Chapter 7

Accounting on the CM-5 System

This chapter provides information about how CM-5 accounting works (see Section 7.1), error handling (see Section 7.2), command syntax (see Section 7.3), and obtaining useful information from its files (Section 7.4).

7.1 Overview

This section describes what kind of information CM-5 accounting can provide and how this information is gathered.

7.1.1 What Can CM-5 Accounting Do?

CM-5 system accounting is an enhanced version of SunOS accounting, capable of generating detailed accounting information for all processes that run on the CM-5 system — CM-5 processes as well as UNIX-only processes. The accounting system can be used for resource monitoring as well as process accounting, enabling a site to charge for CM-system usage and determine resource usage patterns. Accounting data measures resource usage on a per-process basis. For instance, the accounting records can be used to print raw data that you can parse further to answer the following questions:

- How much time did a user spend on the PNs?
- How much time did a user spend on the PMs?
- How much I/O did a user do, and how long did it take?
- Which command used the most PM resources?

Copyright © 1995 Thinking Machines Corporation

Processes can be identified for accounting by various means, including the usual UNIX group ID, user ID, and file type (i.e., regular file or socket) or device name. CM-5 processes can also be identified by an account ID, which can be set by the environment variable **CM_ACCOUNT_ID**. See the following section, *User Verification*, for details.

All aspects of CM-5 accounting are controlled by CMOST utilities. Utilities enable you to manage the accounting system and its files, extract useful information from the logs, and format reports. These utilities are described in detail in Sections 7.3 and 7.4.

User Verification

If the **-V** (uppercase V) verification switch is included on the **ts-daemon** command line when timesharing is started on a partition, the **CM_ACCOUNT_ID** environment variable can be used to verify that a user on that partition does indeed have permission to run jobs on the CM-5 system: when a process attempts to register with a **ts-daemon** that was started with the **-V**, flag, **ts-daemon** checks the user's UID and **CM_ACCOUNT_ID** by a site-supplied executable (specified as an argument to **ts-daemon -V**). If the check is valid, the process is allowed to register. If the check is invalid, a message is printed to the accounting log file and registration is denied.

For more information about user verification, see the "Starting the Timesharing Daemon (**ts-daemon**)" subsection of Section 3.2.3 in Chapter 3.

7.1.2 What Does CM-5 Accounting Not Do?

We do not supply a utility to calculate chargeback to the user.

7.1.3 How Does CM-5 Accounting Work?

CM-5 accounting runs on a per-partition basis — each partition manager maintains its own accounting system, tracking the processes that run on the partition's PNs and/or on the PM itself. One partition's accounting does not affect another partition's accounting; for example, accounting in one partition can be turned off without affecting another partition's accounting.

Copyright © 1995 Thinking Machines Corporation

A partition's accounting records are stored in kernel memory until they are archived to disk. Each partition manager *must* write to an individual file to prevent interleaving of process records from different partitions (which would render the records unusable). The files can be located in a central directory for convenience. The following subsections describe the archival process.

Process Accounting

The kernel on each partition manager maintains the accounting records for processes that run on that partition's PNs and/or that PM. If CM-5 accounting is turned on for a particular partition, the following events occur upon termination of each process that runs on its PNs or its PM:

- The PM's kernel queues the process's accounting records in memory. Each record takes up approximately 1 Kbyte of memory. By default, each partition is reserved 0.5 Mbytes of internal memory to be used by accounting — for the queued accounting records of already-exited processes as well as queued records for processes that are still running . The amount of memory reserved for accounting is configurable, however, in the range of 0.15 Mbyte to 1.5 Mbyte.

  Under normal usage of the partition, the amount of memory allotted for accounting is not in danger of being totally consumed; only if the accounting daemon goes down or is sleeping is there such a danger. If this amount of memory is close to being totally consumed (within a configurable percentage: default = 90% consumed), a warning appears on the system console. If the amount of memory allotted for accounting is totally consumed, a message appears on the system console, and accounting records for new processes are lost, as they are not allocated data storage structures (although records for processes initiated before memory consumption continue to be updated and, when completed, are enqueued). Accounting for new processes is resumed when memory is released by freeing records; see Section 7.3.5, Freeing Accounting Records for Reuse.

- The kernel writes the queued records to disk, on behalf of the accounting daemon. The accounting daemon initiates the archiving via an **`ioctl()`** on the pseudo-device **`/dev/cmac`**. (This **`ioctl()`** never returns, although it is interruptible by the user.) In this manner, the kernel uses the daemon's timeslices to transfer data from the ker-

Copyright © 1995 Thinking Machines Corporation

nel directly to the archive file. After the transfer, the kernel's accounting records are returned to the free list for re-use.

All process accounting records are stored in one binary format file, *acctfile*. The records are stored in chronological order based on the time the process exited. Newly archived records are always appended to the file in the order they were enqueued.

When the file system to which accounting writes is close to being full (within a configurable percentage: default = 90% full), a message is sent to the system console. When the file system becomes 100% full, writes initiated by the accounting daemon will fail, and an appropriate error message will be sent to the log and the system console.

The events above are also instigated upon a request to write synchronization records (useful to record accounting information for processes that run long-term). When a synch request occurs, all other accounting activities are momentarily suspended while the records in the queue are written to disk and then records for currently running processes are written to disk. Accounting then resumes normal operation.

I/O Accounting

Each I/O device used by a process has a separate I/O device entry in that process' accounting record, enabling I/O summaries to be reported on a per-device basis, as follows:

- I/O to/from an SDA logical device (file system)
- I/O to/from a device connected to the IOBA (DataVault, VMEIO host computer, CM-IOPG)
- I/O to/from an ITS
- I/O to/from a socket (CM5-HIPPI)
- I/O to/from a UNIX file (or any file not on an SDA, ITS, socket device, or IOBA device)
- I/O to a particular character device or block device

Each time a process performs an I/O operation, the process' I/O record list is examined. If an I/O record exists for one of the six categories listed above, the record is updated. Otherwise, a new I/O record is created and allocated in much the same way that process accounting records are allocated.

Copyright © 1995 Thinking Machines Corporation

When the process' accounting record is enqueued, all of its I/O records are still attached and enqueued with it. Immediately after the process' accounting record has been written to disk, each I/O record is written to the appropriate device's entry in the record. Afterwards, all the I/O records are returned to their free list for re-use by other processes.

When the accounting daemon starts up, it automatically creates an auxiliary accounting data file, *acctfile*`.dev`. This binary-format file contains a snapshot of the `/dev` directory and is used for correlating a logical device name with the major and minor device numbers of the devices in the accounting records. A new snapshot is appended every time the daemon receives a `SIGHUP`.

7.1.4 What Information is Accounted?

As stated in the previous subsection, a partition's accounting records are kept in two binary-format files: one for all the process data received from the kernel, and one for the environment information extracted from the kernel data on a device-dependent basis. The records are of fixed length and have an 8-byte header. See Table 7 and Table 8, respectively, for list the fields that make up each process accounting and I/O accounting record.

Note that the PN memory fields (e.g., pn_last_stack, pn_last_pstack), refer to memory allocated per PN; the parallel PN memory fields (e.g., pn_last_pstack) are filled in only for CM-5 partitions that have vector units. In addition, the serial read and write fields (e.g., bytes_read) refer to I/O to and from CP memory (via a `read()` or `write()` system call) while the parallel read and write fields (e.g., bytes_pread) refer to I/O to and from a buffer striped across all the PNs (via a `cm_read()` or `cm_write()` system call).

Copyright © 1995 Thinking Machines Corporation

Table 7. Information kept by the process accounting record and reported by the extract command (see Figure 27 in Section 7.4.2). For the record structure, see `/usr/include/sys/cmacct.h`, typedef `CMAC_Kernel_Account_Data`.

Field	Description
key	Information identifying the process: its process ID and time of creation. Reported by the Pid, Host, and Timestamp fields of **`extract`**.
hostname	The name of the PM on which the job executed. Reported by the host field of **`extract`**.
timestamp	The time this record was written. Reported by the Stime field of **`extract`**.
cmaccount_flag	The status of the record, i.e., whether it was written upon the process's exit or upon a request to sync accounting. **`CMAC_FLAG_EXIT`** has the value 4. Reported by the CMflg field of **`extract`**.
exit_status	The process's exit status. Reported by the Exit field of **`extract`**.
command	The name of the program executed. Reported by the Cmmd field of **`extract`**.
tty	The controlling tty. Reported by the tty field of **`extract`**.
in_baseline_op	This field is for internal processing only and is not reported by **`extract`**.
sid	The POSIX-compliant session ID†. Reported by the Sid field of **`extract`**.

(continued)

† See **`setsid(2)`** for details.

Copyright © 1995 Thinking Machines Corporation

Table 7, continued.

Field	Description
uid, euid	The process's user ID (the user number — third field in **/etc/passwd** — of the person who created the process) and effective user ID (the number that determines what resources the process had permission to access; this is the same as uid, unless the program is a setuid process). Reported by the Uid and Euid fields of **extract**.
gid, egid	The process's group ID and effective group ID (see entry for effective_uid). Reported by the Gid and Egid fields of **extract**.
num_children	The number of children created by the process (the connections can be made via the parent key). Reported by the Forks field of **extract**.
io_ctr	The number of I/O records associated with this process record. Reported by the Iorecs field of **extract**.
parent_key	The key of this process's parent process (if any). The parent key is always created when a process is forked off, assuming accounting is active. It is described in **cmsys/cmacct-types.h**. Reported by the Parentid field of **extract**.
account	The 16-character ID under which to account this process (setable via environment variable **CM_ACCOUNT_ID**; if not set, this field is NULL). Reported by the Acct field of **extract**.
cp_utime	Time spent in user mode on the CP. Reported by the Real field of **extract**.
cp_systime	Time spent in system mode of the CP. Reported by the Sys field of the **extract**.
cp_eltime	Difference between the process's time of creation and the time_stamp field. Reported by the User field of **extract**.
cp_statmem_int	Process's use of the CP's static memory: (text + data size) * time_used. Reported by the Cptxt field of **extract**.
cp_stack_int	Process's use of the CP's stack memory: stack_ size * time_used. Reported by the Cpstk field of **extract**.
cp_last_stack	The most recent size of the CP stack allocated for the process. This field is not reported by **extract**.

(continued)

Copyright © 1995 Thinking Machines Corporation

Table 7, continued.

Field	Description
cp_heap_int	Process's total use of the CP's heap memory: heap_size * time_used. Reported by the Cpdat field of **extract**.
cp_last_static	The most recent size of the CP static memory allocated for this process. This field is used internally and is not reported by **extract**.
cp_max_static	The maximum size of the CP static memory allocated for this process. Reported by the Cpmtxt field of **extract**.
cp_max_stack	The maximum size of the CP stack allocated for this process. Reported by the Cpmstk field of **extract**.
cp_max_heap	The maximum size of the CP heap allocated for this process. Reported by the Cpmdat field of **extract**.
cp_last_heap	The most recent size of the CP heap allocated for this process. This field is not reported by **extract**.
pn_cm_pid	The process's PID on the Connection Machine. Reported by the Cmpid field of **extract**.
pn_partition_size	The number of PNs in the partition. Reported by the Partsz field of **extract**.
pn_now_in_use	This field is for internal processing only and is not reported by **extract**.
pn_part	The kind of hardware the partition contains: Vector Unit, No Vector Unit, or SDA. See **cmacct-types.h**. Reported by the Partnm field of **extract**.
pn_utime	The amount of user time on the PNs. Reported by the Pnusr field of **extract**.
pn_eltime	The amount of elapsed time on the PNs. Reported by the PNReal field of **extract**.
pn_start_utime	This field is for internal use only (not reported by **extract**).
pn_last_static	The most recent size of the PN static memory allocated for this process. This field is not reported by **extract**.
pn_max_static	The maximum size of the PN static memory allocated for this process. Reported by the Pnmtxt field of **extract**.
pn_statmem_int	Process's total use of the PN static memory: (text + data size) * time_used. Reported by the Pntxt field of **extract**.

(continued)

Copyright © 1995 Thinking Machines Corporation

Table 7, continued.

Field	Description
pn_last_stack	The most recent size of the PN stack allocated for this process. This field is used internally and is not reported by **extract**.
pn_max_stack	The maximum size of the PN stack allocated for this process. Reported by the Pnmstk field of **extract**.
pn_stack_int	Process's total use of the PN stack memory: stack_size * time_used. Reported by the Pnstk field of **extract**.
pn_last_heap	The most recent size of the PN heap allocated for this process. This field is not reported by **extract**.
pn_max_heap	The maximum size of the PN heap allocated for this process. Reported by the Pnmdat field of **extract**.
pn_heap_int	Process's total use of the PN heap: heap_size * time_used. Reported by the Pndat field of **extract**.
pn_last_pstack	The most recent size of the parallel stack allocated for this process. This field is used internally and is not reported by **extract**.
pn_max_pstack	The maximum size of the parallel stack allocated for this process. Reported by the Ppmstk field of **extract**.
pn_pstack_int	Process's total use of the parallel stack: parallel_stack_size * time_used. Reported by the Ppstk field of **extract**.
pn_last_pheap	The most recent size of the parallel heap allocated for this process. This field is not reported by **extract**.
pn_max_pheap	The maximum size of the parallel heap allocated for this process. Reported by the Ppmdat field of **extract**.
pn_pheap_int	Process's total use of the parallel heap: parallel_stack_size * time_used. Reported by the Ppdat field of **extract**.

Copyright © 1995 Thinking Machines Corporation

Table 8. Information kept by the I/O usage record and reported by the extract command (see Figure 27 in Section 7.4.2). For the record structure, see `/usr/include/sys/cmacct.h`, typedef `CMAC_Kernel_Account_IO_Data`.

Field	Description
devnumber	The file system's device number. Reported by the Dev field of **`extract`**, in angle brackets.
rectype	The type of I/O record: SDA, ITS, IOBA, Socket, File, or Device. Reported by the Dev field of **`extract`**, in square brackets.
num_opened	Number of times the device is currently open. Reported by the Open field of **`extract`**.
real_num_opened	Number of times the device was opened. Reported by the Rlopen field of **`extract`**.
last_open	This field is for internal use only (not reported by **`extract`**).
time_opened	Cumulative amount of time any file descriptor was open. Reported by the Timeopen field of **`extract`**.
num_reads	Number of serial read calls. Reported by **`extract`**'s Reads field.
num_writes	The number of serial write calls. Reported by the Writes field of **`extract`**.
read_time	The cumulative time spent doing serial reads. Reported by the [Reads] Timeread field of **`extract`**.
write_time	The cumulative time spent doing serial writes. Reported by the [Writes] Timewrite field of **`extract`**.
bytes_read	The cumulative number of bytes read by serial read operations. Reported by the [Reads] Bytes field of **`extract`**.
bytes_written	The cumulative number of bytes written by serial write operations. Reported by the [Writes] Bytes field of **`extract`**.
num_preads	The number of parallel read calls. Reported by the PReads field of **`extract`**.
num_pwrites	The number of parallel write calls. Reported by the PWrites field of **`extract`**.
pread_time	The cumulative time spent doing parallel reads. Reported by the [PReads] Timeread field of **`extract`**.
pwrite_time	The cumulative time spent doing parallel writes. Reported by the [PWrites] Timewrite field of **`extract`**.
bytes_pread	The cumulative number of bytes read by parallel read operations. Reported by the [PReads] Bytes field of **`extract`**.
bytes_pwrite	The cumulative number of blocks written by parallel write operations. Reported by the [PWrites] Bytes field of **`extract`**.

Copyright © 1995 Thinking Machines Corporation

7.2 Error Handling

All errors generated by the accounting system are logged to the console and to **/var/adm/messages**. The accounting error messages and their meanings are as follows:

`CM_accounting: acctfile unexpectedly truncated to` *d* `bytes.`
: This message appears if the accounting file is too short to contain the header, e.g., an empty file.

`CM_accounting: version mismatch` *version1* `(`*version* `in` *acctfile*`).`
: This message appears if the accounting software and the accounting log file are of different versions.

`CM_accounting: log file` *acctfile* `more than %`*d* `consumed.`
: This message appears when the log file has reached its warning percentage of available disk space remaining. (See **cmacct -F** in Section 7.3).

`CM_accounting: archive-log file system full.`
: This message appears when the file system in which the accounting log resides is 100% full.

`WARNING! accounting daemon unable to access` *acctfile*`.`
: This message appears when the CP kernel is unable to write on the daemon's behalf. This message usually indicates that NFS is down. This message is repeated every time a process exits until the situation is remedied.

`CM_accounting enabled [ruid` *d*`]`
: This message appears when the accounting system is turned on. The real user ID of the person issuing the **cmacct -o** command is logged.

`CM_accounting disabled [ruid` *d*`]`
: This message appears when the accounting system is turned off. The real user ID of the person issuing the **cmacct -k** command is logged.

`CM_accounting: reached highwater mark of` *d* `bytes[`*d*`]`
: This message appears when the accounting memory limit (shown by the first *d* value) is reached. The *d* value in square brackets is the amount of memory that has actually been allocated.

`CM_accounting: cannot allocate` *u* `bytes.`
: This message appears if the kernel heap has been exhausted.

Copyright © 1995 Thinking Machines Corporation

`CM_accounting: using free list.`
: This message appears when a memory-crunch situation (see the highwater mark message) has been rectified; that is, memory has become available to the accounting system.

`CM_accounting: memory more than %`*d* `consumed.`
: This message appears when the resource-warning percentage has been reached. (See **`cmacct -M`** in Section 7.3).

7.3 Using the System

7.3.1 cmacct Syntax

The **`/usr/etc/cmacct`** command is used to control all aspects of the accounting system. Execute **`cmacct`** individually on the PM of the partition for which you want to starting accounting, using the **`rsh`** command from the system control processor. The syntax of **`cmacct`** is:

`cmacct [`*operations_flags*`] [`*tuning_flags*`]`

where *operations_flags* are the following:

`-d``[aemon]` Starts the accounting logging daemon explicitly, as would be needed if the daemon crashed. The accounting logging daemon is backgrounded automatically.

`-k``[ill]` Stops accounting on the PM on which it is executed and kills the accounting logging daemon.

`-i``[nfo]` Reports the current status of the accounting system. This is the only **`cmacct`** flag that can be executed without root privileges. See Section 7.3.2 for more information.

`-o``[pen]` *acctfile*
: Activates accounting facilities by opening the log file *acctfile* and starting the accounting logging daemon (backgrounded automatically).

  If **`cmacct -o`** is executed when accounting is already running, and the *acctfile* argument is a filename different from the one currently being used, the log

Copyright © 1995 Thinking Machines Corporation

file is rotated; that is, logging goes to the new *acctfile*, and the previous one is not deleted.

If **cmacct -o** is executed when accounting is already running, but the *acctfile* argument is the same filename as the one currently being used, there is no effect (information continues to be appended to *acctfile*).

-s[ync] Generates an accounting snapshot of the processes that are currently running, for the purpose of including accounting records for processes that run long-term. Records generated by a sync are flagged so that they can be differentiated from a record generated when a process exits.

and *tuning_flags* are the following:

-a[llocation] *bytes*
The amount of memory to allocate for use by the accounting system. This size is specified in *bytes* and can be in the range 150 Kbytes to 1500 Kbytes, inclusive. The default is 500 Kbytes. **cmacct** interprets a suffix of **k** or **K** to mean to multiply the preceding number by 1024 and a suffix of **m** or **M** to mean to multiply by 1024 x 1024.

-M[emory] *warning_percentage*
Sets the accounting memory's warning percentage: when the memory allotted to accounting has been consumed to the *warning_percentage* percent mark, a message is sent to the system console. The default *warning_percentage* is 90.

-F[ile system] *warning_percentage*
Sets the file system's warning percentage: when the file system to which accounting writes becomes full to the *warning_percentage* percent mark, a message is sent to the system console. The default *warning_percentage* is 90.

-l[imit] Limits reporting to CM-5 processes. The default is to report all processes, including UNIX-only ones.

Copyright © 1995 Thinking Machines Corporation

-u[nlimit] Record accounting information for all processes, including UNIX-only ones. This is the default behavior.

7.3.2 Obtaining Information About the Accounting System's Current State

To obtain information about the current state of a PM's accounting system, execute **cmacct -i** on that CP. **cmacct -i** can be executed by anyone; root privileges are not required.

When accounting is not turned on, the **cmacct -i** output is similar to the following:

```
syscon# rsh CP_name cmacct -i
Accounting status for Mon Nov  8 12:53:05 1993
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Daemon never enabled
Accounting never enabled
No logfile set since boot time
Queue Status:     0 Data records     0 I/O records on queue
Dynamic Memory Usage:
               0 Data records allocated <0 bytes>
               0 I/O records allocated <0 bytes>
Maximal memory allocation: <512000 bytes>
Last file system usage at 0%
Warnings set to:    File system  90%          Memory  90%
Logging event type: CP [X]        PN [X]    IO [X]
Logging all processes

syscon#
```

When accounting is turned on, **cmacct -i** produces output similar to the following:

```
syscon# rsh CP_name cmacct -i

Accounting status for Mon Nov 8 12:54:00 1993
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Daemon process [287] active from Mon Nov 8 12:53:52 1993

Accounting enabled from Mon Nov 8 12:53:46 1993
```


Copyright © 1995 Thinking Machines Corporation

```
Current account logfile is      /tmp/foo
Queue Status:     0 Data records     0 I/O records on queue
Dynamic Memory Usage:    50 Data records allocated <18800 bytes>
                         50 I/O records allocated <7200 bytes>
Maximal memory allocation: <512000 bytes> Last file system usage
at 78%
Warnings set to:    File system  90%          Memory  90%
Logging event type: CP [X]     PN [X]     IO[X]
Logging all processes

syscon#
```

7.3.3 Starting and Tuning the Accounting System

Starting Accounting

IMPORTANT

Before attempting to start the accounting system on a CP, make sure the device **/dev/cmac** has been created. If it has not yet been created, run the following as root:

```
syscon# rsh CP_name mknod /dev/cmac c 111 0
```

To start the accounting system on a PM, log in to the system control processor as root and execute the following:

```
syscon#   rsh PM_name cmacct -o acctfile
```

If you do not specify a complete pathname, *acctfile* is placed in the current directory in the local file system. We recommend that the system date be incorporated into the name of *acctfile* to facilitate log rotation and history.

The following events occur upon executing **cmacct -o**:

(1) A new accounting daemon is forked off.

Copyright © 1995 Thinking Machines Corporation

(2) The log file is checked; if it exists, it is opened and the version verified; if it does not exist, it is created.

(3) Memory for accounting is allocated and initialized (default amount is overridden if **-a** is also specified).

(4) Parent of the forked process returns to user; does not wait for child process (accounting daemon).

Tuning Accounting

The **cmacct** flags **-F**, **-M**, **-l**, **-u**, and **-a** can change the behavior of the accounting system while it is still running.

- Supply the **-F** *percentage* flag to cause a "file system almost full" warning message to be generated at a percentage-full different from the 90% default. To disable file-system warnings, supply **0** as **-F**'s argument.

  Accounting records continue to be written until the file system is 100% full. When the file system becomes 100% full, writes initiated by the accounting daemon will fail, and an appropriate error message will be sent to the log and the system console. Therefore, if *warning_percentage* is 0, no warning is given before the accounting daemon's writes fail.

- Supply the **-M** *warning_percentage* flag to cause a memory-consumption warning to be generated at a percentage-consumed different from the 90% default. To disable memory-consumption warnings, supply **0** as **-M**'s argument.

  Normally, the amount of memory allotted for accounting is not in danger of being totally consumed; only if the accounting daemon goes down or is sleeping is there such a danger. If the amount of memory allotted for accounting is totally consumed, a message appears on the system console, and accounting records for new processes are lost, as they are not allocated data storage structures (although records for processes initiated before memory consumption continue to be updated and, when completed, are enqueued). Accounting for new processes is resumed when memory is released by freeing records; see Section 7.3.5.

Copyright © 1995 Thinking Machines Corporation

- Supply the **-l** flag to cause the accounting system to log CM-5 processes only. That is, when the **-l** flag is supplied, processes that do not register with the timesharing daemon (**ts-daemon**) are not accounted. (For example, a process that only performs serial I/O to or from the SDA or DataVault would not be accounted). By default, the accounting system logs all processes, including UNIX-only processes.

- Supply the **-u** flag to cause the accounting system to log all processes that run on the CM-5 system, including UNIX-only processes. Since this behavior is the default, the **-u** flag alters the accounting system's behavior only if **-l** had been previously specified.

- **-a**[llocation] *Mbytes*
  The amount of memory to allocate for use by the accounting system. *Mbytes* can be 0.15 to 1.5, inclusive. The default is 0.5 Mbyte. Note that if you execute **-a** while accounting is currently running and specify an *Mbytes* value less than the amount currently consumed, `CM_accounting: memory more than %d consumed` warnings will be generated.

7.3.4 Rotating Accounting Files

To change the file to which accounting records are written without stopping and restarting the accounting system, execute **cmacct -o** *acctfile*, specifying a filename different than the previous one. To facilitate log rotation, we recommend incorporating the date into the name of *acctfile*: for example, **foo.nov8**.

When you execute **cmacct -o** with a different filename when accounting is already running, the following events occur:

- The records that are on the kernel's queue are sync'ed to the "old" file, which is then closed. *acctfile*`.dev` is also closed.

- The new *acctfile* is opened and becomes the current log file. A new *acctfile*`.dev` is created by the accounting daemon.

The frequency with which you rotate the accounting log file depends on system usage and the accounting conventions of the site. We recommend that a **crontab** entry be used to rotate the accounting log file regularly.

Copyright © 1995 Thinking Machines Corporation

7.3.5 Freeing Accounting Records for Reuse

Accounting records wait in a queue in PM memory until the accounting daemon writes them to *acctfile*. The default maximum memory allocation for this queue is 0.5 Mbyte (configurable using **cmacct -a** to 0.15−1.5 Mbytes, inclusive); if this memory is close to being used up — an indication that, for some reason, the accounting daemon is not operational — a message will appear on the system console:

```
CM_accounting: memory more than warning_percentage consumed.
```

The message above appears on the console when the memory allotted to accounting has been consumed to the *warning_percentage* mark (default = 90% consumed, configurable by **cmacct -M**).

To prevent accounting from failing, you must free the accounting records by causing the accounting daemon to become operational once again. To do this, first execute **cmacct -i** to find out whether there is an active accounting daemon:

- If no accounting daemon is active, execute **cmacct -d** to start one.
- If an accounting daemon is active, use **ps** to determine its current state:
  - If the daemon was suspended, allow it to resume.
  - If the daemon is hung, send it a SIGHUP. If it does not respond to the SIGHUP, kill the daemon by executing **cmacct -k** and start up a new one by executing **cmacct -d**.

Once the records are freed, accounting will again be fully operational, and another message will appear on the system console:

```
CM_accounting: using free list.
```

Note, however, that this message does not appear if you have to kill the daemon and restart it, because this results in all state regarding memory usage being lost.

7.3.6 Shutting Down Accounting

To shut down the accounting system on a PM, log in to the system control processor as root and execute the following:

Copyright © 1995 Thinking Machines Corporation

```
syscon# rsh PM_name cmacct -k
```

cmacct -k first writes the contents of the PM's accounting queue to disk, syncs current processes, turns off accounting in the kernel, and releases all memory used by accounting. Then it kills the accounting daemon on that PM and closes the PM's accounting file.

7.4 Obtaining Information from the Accounting Records

CMOST provides two utilities that generate information from the accounting records. Section 7.4.1 describes the utility that dumps the accounting records, and Section 7.4.2 describes the utility that organizes this information into useful data.

7.4.1 Printing Accounting Records

Run **/usr/etc/extract** to parse and print the binary accounting records in ASCII form.

The syntax of **extract** is:

extract [**-s**] *file...*

where

-s[ync]	Generate a sync output only; that is, show only the records generated by execution of **cmacct -s**.
file...	The file(s) from which to extract the accounting records.

extract's output is organized in chronological order by process ID. Figure 27 shows the **extract** output for one process ID.

Table 7 and Table 8 in Section 7.1.4 describe each field of **extract**'s output. Note, in particular, that the device names (shown in square brackets) are the names of the special files in the **/dev** directory. **extract**'s output also shows their corresponding major and minor device driver numbers.

To produce **extract**'s output in report format, see Section 7.4.2.

Copyright © 1995 Thinking Machines Corporation

7.4.2 Obtaining Summaries for Billing or Resource Management

Run the CM-5 accounting post-processor **/usr/etc/report.csh** to produce a summary of the accounting information by account ID, user ID, or file system device. (**report.csh** is a wrapper that invokes an **awk** script. If you wish, you can modify this script to suit your accounting and resource monitoring needs and run the **awk** script directly; see Appendix E.)

To run **report.csh**, execute the following pipeline:

```
% extract file... | report.csh [-a] [-d] [-u]
```

where

file...	The accounting file(s) from which to build the summary report.
-a	Summarize data by account ID.
-d	Summarize data by device.
-u	Summarize data by user ID.

If you have already run the **extract** utility and directed its output to a file, run **report.csh** using the following syntax:

```
% report.csh [-a] [-d] [-u] file...
```

where *file...* is the name of the file(s) that contain the output generated by **extract**.

Figure 28 contains sample **report.csh** output.

Copyright © 1995 Thinking Machines Corporation

```
syscon# rsh PM_name extract acctfile
Pid:288    Host:poulenc.think.com
           Parentid:0            Timestamp:752781240.287691
           Exit:0  Forks:0       Stime:752781240.647691
           Uid:0   Euid:0        Gid:1      Egid:1        Sid:244
           Acct:NULL             Cmmd:cmacct
           Tty:<12,0> Iorecs:5   CMflg:4
           Real:   0.350000      Sys:    0.150000   User:   0.030000
           Cpmtxt: 139264        Cpmstk: 32768      Cpmdat: 57344
           Cptxt:  48740         Cpstk:  11468      Cpdat:  18676
           Cmpid:  0             Partsz: 0          Partnm: 0
           PNReal: 0.000000      PNUser: 0.000000
           Pnmtxt: 0             Pnmstk: 0          Pnmdat: 0
           Pntxt:  0             Pnstk:  0          Pndat:  0
           Ppmstk: 0             Ppmdat: 0
           Ppstk:  0             Ppdat:  0
     Dev[cmac <111,0>]  Open:1  Rlopen:0   Timeopen:     0.30000
           Reads:        0     Bytes:        0  Timeread:     0.0
           Writes:       0     Bytes:        0  Timewrite:    0.0
           PReads:       0     Bytes:        0  Timeread:     0.0
           PWrites:      0     Bytes:        0  Timewrite:    0.0
     Dev[zero <3,12>]   Open:1  Rlopen:0   Timeopen:        0.240000
           Reads:        0     Bytes:        0  Timeread:     0.0
           Writes:       0     Bytes:        0  Timewrite:    0.0
           PReads:       0     Bytes:        0  Timeread:     0.0
           PWrites:      0     Bytes:        0  Timewrite:    0.0
     Dev[File]          Open:8  Rlopen:0   Timeopen:     0.180000
           Reads:        4     Bytes:        842   Timeread:  0.0
           Writes:       0     Bytes:        0     Timewrite: 0.0
           PReads:       0     Bytes:        0     Timeread:  0.0
           PWrites:      0     Bytes:        0     Timewrite: 0.0
     Dev[console <0,0>] Open:8  Rlopen:0   Timeopen:     0.350000
           Reads:        0     Bytes:        0     Timeread:  0.0
           Writes:       1     Bytes:        587   Timewrite: 0.10000
           PReads:       0     Bytes:        0     Timeread:  0.0
           PWrites:      0     Bytes:        0     Timewrite: 0.0
     Dev[Socket]        Open:1  Rlopen:0   Timeopen:     0.30000
           Reads:        0     Bytes:        0     Timeread:  0.0
           Writes:       0     Bytes:        0     Timewrite: 0.0
           PReads:       0     Bytes:        0     Timeread:  0.0
```

Figure 27. Partial output — output for one process ID — from the extract utility.

Copyright © 1995 Thinking Machines Corporation

```
syscon# extract acctfile | report.csh -u
     acctfile:     1 data records        5 I/O records
     total:        1 data records        5 I/O records
Generating accounting summary ...
                ####################################
                #                                  #
                #    ####################################
                #    # CM5 Accounting Job Summary Report #
                #    ####################################
                #                                  #
                #    ####################################
                #    #   Mon Nov 8 12:56:50 EDT 1993    #
                #    ####################################
                #                                  #
                ####################################


####################################################################
SUMMARY BY UID
####################################################################


User ID:               0                Job Total:       1
 Total CP Time (seconds):       Real [    0.35] Usr  [    0.03] Sys   [    0.15]
 Total PN Time (seconds):       Real [       0] Usr  [       0]
 Maximum CP Size (bytes):       Text [  139264] Data [   57344] Stack [   32768]
 CP Integral (byte secs):       Text [   48740] Data [   18676] Stack [   11468]
 Maximum PN Sizes (bytes):      Text [       0] Data [       0] Stack [       0]
 PN Integral (byte secs):       Text [       0] Data [       0] Stack [       0]
 Maximum ParPN Sizes (bytes):   Data [       0] Stack [       0]
 ParPN Integral (byte secs):    Data [       0] Stack [       0]
====================================================================
 Device:                        <zero>
  Open:         1       Timeopen:      0.24
  Read:         0       Timeread:         0     Bytes:          0
  Write:        0       Timewrite:        0     Bytes:          0
  Pread:        0       Timepwrite:       0     Bytes:          0
  Pwrite:       0       Timepwrite:       0     Bytes:          0
====================================================================
 Device:                        <cmac>
  Open:         1       Timeopen:       0.3
  Read:         0       Timeread:         0     Bytes:          0
  Write:        0       Timewrite:        0     Bytes:          0
  Pread:        0       Timepwrite:       0     Bytes:          0
  Pwrite:       0       Timepwrite:       0     Bytes:          0
====================================================================
 Device:                        <console>
  Open:         8       Timeopen:      0.35
  Read:         0       Timeread:         0     Bytes:          0
  Write:        1       Timewrite:      0.1     Bytes:        587
  Pread:        0       Timepwrite:       0     Bytes:          0
  Pwrite:       0       Timepwrite:       0     Bytes:          0
```

continued

Figure 28. A sample accounting report sorted by user ID (page 1 of 2).

Copyright © 1995 Thinking Machines Corporation

```
=========================================================================
 Device:                          <Socket>
  Open:          1       Timeopen:        0.3
  Read:          0       Timeread:          0     Bytes:           0
  Write:         0       Timewrite:         0     Bytes:           0
  Pread:         0       Timepwrite:        0     Bytes:           0
  Pwrite:        0       Timepwrite:        0     Bytes:           0
=========================================================================
 Device:                          <File>
  Open:          8       Timeopen:       0.18
  Read:          4       Timeread:          0     Bytes:         842
  Write:         0       Timewrite:         0     Bytes:           0
  Pread:         0       Timepwrite:        0     Bytes:           0
  Pwrite:        0       Timepwrite:        0     Bytes:           0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

syscon#
```

Figure 28, cont'd. A sample accounting report sorted by user ID.

Copyright © 1995 Thinking Machines Corporation