set the spacct_excluse_enable attribute to true.
- Set the acct_excluse_enable attribute to true for the nodes in your system.
- Set the keyword spacct_excluse_enable to true in the machine stanzas of your LoadLeveler administration file.

To change an accounting attribute, you should use the `spacctnd` command (or `smit node_data`). For example, to disable accounting on node 1 you could use the following command:

```
# spacctnd -e false -l 1
f01n01: rc.ntp: Starting ntp daemon(xntpd)
f01n01: Updating collection sup.admin from server cwst1.
f01n01: File Changes:  0 updated, 0 removed, 0 errors.
f01n01: Updating collection user.admin from server cwst1.
f01n01: File Changes:  6 updated, 0 removed, 0 errors.
f01n01: Updating collection node.root from server cwst1.
f01n01: File Changes:  0 updated, 0 removed, 0 errors.
f01n01: /etc/auto/startauto: The automount daemon is already running on this system.
cws1: rc.ntp: Starting ntp daemon(xntpd)
cws1: 0513-029 The supfilesrv Subsystem is already active.
cws1: Multiple instances are not supported.
cws1: /etc/auto/startauto: The automount daemon is already running on this system.
```

Note that the SP services are reconfigured in the affected node and in the control workstation (the accounting master).

4.1.1.2 Setting up the environment with PSSP

We recommend you set the value of the MAILCOM variable, both in the control workstation and in all the nodes, as described in Section 3.2.2, “Setting up the environment” on page 66. Depending on your needs, you could modify the /etc/environment only on the control workstation and then add that file to the user.admin file collection.

4.1.1.3 Updating the crontab entries with PSSP

With PSSP, you do not need to manually enter the accounting crontab entries on the control workstation and on the nodes. For example, the accounting crontab entries on one of the nodes (f01n05) and on the control workstation, right after the execution of the `spsitenv` command, are shown in the following display:

```
# rsh f01n05 crontab -l | tail -3
0 1 * * 4 /usr/sbin/acct/dodisk
5 * * * * /usr/sbin/acct/ckpacct
0 2 * * 1-6 /usr/lpp/ssp/bin/nrunacct 2>/var/adm/acct/nite/accter
# crontab -l | tail -2
0 4 * * 1-6 /usr/lpp/ssp/bin/crunacct 2>/var/adm/cacct/nite/accterr
15 5 1 * * /usr/lpp/ssp/bin/cmonacct
```

Comparing these entries with those for standard AIX, you should note that:

- A node does not run the `runacct` command daily, but instead runs its PSSP counterpart, `nrunacct`. The `nrunacct` command only generates the daily usage data files (lineuse, reboots, daytacct, daycms, and loginlog), but no aggregate files or reports.
- In the same manner, the `monacct` command does not run on the nodes.
- The accounting master runs its own version of `runacct` and `monacct`. The `crunacct` command gathers the data generated by `nrunacct` on all the nodes (using NFS). With that information, it generates reports and aggregate files per class. The `cmonacct` command generates monthly accounting files and monthly reports for each accounting class.
- Neither the `dodisk` command or the `ckpacct` commands are set to run on the control workstation. PSSP accounting does *not* gather usage data on the control workstation (no users should be using it anyway). If you need to gather accounting data for the control workstation, you may set up standard AIX accounting on it.
- PSSP accounting runs under the root user, not the adm user.
- The accounting jobs are set up to run during the morning hours of each day, not before midnight, as in standard AIX (which means that the accounting files contain data from the previous day).

To modify the accounting crontab entries on the nodes, you have two options.

The first option is to have a single copy of the crontab file and propagate it to all the nodes through supper (probably using the node.root collection, since the crontab file on the control workstation and those on the nodes are not the same). This may work fine if all your nodes can execute the same cron jobs (that is, if the accounting master is the control workstation *and* all nodes have accounting enabled). For this option to work, you would also need to remove the /usr/lpp/ssp/config/cron_template file from all nodes.

Your second option is to modify the /usr/lpp/ssp/config/cron_template file. This file is used by accounting service configuration to create the crontab entries. The default contents of that file follow:

```
# cat /usr/lpp/ssp/config/cron_template
# This is the crontab template file. It is read by acct_config
# to update the root crontab file with accounting-related entries.

DODISK          0 1 * * 4
CHKPACCT        5 * * * *
NRUNACCT        0 2 * * 1-6
CRUNACCT        0 4 * * 1-6
CMONACCT        15 5 1 * *
```

Note that the file does not contain full crontab entries, just a initial keyword specifying the command to be executed, followed by the specification of how the command should run. When editing this file, you should not modify the keywords.

You would need to modify this file on all nodes, as well as on the control workstation. Therefore, the best solution would be to edit it on the control workstation and add it to a supper file collection. A file collection like user.admin would work, since this file can be the same everywhere. It is the accounting service configuration process that decides which jobs to add and which not.

After you have modified the file and propagated the changes to all the nodes, you need to run the following command on the nodes and on the control workstation for those changes to take effect immediately:

```
/usr/lpp/ssp/install/bin/services_config
```

This command runs automatically when a node boots. This is why it was recommended that you remove the cron_template file if you decide to use file collections to propagate the root's crontab file.

4.1.1.4 Setting up disk accounting with PSSP

To set up disk accounting, you need to set the attribute account to true on all file systems that are going to be accounted, as explained in Section 3.2.7.1, “Setting up fast mode disk accounting” on page 71. Since you will have to do it node by node you could use something like the following command:

```
rsh f01n01 chfs -t yes /home
```

Note that PSSP does not expect you to use slow mode disk accounting, only fast mode. If you really need to run slow mode disk accounting, you should investigate the possibility of doing so from the control workstation.

4.1.1.5 Setting up queue accounting with PSSP

To set up queue accounting, you need to specify the name of the printer accounting file (/var/adm/qacct), as discussed in Section 3.2.8, “Setting up queue accounting” on page 72. You may use the `chque` command to do so, as in the following example:

```
rsh f01n01 chque -qlp0 -a'acctfile = /var/adm/qacct'
```

4.1.1.6 Defining the billing periods with PSSP

You should edit the /etc/acct/holidays on the control workstation to correctly specify your company’s definition of prime and non-prime hours. Please refer to Section 3.2.9, “Defining the billing periods” on page 76, for a detailed explanation of that file’s contents.

Since /etc/acct/holidays is automatically put in the user.admin file collection, any modifications done on the control workstation will be automatically propagated to all nodes in the system.

4.1.2 The output files

In this section, we will discuss the most important permanent files generated by PSSP accounting. For a detailed description of accounting files, see Section 3.3, “Reading the accounting files” on page 78.

4.1.2.1 The files on the nodes

A list of the accounting files usually present on the SP nodes appears on the following display:

```
# rsh f010n01 cd /var/adm \; ls -lR acct dtmp fee pacct wtmp
-rw-r--r--   1 root     system         0 Jun 22 00:00 dtmp
-rw-rw-r--   1 adm      adm            0 Jun 28 01:01 fee
-rw-rw-r--   1 adm      adm       149240 Jun 28 22:11 pacct
-rw-rw-r--   1 adm      adm           64 Jun 28 01:00 wtmp

acct:
total 196
drwxr-xr-x   2 adm      adm          512 Jun 28 21:54 fiscal
drwxr-xr-x   2 adm      adm        70144 Jun 28 21:52 nite
drwxr-xr-x   2 adm      adm        26112 Jun 28 21:55 sum

acct/fiscal:
total 0

acct/nite:
total 6
-rw-r--r--   1 adm      adm          144 Jun 28 01:00 accterr
-rw-rw-r--   1 adm      adm          754 Jun 28 01:01 active
-rw-r--r--   1 adm      adm            0 Jun 22 00:00 dacct
-rw-rw-r--   1 adm      adm            5 Jan 03 18:04 jobcharge
-rw-rw-r--   1 adm      adm            0 Jun 28 01:00 jobrecs
-rw-rw-r--   1 adm      adm            9 Jun 28 01:00 lastdate
-rw-rw-r--   1 adm      adm            0 Jun 28 01:00 log
-rw-rw-r--   1 adm      adm          128 Jun 28 01:00 owtmp
-rw-rw-r--   1 adm      adm            0 Jun 28 01:00 wtmperror

acct/sum:
total 10
-rw-rw-r--   1 root     system      4919 Jun 28 01:01 loginlog
```

Most of the files are those found in standard AIX accounting, except for the following:

- The /var/adm/acct/nite/jobcharge contains the value of the node's acct_job_charge attribute, which is used to calculate the exclusive use fees.
- The /var/adm/acct/nite/jobrecs contains start job and end job records for each job step run in non-shared mode. The contents of this file are used to generate the extra service fees (accumulated temporarily in the /var/adm/fee file).

The excluded process accounting records

One set of files that does not appear in the preceding display is the /var/adm/acct/nite/excl_Spacct*i.date* files. These files are generated daily, one for each pacct*i* file in the /var/adm directory. The excl_Spacct*i.date* files are in binary process accounting format (described in Appendix B.3, "The pacct file" on page 154), and contain the process accounting records for the job steps executed in non-shared mode. These records are kept in a separate file because they are excluded from accounting.

In other words, these records are *not* used to compute the daily total accounting files nor the command summary files. This is why the accounting system keeps those files in the /var/adm/acct/nite directory; you might need the information there. As an example of how to use those files, you could use the following script that merges the contents of those files to the daily command summary and then removes them:

```
#! /bin/bsh
PATH=/usr/lpp/ssp/bin:/usr/sbin/acct:/bin:/usr/bin:/etc
export PATH
umask 003
_adm=${ACCTDIR:-/var/adm}
_nite=${_adm}/acct/nite
_sum=${_adm}/acct/sum
_date="`date +%Y +%m%d`"
cd ${_adm}/acct
cp sum/daycms${_date} sum/daycms-excl${_date}
acctcms nite/excl_Spacct*.${_date} > sum/excl${_date}
acctcms -s sum/daycms-excl${_date} sum/excl${_date} > sum/daycms${_date}
rm -f sum/daycms-excl${_date} sum/excl${_date}
rm -f nite/excl_Spacct*.${_date}
```

You should call such a script /var/adm/nsiteacct, since the `nrunacct` command automatically executes a file with that name (like `runacct` does with the /var/adm/siteacct file).

Note

The standard nrunacct script does not remove the excl_Spacct* files. You should have an automatic way to remove those files, otherwise the space in the /var file system will eventually be exhausted. One way to remove those files is by uncommenting the following line in the /usr/lpp/ssp/bin/nrunacct script:

```
# rm -f /var/adm/acct/nite/excl_Spacct*
```

4.1.2.2 The files on the accounting master

The accounting master keeps its files on the /var/adm/cacct directory. A listing with the relevant files and directories is shown in the following display:

```
# cd /var/adm/cacct
# ls -l . nite nite/default sum sum/default
.:
total 13
drwxr-xr-x   4 adm      adm          512 Jun 28 10:07 1/
drwxr-xr-x   4 adm      adm          512 Jun 28 10:07 3/
drwxr-xr-x   4 adm      adm          512 Jun 28 10:07 5/
drwxr-xr-x   4 adm      adm          512 Jun 28 10:07 6/
drwxr-xr-x   4 adm      adm          512 Jun 28 10:07 7/
-rw-rw-r--   1 adm      adm          352 Jun 29 12:09 active
drwxr-xr-x   3 adm      adm          512 Jun 28 10:07 fiscal/
-rw-rw-r--   1 adm      adm            9 Jun 29 12:09 lastcycle
-rw-rw-r--   1 root     system         9 Jun 29 11:59 lastdate
drwxr-xr-x   3 adm      adm          512 Jun 29 04:00 nite/
drwxr-xr-x   2 adm      adm          512 Jun 28 10:07 node_mnt/
-rw-rw-r--   1 root     system         9 Jun 29 12:09 statefile
drwxr-xr-x   3 adm      adm          512 Jun 29 12:09 sum/

nite:
total 1
-rw-r--r--   1 root     system         0 Jun 29 04:00 accterr
drwxr-xr-x   2 adm      adm          512 Jun 29 12:09 default/

nite/default:
total 22
-rw-rw-r--   1 root     system      5411 Jun 29 12:09 cms
-rw-rw-r--   1 root     system      5411 Jun 29 12:09 daycms

sum:
total 29
drwxr-xr-x   2 adm      adm          512 Jun 29 12:09 default/
-rw-rw-r--   1 root     system       249 Jun 29 12:09 loginlog
-rw-rw-r--   1 root     system     13654 Jun 29 12:09 rprt20000629

sum/default:
total 24
-rw-rw-r--   1 adm      adm         5520 Jun 29 12:09 cms
-rw-rw-r--   1 adm      adm         5520 Jun 29 12:09 daycms
-rw-rw-r--   1 adm      adm          288 Jun 29 12:09 tacct
-rw-rw-r--   1 adm      adm          288 Jun 29 12:09 tacct20000629
```

Note that there are several differences between the standard AIX /var/adm/acct structure and the /var/adm/cacct structure:

- The state files that control the execution of crunacct are kept in the /var/adm/cacct directory, not the nite subdirectory.
- A subdirectory is kept for every node on the system; these subdirectories temporarily house the files that were generated on that node. The directories are identified by the corresponding node number. They are usually devoid of files.
- On the /var/adm/cacct/nite directory, there is one subdirectory per account class (default being the only class in the example above). These

subdirectories contain the daily command summary (daycms) and accumulated command summary (cms) reports for the 50 largest commands.

- The sum/loginlog file contains the last login information (taking all nodes into consideration). A sample output follows:

```
# cat loginlog
00-00-00 adm 1 3 5 6 7
00-00-00 bin 1 3 5 6 7
00-00-00 daemon 1 3 5 6 7
00-00-00 guest 1 3 5 6 7
00-00-00 imnadm 1 3 5 6 7
00-00-00 lpd 1 3 5 6 7
00-00-00 nobody 1 3 5 6 7
00-00-00 supman 1 3 5 6 7
00-00-00 sys 1 3 5 6 7
00-00-00 uucp 1 3 5 6 7
00-06-29 gonzalo 5
00-06-29 root 1 5 6
```

  The contents of loginlog are indicating that the last day user gonzalo logged on was 06/29/2000, onto node 5. It also shows that the last day user root logged on was 06/29/2000 and that she logged on nodes 1, 5, and 6 that day.

- The sum/rprt*date* is the daily report for that date (note that *date* is in a *yyyymmdd* format). It is generated by the `/usr/lpp/ssp/bin/cprdaily` command and contains:
  - Reboots and lineuse information for all nodes
  - Daily total accounting report for each class
  - Daily command summary report for each class
  - Accumulated command summary report for each class
  - The last login information
- On the /var/adm/cacct/sum directory there is one subdirectory per account class (default being the only class in the example above). These subdirectories contain:
  - The daily command summary in binary format (daycms)
  - The accumulated command summary in binary format (cms)
  - The daily total accounting files in binary format (tacct*date*)
  - The accumulated total accounting file in binary format (tacct)

The /var/adm/cacct/fiscal directory follows the same general idea. The directory contains the monthly reports (fiscrpt*mm*). And there is a subdirectory with the monthly command summary files (cms*mm*) and monthly total accounting files per class (tacct*mm*).

4.2 Accounting using LoadLeveler

If you are using LoadLeveler on your network, you can collect resource usage data for your jobs through LoadLeveler's accounting feature. When you activate accounting, LoadLeveler starts accumulating resource consumption data for completed serial and parallel jobs.

In this section we discuss the steps to get LoadLeveler up and running. For additional information, please refer to *LoadLeveler Using and Administering, Version 2.2*, SA22-7311.

4.2.1 The accounting data

LoadLeveler accumulates the data returned by the wait3 system call. This means that LoadLeveler accounting suffers from the constraints imposed by that system call. In particular, if the job step spawns a process without waiting for it to end, or if the step spawns a process on a remote machine through rsh, LoadLeveler will not collect accounting data for those spawned processes.

Besides all job and job step attributes, LoadLeveler collects the following resource consumption data per job step:

- Step User Time: The total amount of time running in user mode.
- Step System Time: The total amount of time spent by the system executing on behalf of the processes.
- Step Total Time: The sum of Step User Time and Step System Time.
- Step maxrss: The maximum size, in KB, of the used resident set size.
- Step ixrss: The integral of the amount of memory used by the text segments (which could have been shared with other processes) while the processes were executing. This value is expressed in KB times seconds and is calculated by adding the number of text memory pages in use each time the internal system clock ticks, and adjusted to one-second intervals.
- Step idrss: The integral of the amount of memory in the data segment of the processes (KB times seconds).
- Step isrss: Always 0 with AIX (should be the integral of the amount of memory in the stack segment of the processes).
- Step minflt: The number of page faults serviced without any I/O activity (the page was still in memory).
- Step majflt: The number of page faults serviced that required I/O activity.

- Step nswap: The number of times a process was swapped out of main memory.
- Step inblock: The number of times the file system performed actual disk reads for the processes.
- Step oublock: The number of times the file system performed actual disk writes.
- Step msgsnd: The number of IPC messages sent.
- Step msgrcv: The number of IPC messages received.
- Step nsignals: The number of signals delivered.
- Step nvcsw: The number of times a context switch resulted because a process voluntarily gave up the processor before its time slice was completed. This usually occurs while the process waits for availability of a resource.
- Step nivcsw: The number of times a context switch resulted because a higher priority process ran or because the current process exceeded its time slice.

For parallel jobs, a job step is composed of several executions in different machines. LoadLeveler aggregates the resource usage of all executions in the following manner:

1. For memory usage (maxrss, ixrss, idrss, and isrss), the job step stores the maximum value from all executions.
2. For all other values (CPU time and counters), the job step stores the sum of all executions.

4.2.2 The history file

The accounting data is gathered by the machine responsible for scheduling the job. The accounting data is kept in the history file in LoadLeveler's spool directory.

```
# ls -l ~loadl/spool/
total 60
-rw-rw----   1 loadl    loadl      14220 Jun 30 14:33 history
-rw-rw----   1 loadl    loadl          0 Jun 30 10:52 job_queue.dir
-rw-rw----   1 loadl    loadl      16384 Jun 30 14:33 job_queue.pag
```

Unlike AIX accounting, LoadLeveler does not summarize the resource usage data. This means that you can extract individual job data from the history file.

For example: in the following display we extract the usage data for a particular job using the llsummary command:

```
# llsummary -l -j cws1.4 | tail -21
     Step User Time:   0+00:00:00.30000
   Step System Time:   0+00:00:00.40000
    Step Total Time:   0+00:00:00.70000
        Step maxrss: 964
         Step ixrss: 1996
         Step idrss: 1564
         Step isrss: 0
        Step minflt: 331
        Step majflt: 0
         Step nswap: 0
       Step inblock: 0
       Step oublock: 0
        Step msgsnd: 0
        Step msgrcv: 0
      Step nsignals: 0
         Step nvcsw: 1
        Step nivcsw: 17
          Step Cpus: 0
Step Virtual Memory: 0 megabytes
   Step Real Memory: 0 megabytes
Step Adapter Memory: 0 bytes
```

Note

In the preceding display, the last four lines of the llsummary output show the usage of consumable resources, a new scheduling facility available with LoadLeveler 2.2. Those values do not necessarily express actual resource usage.

4.2.2.1 Collecting history files

Each scheduling machine keeps the accounting records for the jobs it has scheduled, and the llsummary command reads the contents of the local history file to extract job resource information. This means that if you have several scheduling machines, you need to run the command in each machine to get the accounting information stored there.

LoadLeveler allows you to merge all history files into a single global history file. To do so, you should use the llacctmrg commands as follows:

```
# llacctmrg
llacctmrg: History transferred successfully from cws1.itsc.austin.ibm.com (14220 bytes)
# ls -l ~loadl/spool/
total 60
-rw-r--r--   1 root     system     14220 Jun 30 14:46 globalhist.200006301446
-rw-rw----   1 loadl    loadl          0 Jun 30 14:46 history
-rw-rw----   1 loadl    loadl          0 Jun 30 10:52 job_queue.dir
-rw-rw----   1 loadl    loadl      16384 Jun 30 14:33 job_queue.pag
```

By default, the `llacctmrg` commands collects the history files from all scheduling machines in your cluster, creating a file named globalhist.*yyyymmddHHMM* in the spool directory of the local machine. It also removes the existing history files, creating new empty files.

To obtain reports from a global history file, you should specify the file's name in the `llsummary` command.

4.2.3 Setting up accounting

To set up LoadLeveler accounting you just need to modify the ACCT keyword on LoadLeveler's configuration file. This keyword accepts one or more flags. The possible values are:

- A_OFF: Turns accounting off (default value).
- A_ON: Turns accounting on.
- A_DETAIL: Enables extended accounting, where detailed resource consumption by machine and by event is stored in the history file. You have to set this flag if you want to use the -x flag of the `llq` and `llsummary` commands.
- A_VALIDATE: Turns account validation on. LoadLeveler allows you to gather usage information base on *account numbers*. With account validation on, you force your users to specify a valid account number in their jobs (using the account_no keyword). The valid account numbers are specified in the user stanza (with the account keyword).

The default value of the ACCT keyword can be seen in the following display:

```
$ grep -p accounting LoadL_config
#
# Specify accounting controls
#
ACCT                    = A_OFF
ACCT_VALIDATION         = $(BIN)/llacctval
GLOBAL_HISTORY          = $(SPOOL)
```

The other accounting controls are the ACCT_VALIDATION and GLOBAL_HISTORY keywords. The former indicates which subroutine will validate the users' account number (if A_VALIDATE is set). The latter indicates the default directory to store the global history files.

The following display shows the modified contents of the configuration file, followed by the command to force cluster reconfiguration:

```
$ grep -p accounting LoadL_config
#
# Specify accounting controls
#
ACCT                    = A_ON A_DETAIL
ACCT_VALIDATION         = $(BIN)/llacctval
GLOBAL_HISTORY          = $(SPOOL)

$ llctl -g reconfig
llctl: Sent reconfig command to host cws1.itsc.austin.ibm.com.
```

4.2.4 Extracting accounting information

You should use the `llsummary` command to obtain job usage information. By default, `llsummary` provides you with four reports. These reports group the accounting data by user, by class, by LoadLeveler group, and by account number. For example:

```
# llsummary
       Name   Jobs   Steps        Job Cpu    Starter Cpu     Leverage
   ausres04      1       1     0+00:00:00     0+00:00:00  (undefined)
      TOTAL      1       1     0+00:00:00     0+00:00:00  (undefined)

      Class   Jobs   Steps        Job Cpu    Starter Cpu     Leverage
   No_Class      1       1     0+00:00:00     0+00:00:00  (undefined)
      TOTAL      1       1     0+00:00:00     0+00:00:00  (undefined)

      Group   Jobs   Steps        Job Cpu    Starter Cpu     Leverage
   No_Group      1       1     0+00:00:00     0+00:00:00  (undefined)
      TOTAL      1       1     0+00:00:00     0+00:00:00  (undefined)

    Account   Jobs   Steps        Job Cpu    Starter Cpu     Leverage
       NONE      1       1     0+00:00:00     0+00:00:00  (undefined)
      TOTAL      1       1     0+00:00:00     0+00:00:00  (undefined)
```

The reports display for each category item:

- Jobs: The number of jobs
- Steps: The number of job steps
- Job Cpu: The accumulated CPU time for the job steps

- Starter Cpu: The accumulated CPU time for the LoadLeveler starter processes
- Leverage: The ratio of Job Cpu to Starter Cpu

You can also select a specific report using the -d flag. For example, you can ask for usage data categorized by user as follows:

```
# llsummary -d user | head
       Name   Jobs   Steps        Job Cpu    Starter Cpu     Leverage
      user1    168     168     2+15:50:42     0+00:00:58       3962.8
      user2    117     117     0+03:45:39     0+00:01:05        208.3
      user3    147     147   185+19:31:57     0+00:00:42     382245.6
      user4    269     269    74+06:47:03     0+00:01:05      98738.8
      user5    190     190    27+20:53:23     0+00:00:52      46307.8
      user6     74      74    44+12:44:33     0+00:00:13     295959.5
      user7     46      46    37+18:54:38     0+00:00:19     171835.7
      user8     80      80     7+05:27:39     0+00:00:17      36732.9
      user9      3       3     1+01:52:27     0+00:00:02      46573.5
```

The possible values for the -d option are `user`, (LoadLeveler) `group`, `unixgroup`, `class`, `account`, `day`, `week`, `month`, `jobid`, `jobname`, `allocated`. Another sample output, now grouping the data by month, follows:

```
# llsummary -d month
      Month   Jobs   Steps        Job Cpu    Starter Cpu     Leverage
    07/1999    350     350    32+23:35:05     0+00:02:19      20501.5
    08/1999    951     971   154+12:28:28     0+00:06:37      33628.5
    09/1999    446     446   122+07:38:25     0+00:04:14      41607.5
    10/1999    401     401   128+11:00:00     0+00:02:57      62705.1
    11/1999    986     986   142+16:44:44     0+00:05:53      34926.6
    12/1999    928     928   120+14:18:38     0+00:06:17      27638.0
    01/2000    796     796    94+16:51:51     0+00:06:59      19528.2
    02/2000    729     734    98+18:12:19     0+00:03:51      36938.3
    03/2000    263     263   124+11:58:47     0+00:00:50     215134.5
    04/2000    294     294   103+07:09:45     0+00:00:52     171634.3
    05/2000    323     323   123+07:16:26     0+00:01:33     114552.5
    06/2000    279     279   100+02:58:08     0+00:01:47      80847.6
      TOTAL   6746    6771  1346+04:53:20     0+00:44:15      43808.7
```

You can also ask for a different kind of report by using the -r flag. The possible values for this option are `resource` (default), `avgthroughput`, `maxthroughput`, `minthroughput`, `throughput`, and `numeric`. With the throughput reports, you can obtain several statistics about your jobs. The following example shows, categorized by class, the average time in queue, the average time in the system, and the average CPU time for all jobs:

```
# llsummary -r avgthroughput -d class
      Class   Jobs   Steps   AvgQueueTime    AvgRealTime     AvgCPUTime
ParallelJobs  3110    3115     0+03:18:53     0+00:16:38     0+00:17:34
   LargeJobs   661     661     0+08:03:27     0+10:57:49     0+07:52:46
VeryLargeJob   775     775     1+16:30:14     1+02:05:04     0+17:54:39
   SmallJobs    75      75     0+05:03:03     0+01:33:45     0+01:09:22
    HugeJobs   265     265     4+01:49:46     2+04:52:08     1+11:35:25
VerySmallJob   291     291     0+03:58:27     0+00:40:59     0+00:32:36
  MediumJobs   291     291     0+06:49:07     0+00:25:45     0+00:22:11
AboveAvgJobs   613     622     0+07:28:03     0+04:08:48     0+03:15:01
     WSPJobs    48      48     0+00:28:29     0+01:34:15     0+01:34:27
    TestJobs    32      32     1+03:13:03     0+13:32:36     0+13:21:46
         ALL  6161    6175     0+13:17:00     0+07:25:39     0+05:13:56
```

Please refer to the *LoadLeveler Using and Administering, Version 2.2*, SA22-7311, to obtain additional information about the report options of the `llsummary` command.

4.2.4.1 Using the LoadLeveler APIs

LoadLeveler provides one function, GetHistory, to extract accounting data from the history files. If `llsummary`'s options do not attend your needs for extracting usage data, then you will need to build your own program using that routine. GetHistory is documented in *LoadLeveler Using and Administering, Version 2.2*, SA22-7311, and the samples/llphist directory contains a sample program that uses it.

Chapter 5. Third-party accounting solutions

The AIX base operating system provides all the basics needed to start and collect accounting information on your system, but you can be in situations where the information is either not complete, or the output format is not as elegant as you would like.

The solution to these situations might be to develop your own displays or applications to format the data coming from the accounting tool. You might also look for third party solutions. We have browsed the Internet, in search of such software, and found some that you might find interesting. We did not have time to test the software we found, and the information presented here was extracted from the software's respective Web sites, with the agreement of the various companies.

All of the products mentioned here are supported on AIX version 4.3.3. For each of them, we give a summary of their functions, their advantages compared to AIX, and an how they appear (if possible).

5.1 COSchargeback

COSchargeback is a member of OSM's COSMOS products, so it has a general flexibility in terms of being able to be easily configured to comply with an IT organization's existing policy. If such policies do not already exist, COSchargeback provides an ideal setting for IT managers to implement new policy in the areas of:

- Resource management
- Resource accounting
- IT cost allocation
- IT cost monitoring

COSchargeback is a potent resource accounting and back-charging management solution for UNIX computing environments. In addition to its prime purpose of accounting for system resource usage (across multiple servers) and producing consolidated reports for billing users, you may also collect information to be used in planning the needs of future computing capacity.

COSchargeback provides a cost-effective charge-back solution that:

- Charge users, groups or business units for the system resources they consume

- Understand how your IT budget is being spent
- Identify which business units use your resources more than others
- Implement a charge-back scheme to make business units accountable
- Bill clients accurately for their use of a shared resource in an ASP service

5.1.1 Overview

COSchargeback is a resource accounting and chargeback software product. It allows IT managers to measure and calculate exactly the amount of system resources each department or business unit uses and, if required, how much each should be charged.

Particularly in the case of server consolidation projects and the world of ASPs, the cost of resources consumed must be allocated accurately and fairly to the consumers. A resource accounting and chargeback policy implemented through COSchargeback provides:

- Information to business unit heads on their use of IT and its associated cost
- Improved communication and alignment between the IT department and business units
- Opportunities to improve service levels
- Information to IT managers about which items could be targeted for cost reduction or outsourcing
- Information to IT managers about past investment decisions
- Information to accurately monitor and control costs
- Information to enable capacity and upgrade planning

5.1.2 Features

Audit information is collected from any defined node or system on the network. Each node has one or more accounting programs, each gathering data from the services running on that system. Resource usage can be attributed to individual users, defined groups, departments or business units. COSchargeback:

- Collects information from 13 versions of UNIX
- Gathers information from databases and applications
- Uses standard audit data, imposing no additional system overhead

- Collects data automatically and allows the collection times to be completely customized
- Acquires information about users on a network-wide basis, as opposed to a single "logged in" user
- Allows any number of charging rates to be defined
- Calculates usage across different users, groups or departments
- Supports the production of extensive reports
- Integrates with Microsoft Excel for extensive data analysis
- Enables day-to-day usage information to be used for trend analysis and capacity planning
- Provides business units with a view of accounting charges for each defined billing period
- Allows reports to be easily produced for all systems across the network from a single host

5.1.3 Chargeback software components

OSM's charge-back software - COSchargeback – has two software components: Accountants (or agents) and the Console.

Here are the definitions for some essential elements of the COSchargeback solution:

- Accountants (agents)

  On every COSchargeback node, each resource for which accounting is necessary requires an Accountant. The Accountant keeps track of how Services have been used over a specified period by collecting audit data. Typically, a COSchargeback node has Accountants for each:

  - Operating system
  - Database server
  - Database instance
  - Application
- Services

  A COSchargeback Service consists of a number of similar Resource Types grouped together under a unique name.

- Resource Types

  A Resource Type is the smallest measurable component within a COSchargeback node, such as:

  - Sessions
  - Connections
  - CPU Usage
  - System Memory
  - Disk
  - Processors Count
  - Characters Read
  - Characters Written
  - Blocks Read
  - Blocks Written
  - Statements
  - Committed Transactions.

- Resource Manager

  COSchargeback's Resource Manager gathers audit information from target resources contained within accountants (or agents) and services. The information collected includes:

  - How much of the resource has been used.
  - Who, or what, used the resource.
  - Where on the network the resource was used.
  - When the resource was used.

- COSchargeback Console

  The COSchargeback Console forms a central point of data collection from all COSchargeback accountants. All data is stored here and can be queried by COSchargeback's reporting tools and with Microsoft Excel.

- Time-banding

  When using COSchargeback, IT managers can define any number of different time-bands, allowing for the cost of usage to vary according to time and date.

- Reporting

  Through the Output Format Description tool, users can define the look and format of COSchargeback reports. There are standard facilities to produce useful reports directly from COSchargeback and, additionally, there are facilities to export the data in a format that can be used by third party software, such as Microsoft Excel.

- Chargeback Options

  Every Resource Type can have a charge description applied to it, in order to improve cost allocation and chargeback.

- Integration with Microsoft Excel

  COSchargeback integrates with Microsoft Excel through an Excel menu option that automatically retrieves COSchargeback data from pre-existing Excel. The data can then be worked on and analyzed extensively.

- Application Integration

  COSchargeback integrates with any application that can report information to the standard UNIX system accounting processes. In addition, OSM is able to integrate COSchargeback to other applications through its customizable API.

- COSchargeback and COSMOS

  Installation of other products from the COSMOS operations management suite is not a prerequisite for installing COSchargeback. However, adding other COSMOS products brings further benefits to COSchargeback in functionality and control.

- Integration with BMC Software's PATROL

  COSMOS integrates with the PATROL environment, providing proactive event management by means of a dedicated Knowledge Module (KM) available from OSM.

To get more information about this product, visit their web site at `http://www.coschargeback.com`.

5.2 UNISOL® JobAcct^{TM}

UNISOL JobAcct is used by organizations to provide system access and services to other organizations. UNISOL JobAcct is also used by organizations that need to distribute a user's work across multiple projects while maintaining reasonable control over the way system resource usage is distributed among the different projects. UNISOL JobAcct's ability to produce reports for many or all machines on the network make it easy to track all system resources available.

UNISOL JobAcct can be easily customized to fit your local installation requirements and can greatly simplify the resource chargeback task faced by many installations, while meeting government and government contractor reporting criteria.

5.2.1 Overview

UNISOL JobAcct is a job accounting and chargeback software package for UNIX and Windows NT systems and Oracle databases, and it was the first true project level charge-back product, first introduced in 1988.

UNISOL JobAcct performs actual cost accounting and chargeback for each user, group, project and cost center, or, proportional chargeback on a project/department level. Accounting can be restricted to specific commands or can include all commands, with individual charges broken down into:

- Connect time
- CPU usage
- Disk usage
- Memory usage
- Disk I/O operations
- Pages printed
- Miscellaneous charges
- Database usage

Reports are produced with charges itemized for each user, group, project charge-back number and cost center, and can include addresses for direct mailing to users and their organizations. Reports can be easily added or modified with awk or other report writing programs. ASCII reports (flat files) can also be generated and easily reformatted for use with databases or other financial programs.

Features of UNISOL JobAcct include:

- Allocation of charges to many project charge-back numbers by using user-level commands to change project charge-back numbers within a terminal session. System-wide or personal aliases can be used to refer to charge-back numbers.
- Charging different rates for prime-time, non-prime-time and reduced-charge hours, for each charge-back number.
- Variable accounting periods (in one-day intervals) for each project, user or group.
- Flexible distribution of disk charges among many charge-back numbers. Disk usage accounting includes minimum and maximum charges as well as standard disk usage averaging.
- ISP Accounting. Allows for set amounts of available resources for a minimum charge per month while charging for extra resources. Accomodates ISP-type accounting by providing a set of discounted

resources for a minimum monthly fee while charging different rates beyond the minimum resource usage.

- University Accounting. Keeps track of budgets by individual users and disables login or produces a warning when budget is exceeded.
- Users may individually (pre)view their accounting charges (up to the last available accounting summary) for each billing period.
- Reports may be easily combined/produced for all/many systems on the network from a single host.
- Generation of reports indicating logins of users and teletypes, and commands executed by each user (both detailed and summary reports).
- Summary reporting for a top-down view of the overall system or charge-back level accounting charges, including percentage information (of both resource usage and revenue sources) for each chargeable field, and generate pie-charts and bar-charts of system usage at the charge-back number level.
- Direct account billing and project-level budget tracking and reporting.
- A user-friendly Motif graphical user interface (Open Look for Sun Systems) simplifies the configuration of JobAcct control tables and the generation of standard and/or ad-hoc reports.
- Wildcards can be used when configuring user project files, thus increasing flexibility and reducing the amount of time spent updating project configuration files.

In order to perform budgeting and accounting by project, UNISOL JobAcct replaces the standard UNIX accounting system. A collector copy of JobAcct must be run on every machine for which accounting needs to be collected. A master version of the software runs on the server responsible for generating reports for all of the machines on the network.

5.2.2 Oracle database accounting

UNISOL JobAcct tracks database usage (for Oracle databases) for the following resources:

- Session connect time
- CPU usage
- Disk I/O operations
- Memory usage, and
- SQLNet I/O

Unlike other implementations of Oracle database accounting, UNISOL JobAcct does not use sampling to collect database usage data because that impacts system performance and may miss data if the sampling interval is too large. UNISOL JobAcct implements Oracle accounting by using database triggers to capture Oracle session activity when the session ends and saves the accounting information within the database itself. The database accounting data is available for inclusion in all of JobAcct's standard reports.

5.2.3 UNISOL JobAcct user interface

UNISOL JobAcct provides user-level commands for the generation of summary data and reports, suitable for inclusion in shell-scripts or execution from cron. An Application Management Interface is also available for both the curses and X-Windows environments (Motif or Open Look) for the effortless maintenance of product tables and control files, and for the generation of usage and billing reports. The Motif and Open Look interfaces also allow cursor manipulation of the menus and forms, and uses mnemonics and user-defined accelerators. In addition, context-sensitive help makes learning how to use the user interface an easy task.

Figure 23 shows a Motif display with the main product window, the menu bar, and an expanded Management menu showing the functions used to manage system-wide projects, user and project assignments and other product control files:

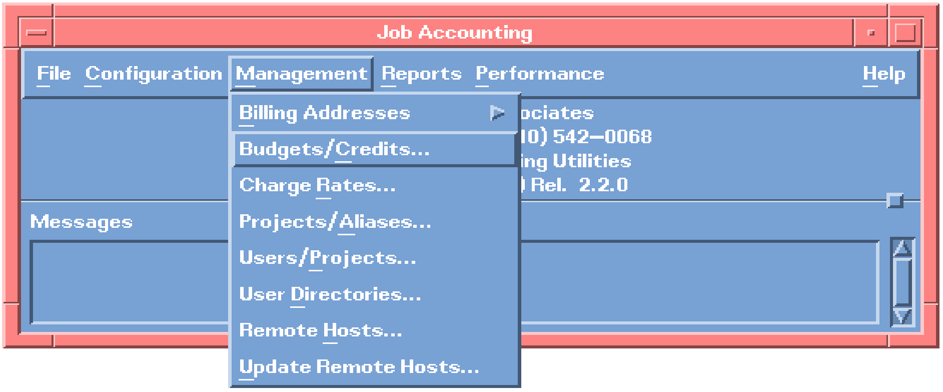


Figure 23. UNISOL JobAcct management menu

5.2.4 UNISOL JobAcct reports

Reports are produced with charges itemized for each user, group, project charge-back number and cost centers, and can include addresses for direct mailing to users and their organizations. New reports can be easily added (or charge-back reports can be easily changed) with awk or other report writing programs.

The reports currently available through JobAcct are:

- Detailed process accounting report
- Per-user command-usage report
- Detailed user login report
- Summary charge-back report
- Per-user charge-back report (default)
- Per-group charge-back report
- Per-chargenumber charge-back report
- Per-chargegroup charge-back report
- Top-level resource usage report
- Top-level fiscal report
- Top-level pie-charts showing system utilization (postscript printers)
- Top-level bar-charts showing system utilization (X displays)
- Budget Status reports (by uid or project) and automatic low-budget notification reports.

ASCII reports (flat files) can also be generated and easily reformatted for inclusion into databases or other financial programs.

The following Motif display shows the types of Summary Reports available from the Application Management Interface of UNISOL JobAcct:

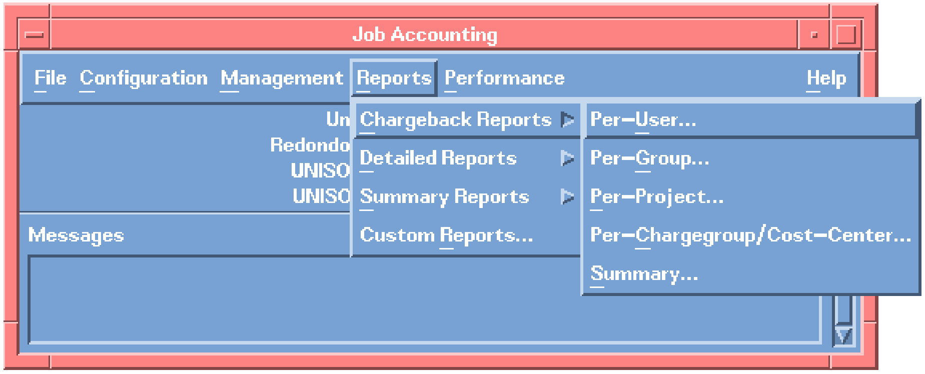
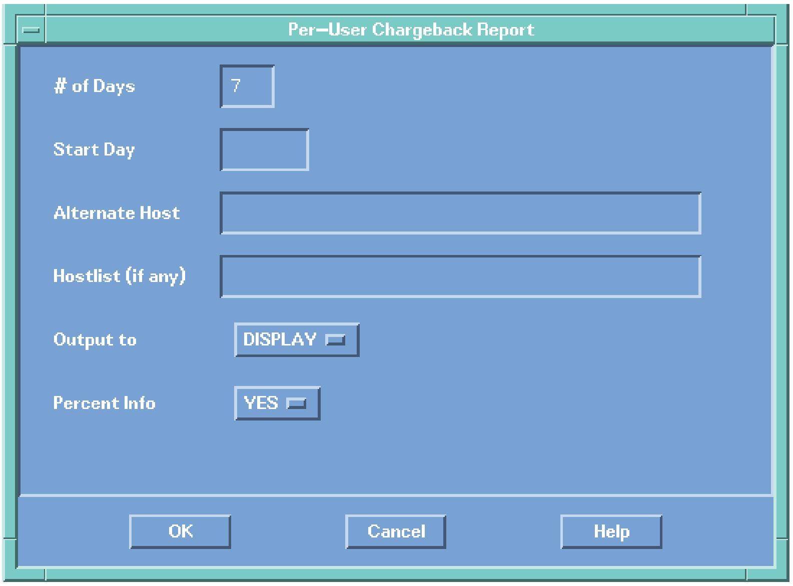


Figure 24. UNISOL JobAcct Summary Reports

Reports can be displayed on the display, sent to the printer, or saved in a file. Reports can also be produced for any specific range of days and hosts providing a flexible accounting period and reporting scheme. Figure 25 shows a Motif display with a weekly per-user charge-back report:


Per–User Chargeback Report

# of Days: 7
Start Day:
Alternate Host:
Hostlist (if any):
Output to: DISPLAY
Percent Info: YES

Figure 25. UNISOL JobAcct Chargeback Report

5.2.5 Performance monitoring

This optional module monitors system activity and summarizes system performance data at user-selected intervals.

Under normal operation, data is automatically collected (from system startup time) into daily summary reports and data may be independently analyzed for each day under user control and may be summarized and displayed to the display or printed.

Performance data summaries can be kept on-line for months allowing for the easy comparison of system performance from different time periods, or can be archived to tape for retrieval and analysis at a later time. Ad hoc monitoring and reporting is also supported providing live system monitoring and the generation of short-term reports.

The information that is recorded, reported, and analyzed by the System Performance Monitoring option includes:

- Processes (in run queue, blocked or swapped out)
- Memory Usage (used, available)
- Paging Activity (pages in, out)
- Swap Space Usage,
- Disk I/O (number of operations)
- Faults (interrupts, system calls, context switches)
- CPU usage (user, system, idle time)
- System Tables (files, inodes, processes, texts)
- User Logins
- Load Average

The summarized data can be further analyzed to provide an insight into system resource utilization and possible expansion needs, while providing graphs of interesting events suitable for inclusion in management reports or computer equipment expansion justifications.

X-Windows release of Performance Monitoring provides clear graphical representations of the collected information, and supports zoom capability and Postscript or Laserjet formatted graphs.

To get more information about this product, visit `http://www.unisol.com`.

5.3 CIMS for UNIX

CIMS for UNIX is an enterprise-wide solution for system accounting, resource management, project accounting, software tracking, capacity reporting, and chargeback. It integrates information from diverse UNIX platforms, creating presentation graphics for department heads, financial managers, project managers, and system administrators. CIMS for UNIX creates colorful pie charts, bar charts, area graphs, and line plots in two dimensions through its point-and-click GUI.

5.3.1 Overview

CIMS for UNIX collects detailed information on resources consumed, including CPU time, connect time, I/Os, pages printed, software package usage, and disk storage. It displays resources used and resource charges in flexible textual or graphic reports. CIMS for UNIX tracks department activity, user sessions, project accounts, software packages, databases, in-house applications, file systems, servers, and workstations. CIMS for UNIX also collects usage information from Oracle, DB2, and other databases to provide resource usage details not found in UNIX accounting files. With CIMS for UNIX, sites can track and cost-justify databases and equitably allocate database costs to departments, projects, and users.

This product currently utilizes a UNIX Motif GUI interface.

5.3.2 Benefits

Here are the main benefits of using CIMS for UNIX:

- Correctly allocates IT costs to resources and or projects.
- Accurately collects UNIX metrics for chargeback.
- Identifies applications and resources used by projects.
- Shows the software packages accessed by users.
- Tracks database usage trends for Oracle and DB2.
- Tracks usage trends on specific servers and workstations.

Multiple scenarios can be established, when time comes to deploy this software:

- Collects system accounting information from multiple UNIX platforms, including multiuser systems, servers, and workstations.
- Integrates information from diverse UNIX systems to provide enterprise-wide reporting of resource usage from a single server.
- Delivers enterprise chargeback when used with CIMS Desktop, CIMS for MVS, CIMS for NT, and CIMS for OpenVMS.

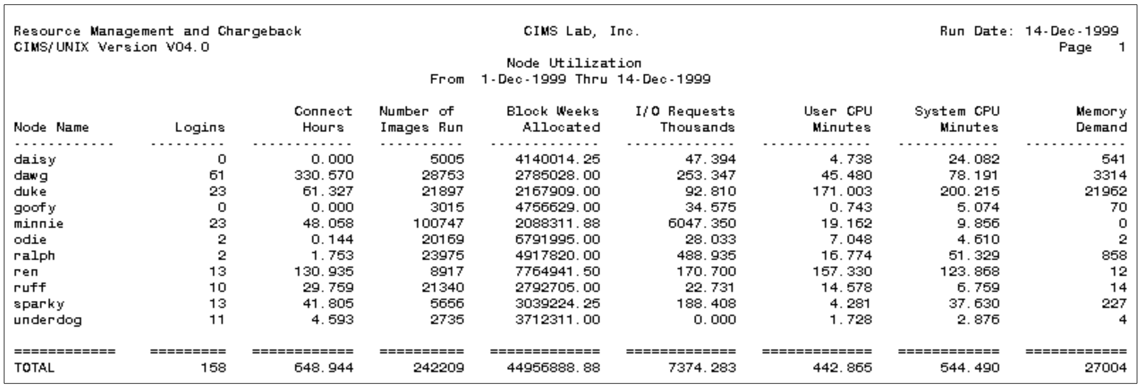
5.3.3 Sample reporting

CIMS UNIX provides native reporting capabilities as well as integration with CIMS Desktop to produce a wide range of reports. A list of the reports that can be produced using CIMS follows.

First let’s start with the CIMS UNIX Native reports:

- Node Utilization by Node
- Node Utilization by User
- DB2 Database Activity
- Oracle Database Activity
- Oracle User I/O Activity

Here is an example of what the Node Utilization by Node looks like:



```
Resource Management and Chargeback                          CIMS Lab, Inc.                                   Run Date: 14-Dec-1999
CIMS/UNIX Version V04.0                                                                                                   Page    1
                                                          Node Utilization
                                                 From  1-Dec-1999 Thru 14-Dec-1999

                                 Connect   Number of      Block Weeks    I/O Requests        User CPU     System CPU         Memory
Node Name          Logins          Hours   Images Run       Allocated       Thousands         Minutes        Minutes         Demand
------------    ---------   ------------   ----------   -------------   -------------   -------------   ------------   ------------
daisy                   0          0.000         5005      4140014.25          47.394           4.738         24.082            541
dawg                   61        330.570        28753      2785028.00         253.347          45.480         78.191           3314
duke                   23         61.327        21897      2167909.00          92.810         171.003        200.215          21962
goofy                   0          0.000         3015      4756629.00          34.575           0.743          5.074             70
minnie                 23         48.058       100747      2088311.88        6047.350          19.162          9.856              0
odie                    2          0.144        20169      6791995.00          28.033           7.048          4.610              2
ralph                   2          1.753        23975      4917820.00         488.935          16.774         51.329            858
ren                    13        130.935         8917      7764941.50         170.700         157.330        123.868             12
ruff                   10         29.759        21340      2792705.00          22.731          14.578          6.759             14
sparky                 13         41.805         5656      3039224.25         188.408           4.281         37.630            227
underdog               11          4.593         2735      3712311.00           0.000           1.728          2.876              4

============    =========   ============   ==========   =============   =============   =============   ============   ============
TOTAL                 158        648.944       242209     44956888.88        7374.283         442.865        544.490          27004
```

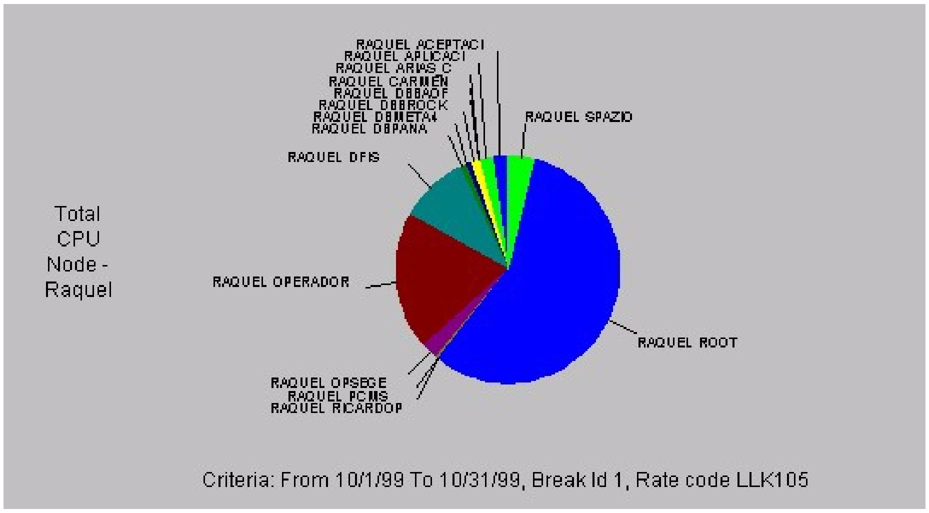
Figure 26. Example of the Node Utilization by node report

Now, let’s see the more advanced possibilities with the CIMS desktop:

- Charges by Cost Center
- Charges by Node
- Charges by Specific Node
- Charges by Resource Group
- System Connect Time by Cost Center
- Oracle Connect Time By Instance
- Oracle Connect Time By Node
- Oracle Disk Sorts by Cost Center & Instance

- Oracle Records by CPU
- Oracle Session CPU by Cost Center
- Oracle Session CPU by Instance
- Charges by Oracle Instance

Figure 27 shows an example of what the Charges by Specific Node looks like:



Criteria: From 10/1/99 To 10/31/99, Break Id 1, Rate code LLK105

Figure 27. Example of the charges by specific node report

To get more information about this product, visit `http://www.cimslab.com`.

Appendix A. Audit events

Table 7 lists all known events in AIX 4.3.3. A description of each event is also provided.

Table 7. Known events in AIX 4.3.3

USER/SYSTEM	AUDIT EVENT	Description
fork	PROC_Create	Creates a new process.
exit	PROC_Delete	Terminates the calling process.
exec	PROC_Execute	Executes a new program.
setuidx	PROC_RealUID	
	PROC_AuditID	
	PROC_SetUserIDs	
setgidx	PROC_RealGID	
accessx	FILE_Accessx	Determines the accessibility of a file.
statacl	FILE_StatAcl	Retrieves the access control information for a file.
statpriv	FILE_StatPriv	
revoke	FILE_Revoke	
frevoke	FILE_Frevoke	Revokes access to a file by other processes.
usrinfo	PROC_Environ	Change various piece of user information data.
sigaction	PROC_SetSignal	Action to take upon delivery of signal.
setrlimit	PROC_Limits	Controls max system resource consumption.
nice	PROC_SetPri	Use of the nice function.
setpri	PROC_Setpri	Sets fixed priority for process.
setpriv	PROC_Privilege	Changes one or more privilege vectors for process.
settimer	PROC_Settimer	Sets current value for a specified system wide timer.
adjtime	PROC_Adjtime	Changes system clock.

USER/SYSTEM	AUDIT EVENT	Description
ptrace	PROC_Debug	Traces the execution of another process.
kill	PROC_Kill	Sends a signal to a process or group of processes.
setpgid	PROC_setpgid	Sets the process id group.
ld_loadmodule	PROC_Load	Loads new object module into process address space.
	PROC_LoadMember	
	PROC_LoadError	
setgroups	PROC_SetGroups	Change process concurrent group set.
sysconfig	PROC_Sysconfig	Calls to the sysconfig subroutine.
audit	AUD_It	
auditbin	AUD_Bin_Def	Modification of auditbin.
auditevents	AUD_Events	Modification of Events.
auditobj	AUD_Objects	Modification of auditobj.
auditproc	AUD_Proc	
acct	ACCT_Disable	Disables system accounting.
	ACCT_Enable	Enables system accounting.
open and create	FILE_Open	Calls to the open subroutine.
	TCB_Leak	
	TCB_Mod	
	TCB_Exec	
read	FILE_Read	Reads from file descriptor.
write	FILE_Write	Writes data to descriptor.
close	FILE_Close	Closes open file descriptor.
link	FILE_Link	Creates new directory entry for file.
unlink	FILE_Unlink	Removes a file system object.
rename	FILE_Rename	Changes name of a file system object.

USER/SYSTEM	AUDIT EVENT	Description
chown	FILE_Owner	Changes file ownership.
chmod	FILE_Mode	Changes file mode.
fchmod	FILE_Fchmod	Changes file permission for file descriptor.
fchown	FILE_Fchown	Changes ownership for file descriptor.
truncate	FILE_Truncate	Calls to the truncate subroutine.
symlink	FILE_Symlink	Creates symbolic link.
pipe	FILE_Pipe	Creates unnamed pipe.
mknod	FILE_Mknod	Calls to the mknod subroutine.
fcntl	FILE_Dupfd	Duplicates file descriptor.
fscntl	FS_Extend	Extends file system.
mount	FS_Mount	Connects file system to named directory.
umount	FS_Umount	Disconnects mounted file system.
chacl	FILE_Acl	Changes file access control list (ACL)
	FILE_Facl	Changes ACL for file descriptor.
chpriv	FILE_Privilege	Calls to the chpriv subroutine.
	FILE_Chpriv	Changes privilege control list.
	FILE_Fchpriv	Changes PCL for file descriptor.
chdir	FS_Chdir	Changes current working directory.
fchdir	FS_Fchdir	Changes current working directory by file descriptor.
chroot	FS_Chroot	Changes meaning of “/” for current process.
rmdir	FS_Rmdir	Removes directory object.
mkdir	FS_Mkdir	Creates directory.
utimes	FILE_Utimes	Calls to the utimes subroutine.
stat	FILE_Stat	Calls to the stat subroutine.
msgget	MSG_Create	Creates new message queue.

USER/SYSTEM	AUDIT EVENT	Description
msgrcv	MSG_Read	Receives message from message queue.
msgsnd	MSG_Write	Sends message on message queue.
msgctl	MSG_Delete	Removes message queue.
	MSG_Owner	Changes ownership and access right of message queue.
	MSG_Mode	Queries access rights of message queue.
semget	SEM_Create	Creates new semaphore set.
semop	SEM_Op	Increases or decreases one or more semaphore.
semctl	SEM_Delete	Deletes semaphore set.
	SEM_Owner	Changes ownership and access rights for semaphore set.
	SEM_Mode	Queries semaphore set access rights.
shmget	SHM_Create	Creates new shared memory segment.
shmat	SHM_Open	Calls to the shmat subroutine with Open option.
shmat	SHM_Detach	Calls to the shmat subroutine with Detach option.
shmctl	SHM_Close	Closes shared memory segment.
	SHM_Owner	Changes ownership and access rights for shared memory segment.
	SHM_Mode	Queries access rights of shared memory segment.
tcpip user level	TCPIP_config	Logs changes to TCP/IP interface.
	TCPIP_host_id	Logs attempts to change system hostname.
	TCPIP_route	Logs changes to routing table.
	TCPIP_connect	Calls to the connect subroutine.
	TCPIP_data_out	Data sent.

USER/SYSTEM	AUDIT EVENT	Description
	TCPIP_data_in	Data received.
	TCPIP_access	
	TCPIP_set_time	Logs attempt to change system time via network.
tcpip kernel level	TCP_ksocket	Calls to the kernel TCPIP kernel services.
	TCP_ksocketpair	
	TCP_kclose	
	TCP_ksetopt	
	TCP_kbind	
	TCP_klisten	
	TCP_kconnect	
	TCP_kaccept	
	TCP_kshutdown	
	TCP_ksend	
	TCP_kreceive	
tsm	USER_Login	Calls to the Terminal State Management service.
	PORT_Locked	
	TERM_Logout	
rlogind/telnetd	USER_Exit	
sysck	SYSCK_Check	Calls to the sysck function.
	SYSCK_Update	
	SYSCK_Install	
	SYSCK_Delete	
tcbck	TCBCK_Check	Calls to the tcbck function.
	TCBCK_Update	
	TCBCK_Delete	

USER/SYSTEM	AUDIT EVENT	Description
usrck	USER_Check	Calls to the usrck function.
	USRCK_Error	
logout	USER_Logout	Calls to the logout subroutine.
chsec	PORT_Change	
chuser	USER_Change	Calls to the chuser subroutine.
rmuser	USER_Remove	Removes a user.
mkuser	USER_Create	Creates a user.
setgroups	USER_SetGroups	Calls to the setgroups subroutine.
setsenv	USER_SetEnv	Calls to the setenv subroutine.
su	USER_SU	Calls to the su subroutine.
grpck	GROUP_User	Calls to the grpchk subroutine.
	GROUP_Adms	
chgroup	GROUP_Change	Calls to the chgroup subroutine.
mkgroup	GROUP_Create	Calls to the mkgroup subroutine.
rmgroup	GROUP_Remove	Calls to the rmgroup subroutine.
passwd	PASSWORD_Change	Changes a user password.
pwdadm	PASSWORD_Flags	Calls to the pwdadm subroutine.
pwdck	PASSWORD_Check	Calls to the pwdck subroutine.
	PASWORD_Ckerr	
startsrc	SRC_Start	Starts a system resource controller.
stopsrc	SRC_Stop	Stops a system resource controller.
addssys	SRC_Addssys	Calls to the addsys subroutine.
chssys	SRC_Chssys	Calls to the chssys subroutine.
addserver	SRC_Addserver	Calls to the addserver subroutine.
chserver	SRC_Chserver	Calls to the chserver subroutine.
rmsys	SRC_Delssys	Calls to the rmsys subroutine.
rmserver	SRC_Delserver	Calls to the rmserver subroutine.

USER/SYSTEM	AUDIT EVENT	Description
enq	ENQUE_admin	Calls to the enq subroutine.
qdaemon	ENQUE_exec	Calls to the qdaemon subroutine.
sendmail	SENDMAIL_Config	Calls to the sendmail fucntion.
	SENDMAIL_ToFile	
	MAIL_ToUser	
at	AT_JobAdd	Calls to the at function.
	At_JobRemove	
cron	CRON_JobRemove	Calls to the cron function.
	CRON_JobAdd	
	CRON_Start	Start of a cron job.
	CRON_Finish	End of a cron job.
nvload	NVRAM_Config	Access to the NVRAM.
cfgmgr	DEV_Configure	Calls to the cfgmgr function.
chdev and mkdev	DEV_Change	Device changed.
mkdev	DEV_Create	Device created.
	DEV_Start	Device started.
installp	INSTALLP_Inst	Calls to the installp function.
	INSTALLP_Exec	
rmdev	DEV_Stop	Device stopped.
	DEV_Unconfigure	Device unconfigured.
	DEV_Remove	Device removed.
DSMIT	DSMIT_start	Calls to the dsmit function.
	DSMIT_end	
lchangelv	LVM_ChangeLV	Calls to the lvm function.
lextendlv	LVM_ChangeLV	
lreducelv	LVM_ChangeLV	
lchangepv	LVM_ChangeVG	

USER/SYSTEM	AUDIT EVENT	Description
ldeletepv	LVM_ChangeVG	
linstallpv	LVM_ChangeVG	
lcreatelv	LVM_CreateLV	
lcreatevg	LVM_CreateVG	
ldeletepv	LVM_DeleteVG	
rmlv	LVM_DeleteLV	
lvaryoffvg	LVM_VaryoffVG	
lvaryonvg	LVM_VaryonVG	
logical volume operations	LVM_AddLV	
	LVM_KDeleteLV	
	LVM_ExtendLV	
	LVM_ReduceLV	
	LVM_KChangeLV	
	LVM_AvoidLV	
physical volume operations	LVM_MissingPV	
	LVM_AddPV	
	LVM_AddMissPV	
	LVM_DeletePV	
	LVM_RemovePV	
	LVM_AddVGSA	
	LVM_DeleteVGSA	
volume group operations	LVM_SetupVG	
	LVM_DefineVG	
	LVM_KDeleteVG	
	LVM_ChgQuorum	

USER/SYSTEM	AUDIT EVENT	Description
	LVM_Chg1016	
	LVM_UnlockDisk	
	LVM_LockDisk	
backup and restore	BACKUP_Export	Calls to the backup/restore function.
	BACKUP_Priv	
	RESTORE_Import	
shell	USER_Shell	Access to the shell.
reboot	USER_Reboot	Calls to the reboot function.
	PROC_Reboot	

Appendix B. Internal structure of the accounting files

In this appendix, we describe the internal structure of the binary accounting files. The structure of most the accounting files can be found in the /usr/include directory (contained in the bos.adt.include fileset), with the single exception of the command summary files.

B.1 The tacct file

The structure of the tacct files can be found in /usr/include/sys/tacct.h. Note that the original file indicates that the ta_io values are given in 512 byte values, which is not true. They are given in 1024 byte values.

```
/* Float arrays below contain prime and non-prime components */
struct  tacct   {
        uid_t           ta_uid;         /* userid */
        char            ta_name[8];     /* login name */
        float           ta_cpu[2];      /* cum. cpu time (mins) */
        float           ta_kcore[2];    /* cum kcore-mins */
        float           ta_io[2];       /* cum. chars xferred (KB) */
        float           ta_rw[2];       /* cum. blocks read/written */
        float           ta_con[2];      /* cum. connect time (mins) */
        float           ta_du;          /* cum. disk usage */
        long            ta_qsys;        /* queueing 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 */
};
```

B.2 The wtmp file

The structure of the utmp and wtmp files can be found in /usr/include/utmp.h.

```
struct utmp
  {
        char ut_user[8] ;               /* User login name */
        char ut_id[14] ;                /* /etc/inittab id */
        char ut_line[12] ;              /* device name (console, lnxx) */
        short ut_type ;                 /* type of entry */
        pid_t ut_pid ;                  /* process id */
        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.
                                         */
        time_t ut_time ;                /* time entry was made */
        char ut_host[16] ;              /* host name */
  } ;

/*      Definitions for ut_type                                       */

#define EMPTY           0
#define RUN_LVL         1
#define BOOT_TIME       2
#define OLD_TIME        3
#define NEW_TIME        4
#define INIT_PROCESS    5       /* Process spawned by "init" */
#define LOGIN_PROCESS   6      /* A "getty" process waiting for login */
#define USER_PROCESS    7       /* A user process */
#define DEAD_PROCESS    8
#define ACCOUNTING      9
```

B.3 The pacct file

The structure of the pacct files can be found in /usr/include/sys/acct.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 */
        uid_t   ac_uid;                 /* Accounting user ID */
        gid_t   ac_gid;                 /* Accounting group ID */
        dev_t   ac_tty;                 /* control typewriter */
        time_t  ac_btime;               /* Beginning time */
        comp_t  ac_utime;              /* acctng user time in seconds */
        comp_t  ac_stime;              /* acctng system time in seconds
*/
```

```
       comp_t  ac_etime;              /* acctng elapsed time in seconds
*/
        comp_t  ac_mem;                 /* memory usage */
        comp_t  ac_io;                  /* chars transferred */
        comp_t  ac_rw;                  /* blocks read or written */
        char    ac_comm[8];             /* command name */
};

#define AFORK   0001            /* has executed fork, but no exec */
#define ASU     0002            /* used super-user privileges */
#define ACOMPAT 0004            /* used compatibilty mode */
#define ACORE   0010            /* dumped core */
#define AXSIG   0020            /* killed by signal */
#define ACCTF   0300            /* record type: 00 = acct */

/*
 * 1/AHZ is the granularity of the data encoded in the various
 * comp_t fields. This is not necessarily equal to hz.
 */

#define AHZ 64
```

B.4 The qacct file

The structure of the qacct file can be found in /usr/include/sys/accrec.h.

```
/* 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;
```

B.5 The cms file

The `acctcms` command can work with two types of command summary files, either containing total cms records or prime/non-prime cms records. The cms files used by the accounting system are of the second type. To work with total cms record formats, you should use the -t flag.

```
#define CMDSIZE  9
```

```
/*  Total cms records format */
struct tcms {
        char    tcm_comm[CMDSIZE];      /* command name */
        ulong   tcm_pc;         /* number of processes */
        float   tcm_cpu;        /* cpu time(min) */
        float   tcm_real;       /* real time(min) */
        float   tcm_kcore;      /* kcore-minutes */
        ulong   tcm_io;         /* chars transferred */
        ulong   tcm_rw;         /* blocks read */
} ;
struct tcms     *tcm;

/* prime/nonprime CMS record format */
struct pcms {
        char    pcm_comm[CMDSIZE];      /* command name */
        ulong   pcm_pc[2];      /* number of processes */
        float   pcm_cpu[2];     /* cpu time(min) */
        float   pcm_real[2];    /* real time(min) */
        float   pcm_kcore[2];   /* kcore-minutes */
        float   pcm_io[2];      /* chars transferred */
        float   pcm_rw[2];      /* blocks read */
} ;
```

Appendix C. Special notices

This publication is intended to help system engineers, IT architects, and consultants understand the capabilities of the AIX operating system in terms of auditing and accounting. The information in this publication is not intended to be the specification of any programming interfaces that are provided by the AIX operating system. See the base documentation of AIX for more information.

References in this publication to IBM products, programs or services does not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only products, programs, or services of IBM may be used. Any functionally equivalent program that does not infringe any IBM intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

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

IBM ®
PAL
Redbooks
RMF
SP0
TCS
Netfinity
RACF
Redbooks Logo
RS/6000
System/39

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

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

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Lotus Notes is a registered trademark of Lotus Development Corporation.

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

Appendix D. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 163.

- *AIX 4.3 Elements of Security Effective and Efficient Implementation*, SG24-5962
- *Understanding IBM RS/6000 Performance and Sizing*, SG24-4810

D.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.3 Other resources

These publications are also relevant as further information sources:

- *AIX Version 4.3 Commands Reference*, SBOF-1877

D.4 Referenced Web sites

These Web sites are also relevant information sources:

- Presentation for the UNISOL® JobAcctTM product: `http://www.unisol.com`
- Presentation of the CIMS for UNIX product: `http://www.cimslab.com/cimsunix.html`
- Presentation of the Coschargeback product: `http://www.coschargeback.com`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** **ibm.com**/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

□ Invoice to customer number

□ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Abbreviations and acronyms

ACE	Access Control Entries
ACL	Access Control List
ADSM	ADSTAR Distributed Storage Manager
AFS	Andrew File System
AIX	Advanced Interactive eXecutive
APA	All Points Addressable
API	Application Programming Interface
APPC	Advanced Program-to-Program
APPN	Advanced Peer-to-Peer Networking
ARC	Advanced RISC Computer
ARPA	Advanced Research Projects Agency
ASCII	American National Standard Code for Information Interchange
ATE	Asynchronous Terminal Emulation
ATM	Asynchronous Transfer Mode
AVI	Audio Video Interleaved
BDC	Backup Domain Controller
BIND	Berkeley Internet Name Domain
BNU	Basic Network Utilities
BOS	Base Operating System
BRI	Basic Rate Interface
BSD	Berkeley Software Distribution
BUMP	Bring-Up Microprocessor
C-SPOC	Cluster single point of control
CDE	Common Desktop Environment
CDMF	Commercial Data Masking Facility
CDS	Cell Directory Service
CERT	Computer Emergency Response Team
CGI	Common Gateway Interface
CHAP	Challenge Handshake Authentication
CIDR	Classless InterDomain Routing
CMA	Concert Multithreaded Architecture
CO	Central Office
COPS	Computer Oracle and Password System
CPI-C	Common Programming Interface for Communications
CSNW	Client Service for NetWare
CSR	Client/Server Runtime
DAC	Discretionary Access Controls
DARPA	Defense Advanced Research Projects Agency
DASD	Direct Access Storage Device
DBM	Database Management
DCE	Distributed Computing Environment

DCOM	Distributed Component Object Model
DDE	Dynamic Data Exchange
DDNS	Dynamic Domain Name System
DES	Data Encryption Standard
DFS	Distributed File System
DHCP	Dynamic Host Configuration Protocol
DLC	Data Link Control
DLL	Dynamic Load Library
DNS	Domain Name System
DTS	Distributed Time Service
EGID	Effective Group Identifier
EMS	Event Management Services
EPROM	Erasable Programmable Read-Only
ERP	Enterprise Resources Planning
ESCON	Enterprise System Connection
ESP	Encapsulating Security Payload
EUID	Effective User Identifier
FAT	File Allocation Table
FDDI	Fiber Distributed Data Interface
FIFO	First In/First Out
FIRST	Forum of Incident Response and Security
FQDN	Fully Qualified Domain Name
FSF	File Storage Facility
FTP	File Transfer Protocol
FtDisk	Fault-Tolerant Disk
GDA	Global Directory Agent
GDI	Graphical Device Interface
GDS	Global Directory Service
GID	Group Identifier
GSNW	Gateway Service for NetWare
GUI	Graphical User Interface
HACMP	High Availability Cluster Multiprocessing
HAL	Hardware Abstraction Layer
HCL	Hardware Compatibility List
IBM	International Business Machines
ICCM	Inter-Client Conventions Manual
IDE	Integrated Drive Electronics
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IIS	Internet Information Server
IMAP	Internet Message Access Protocol
IPC	Interprocess Communication

IPL	Initial Program Load
IPX	Internetwork Packet eXchange
ISA	Industry Standard Architecture
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ISS	Interactive Session Support
ITSEC	Initial Technology Security Evaluation
ITSO	International Technical Support Organization
ITU	International Telecommunications Union
IXC	Inter Exchange Carrier
JFS	Journaled File System
JIT	Just-In-Time
LAN	Local Area Network
LCN	Logical Cluster Number
LDAP	Lightweight Directory Access Protocol
LFS	Log File Service (Windows NT)
LFS	Logical File System **(**AIX**)**
LFT	Low Function Terminal
LOS	Layered Operating System
LP	Logical Partition
LPC	Local Procedure Call
LPD	Line Printer Daemon
LPP	Licensed Program Product
LRU	Least Recently Used
LSA	Local Security Authority
LUID	Login User Identifier
LV	Logical Volume
LVDD	Logical Volume Device Driver
LVM	Logical Volume Manager
MBR	Master Boot Record
MFT	Master File Table
MMC	Microsoft Management Console
MOCL	Managed Object Class Library
MPTN	Multiprotocol Transport Network
MSS	Maximum Segment Size
MWC	Mirror Write Consistency
NBF	NetBEUI Frame
NBPI	Number of Bytes per I-node
NCP	NetWare Core Protocol
NCS	Network Computing System
NCSC	National Computer Security Center
NDIS	Network Device Interface Specification
NDS	NetWare Directory Service
NETID	Network Identifier
NFS	Network File System
NIM	Network Installation Management
NIS	Network Information System

NIST	National Institute of Standards and Technology
NLS	National Language Support
NNS	Novell Network Services
NSAPI	Netscape Commerce Server's Application
NTFS	NT File System
NTLDR	NT Loader
NTP	Network Time Protocol
NTVDM	NT Virtual DOS Machine
NVRAM	Non-Volatile Random Access Memory
NetBEUI	NetBIOS Extended User Interface
NetDDE	Network Dynamic Data Exchange
OCS	On-Chip Sequencer
ODBC	Open Database Connectivity
ODM	Object Data Manager
OLTP	OnLine Transaction Processing
ONC	Open Network Computing
OS	Operating System
OSF	Open Software Foundation
PAL	Platform Abstract Layer
PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PBX	Private Branch Exchange
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card
PDC	Primary Domain Controller
PDF	Portable Document Format
PDT	Performance Diagnostic Tool
PEX	PHIGS Extension to X
PFS	Physical File System
PHIGS	Programmer's Hierarchical Interactive Graphics System
PMTU	Path Maximum Transfer Unit
POP	Post Office Protocol
POSIX	Portable Operating System Interface for Computer Environment
POST	Power-On Self Test
PP	Physical Partition
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
PReP	PowerPC Reference Platform
PSN	Program Sector Number
PSSP	Parallel System Support Program
PV	Physical Volume
PVid	Physical Volume Identifier
QoS	Quality of Service
RACF	Resource Access Control Facility

RAID	Redundant Array of Independent Disks
RAS	Remote Access Service
RFC	Request for Comments
RGID	Real Group Identifier
RMSS	Reduced-Memory System Simulator
ROLTP	Relative OnLine Transaction Processing
ROS	Read-Only Storage
RPC	Remote Procedure Call
SAK	Secure Attention Key
SAM	Security Account Manager
SATAN	Security Analysis Tool for Auditing
SCSI	Small Computer System Interface
SDK	Software Developer's Kit
SFG	Shared Folders Gateway
SID	Security Identifier
SLIP	Serial Line Internet Protocol
SMB	Server Message Block
SMIT	System Management Interface Tool
SMP	Symmetric Multiprocessor
SMS	Systems Management Server
SNA	Systems Network Architecture
SNAPI	SNA Interactive Transaction Program
SNMP	Simple Network Management Protocol
SP	System Parallel
SPX	Sequenced Packet eXchange
SRM	Security Reference Monitor
SSA	Serial Storage Architecture
SSL	Secure Sockets Layer
SVC	Serviceability
SWS	Silly Window Syndrome
TAPI	Telephone Application Program Interface
TCB	Trusted Computing Base
TCSEC	Trusted Computer System Evaluation
TDI	Transport Data Interface
TTL	Time to Live
UDP	User Datagram Protocol
UID	User Identifier
UMS	Ultimedia Services
UNC	Universal Naming Convention
UPS	Uninterruptable Power Supply
UTC	Universal Time Coordinated
UUCP	UNIX to UNIX Communication Protocol
UUID	Universally Unique Identifier
VAX	Virtual Address eXtension
VCN	Virtual Cluster Name
VFS	Virtual File System

VG	Volume Group
VGDA	Volume Group Descriptor Area
VGSA	Volume Group Status Area
VGid	Volume Group Identifier
VMM	Virtual Memory Manager
VPD	Vital Product Data
VPN	Virtual Private Network
VRMF	Version, Release, Modification, Fix
VSM	Virtual System Management
W3C	World Wide Web Consortium
WAN	Wide Area Network
WINS	Windows Internet Name Service
WOW	Windows-16 on Win32
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
WinMSD	Windows Microsoft Diagnostics
XCMF	X/Open Common Management Framework
XDM	X Display Manager
XDMCP	X Display Manager Control Protocol
XDR	eXternal Data Representation
XNS	XEROX Network Systems
XPG4	X/Open Portability Guide

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at ibm.com/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6020-00 Auditing and Accounting on AIX
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your e-mail address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	 O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. ibm.com/privacy/yourprivacy/

IBM

Redbooks

Auditing and Accounting on AIX

Auditing and Accounting on AIX

Comprehensive guide to auditing and accounting on your AIX-based system

Step-by-step instructions on auditing your system

Find the most effective way to use accounting to track system resources

Auditing and Accounting on AIX is your comprehensive guide to setting up, maintaining, and troubleshooting the advanced auditing and accounting features on your AIX systems. Generously illustrated instructions will guide 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 will plug into the AIX suite. A chapter specific to SP solutions is provided.

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 a list of rules of thumb 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.

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

SG24-6020-00 ISBN 0738418501