Accounting [10]

The UNICOS system provides two types of system accounting, standard UNIX System V accounting or Cray system accounting (CSA). You may use one or the other of these accounting packages at your site. To help you decide which accounting package to use, see Section 10.3, page 243, which describes the unique features of CSA.

This chapter describes CSA, which is the more complete and frequently used of the two accounting types. It includes the following:

- An overview of CSA, including unique CSA features, descriptions of directories and files, and the `/usr/lib/acct/csarun` primary daily accounting shell script.
- Procedures to follow so that you can set up CSA and execute daily accounting procedures that result in the generation of a variety of reports.

Your accounting configuration file is located in `/etc/config/acct_config`. A sample file is provided at the end of this chapter; the sample file may differ slightly from the one included with your system.

For information on using standard UNIX System V accounting, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

10.1 Related accounting documentation

The following publications contain more detailed information about the material covered in this section:

- *UNICOS Resource Administration*, Cray Research publication SG–2302, "Accounting" chapter
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `acctdusg`(8), `chargefee`(8), `csaboots`(8), `csabuild`(8), `csacon`(8), `csacrep`(8), `csadrep`(8), `csaedit`(8), `csajrep`(8), `csaline`(8), `csanqs`(8), `csapacct`(8), `csaperiod`(8), `csarecy`(8), `csarun`(8), `csaverify`(8), `devacct`(8), `diskusg`(8), `dodisk`(8), `nulladm`(8), `runacct`(8), and `setacid`(8) man pages
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `acctcom`(1), `ja`(1), `last`(1B), and `who`(1) man pages

10.2 Concepts and terminology

The following concepts and terms are important in CSA:

Term	Description
Daily accounting	Unlike the standard daily accounting, CSA's accounting can be run as many times as necessary during a day. However, this feature is still referred to as daily accounting.
Periodic accounting	Accounting similar to the standard UNICOS monthly accounting. CSA, however, lets system administrators specify the time periods for which "monthly" or cumulative accounting will be run. Thus, periodic accounting can be run more than once a month.
Recycled data	By default, accounting data for active sessions is recycled until the session terminates. CSA reports data for only terminated sessions, unless you invoke the `csarun` command with the `-A` option. `csarun` places recycled data into data files in the `/usr/adm/acct/day` directory. These data files are suffixed with 0; for example, per-process accounting data for active sessions from previous accounting periods is in the `/usr/adm/acct/day/pacct0` file. For information about recycled data, see *UNICOS Resource Administration*, Cray Research publication SG–2302.
Session	CSA organizes accounting data by sessions and boot times, and then it places the data into a session record file. For non-NQS jobs, a *session* consists of all accounting data for a given job ID during one boot period. A *session* for an NQS job consists of the accounting data for all job IDs associated with the job's NQS sequence number/machine name identifier. NQS jobs may span multiple boot periods. If a job is restarted, it has the same job ID associated with it during all boot periods in which it runs. Rerun NQS jobs

	have multiple job IDs. CSA treats all phases of an NQS job as being in the same session.
Uptime/boot period	A period delineated by the system boot times found in `/etc/csainfo`. The `csaboots` command writes to this file during system boot.

10.3 Unique features of CSA

Like the UNIX System V accounting package, CSA provides methods to collect per-process data, record connect sessions, monitor disk usage, and charge fees. However, CSA also provides other facilities not available from the standard accounting package, including the following:

- Per-job accounting.
- Device accounting; categories include logical, block, and character special devices. Disk usage information is not available on a job basis; however, to bill disk usage on a user or account ID basis, you can use output from the `dodisk` command.

  Note: The system overhead for device accounting is fairly low. However, the amount of accounting data produced in the worst cases is more than double that produced by standard accounting. The more device accounting data is kept, the more file system space is required. If one device is accounted for, processes that use that device generate twice as much accounting data as a process that did not use the device or the same process without device accounting. However, for 1 to `NODEVACCT` device types, the maximum size of the accounting data does not increase, except that more processes may use one of the devices.
- Daemon accounting (for NQS and the tape subsystem); accounting information is available from the NQS and online tape daemons. Data is written to the `nqacct` and `tpacct` files, respectively, in the `/usr/adm/acct/day` directory. The NQS and online tape daemons also must enable accounting.
- Device usage by account ID.
- Arbitrary accounting periods; for example, you can set your accounting period to be from 4 A.M. to 4 P.M. rather than using the default period.
- Flexible system billing unit (SBU) scheme.
- One file that contains all data from the accounting period.

- Archiving of accounting data, so you can move the data to a front-end system and merge it with your other accounting information.
- Capability to perform additional site-specific processing during daily accounting.
- Error recovery and automatic repairing of file systems.

For detailed information on these facilities, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

10.4 Accounting directories and files

This section provides a brief overview of the CSA file and directory structure. The following directories apply to both UNIX System V and CSA and are the main accounting directories. For a more complete description of the files and directories, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

Note: Consider configuring the `/usr/adm` directory as another file system so that if `/usr/adm` fills up, other directories (such as `/usr/mail`) are still usable.

The `/tmp` directory also is used while the `csarun` script is running. (For information about the `/tmp` directory, see Chapter 5, page 51.)

Directory	Description
`/usr/lib/acct`	Contains all of the programs and scripts used to run either CSA or UNIX System V accounting. (For a complete list of programs and scripts, see *UNICOS Resource Administration*, Cray Research publication SG–2302.) The only CSA program not located here is `/etc/csaboots` (see the `csaboots`(8) man page), which records boot times at system startup. Programs used only by CSA begin with the characters `csa`. This directory also may contain the `csa.archive1`, `csa.archive2`, `csa.fef`, and `csa.user` user-exit scripts if you enable them. (To determine whether your UNICOS release level allows these scripts, see *UNICOS Resource Administration*, Cray Research publication SG–2302.)

`/usr/adm/acct/day`

Contains the following current and recycled accounting files for per-process, disk, and daemon accounting:

- `dtmp` (disk accounting data)
- `nqacct*` (NQS daemon accounting data)
- `pacct*` (per-process accounting data)
- `tpacct*` (tape daemon accounting data)

Note: Accounting files in `/usr/adm/acct/day` that include the suffix 0 in their file names, contain data from sessions that did not complete during the previous accounting periods.

During CSA data processing, sites may select to archive the raw and/or processed data offline. For a description of how to do this, see the "Accounting" chapter in *UNICOS Resource Administration*, Cray Research publication SG–2302. By default, all raw data files are deleted after use and are not archived.

`/usr/adm/acct/fiscal`

Contains periodic files created by `csaperiod` (CSA) or `monacct` (System V). Within this directory, the `rpt/MMDD/hhmm/rprt` file contains a variety of CSA periodic reports.

`/usr/adm/acct/nite`

Contains files that are reused daily by `csarun` (CSA) or the `runacct` (System V) procedure. Contains processing messages and errors (files that have names beginning with `E` and ending with the date and time).

`/usr/adm/acct/sum`

Contains cumulative summary files updated by `csarun` (CSA) or `runacct` (System V). Within this directory, the

`rpt/MMDD/hhmm/rprt` file contains a variety of CSA daily reports.

`/usr/adm/acct/work`

Contains temporary files used by daily accounting procedures.

The following are the main basic accounting files:

File	Description
`/etc/config/acct_config`	Accounting configuration file; contains the configurable parameters used by the accounting commands.
`/etc/csainfo`	Contains system boot time, written to this file by `/etc/csaboots`.
`/etc/wtmp`	Contains login and logout records for users. Records tty, process ID, type of process, and connect-time accounting data. For information about fixing `wtmp` errors, see Section 10.9, page 254.
`/usr/adm/acct/day/pacct`	Contains data file written by the UNICOS kernel. Source of all process accounting for both CSA and System V accounting.
`/usr/adm/acct/nite/statefile`	Contains the name of the next reentrant state so that the `csarun` accounting script can be restarted at a specified point. For descriptions of CSA states (files that have names beginning with `E` and ending with the date and time contain errors), see Section 10.8, page 252.

CSA outputs the following six data files:

File	Description
`/tmp/AC.`*MMDD*`/`*hhmm*`/Super-record`	Session record file; this file usually is deleted after CSA has used it.
`/usr/adm/acct/fiscal/data/`*MMDD*`/`*hhmm*`/pdacct`	Consolidated periodic data.
`/usr/adm/acct/fiscal/data/`*MMDD*`/`*hhmm*`/cms`	Periodic command usage data.
`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/cacct`	Consolidated daily data; if you specify the `-r` option, `csaperiod` deletes this file.
`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/cms`	Daily command usage data; if you specify the `-r` option, `csaperiod` deletes this file.
`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/dacct`	Daily disk usage data; if you specify the `-r` option, `csaperiod` deletes this file.

Note: Occasionally, sites run on numerous `/` and `/usr` file systems and want to maintain the same accounting files throughout. The easiest way to accomplish this is to put `/usr/adm` or `/usr/adm/acct` on a separate file system and to mount this file system along with each different system. You also must copy two other files, `/etc/csainfo` and `/etc/wtmp`, from the previously booted `/` file system. You must copy these files to the new `/` file system before it is brought up. If you do not copy `/etc/csainfo` file to the new `/` file system correctly, `csarun` may abort abnormally. When `/etc/wtmp` is not copied, incorrect connect-time data is reported.

10.5 Daily operation overview of CSA

When the UNICOS system is run in multiuser mode, accounting behaves as described in the following steps and shown in Figure 2, page 250. However, if you modify CSA to meet your own requirements, the following steps may not reflect the process at your site:

1. `/etc/csaboots`

   Writes boot record to `/etc/csainfo`, which contains a record of every system boot; executed by `/etc/rc` on entering multiuser state.

2. `/usr/lib/acct/startup`

   a. Executed by `/etc/rc` on entering multiuser state (see `acctsh`(8) for additional information).

   b. `acctwtmp` adds a boot record to `/etc/wtmp`.

   c. `turnacct` starts per-process accounting.

   d. `turnacct` enables daemon accounting if it is enabled in the `acct_config` file. By default, `/usr/lib/acct/startup` enables daemon accounting.

   e. `remove` cleans up previous day's files.

3. When they are started by `/etc/rc`, the NQS and tape daemons enable daemon accounting.

4. `/usr/lib/acct/ckpacct`

   a. Executed by `cron` every hour to check size of `/usr/adm/acct/day/pacct`. If `pacct` gets too large, a new file is started. The new file(s) will have a `.x` suffix; `.x` is .1, .2, .3, and so on.

   b. Verifies that at least 500 free data blocks exist in the file system that contains the `/usr/adm/acct` directory; if the file system is full, `ckpacct` will turn off accounting.

5. `/usr/lib/acct/ckdacct`

   a. Executed by `cron` every hour to check size of daemon accounting files. If an accounting file gets too large, a new file is started.

   b. Verifies that at least 500 free data blocks exist in the file system that contains the `/usr/adm/acct` directory. `ckdacct` turns off daemon accounting if the file system is full.

6. The `cron` utility runs the `dodisk` script periodically to generate a snapshot of the amount of disk space being used by each user.

7. `/usr/lib/acct/csarun` (also see Section 10.7, page 251)

   a. Executed by `cron` at specified times.

b. Processes active accounting files, combining data from `pacct`, `/etc/wtmp`, `nqacct`, and `tpacct`.

c. Produces accounting reports and a consolidated data file.

8. `/usr/lib/acct/shutacct`

a. Executed by `/etc/shutdown`.

b. `acctwtmp` writes shutdown reason in `/etc/wtmp`.

c. `turnacct` stops per-process accounting.

d. `turndacct` stops daemon accounting.

9. (Optional) `/usr/lib/acct/chargefee`

a. Creates a fee file; a site must invoke this (see the `chargefee`(8) man page).

b. (Optional) `/usr/lib/acct/csaperiod`

i. Runs periodic accounting and is executed by `cron` to process consolidated accounting data from previous accounting periods.

ii. Produces a consolidated periodic accounting file and an ASCII report.

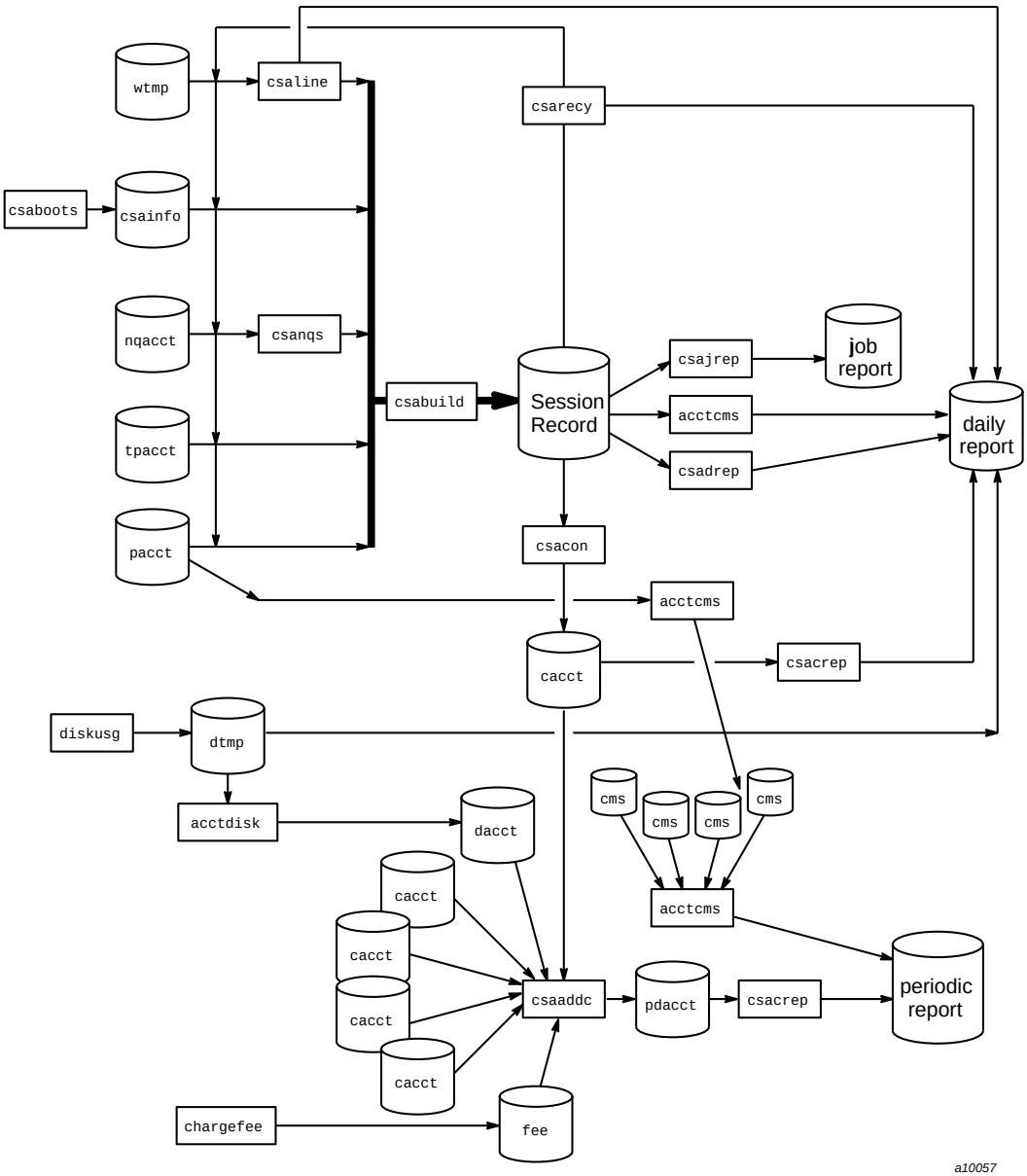

Figure 2. Daily operation overview of CSA

10.6 Customizing your system billing procedure

UNICOS system accounting has been designed to be easy for you to customize to meet your site's requirements. For instance, you may want to change the way the system charges system billing units (SBUs) for various kinds of services, to define the weighting factor used in calculating SBUs for various system services (such as tape requests, NQS requests, or connect time), to change the definition of the memory integral to be used in connection with memory charges for multitasking programs, and so on. You may want to set prime time equal to nonprime time and charge based on NQS queue usage. To do so, you must modify the `/etc/config/acct_config` file (a sample file is included beginning on page Section 10.13, page 262 of this chapter). By default, all SBUs are set to 0.0. Read the commented instructions throughout the file to determine which lines of the `acct_config` file you want to modify for your site's needs. You also can use your local SBU calculations by modifying the default algorithms defined in `/usr/src/cmd/acct/lib/acct/user_sbu.c`, compiling, and relinking the accounting programs.

10.7 The `csarun` command

The `/usr/lib/acct/csarun` command (see the `csarun`(8) man page) is the primary daily accounting shell script. It processes connect, per-process, and daemon accounting files and usually is initiated by `cron` during nonprime hours. `csarun` also contains four user-exit points, allowing you to tailor the daily run of accounting to your specific needs (for information on setting up user exits callable from `csarun` and callable from `runacct`, see *UNICOS Resource Administration*, Cray Research publication SG–2302).

If errors occur, `csarun` does not damage files. It contains a series of protection mechanisms that try to recognize an error, provide intelligent diagnostics, and terminate processing in such a way that `csarun` can be restarted with minimal intervention.

You also can interactively process data for a new accounting period by executing the following command. Before executing `csarun` in this manner, ensure that the previous invocation completed successfully by looking at the `active` and `statefile` files in `/usr/adm/acct/nite` . Both files should specify that the last invocation completed successfully.

```
nohup csarun 2> /usr/adm/acct/nite/fd2log &
```

Note: All command lines you enter that include I/O redirection, such as `2>`, require that you use either the `ksh` or `sh` shell; the `csh` shell will not accept the same I/O redirection command syntax as the `ksh` and `sh` shells.

The csarun error and status messages are placed in the /usr/adm/acct/nite directory. The progress of a run is tracked by writing descriptive messages to the active file. Diagnostic output during the execution of csarun is written to fd2log. The lock and lock1 files prevent concurrent invocations of csarun; csarun aborts if these two files exist when it is invoked. The clastdate file contains the month, day, and time of the last two executions of csarun.

Errors and warning messages from programs called by csarun are written to files that have names that begin with E and end with the current date and time. For example, Ebld.11121400 is an error file from csabuild for a csarun invocation on November 12, at 14:00.

If csarun detects an error, it sends an informational message to the operator by using msgi(1), sends mail to root and adm, removes the locks, saves the diagnostic files, and terminates execution. When csarun detects an error, it will send mail either to MAIL_LIST if it is a fatal error, or to WMAIL_LIST if it is a warning message, as defined in the /etc/config/acct_config configuration file.

10.8 CSA accounting states

During daily execution, csarun writes its starting time into nite/clastdate. The main csarun processing is divided into several separate, restartable states (see the following list). At the conclusion of each state, csarun writes the name of the next state into nite/statefile. The csarun procedure also writes descriptive messages into nite/active and any diagnostic messages into nite/fd2log.

If daily accounting does not complete successfully, check the active, fd2log, and statefile files. You may then restart csarun from the current state, or you may specify the state at which to restart.

Example:

csarun 0415	Restarts accounting for April 15, using the time and state specified in clastdate and statefile.
csarun 0415 0400 CMS	Restarts at the specified time and at the CMS state.

If csarun is run without arguments, the previous invocation must have terminated normally. If not, csarun will abort.

The following is a list of CSA accounting states in the order in which they occur:

State	Description
COMPLETE	Ensures that accounting successfully completed the last time it was run.
SETUP	The current accounting files are switched by using the `turnacct` and `turndacct` files. These files are then moved to the `/usr/adm/acct/work/`*MMDD*`/`*hhmm* directory. File names are prefaced with `W`. The `/etc/wtmp` and `/etc/csainfo` files also are moved to this directory.
WTMPFIX	The `wtmpfix`(8) command checks the `wtmp` file in the work directory for accuracy. Some date changes cause `csaline`(8) to fail; therefore, `wtmpfix` tries to adjust the time stamps in the `wtmp` file if a data change record appears. If `wtmpfix` cannot fix the `wtmp` file, you must repair the `wtmp` file manually, as described in Section 10.9, page 254.
VERIFY	By default, per-process, NQS, and tape accounting files are checked for valid data. Records with data that is not valid are removed. Names of bad data files are prefixed with `BAD.` in the `/usr/adm/acct/work/*` directory.
PREPROC	NQS and connect time (`wtmp`) accounting files are run through preprocessors. File names of preprocessed files are prefixed with a `P` in the `/usr/adm/acct/work/`*MMDD*`/`*hhmm* directory.
ARCHIVE1	First user exit of the `csarun` script. You can use this script to archive the raw and preprocessed accounting files. The shell `.` command executes the `/usr/lib/acct/csa.archive1` script, if it exists. By default, this feature is disabled.
BUILD	The per-process, NQS, tape, and connect accounting data is organized into a session record file.
ARCHIVE2	Second user exit of the `csarun` script. You can use this script to archive the session record file. The shell `.` command executes the `/usr/lib/acct/csa.archive2` script, if it exists. By default, this feature is disabled.
CMS	Produces a command summary file in `cacct.h` format. The `cacct` file is put into the `/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm* directory for use by `csaperiod`.

REPORT	Generates the daily accounting report and puts it into /usr/adm/acct/sum/rpt/*MMDD*/*hhmm*/rprt. A consolidated data file, /usr/adm/acct/sum/data/*MMDD*/*hhmm*/cacct, also is produced from the session record file. Accounting data for unfinished sessions also is recycled.
DREP	Generates a daemon usage report based on the session file. This report is appended to the daily accounting report, /usr/adm/acct/sum/rpt/*MMDD*/*hhmm*/rprt.
FEF	Third user exit of the csarun script. You can use this script to execute a front-end formatter. The shell . command executes the script /usr/lib/acct/csa.fef, if it exists.
USEREXIT	Fourth user exit of the csarun script. You can use this script to execute local accounting programs. The shell . command executes the /usr/lib/acct/csa.user script, if it exists.
CLEANUP	Cleans up temporary files, removes the locks, and then exits.
COMPLETE	Ensures that accounting successfully completed.

10.9 Fixing wtmp errors

The wtmp files generally cause the highest number of errors in the day-to-day operation of the accounting subsystem. When the date is changed, and the UNICOS system is in multiuser mode, a set of date change records is written into the /etc/wtmp file. When a date change is encountered, the wtmpfix (see the fwtmp(8) man page) program adjusts the time stamps in the wtmp records.

Some combinations of date changes and reboots, however, slip by wtmpfix and cause csaline to fail. The following steps show how to repair a wtmp file:

```
$ cd /usr/adm/acct/work/MMDD/hhmm
$ /usr/lib/acct/fwtmp < Wwtmp > xwtmp
$ ed xwtmp


(delete corrupted records)

$ /usr/lib/acct/fwtmp -ic < xwtmp > Wwtmp


(restart csarun at the wtmpfix state)
```

If the wtmp file is beyond repair, create a null Wwtmp file. This prevents any charging of connect time.

10.10 Verifying data files

To verify data files, use the csaedit, csapacct, and csaverify commands. csaedit and csapacct verify and delete bad data records; csaverify only flags bad records. By default, csaedit and csaverify are invoked in csarun to verify the data files.

These commands may allow files that contain bad data, such as very large values, to be verified successfully.

10.11 Editing data files

You can use the csaedit and csapacct commands to verify and remove records from various accounting files. The following example shows how you can use csapacct to verify and remove bad records from a per-process (pacct) accounting file.

In this example, csapacct is invoked with verbose mode enabled (valid data records are written to the pacct.NEW file):

```
$ /usr/lib/acct/csapacct -v pacct pacct.NEW
```

The output produced by this command line is as follows:

```
Bad record - starting byte offset is 077740 (32736)
   invalid pacct record - bad base parent process id 97867
Found the next magic word at byte offset 0100130, ignored 120 bytesFound 394 BASE records
Found 4 EOJ records
Found 1 MTASK (multitasking) records
Found 0 ERROR records
Found 0 IO records
Found 0 SDS records           # not on CRAY-2, J90 systems
Found 0 MPP records           # not on CRAY-2, J90 systems
Found 0 PERFORMANCE records   # not on CRAY-2 systems
Outputted records for 398 processes
Ignored 120 bytes from the input file
```

You can use csaedit and csapacct in conjunction with csaverify , by first running csaverify and noting the byte offsets of the first bad record. Next, execute csaedit or csapacct and remove the record at the specified offset.

The following example shows how you can verify and then edit a bad `pacct` accounting file:

1. Verify the `pacct` file by using the following command line; the following output is received:

```
$ /usr/lib/acct/csaverify -P pacct

/usr/lib/acct/csaverify: pacct: invalid pacct record - bad base parent process id
    97867 byte offset: start = 077740 (32736)  word offset: start = 07774 (4092)
/usr/lib/acct/csaverify: pacct: invalid pacct record - bad magic word 03514000
   byte offset: start = 0100070 (32824)  word offset: start = 010007 (4103)
```

2. Delete the record found at byte offset 32736, as follows (valid records are written to `pacct.NEW`):

```
$ /usr/lib/acct/csapacct -o 32736 pacct pacct.NEW
```

3. Reverify the new `pacct` file to ensure that all the bad records have been deleted, as follows:

```
$ /usr/lib/acct/csaverify -P pacct.NEW
```

You can use `csaedit` to produce an abbreviated ASCII version of some of the daemon accounting files and `acctcom` to generate a similar ASCII version of `pacct` files.

10.12 Data recycling

A system administrator must correctly maintain recycled data to ensure accurate accounting reports. Data recycling allows CSA to bill sessions properly that are active during multiple accounting periods. By default, the `csarun` script reports data only for sessions that terminate during the current accounting period. Through data recycling, CSA preserves data for active sessions until the sessions terminate.

In the `Super-record` file, `csabuild` flags each session as being either active or terminated. `csarecy` reads the `Super-record` file and recycles data for the active sessions. `csacon` consolidates the data for the terminated sessions, which `csaperiod` uses later. `csarun` invokes `csabuild`, `csarecy`, and `csacon`.

The `csarun` command puts recycled data in the `/usr/adm/acct/day` directory. Data files with names suffixed with 0 contain recycled data. For example, `ctime0`, `nqacct0`, `pacct0`, `tpacct0`, `usacct0`, and `uptime0` are generally the recycled data files that are found in `/usr/adm/acct/day`.

Usually, an administrator should not have to purge the recycled accounting data manually. This purge should be necessary only if accounting data is missing. Missing data can cause sessions to recycle forever and consume valuable CPU cycles and disk space.

Recycling unnecessary data can consume a lot of disk space and CPU time. The session file and recycled data can occupy a vast amount of disk space on the file systems that contain `/tmp` and `/usr/adm/acct/day`. Sites that archive data also require additional offline media. `csarun` uses wasted CPU cycles to reexamine and recycle the data. Therefore, to conserve disk space and CPU cycles, you should purge unnecessary recycled data from the accounting system.

For detailed information about data recycling, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

Procedure 33: Setting up CSA

This procedure shows you how to ensure that accounting is started and terminated properly at system boot and shutdown time. The procedure also shows you how to configure accounting parameters, gather disk usage information, and schedule daily and periodic accounting runs automatically.

Note: Before you begin this procedure, determine your company's system billing units (SBUs). For detailed information about making site-specific modifications, see the "Tailoring CSA" section of the "Accounting" chapter in *UNICOS Resource Administration*, Cray Research publication SG–2302.

1. Modify configurable accounting parameters manually or by using the menu system. Following are some of the types of changes you might want to make; parameters that pertain to USCP, MPP, and SDS accounting do not apply to CRAY J90:

- Setting up SBUs for per-process accounting (`pacct`) data; by default, no SBUs are calculated. You can set several variables (for example, weighting factors are set through over a dozen variables, including `P_BASIC`, `P_TIME`, and `P_STIME`). (For information, see *UNICOS Resource Administration*, Cray Research publication SG–2302.)
- Charging for NQS jobs (`NQS_TERM_`*xxx* variables); also see step 4.

- Modifying the file system on which `/usr/adm/acct` resides; the default is `/usr` (`ACCT_FS` variable).
- Working with an alternative accounting configuration file (the `ACCTCONFIG` variable).
- Compiling a list of users to whom mail is sent when a warning or error is detected. The default is `root` (`/`) and `adm` (`WMAIL_LIST` and `MAIL_LIST` variables).
- Enabling NQS and tape daemon accounting at system startup (`NQS_START` and `TAPE_START` variables).
- Changing the minimum number of free blocks (the `MIN_BLKS` variable) needed in `ACCT_FS` to enable accounting or to run `csarun` or `csaperiod`. The default is 500 free blocks.

  If you are using the menu system, select the `Configure System->Accounting Configuration` menu. Enter your changes, and then activate the new configuration. A sample menu screen follows:

```
Configure System
    ->Accounting Configuration
```

```
                        Accounting Configuration


S-> Edit accounting configuration ...
    Import accounting configuration ...
    Activate accounting configuration ...


    Keys:   ^? Commands   H Help   Q Quit   V ViewDoc   W WhereAmI
```

  If you are not using the menu system, edit the `/etc/config/acct_config` file.

2. Accounting is started by default each time `/etc/rc` is invoked. Make sure that `csaboots` is invoked from `rc`, and not from `/etc/inittab` or `/etc/brc`. If necessary, add the following lines to the `/etc/rc` script in the Accounting script section, just before the `/usr/lib/acct/startup` script section:

```
# Accounting.
#
if [ -x /etc/csaboots ]; then
        echolog "Writing accounting boot time."
        x /etc/csaboots -v >> $RC_LOG
fi
```

3. Ensure that the following lines are included in `/etc/shutdown` to turn off per-process accounting and daemon accounting before the system is brought down:

```
if [ -x /usr/lib/acct/shutacct ]
then
        /usr/lib/acct/shutacct
        echo "Process accounting stopped."
fi
```

4. (Optional) Enable daemon accounting at system start-up time, as follows:

   a. Ensure that the variables for the subsystems for which you want to enable daemon accounting are set to `on` in `/etc/config/acct_config` by editing the `/etc/config/acct_config` file or by using the `NQS Configuration` menu. Set the `NQS_START` and `TAPE_START` parameters to `on` to enable NQS and online tapes, respectively.

   b. If necessary, enable accounting from the daemon's side (required for NQS and tape).

      To turn on NQS accounting, do **one** of the following actions:

      - Insert the line `set accounting on` in the `/etc/config/nqs_config` and `/etc/config/NQS.startup` file (recommended).

        or

      - Turn on NQS accounting by using the `qgmr set accounting on` command.

      To turn on tape accounting, execute the `/usr/lib/tp/tpdaemon` by using the `-c` command-line option from `/etc/config/daemons` or from the `System Daemons Table` menu.

5. (Optional) If you plan to gather disk usage statistics, create or modify the `/etc/checklist` file. This file contains a list of file systems (full path

names) for which `dodisk` will collect information. One special file name is listed on each line. A sample `/etc/checklist` file follows:

```
# more /etc/checklist
       /dev/dsk/home
       /dev/dsk/tmp
       /dev/dsk/filesystemA
       /dev/dsk/filesystemB
```

Generally, `root` executes `dodisk` through `cron` (see the next step). `csarun` incorporates the disk usage data into the daily accounting report.

6. Ensure that entries similar to the following are included in `/usr/spool/cron/crontabs/root` so that `cron` automatically runs daily accounting. The `ckdacct` and `ckpacct` scripts check and limit daemon and standard accounting files sizes.

Caution: The `dodisk` script **must** run at least 1 hour before the `csarun` script, so that the `dodisk` script has time to complete before `csarun` tries to access that data. You also **must** invoke `dodisk` with either the `-a` or `-A` option; if you do not, `csaperiod` aborts when it tries to merge the disk usage information with other accounting data.

```
0 23 * * 0-6 /usr/lib/acct/dodisk -a -v 2> /usr/adm/acct/nite/dk2log
0 0 * * 0-6 /usr/lib/acct/csarun 2> /usr/adm/acct/nite/fd2log
0 * * * * /usr/lib/acct/ckdacct nqs tape
0 * * * * /usr/lib/acct/ckpacct
```

If you want to run periodic accounting, ensure that an entry similar to the following is included in `/usr/spool/cron/crontabs/root`; this command generates a periodic report on all consolidated data files found in `/usr/adm/acct/sum/data/*` and then deletes those data files:

```
15 5 1 * * /usr/lib/acct/csaperiod -r 2> /usr/adm/acct/nite/pd2log
```

7. Ensure that the following lines are included in `/usr/lib/acct/csarun` if you want the following information:

   a. To get trace information, ensure that the following line is the first line after the first set of comment lines in the file:

      ```
      set-xS
      ```

   b. To get the SBU report , add the `b` option to the `csacrep` line, as follows:

```
csacrep -hucwb < ${CDATA}/cacct > ${CRPT}/conrpt 2> ${NITE}/Ecrpt.${DTIME}
```

8. Update the `/usr/lib/acct/holidays` file, which should reflect your site's prime/nonprime time and holiday schedules. The year field must contain either the current year or the wildcard character symbol, *, which specifies that the current year should be used. Following is a sample `holidays` file:

```
# USMID @(#)acct/src/acct/holidays
#
#       (c) Copyright Cray Research, Inc.
#       Unpublished Proprietary Information.
#       All Rights Reserved.
#
# Prime/Nonprime Table for UNICOS Accounting System
#
# Curr  Prime   Non-Prime
# Year  Start   Start
#
  199x   0730    1730

(Accounting software references this line to confirm current year)

#
# Day of        Calendar        Company
# Year          Date            Holiday
#
    1           Jan 1           New Year's Day
  146           May 25          Memorial Day
  184           Jul 2           Independence Day Thursday
  185           Jul 3           Independence Day Friday
  186           Jul 4           Independence Day
  251           Sep 7           Labor Day
  331           Nov 26          Thanksgiving Day
  332           Nov 27          Thanksgiving Friday
  359           Dec 24          Christmas Eve
  360           Dec 25          Christmas Day
  366           Dec 31          New Year's Eve day
```

9. Label file systems with accounting types while they are mounted by using the `devacct`(8) command. If a file system does not contain a device type label, device accounting ignores it.

10.13 Daily CSA reports

The `csarun` script generates various daily reports, all of which are placed in a file named `/usr/adm/acct/sum/rpt/`*MMDD*`/`*hhmm*`/rprt` (for example, `0415/2000/rprt`). By default, the report includes statistics only for sessions that have terminated. The reports include the following:

- Interactive connect time by ttyp.
- CPU usage by user ID and account ID. For a description of the fields in this report, see the "Accounting" chapter of *UNICOS Resource Administration*, Cray Research publication SG–2302.
- A listing of active interactive and batch jobs by job ID.
- Disk usage by user ID and account ID.
- Command summary data by total CPU time used. For a description of the fields in this report, see the "Accounting" chapter of *UNICOS Resource Administration*, Cray Research publication SG–2302.
- Last interactive login information by date.
- Job mix (interactive versus NQS), tape, and NQS usage.

Many of the periodic reports that `csaperiod` generates are similar to the preceding reports; however, not all of the reports listed have a periodic equivalent. Periodic reports are located in `/usr/adm/acct/fiscal/rpt/`*MMDD*`/`*hhmm*`/rprt`.

A sample `/etc/config/acct_config` file follows.

```
#
#   SN5228 - acct_config - Edition 40  [Mon Jan  3 14:34:41 CST 1994]
#   Created by Configuration Generator Rev. 80.60
#
#
#   SN5228 - acct_config - Edition 22  [Mon Nov 15 13:15:11 CST 1993]
#   Created by Configuration Generator Rev. 80.60
#
#
#   SN5147 - acct_config - Edition 8  [Mon Jun  7 14:58:30 CDT 1993]
#   Created by Configuration Generator Rev. 80.60
#
# USMID @(#)skl/etc/config/acct_config   70.8    04/20/93 17:17:19
#
#    (C) COPYRIGHT CRAY RESEARCH, INC.
```

```
#    UNPUBLISHED PROPRIETARY INFORMATION.
#    ALL RIGHTS RESERVED.
#
#
#    This file contains the parameter labels and values used by the
#    accounting software.
#
################################################################################
#
#    Connect time SBUs.
#  The following section contains the labels and values which pertain to
#  connect time accounting.
#
################################################################################
#
# The CON_PRIME parameter defines the weighting factor for prime time
# connect time. This is in billing units per second.
CON_PRIME    0.0
# The CON_NONPRIME parameter defines the weighting factor for nonprime
# time connect time.  This is in billing units per second.
CON_NONPRIME    0.0
################################################################################
#
#    Device accounting SBUs (available only on non-Cray2 systems).
#  The following section contains the labels and values which pertain to
#  device accounting.
#
#
################################################################################
#
# The System Administrator's Guide (SG-2113) describes device accounting
# configuration in detail.
#
# For each device type to be billed, 3 fields must be filled out.
#   1) Logical I/O sbu  -  unit per logical I/O request.
#   2) Characters xfer sbu  -  unit per character transferred.
#   3) Device name  -  Device type name. This field must be surrounded by
#                      double quotes if the name contains embedded spaces.
#    Block devices: The name should match the type assigned to
#      the block device. For example, if the device to be
#      mounted on /tmp is a dd49 with logical device cache,
#      then this device should be labeled as "dd49 with ldcache"
#      or numerically 7.
```

```
#    Character devices: The name should match the major device numbers
#      in /usr/src/uts/c1/cf/devsw.c.
#
# Sites may add new types as long as the number assigned to that type does
# not exceed MAXBDEVNO (for block devices) or MAXCDEVNO (for character
# devices). A site may also change the meaning of the default device types.
#
# MAXBDEVNO and MAXCDEVNO may be increased (see /usr/include/sys/param.h),
# but this will increase the size of the largest possible accounting record.
# This in turn may necessitate more disk space for the pacct files and/or
# a larger kernel stack.
#
#
#    Block device SBUs.
# The numeric suffixes for the "BLOCK_DEVICE" labels must be ascending from
# 0 to MAXBDEVNO - 1. MAXBDEVNO is currently 10.
#
#             Logical       Characters           Device
#   label     I/O Sbu        Xfer Sbu        Name
BLOCK_DEVICE0    0.0        0.0        "dd29"
BLOCK_DEVICE1    0.0        0.0        "dd39"
BLOCK_DEVICE2    0.0        0.0        "dd40"
BLOCK_DEVICE3    0.0        0.0        "dd49"
BLOCK_DEVICE4    0.0        0.0        "dd29 with ldcache"
BLOCK_DEVICE5    0.0        0.0        "dd39 with ldcache"
BLOCK_DEVICE6    0.0        0.0        "dd40 with ldcache"
BLOCK_DEVICE7    0.0        0.0        "dd49 with ldcache"
BLOCK_DEVICE8    0.0        0.0        "ssd"
BLOCK_DEVICE9    0.0        0.0        "bmr"
#
#    Character device SBUs.
# The numeric suffixes for the "CHAR_DEVICE" labels must be ascending from
# 0 to MAXCDEVNO - 1. MAXCDEVNO is currently 35. The suffixes must match
# the minor numbers in /dev. These minor numbers are defined in
# /usr/src/uts/c1/cf/devsw.c in the cdevsw[] array.
#
#             Logical       Characters           Device
#   label     I/O Sbu        Xfer Sbu        Name
CHAR_DEVICE0    0.0        0.0        "hpm"
CHAR_DEVICE1    0.0        0.0        "ios-tty"
CHAR_DEVICE2    0.0        0.0        "systty"
CHAR_DEVICE3    0.0        0.0        "memory"
CHAR_DEVICE4    0.0        0.0        "hyperchannel"
```

```
CHAR_DEVICE5    0.0        0.0        "expander tape"
CHAR_DEVICE6    0.0        0.0        "expander printer"
CHAR_DEVICE7    0.0        0.0        "hsx"
CHAR_DEVICE8    0.0        0.0        "error-device"
CHAR_DEVICE9    0.0        0.0        "expander disk"
CHAR_DEVICE10    0.0        0.0        "low speed channel"
CHAR_DEVICE11    0.0        0.0        "bmx tape"
CHAR_DEVICE12    0.0        0.0        "pty-master"
CHAR_DEVICE13    0.0        0.0        "pty"
CHAR_DEVICE14    0.0        0.0        "?"
CHAR_DEVICE15    0.0        0.0        "bmx daemon"
CHAR_DEVICE16    0.0        0.0        "net"
CHAR_DEVICE17    0.0        0.0        "net"
CHAR_DEVICE18    0.0        0.0        "disk control"
CHAR_DEVICE19    0.0        0.0        "secded"
CHAR_DEVICE20    0.0        0.0        "security log"
CHAR_DEVICE21    0.0        0.0        "cpu control"
CHAR_DEVICE22    0.0        0.0        "logger"
CHAR_DEVICE23    0.0        0.0        "disk maintenance"
CHAR_DEVICE24    0.0        0.0        "data migration"
CHAR_DEVICE25    0.0        0.0        "?"
CHAR_DEVICE26    0.0        0.0        "?"
CHAR_DEVICE27    0.0        0.0        "?"
CHAR_DEVICE28    0.0        0.0        "?"
CHAR_DEVICE29    0.0        0.0        "?"
CHAR_DEVICE30    0.0        0.0        "?"
CHAR_DEVICE31    0.0        0.0        "?"
CHAR_DEVICE32    0.0        0.0        "?"
CHAR_DEVICE33    0.0        0.0        "?"
CHAR_DEVICE34    0.0        0.0        "?"
##############################################################################
#
#    Multitasking CPU time SBUs.
#  The following section contains the labels and values which pertain to
#  multitasking.
##############################################################################
#
# The MUTIME_WEIGHT variables define the weighting factors that
# are used to bill user CPU time for multitasking programs. It
# is used in conjunction with the ac_mutime array (see
# /usr/include/sys/acct.h), which defines the amount of user
# CPU time the multitasking program spent with i + 1 CPUs connected.
#
```

```
# MUTIME_WEIGHTi defines the marginal cost for getting the i-th + 1
# CPU at one instant. If the MUTIME_WEIGHT values are set to less
# than 1.0, there will be an incentive for multitasking. If the
# values are set to 1.0, multitasking programs will be charged for
# user CPU time just as all other programs.
#
# There must be an MUTIME_WEIGHT variable for each of the cpus
# available on the machine.
#
MUTIME_WEIGHT0        1.0        # 1 CPU
MUTIME_WEIGHT1        1.0        # 2 CPU
MUTIME_WEIGHT2        1.0        # 3 CPU
MUTIME_WEIGHT3        1.0        # 4 CPU
MUTIME_WEIGHT4        1.0        # 5 CPU
MUTIME_WEIGHT5        1.0        # 6 CPU
MUTIME_WEIGHT6        1.0        # 7 CPU
MUTIME_WEIGHT7        1.0        # 8 CPU
##############################################################################
#
#    NQS SBUs.
#  The following section contains the labels and values which pertain to
#  NQS accounting.
#
##############################################################################
#
#    Set the values to 1 if jobs, or portion of jobs, which
#    terminate with the specified termination code are to be billed.
#    Otherwise, set the value to 0. By default, all portions of a
#    request will have sbus calculated for them.
#
NQS_TERM_EXIT        1        # Request exited
NQS_TERM_REQUEUE    1        # Request requeued for a restart
NQS_TERM_PREEMPT    1        # Request preempted
NQS_TERM_HOLD        1        # Request held
NQS_TERM_OPRERUN    1        # Request rerun by operator
NQS_TERM_RERUN        1        # Request non-operator rerun




#
```

```
#    Set NQS_NUM_QUEUES to be the number of queues for which you want
#    to set sbus.
#
NQS_NUM_QUEUES        3
#
#    Set the sbus associated with each queues. There must be
#    NQS_NUM_QUEUES sbu/queue pairs. The labels' numeric suffixes
#    must be ascending from 0 to NQS_NUM_QUEUES. Thus, if
#    NQS_NUM_QUEUES is 0, no NQS_QUEUEx values need be defined.
#
#    If an sbu value is set to less than 1.0, there is an incentive
#    to run jobs in this queue. If the value is set to 1.0, the
#    jobs will be charged as though it were a normal, non-NQS job.
#    If the sbu is set to 0.0, there is no charge for jobs running
#    in this queue. For queues not listed below, the sbu is set
#    to 1.0.
#
#  label    sbu    queue_name
NQS_QUEUE0    1.0    b_30_5
NQS_QUEUE1    1.0    b_600_1
NQS_QUEUE2    1.0    b_1200_1
#
#    Set NQS_NUM_MACHINES to the number of originating machines for
#    which you want to set sbus.
#
NQS_NUM_MACHINES        2
#
#    Set the sbus associated with each originating machine. There must
#    be NQS_NUM_MACHINES sbu/machine pairs. The sbu values are set
#    in the same manner as those for the queues. Once again, the
#    numeric label suffixes must be ascending from 0 to NQS_NUM_MACHINES.
#    Thus, if NQS_NUM_MACHINES is 0, no NQS_MACHINEx values need be
#    defined.
#
#  label    sbu    machine_name
NQS_MACHINE0    1.0    sn1405
NQS_MACHINE1    1.0    sn2024
##############################################################################
#
#    Pacct SBUs.
#  The following section contains the labels and values which pertain to
#  pacct (kernel) accounting.
#
```

```
#############################################################################
#
#    Set the prime time weighting factors.
#    On non-Cray2 systems:
#       If P_STIME is nonzero, then P_SCTIME and P_INTTIME must be zero.
#       If P_SCTIME and P_INTTIME are nonzero, then P_STIME must be zero.
#       This is so there won't be multiple billing of system cpu time.
#
P_BASIC        0.0    # Basic prime time weighting factor
P_TIME        0.0    # General time weighting factor
P_STIME        0.0    #   System CPU time weighting factor (unit/sec)
P_UTIME        0.0    #   User CPU time weighting factor (unit/sec)
P_ITIME        0.0    #   I/O wait time weighting factor (unit/sec)
#
#    P_SCTIME and P_INTTIME are used only on non-Cray2 systems.
#
P_SCTIME    0.0    #   System call weighting factor (unit/sec)
P_INTTIME    0.0    #   Interrupt time weighting factor (unit/sec)
P_MEM        0.0    # General memory weighting factor
P_XMEM        0.0    #   CPU time memory weighting factor (unit/Kw-min)
P_IMEM        0.0    #   I/O wait time memory weighting factor (unit/Kw-min)
P_IO        0.0    # General I/O weighting factor
P_BYTEIO    0.0    #   I/O char xfer weighting factor (unit/char xferred)
P_PHYIO        0.0    #   Physical i/o req weighting factor (unit/phy i/o req)
P_LOGIO        0.0    #   Logical i/o req weighting factor (unit/log i/o req)
#
#    The following 3 SDS weighting factors are used only on non-Cray2
#    machines.
#
P_SDSMEM    0.0    # SDS memory integral weighting factor (unit/Mw-sec)
P_SDSLOGIO    0.0    #   SDS logical i/o req weighting factor (unit/log req)
P_SDSBYTEIO    0.0    #   SDS char xferred (unit/char transferred)
#
#    Set the non-prime time weighting factors.
#    On non-Cray2 systems:
#       If NP_STIME is nonzero, then NP_SCTIME and NP_INTTIME must be zero.
#       If NP_SCTIME and NP_INTTIME are nonzero, then NP_STIME must be zero.
#       This is so there won't be multiple billing of system cpu time.
#
NP_BASIC    0.0    # Basic non-prime time weighting factor
NP_TIME        0.0    # General time weighting factor
NP_STIME    0.0    #   System CPU time weighting factor (unit/sec)
NP_UTIME    0.0    #   User CPU time weighting factor (unit/sec)
```

```
NP_ITIME    0.0    #   I/O wait time weighting factor (unit/sec)
#
#    NP_SCTIME and NP_INTTIME are used only on non-Cray2 systems.
#
NP_SCTIME    0.0    #   System call weighting factor (unit/sec)
NP_INTTIME    0.0    #   Interrupt time weighting factor (unit/sec)
NP_MEM        0.0    # General memory weighting factor
NP_XMEM        0.0    #   CPU time memory weighting factor (unit/Kw-min)
NP_IMEM        0.0    #   I/O wait time memory weighting factor (unit/Kw-min)
NP_IO        0.0    # General I/O weighting factor
NP_BYTEIO    0.0    #   I/O char xfer weighting factor (unit/char xferred)
NP_PHYIO    0.0    #   Physical i/o req weighting factor (unit/phy i/o req)
NP_LOGIO    0.0    #   Logical i/o req weighting factor (unit/log i/o req)
##############################################################################
#
#    Tape SBUs.
#  The following section contains the labels and values which pertain to
#  tape accounting.
#
##############################################################################
#
#    The following section sets the sbu values for each of the
#    TP_MAXDEVGRPS tape device groups. TP_MAXDEVGRPS is defined
#    in /usr/include/acct/dacct.h. At this time, only 2 device
#    groups are used: TAPE and CART. However, there must
#    be TP_MAXDEVGRPS "TAPE_SBU" variables defined. The TAPE_SBU
#    numeric suffix must be ascending from 0 to TP_MAXDEVGRPS - 1.
#
#    The fields are:
#      Device_group    Device group name
#      Mount        Billing unit per mount
#      Reserve    Billing unit per reserve second
#      Read        Billing unit per byte read
#      Write        Billing unit per byte written
#
#    Note:  On Cray2 systems, TAPE_SBU0 is always for tape devices,
#      and TAPE_SBU1 is always for cart devices.
#
#
#        Device
#        Group    Mount        Reserve        Read        Write
TAPE_SBU0    TAPE    0.0        0.0        0.0        0.0
TAPE_SBU1    CART    0.0        0.0        0.0        0.0
```

```
TAPE_SBU2    SILO    0.0        0.0        0.0        0.0
TAPE_SBU3    UNUSED    0.0        0.0        0.0        0.0
TAPE_SBU4    UNUSED  0.0             0.0             0.0            0.0
TAPE_SBU5    UNUSED  0.0             0.0             0.0            0.0
TAPE_SBU6    UNUSED  0.0             0.0             0.0            0.0
TAPE_SBU7    UNUSED  0.0             0.0             0.0            0.0
##############################################################################
#
#    USCP SBUs.
#  The following section contains the labels and values which pertain to
#  USCP accounting.
#
##############################################################################
#
#
#    The USCP_MAXMF parameter defines the number of mainframes
#    for which sbus are to be set. This value must be at least 1.
#
USCP_MAXMF    2
```

(Set this to 0 for CRAY J90 systems.)

```
#
#    The following parameters set the sbu values for each of the
#    USCP_MAXMF mainframes. Sbus must be set for each of the
#    US_MAXTTYPE transfer types. US_MAXTTYPE is defined in
#    /usr/include/acct/dacct.h.
#
#    Mainframes not listed below are given sbu values of 0.0.
#
#      USCP_MAINFRAME sets the 2 character mainframe identifier.
#
#      The following parameters set the runtime and sectors transferred
#      sbus for each of the various transfer types. Runtime sbus
#      are in units per second. Sectors transferred (xfer) sbus
#      are in units per sectors transferred.
#
#      USCP_INTER sets the sbus for interactive disposes and fetches
#        (obsolete).
#      USCP_DISPOSE sets the sbus for disposes.
#      USCP_FETCH sets the sbus for fetches.
#      USCP_GET sets the sbus for gets.
#      USCP_PUT sets the sbus for puts.
#      USCP_SAVE sets the sbus for saves.
```

```
#
USCP_MAINFRAME0        SB
#            runtime        xfer
USCP_INTER0        0.0        0.0
USCP_DISPOSE0        0.0        0.0
USCP_FETCH0        0.0        0.0
USCP_GET0        0.0        0.0
USCP_PUT0        0.0        0.0
USCP_SAVE0        0.0        0.0
USCP_MAINFRAME1        YJ
#            runtime        xfer
USCP_INTER1        0.0        0.0
USCP_DISPOSE1        0.0        0.0
USCP_FETCH1        0.0        0.0
USCP_GET1        0.0        0.0
USCP_PUT1        0.0        0.0
USCP_SAVE1        0.0        0.0
################################################################################
#
#    Miscellaneous parameters which are sometimes reset.
# The following section contains miscellaneous parameters that can be
# reset by the site.
#
################################################################################
#
# The ACCT_FS parameter defines the file system on which /usr/adm/acct
# resides. It is used when checking the amount of free space on /usr/adm/acct.
#
ACCT_FS        /usr
#
# The HOLIDAY_FILE parameter defines the location of the holidays file.
# This parameter should be an absolute pathname.
#
HOLIDAY_FILE    /usr/lib/acct/holidays
#
# The MAIL_LIST parameter is a list of users to whom mail is sent
# if errors are detected in the various shell scripts.
#
MAIL_LIST    "root adm"
#
# The MEMINT parameter is used to select the memory integral.
#
MEMINT        2
```

```
#
# The MIN_BLKS parameter sets the minimum number of free blocks on the
# ACCT_FS filesystem that need to be available. If less than MIN_BLKS
# free blocks is available, accounting is disabled, or processing via
# runacct or csarun is halted.
#
MIN_BLKS    500
#
# The NQS_START parameter enables or disables NQS accounting when
# /usr/lib/acct/startup is executed. Valid values are "on" and "off".
# If NQS accounting is enabled here, it must also be enabled by NQS
# via the qmgr(8) "set accounting on" command.
#
NQS_START    on
#
# The NUM_HOLIDAYS parameter sets the upper limit on the number of
# holidays that can be defined in HOLIDAY_FILE.
#
NUM_HOLIDAYS    20
#
# The PERF_NAME0 parameter sets the type which is to be specified with
# devacct(1M) when enabling and disabling performance accounting.
#
PERF_NAME0    perf_01
#
# The TAPE_START parameter enables or disables tape accounting when
# /usr/lib/acct/startup is executed. Valid values are "on" and "off".
# If tape accounting is enabled here, the "-c" option must be used
# when starting the tape daemon, tpdaemon(8).
#
TAPE_START    on
#
# The USCP_START parameter enables or disables uscp accounting when
# /usr/lib/acct/startup is executed. Valid values are "on" and "off".
# Uscp accounting does not need to be enabled by the uscp daemon.
#
USCP_START    on
```

(Set this to off for CRAY J90 systems.)

```
##############################################################################
#
#    Miscellaneous parameters generally not reset.
# The following section contains miscellaneous parameters that are not
```

```
# generally changed by a site. Care must be used when some of these are
# modified.
#
##############################################################################
#
# The A_SSIZE parameter is the maximum number of sessions in 1
# accounting run that can be processed by acctprc1(1M).
#
A_SSIZE        10000
#
# The A_TSIZE parameter is the maximum number of tty line names
# in 1 accounting run that can be processed by acctcon1(1M) and
# csaline(1M).
#
A_TSIZE        1000
#
# The A_USIZE parameter is the maximum number of distinct login
# names in 1 accounting run that can be processed by acctprc1(1M)
# and acctprc2(1M).
#
A_USIZE        5000
#
# The ACCTOFF string is written to /etc/wtmp when accounting is
# turned off by shutacct(1M). This string should be a maximum
# of 11 characters.
#
ACCTOFF        acctg off
#
# The ACCTON string is written to /etc/wtmp when accounting is
# turned on by startup(1M). This string should be a maximum of
# 11 characters.
#
ACCTON        acctg on
#
# The BUILD_MAXFILES parameter sets the upper limit on the number
# of files which can be processed by csabuild(1M).
#
BUILD_MAXFILES    200
#
# The MAX_CPUS parameter sets the upper limit on the number of
# cpus a machine can have. This value must be at least as large
# as the number of cpus on your machine.
#
```

```
MAX_CPUS    32
#
# The MAXICYLS parameter sets the upper limit on the number disk
# cylinders involved in the inode region that must be read by the
# Cray2 version of diskusg(1M). This is an expected worse case
# based on 8000 inode sectors / 16 regions = 512 sectors with the
# dd49's being the smallest at 360 sector/cylinder, requiring 2
# reads per area for an average distribution.
#
MAXICYLS    32
#
# The MAXILIST parameter sets the maximum number of ilist expected
# in a filesystem. This parameter is used only by the Cray2
# version of diskusg(1M).
#
MAXILIST    200
#
# The MAXUSERS_DISK parameter sets the upper limit on the number
# of user id/account id pairs that can be handled by the Cray2
# version of diskusg(1M).
#
MAXUSERS_DISK    5000
# The NCLUSTER parameter sets the upper limit on the number of
# partitions in a filesystem. This parameter is used only by the
# Cray2 version of diskusg(1M).
#
NCLUSTER    100
# The NSYS parameter sets the upper limit on the number of different
# reasons a wtmp record can be written. Generally, these records are
# written by acctwtmp(1M) when accounting is turned on or off.
#
NSYS            20
##############################################################################
#
#    User defined labels.
# The following section contains user defined labels and values.
# These labels are used in the site tailored sbu routines. The
# format of the following lines must be:
#    label        value
#
##############################################################################
```