Log Files [9]

This chapter describes several log files that are important for you to monitor. Information found in these files can help you determine appropriate actions. You can access the logs described in this chapter through normal file manipulation commands such as `tail`(1), `cat`(1), `pg`(1), and `more`(1).

This chapter describes the following:

- The `/etc/boot.log` file
- The `/etc/rc.log` file
- The `/etc/syslog.conf` file and the `syslog` daemon, `/etc/syslogd`, which works with the `/etc/syslog.conf` file to record entries into the following system log files:
  - `/usr/adm/sulog`
  - `/etc/dump.log`
  - `/usr/adm/nu.log`
  - `/usr/adm/sa/sa`*DD*
  - `/usr/adm/sl/slogfile`
  - `/usr/spool/msg/msglog.log`
  - `/usr/lib/cron/cronlog`
  - `/usr/tmp/nqs.log`
  - `/usr/adm/errfile`
  - `/usr/spool/dm/*`
- Cleaning up system logs

For information about accounting logs and reports, see Chapter 10, page 241.

9.1 Related log files documentation

The following documentation contains information that you will find useful in understanding the material presented in this section:

- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `brc`(8), `cron`(8), `newsys`(8), `reduce`(8), `sar`(8), and `syslogd`(8) man pages
- *CRAY IOS-V Commands Reference Manual*, Cray Research publication SR–2170
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `at`(1), `batch`(1), `cat`(1), `crontab`(1), `date`(1), `logger`(1), `more`(1), `sar`(1), `tail`(1), and `uname`(1) man pages

9.2 `/etc/boot.log` file

The `/etc/boot.log` file records boot dates and times for a system. When the `/etc/rc` script is executed, it appends a record to the `/etc/boot.log` file. The file is composed of output from the `/bin/date` and `uname -a` commands. The format of the `/etc/boot.log` file includes the system name, node, release, version, and hardware information. To determine the last time a system was booted, see this log. The format is as follows:

`date, uname -a` *yy/mm/dd hh:mm system node release version hardware*

Example:

```
# cat /etc/boot.log
93/09/10 08:07 sn1703c cool 8.0.2 CRAY Y-MP
```

For further information, see the `date`(1) and `uname`(1) man pages.

9.3 `/etc/rc.log` file

The `/etc/rc.log` file records the events that occurred the last time the `/etc/rc` (multiuser startup) script was run.

9.4 `/etc/syslog.conf` file

The `syslog` configuration file, `/etc/syslog.conf`, defines the messages that are processed and where they are recorded. An example of the `/etc/syslog.conf` file follows (for a description of the fields, see the `syslogd`(8) and `syslog`(3) man pages):

```
# (Messages processed)               (Stored location)
#
*.debug                                /usr/adm/syslog/debug
#
mail.debug                             /usr/spool/mqueue/syslog
#
kern.debug                             /usr/adm/syslog/kern
#
daemon,auth.debug                      /usr/adm/syslog/auth
#
*.err;kern.debug;daemon,auth.notice;   /usr/adm/syslog/daylog
#
*.alert;kern.err;daemon.err            operator
#
*.alert                                root
```

9.5 System logs

The syslog daemon , /etc/syslogd, provides the UNICOS system with the ability to route messages to regular disk files or to forward them to mail accounts. The /etc/syslogd daemon reads and logs messages into a set of files specified by the administrator in the /etc/syslog.conf configuration file. /etc/syslogd configures itself at start-up time and when it encounters a hang-up signal. The /etc/newsys shell script starts it.

The /usr/ucb/logger command places entries into the system log. For example, if you restart a daemon in the middle of the day, you can log this event by using the following command:

```
# /usr/ucb/logger -p user.info restarted development copy of syslog daemon
```

This section includes information about the following topics:

- Message sources
- Priority levels
- syslog daemon startup
- System log files

9.5.1 Message sources

Messages may be given to the `syslog` daemon, `/etc/syslogd`, from the following sources or facilities:

Source/ Facility	Description
`auth`	Messages that the authorization system (that is, `login`, `su`, or `getty`) generates.
`daemon`	Messages that system daemons (such as `telnetd`, `ftpd`, and `errdaemon`) generate.
`kern`	Messages that the kernel generates. The daemon reads kernel messages from the `/dev/klog` device.
`local0`	Reserved for local use (`local0` through `local7` are available).
`mail`	Messages that the mail system generates.
`mark`	Informational-level messages are sent; default interval is every 20 minutes (set by the `syslogd -m` command).
`user`	Messages that user processes generate. Users write messages (see the `logger`(1) man page) to the named pipe `/dev/log`.

9.5.2 Priority levels

The following eight priority levels (in order of highest to lowest priority) are defined for messages that the system log daemon handles:

Priority	Description
`emerg`	Panic condition, which is usually broadcast to all users
`alert`	Condition that you should correct immediately
`crit`	Critical condition
`err`	Errors
`warning`	Warning message
`notice`	Condition that is not an error condition, but possibly should be specially handled

info	Informational message
debug	Message useful only when debugging a program

9.5.3 syslog daemon startup

The /etc/newsys shell script starts /etc/syslogd and renames any existing log files. As released, the /etc/newsys shell script saves the 10 most recent copies of the log files and deletes the oldest. To preserve more or fewer log files, adjust this limit by editing the /etc/newsys shell script. Two shell functions, quantity() and time_based(), control the preservation of old log files, which are saved under the /usr/adm/syslog/oldlogs directory. A description of the quantity() and time_based() shell functions follows, followed by examples of their use and examples of the /usr/adm/syslog and the /usr/adm/syslog/oldlogs files.

Function	Description
quantity()	Preserves the specified quantity of the specified log files. quantity() is called with at least two arguments. The first is the number of copies to keep. The remaining arguments are the names of the files to be preserved. It will retain *x* copies of each file and delete the oldest.
time_based()	Preserves old log files on the basis of time, rather than system restarts. time_based() is passed at least two arguments. The first is the number of days to preserve files. The remaining arguments are the names of the actual files.

Note: If the base name of the log file consists of more than 6 characters, time_based() will not work. The pattern match in find will fail.

Examples:

```
#
#  Save 20 copies of daylog and debug
#
quantity 20 daylog debug
#
#  Save the last 30 days worth of kern and auth
#
time_based 30 kern auth
```

Examples of system log files follow:

```
# cd /usr/adm/syslog
# ls -lF
total 44

-rw-r--r--  1 root           0 Nov  9 11:03 auth
-rw-r--r--  1 root       15465 Nov  9 15:52 daylog
-rw-r--r--  1 root       15465 Nov  9 15:52 kern
drwxr-xr-x  2 root       11232 Nov  9 11:03 oldlogs/
```

```
# /usr/adm/syslog 5=> tail kern

Nov 9 15:42:39  unicos: NFS server sn218 not responding, giving up
Nov 9 15:42:39  unicos: NFS fsstat failed for server sn218: TIMED OUT
Nov 9 15:42:40  unicos: NFS server sn218 not responding, giving up
Nov 9 15:42:40  unicos: NFS getattr failed for server sn218: TIMED OUT
```

```
# /usr/adm/syslog 6=> cd oldlogs
# /usr/adm/syslog/oldlogs 7=> ls -CF

10-09.5.kern  10-17.6.kern    10-22.3.kern    10-29.1.kern
11-04.1.kern
10-10.0.auth  10-17.7.auth    10-23.0.auth    10-29.2.auth
11-04.2.auth
10-10.0.kern  10-17.7.kern    10-23.0.kern    10-29.2.kern
11-04.2.kern
10-11.0.auth  10-17.8.auth    10-23.1.auth    10-29.3.auth
11-04.3.auth
10-11.0.kern  10-17.8.kern    10-23.1.kern    10-29.3.kern
11-04.3.kern
10-11.1.auth  10-17.9.auth    10-23.2.auth    10-30.0.auth
11-05.0.auth...
10-16.0.auth   10-21.0.auth   10-27.0.auth
11-02.6.auth...
10-16.0.auth   10-21.0.auth   10-27.0.auth    11-02.6.auth    daylog.0
10-16.0.kern   10-21.0.kern   10-27.0.kern    11-02.6.kern    daylog.1
10-16.1.auth   10-21.1.auth   10-27.1.auth    11-03.0.auth    daylog.10
10-16.1.kern   10-21.1.kern   10-27.1.kern    11-03.0.kern    daylog.11
```

9.5.4 `/usr/adm/sulog`

The `/usr/adm/sulog` file contains a line of information for every attempted use of the `/bin/su` command since this version of the file was started. The line indicates whether the attempt was successful. You could monitor this log for attempted system breaching or other malicious use of a system. `root` should own this file, with no `read` or `write` permissions for others. The format of the log is as follows:

`SU` *MM/DD hh.mm flag tty olduser-newuser*

In the following sample `/usr/adm/sulog` file, the entry that contains a minus sign (line 6) indicates an unsuccessful attempt to use the `/bin/su` command:

```
# cat /usr/adm/sulog
SU 03/11 07:00 + console root-adm
SU 03/11 07:59 + ttyp000 guest-root
SU 03/11 08:13 + ttyp001 jones-root
SU 03/11 11:14 + ttyp002 jones-root
SU 03/11 11:33 + ttyp001 smith-root
SU 03/11 12:26 - ttyp001 smith-root
SU 03/11 12:26 + ttyp001 smith-root
```

9.5.5 `/etc/dump.log`

The `/etc/dump.log` file contains the time and reason for each system dump. The system supplies the time and the user supplies the reason. By default, the dump is located in `/etc/dump.log` and can be accessed using the normal file manipulations, such as `tail`, `cat`, and `more`. When the system is changing out of single-user mode, `brc` calls `coredd` to copy a dump file to a file system. To reconfigure the location of the dump, use the menu system. To change the location of this log file, use the `cpdmp -l` command.

> **Note:** This is a system dump log. It is **not** the log created by the `dump` utility (which is the `/etc/dumpdates` file).

An example of an `/etc/dump.log` follows:

```
# cat /etc/dump.log

cpdmp: 035120 blocks on dump device - waiting to be copied
01/26/94 07:27:09   coredd: Copying system dump into /core//04260727.
UNICOS dump copied to=/core//04260727/dump
   dump taken: 04/26/93 at 07:18:51
   reason: PANIC: master.s: EEX interrupt in UNICOS kernel
```

9.5.6 /usr/adm/nu.log

The new user log contains information about new user accounts on the system that are created by using `/etc/nu`. It includes entries about who created the account and the time it was added, information about the default environment settings, and the IDs. The `/etc/nu` command creates this file (for further information about `/etc/nu`, see Chapter 7, page 171).

The following types of user account transactions are recorded into `/usr/adm/nu.log`: `changed`, `added`, `deleted`, and `destroyed`.

An excerpt from a `nu.log` file follows:

```
Text goes  here
# cat /usr/adm/nu.log
jones:co:L B. Jones
jones:ui:840:di:/home/sis/jones:sh:/bin/csh:dr:/
jones:gi:178
jones:ai:0
jones:rg:178:as:100
jones:dc:none:cm:none:pm:none
jones:ic:none:vc:none
jones:pj[b]:100:pj[i]:100
        changed to
jones:co:Lauren B. Jones
jones:ui:840:di:/home/sis/jones:sh:/bin/csh:dr:/
jones:gi:178
jones:ai:0
jones:rg:178:as:100
jones:dc:none:cm:none:pm:none
jones:ic:none:vc:none
jones:pj[b]:100:pj[i]:100
jones:tp:type0[b]:3:tp:type0[i]:3:tp:type1[b]:3:tp:type1[i]:3
jones:tp:type2[b]:3:tp:type2[i]:3:tp:type3[b]:3:tp:type3[i]:3
        by jones on Mon Sept  13 11:51:00 1993
```

9.5.7 /usr/adm/sa/sa*DD*

The sar command uses the /usr/adm/sa/da*DD* data collection file to report system activity. The /usr/lib/sa/sadc and /usr/lib/sa/sa1 commands write data to this file; they must be scheduled by cron to run at frequent intervals (such as every 15 minutes).

The /usr/adm/sa/sa*DD* file is too large and too varied to show a representative example. It is filled with multiple types of reports, each with many different output fields.

For more information about system activity reporting, see the sar(1) and sar(8) man pages and *UNICOS Resource Administration*, Cray Research publication SG–2302.

9.5.8 /usr/adm/sl/slogfile

The /usr/adm/sl/slogfile data file records UNICOS multilevel security (MLS) event information. The reduce command, executable only by the security administrator, reads this data file. The reduce command extracts, formats, and outputs entries from UNICOS MLS event files. The MLS event logging daemon, slogdemon, collects security-relevant records from the operating system by reading the character special pseudo device /dev/slog. An excerpt of the output from the reduce command follows:

```
# /etc/reduce -s 04021300 -u jones -p

Apr  2 14:49:21 1993  Validation        o_lvl: 0  s_lvl: 0  jid:0  pid:17183
    r_ids:[jones(8863),tng(146)]   e_ids:[jones(8863),tng(146)]    ********
    Login uid: jones(8863)
  Login to [jones(8863),tng(146)] : Okay   via 128.162.121.20   on /dev/ttyp042
Apr  2 14:49:21 1993  Setuid Syscall    o_lvl: 0  s_lvl: 0  jid:1255  pid:17183
    r_ids:[jones(8863),tng(146)]   e_ids:[jones(8863),tng(146)]         ********
    Login uid: jones(8863)
   Setuid call from root (0) to jones (8863) was successful::
```

9.5.9 /usr/spool/msg/msglog.log

The /usr/spool/msg/msglog.log file contains messages and replies to and from the operator. Following is an excerpt from a msglog.log file:

```
# cat /usr/spool/msg/msglog.log

Sep 16 09:51:20  Message    1: WARNING THRESHOLD ON /nasc
Sep 16 09:58:59  Message    2: CRITICAL THRESHOLD ON /nasc::

Jun  9 23:34:02  Message daemon stopped
Jun 10 00:43:15  Message daemon started
Jun 10 04:07:49  Message    1: From ghe:  How are you?
Jun 10 04:08:44  Reply    1: good::
Jun 18  12:41:52  Informative: ********* SYSTEM ACCOUNTING RESTARTED for 0618/*
Jun 18 12:48:21  Informative:  *************** SYSTEM ACCOUNTING COMPLETE Thu J*:
```

9.5.10 /usr/lib/cron/cronlog

The /usr/lib/cron/cronlog file reports the status of all commands that cron executes, including at, batch, and crontab. When the UNICOS system is brought to multiuser mode, the old log file is copied to /usr/lib/cron/OLDLOG.

Various types of error messages may be present in the `cronlog` file, including messages about when `cron` was started and stages of job execution. The `cronlog` file has the following format:

`CMD:` *command_executed username process_id job_type start_time username process_id job_type end_time* `rc=`*error return code*

The *job_type* argument can have one of the following values:

`a` = `at`(1) job

`b` = Batch job

`c` = `cron`(8) job

An example of `/usr/lib/cron/cronlog` follows:

```
! *** cron started ***   pid = 3654 Thu Sep 16 17:47:44 1993
! new user (ce) with a crontab Thu Sep 16 17:47:45 1993
! new user (nfs) with a crontab Thu Sep 16 17:47:45 1993
! new user (root) with a crontab Thu Sep 16 17:47:46 1993
>  CMD:      date >>/home/swts/geir/60564.cron/date.log
>  root 3687 c Thu Sep 16 17:48:01 1993
<  root 3687 c Thu Sep 16 17:48:02 1993
>  CMD: /usr/lib/acct/ckpacct
>  root 4291 c Thu Sep 16 18:00:01 1993
>  CMD: /usr/lib/sa/sa1 600 1
>  root 4292 c Thu Sep 16 18:00:01 1993
>  CMD:      date >>/home/swts/geir/60564.cron/date.log
>  root 4293 c Thu Sep 16 18:00:01 1993
<  root 4293 c Thu Sep 16 18:00:02 1993
<  root 4292 c Thu Sep 16 18:00:02 1993
<  root 4291 c Thu Sep 16 18:00:04 1993
>  CMD: $HOME/scripts/runsequence cpuseq b
>  ce 4731 c Thu Sep 16 19:30:01 1993
>  CMD:      date >>/home/swts/geir/60564.cron/date.log
>  root 4732 c Thu Sep 16 19:30:01 1993
<  root 4732 c Thu Sep 16 19:30:01 1993
<  ce 4731 c Thu Sep 16 19:30:12 1993
```

9.5.11 `/usr/tmp/nqs.log`

The Network Queuing System (NQS) log, created by the NQS log daemon, contains NQS activity. Its default location is the ASCII file `/usr/spool/nqs/log` (to change the location of the log file, use the `qmgr` set log_file command; to see where the current log file resides, use the qmgr show parameters command). Access to /usr/spool/nqs is restricted; however, if you have the correct permissions, you can access the NQS log file by using normal file manipulations, such as tail, cat, and more. If you experience problems with NQS, use a tail -f command on this file to observe what NQS is doing.

A sample nqs.log file follows:

```
# cat /usr/tmp/nqs.log

NQS(INFO): local mid = 130
I$nqs_boot(): TZ=CST6CDT
NQS(DEBUG): tra_read():0, pid 4033,  state=0, sequence#=0, tid=0
NQS(DEBUG): gen_shrpri_tree(): completed setudb.
NQS(INFO):  gen_shrpri_tree(): Fair Share turned off, Share_wt sched factor set.
NQS(DEBUG): gen_shrpri_tree(): Sh_Decay_usage =  0.0000, Sh_Run_rate =  1.0000
NQS(DEBUG): gen_shrpri_tree(): Share_basis & SHAREBYACCT = 8
NQS(DEBUG): gen_shrpri_tree(): childcnt = 1, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 2, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 3, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 4, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 5, st[0].childsum = 0
NQS(INFO):  nqs_ldconf(): i = 1NQS(INFO): nqs_ldconf(): Pipe queue gale; Dest count: 1
NQS(INFO):  nqs_ldconf(): Creating new destination 0NQS(INFO):  nqs_ldconf(): batch
NQS(INFO):  nqs_upd.c(): Adding new destn batch to head of queue
NQS(INFO):  upd_addquedes(): Updating queue gale destinations
NQS(INFO):  upd_addquedes(): Destination 0;  832NQS(INFO):  nqs_ldconf(): i = 1
NQS(INFO):  upp_setlogfil(): Logfilename - /usr/spool/nqs/log
NQS(INFO):  upp_setlogfil(): Set/Reset command - $$/usr/spool/nqs/log
NQS(INFO):  netdaemon(): Listening on TCP/IP port: nqs
NQS(INFO):  nqs_rbuild(): Set flag for first time thru spawn.
NQS(INFO):  nqs_boot(): BOOTDONE, Database present.
NQS(INFO):  upp_setchkpntdir(): New directory = /usr/spool/nqs/private/root/chkt
NQS(INFO):  upp_setlogfil(): Logfilename - /usr/spool/nqs/log
NQS(INFO):  upp_setlogfil(): Set/Reset command - #$/usr/spool/nqs/log
NQS(INFO):  upp_setsnapfil(): New pathname = /home/swts/cjd/nqs_snapfile
```

9.5.12 /usr/adm/errfile

The error log is a binary file that contains error records from the operating system. errpt processes error reports from the data. The /etc/errdemon command (see the errdemon(8) man page) reads /dev/error and places the error records from the operating system into either the specified file, or errfile, by default. The /etc/rc (see the brc(8) man page) script starts /etc/errdemon, and /etc/mverr starts a new errfile.

9.5.13 /usr/spool/dm/*

If UNICOS Data Migration Facility (DMF) software is configured on your system, the /usr/spool/dm/dmdlog.*YYMMDD* files record activities that pertain to data migration.

A sample /dm/dmdlog.*YYMMDD* file follows:

```
# cat /usr/spool/dm/dmdlog.930912

        dmdlog.930912

10:55:29 Data Migration daemon 35745 initializing, release level 6100
10:55:29 0 index entries in database
10:55:29 Command request pipe initialized, fd = 7
10:55:29 Kernel request pipe initialized, fd = 8
10:55:29 initmsp: msp fake, pid = 35751, wt_fd = 10, rd_fd = 11
10:55:29 machine id set to 2158163973
10:55:29 First available handle for assignment is 2158163973:1
10:55:30 0 incomplete MSP entries found
10:55:30 0 soft-deleted premigration files found
   .
   .
   .
10:56:35 Counts - permdel,     0,     0,     0,     0,  0,  0
10:56:35 Counts - retrybu,     0,     0,     0,     0,  0,  0
10:56:35 Counts - krecall,    10,    20,    20,     0,  0,  1
10:56:35 Counts - kremove,    28,    28,    28,     0,  0,  1
10:56:35 Counts - kcancel,     0,     0,     0,     0,  0,  0
10:56:35 Counts - invalid,     0,     0,     0,     0,  0,  0
10:56:35 Counts -  pclear,     0,     0,     0,     0,  0,  0
10:56:35 Current mem  = 94437
10:56:35 Stopping daemon processing
10:56:35 Data migration daemon stopped, exit=0
```

Note: The following log files also exist for each file system under data migration:

- `dmloght` (generated by the `dmhit` command)
- `dmlogct` (generated by the `dmmctl` command)
- `dmlogsm` (generated by the `dmfree` command)

9.6 Cleaning up system logs

Some log files are recycled during each reboot, some logs accumulate content slowly and must be cleaned up only occasionally, and some log files accumulate content quickly and should be monitored and cleaned up frequently. This section describes each group of log files.

9.6.1 Log files recycled during each reboot

The following log files are recycled during each reboot; therefore, you do not have to monitor them for space consumption. If any of the log files must be saved, however, you should copy them to a location of your choice before shutting down the system. If you forget their location, most of the log files are linked to `/usr/spool/ccflogs` .

Log files that recycle are as follows:

- `/etc/rc.log` (log file from `init 2` function)
- `/usr/adm/sulog` (including all `su` records)
- `/usr/spool/msg/msglog.log` (messages and replies from and to an operator)
- `/usr/lib/cron/log` (all `cron` entries since reboot)
- `/usr/tmp/nqs.log` (all NQS entries)

9.6.2 Small accumulative log files

The following log files accumulate content slowly, but you should clean them up occasionally so that they do not consume space needlessly:

- `/etc/boot.log` (records boot dates and times for a system)
- `etc/dump.log` (records crash dump dates and dump file locations)

- usr/adm/nu.log (records all /etc/nu output)

9.6.3 Large accumulative log files

The system activity report (sar) data and report log files accumulate content quickly; therefore, you should monitor these and clean them up frequently. If not managed promptly, these log files could potentially saturate the /usr file system. All sar data is saved up to 31 days in /usr/adm/sa/sa*DD*. At the end of each month, you should dump them to a file server or to tape; otherwise, newer collected data will overwrite them. The sar reports (stored in /usr/adm/sa/sar*DD*) are kept only up to 7 days, because the reports usually are not backed up. To change the number of days you want to keep sar data or sar reports, modify /usr/lib/sa/sa2.

You also should monitor the following log files:

- Email log file
- User mail files if not read and cleared
- NQS log files (these can grow quickly)
- errpt files when active disk errors or tape activity exists
- MLS log files, which are located in /usr/adm/sl