Fair-share Scheduler [4]

The fair-share scheduler (also referred to as *fair-share*) controls CPU resources and allows the Cray Research system to be shared among groups in an organized fashion. To accomplish this, the fair-share scheduler assigns the system's CPU resource to the most deserving processes.

Other system resources, such as usage of memory, tape, SSD, and system calls, are monitored by appropriate system components. However, the fair-share scheduler can be configured to add a penalty for usage of these resources as well as the CPU resource.

This section includes information on the following topics:

4.1 Design Objectives

The fair-share scheduler is designed to allow users with similar *share allocations* (the number of shares allocated in the user database, or *UDB*) to utilize similar amounts of the CPU resource, regardless of the number of active processes they have executing. By contrast, traditional UNIX process schedulers allow users with more processes to have a larger percentage of the system than their priority might typically allow. The fair-share scheduler knows the aggregate consumption rate of each user, and it does not allow users with many active processes to utilize CPU resources at a higher rate than those users with only a few active processes.

Another goal of resource scheduling is to provide adequate and predictable response times. However, it is possible to have levels of system loading that create a long response time, regardless of the scheduling mechanism used. It is

your responsibility to control the amount of work in process, using other available methods.

Before the UNICOS 8.0 release, the Network Queuing System (NQS) queue structures were the primary method of controlling the system workload. Beginning with 8.0, the Unified Resource Manager (URM) can be used to control the workload. Using URM ensures that the work the system is expected to do is reasonable, and predictable responsiveness is achievable. For more information on URM, see "Unified Resource Manager (URM)," Chapter 8, page 353.

Under the UNICOS operating system, users are allocated a portion of the CPU resource specified by their share. To improve the fair distribution of resources among users who are allocated equal shares, the concept of *usage history* has been introduced. Usage history allows the scheduler to allocate proportionally more resource to a user who has done less work in the recent past than one who has done more. This rewards users who distribute their work over time and can be valuable in environments where deadlines cause users to do work within a short span of time (which increases the possibility of overloading the system).

The length of time during which past work affects the priority calculation is determined by a *decay factor* (expressed as the half-life of usage history). This controls the rate at which past usage is reduced as a factor in scheduling the user's processes. It can be set anywhere in a range from seconds to many days.

4.2 Fair-share Feature Summary

The fair-share scheduler and the portion of resource control represented by it can be summarized as follows:

- Comprehensive user information is contained in the UDB. A number of utilities and library routines are provided to maintain and view this information and migrate from earlier mechanisms for user validation and control.
- Two system calls, `limits`(2) and `policy`(2), provide an interface between the kernel and user levels of the fair-share scheduler. A daemon, `shrdaemon`(8), updates usage information in the UDB and recovers user information from unplanned system halts. If user-level fair-share mode is enabled, the daemon also updates lnode information in the kernel. The `login`(1) command, the `cron`(8) command, and NQS access the new user information and pass it on to the kernel by using the `setshares`() routine to create *lnodes* (limits nodes) based on the UDB definitions.

- The system tracks usage of users or accounts with a user limits structure called an lnode in kernel memory for each active user name on the system. Users can access their system usage with the `limits`(2) system call.
- Both the user and administrator have displays of scheduling activity available through the `shrview`(1) command (as well as the command `shrmon`(8)). A user's profile can be viewed with the command `udbsee`(1), and the administrator may use features of this command to help generate reports or create source input, which, with further manipulation, can be used by `udbgen`(8) for UDB maintenance.
- At system startup or during operation, the administrator uses the `shradmin`(8) command to set or alter the behavior of the fair-share scheduler to tune the system or prepare for differences in operational emphasis. (The `shrdist`(8) and `shrsync`(8) commands can be used to adjust shares.)
- A fair-share hierarchy can be defined to proportion resources among and within organizations so that a predictable amount of system resources can be allocated to each organization. This method of allocation is dynamic and does not allow a portion of the resource to go unused if some of the organizations are not presently active or are unable to utilize their share. Users and administrators can display the fair-share hierarchy with the `shrtree`(8) utility.
- An optional *Share by Account* mode can be used to assign shares to account IDs rather than to users as in the default *Share by User* mode. With this feature, a user's share allocation and resulting scheduling priority are determined by the user's account ID, which is set initially to the default account ID for the user (in the UDB). Scheduling priorities are determined by the current account ID as users change from one account to another using the `newacct`(1) command.
- Usage history to the degree desired is available the administrator. It allows users or organizations who have equal shares but have utilized unequal amounts of resource to come into balance by encouraging the load to be spread over time rather than in a last-minute flurry of activity before a deadline.
- An optional CPU scheduling mode, *user-level fair-share*, provides a user exit and duplicates kernel functionality at the user level. In this mode, the fair-share scheduler's calculations are performed by the fair-share daemon, `shrdaemon`(8), instead of the kernel. `shrdaemon` replaces the kernel functions that apply the scheduling policy algorithms. This optional mode is enabled with the `USRLEVLFSS` flag in `shradmin`(8).

4.3 Components of Fair-share

Several components, both at the user level and in the kernel, work together to accomplish the objectives of the fair-share scheduler. These are described in the following sections and include the UDB, support functions, user and administrator displays, administrator controls, hierarchical share, share normalizing, and process scheduling.

4.3.1 User Database (UDB)

In resource control, all user profile information must be stored in a comprehensive way. The UDB contains all the user information (factors) used for the scheduler. The UDB factors used for the fair-share scheduler are as follows. (For more information on UDB fields, see the `udbgen`(8) man page.)

Factor	Description
Acids	The UDB stores account IDs (*acids*) as a list of up to 64 (set by `MAXVIDS`) numeric account IDs or account names separated by commas. If acids are used, they must be added to the `/etc/acid` file before `udbgen` is executed. The UDB `acids` field is maintained by the administrator; it has the following format: `acids =|+|- :`*n1*`, `*n2*`, ..., `*nn*`:`
Charges	The UDB stores the long-term accumulated costs from the fair-share scheduler. The UDB `shcharge` field is maintained by the fair-share scheduler; it has the following format: `shcharge :`*vv.vv*`:`
Exit-time	Records the time that the user last completely logged off the system (that is, the last time there were zero processes owned by this user running in the system). The UDB `shextime` field is maintained by the fair-share scheduler; it has the following format: `shextime :`*n*`:`
Resource-group	In a fair-share hierarchy, a user can be a member of an entity known as a *resource group*. Resource groups can be members of other resource groups. Four levels are enabled by default; the maximum

number of levels is limited only by the configuration, but system overhead increases for each additional level. Resource groups can be further divided by account IDs, or *shareholders*, by using specific values in the UDB share-flags field (see the "Shareholders" entry in this list).

Resource groups exist in the UDB with the character "*" as the password, ensuring that no user logs in as a resource group. The UDB `resgrp` field is maintained by the administrator; it has the following format: `resgrp:`*n*`:`

Shareholder

Subdivision of a resource group, used when *Share by Account* mode has been selected; also called an account ID. In Share by Account mode, there must be an entry in the UDB for each account ID that is in use. These entries must correspond with account ID (acid) numbers; that is, they must exist in the `/etc/acid` file before `udbgen` is executed (see the "Acids" entry in this list).

Shares

Each user of the system is allocated a number of shares. This number has meaning only as a proportional value; that is, a share represents the proportion of system resources relative to all other users or accounts within the same group. The actual share values in the UDB are not relevant in any other way. The UDB `shares` field is maintained by the administrator; it has the following format: `shares:`*n*`:`

Share-flags

Certain user entries in the UDB can represent an entity other than a direct user of the system (such as a resource group or a shareholder). The UDB `shflags` field is used to denote these special UDB entries. This field is maintained by the administrator; it has the following format: `shflags =|+|-:`*octal*`:` (The fair-share scheduler now enforces correct usage of the fair-share fields in the UDB.)

Usage

One principal factor that the fair-share scheduler uses for establishing priority is a user's previous usage of the system. When a user is completely

logged off the system, that user's decayed usage of the system is recorded in the UDB. The next time the user logs in to the system, the login process calculates a new decayed-usage value to be installed in the system, based upon the amount of time that has passed since the user last logged out (see the "Exit-time" entry in this list). The UDB `shusage` field is maintained by the fair-share scheduler; it has the following format: `shusage:`*vv.vv*`:`

In addition to the fair-share fields in the UDB, several special user accounts (UDB entries) are necessary for the proper operation of the system, including the `Idle` account, the `UnKnown` or `unknown` account, and resource group entries. (See "Setting up system UDB entries," Section 4.4.4, page 209, for a list of these special UDB entries.)

Idle processes are treated just like any other user of system resources in the fair-share system and, as such, need an entry in the UDB. User ID 11 is reserved in UNICOS for representing the idle usage of the system.

There must also be an entry in the UDB called `UnKnown` or `unknown`; this entry is used when no valid UDB entry for a user can be found.

Each resource group or shareholder (account ID) represented within the fair-share hierarchy must also have an entry in the UDB. Resource groups and shareholders are assigned shares, which are taken to be relative to the shares allocated to other resource groups at the same level, and are a subset of the shares in the next higher level in the fair-share hierarchy.

4.3.2 Support Functions

At the user level, the fair-share daemon `shrdaemon`(8) performs the following activities:

- At 1-minute intervals, `shrdaemon` records in the UDB usage for each user and resource group that has finished execution.
- At 5-minute intervals, `shrdaemon` checkpoints the kernel tables holding fair-share usage information about running users and resource groups.
- When the UNICOS operating system is restarted after an unscheduled shutdown, `shrdaemon` recovers user and group CPU consumption information from the checkpoint file.

At the kernel level, the fair-share scheduler performs the following activities:

- At configurable intervals (4 seconds by default), the fair-share scheduler accumulates charges and updates usage information in the lnodes.

  Note: In user-level fair-share mode, updating usage and lnode information is done by the fair-share daemon rather than by the kernel.

- Every 1/60th of a second, CPU usage is accumulated and stored in the lnodes of processes having had the CPU, and the `p_sharepri` values are adjusted based on usage.

- At 1-second intervals, the share priority of each process is decayed according to the rate determined by the nice value of the process.

The most significant factors affecting placement on the kernel run queue (high or low) are as follows:

- The process's resource usage during the last cycle, relative to other processes.

- The process's proportion of machine shares relative to other users or groups (active lnodes) at that time.

The scheduling calculations can lower, raise, or leave unchanged the position of the process in the run queue. When this evaluation has been accomplished, the processes at the top of the run queue are connected to a CPU.

In addition, the positions of all the processes in the queue are evaluated by decaying their resource consumption at a rate determined by their nice values (processes with smaller nice values move toward the top of the queue faster). This has the effect of gradually moving processes to the top of the queue and ensures that every process will be scheduled to run; note that only processes which have not received the CPU recently actually move up in the queue. It is necessary to balance the rate of migration to the top of the queue and the rate of resource consumption so that relative priorities are remembered for a long enough time period to prevent large numbers of processes from migrating to the top of the queue.

4.3.3 User and Administrator Displays

Fair-share displays are generated by the `shrview`(1) command, the `shrmon`(8) command, and the `shrtree`(8) command. In addition, users can view their share control values by using the `udbsee`(1) command, which displays the content of the UDB (with the exception of sensitive information).

Note: The `shrmon`(8) command will not be available in future releases of the UNICOS operating system. Its functionality has been replaced by the `shrview`(1) command.

4.3.4 Administrator Controls

The `shradmin`(8) command changes scheduling parameters, decay rates, flags, and other useful items. One of the intended uses of `shradmin` is to set appropriate control parameters at startup so that you can alter the behavior of the system to reflect current needs without resorting to recompiling modules or other inefficient mechanisms. It is also possible to change scheduling characteristics during system operation. This facility could be used to change scheduling emphasis during portions of the day when, for example, mostly interactive or batch work is encouraged.

Note: You must run `shradmin` before activating the fair-share daemon, `shrdaemon`(8). For more information, see "Activating the fair-share scheduler," Section 4.4.6, page 212.

You can use the `-n` option of `shradmin`(8) to lend more reliability to the fair-share scheduler information in the UDB. This option specifies the interval, in seconds, to be used for copying lnode information to the UDB. When this feature is enabled, the fair-share daemon writes accumulated usage and charge information from the lnodes to the UDB at the specified interval. For more information, see the `shradmin`(8) man page.

The `shrview`(8) command monitors the operation of the fair-share scheduler at a closer level of detail.

To turn off fair-share scheduling, see "Disabling the fair-share scheduler," Section 4.4.9, page 217.

4.3.5 Fair-share Hierarchy

A flat, or single-level, fair-share mechanism is inadequate for many environments because it does not provide a convenient way to allocate shares among organizations and then to the users within each organization. The *fair-share hierarchy* grants each organization a part of the system, determined by the proportional shares assigned to each. After this is accomplished, the members of each organization can be allocated shares as though that organization had exclusive use of the system. This makes the allocation of shares within an organization easier; however, individual user share values are not comparable across organization boundaries.

Whenever the active user population changes, the shares assigned through the fair-share hierarchy are converted to an internal value known as *machine share*. This is the proportion of the available resources to which a group or user is entitled. (You can see these values in the fair-share displays.) Because they are normalized across organizations with respect to each organization's share proportion, machine-share relative values can be directly compared; user shares cannot. Machine shares are recomputed whenever a user logs in or out or when an organization becomes active or inactive. For more information, see "Share normalizing," Section 4.3.6, page 199.

You can activate the optional Share by Account mode by using `shradmin`(8) to set the `SHAREBYACCT` scheduling flag. In this mode, fair-share determines the relative share based on the account ID rather than the user ID (UID). The initial association is based on the default account ID, which is the first ID in the list of valid account IDs for that user in the UDB entry. In Share by Account mode, the `newacct`(1) command reassociates the user with the new account ID, which allows the user to move within the fair-share hierarchy.

If Share by Account mode is enabled, the fair-share hierarchy must have resource-group lnodes at the head of the hierarchy chain; the hierarchy chain must terminate with shareholder (account ID) lnodes.

You can use the `shrtree`(8) command to display and verify the fair-share hierarchy; see "Using the `shrtree`(8) command," Section 4.4.11.3, page 222, for more information.

4.3.6 Share Normalizing

In Figure 4, page 200, three groups have been given portions of the system resources. Groups G1 and G3 have 25%, while Group G2 has 50%. This illustration is a snapshot of a particular instant in time when the active group membership is as shown. Within each group, shares have been allocated based on some arbitrary range of values. G1 is based on the range 0 through 1000, G2 on the range 0 through 10, and G3 on the range 0 through 100, to show that the normalizing function works in mixed-range situations.

The columns headed "G Sh" show the shares allocated to the currently active users from each group. Column "G %" shows the percentage of the group's resource to which each user is entitled, based on the allocated shares and the active group membership. (The percentage will vary as group members arrive and depart because the goal is to distribute the system proportion fairly according to each user's share among the active users at a given instant.)

The column headed "M %" shows the actual machine share each user should have, based on the group in which the user exists. The totals at the bottom of the figure show that the sum of machine shares from each group adds up to the share allocated to the group.

Group share total = 100%

G1 = 25%			G2 = 50%			G3 = 25%		
G Sh	G%	M%	G Sh	G%	M%	G Sh	G%	M%
700	70	17.5	1	10	5	40	50	12.5
150	15	3.75	2	20	10	25	31.25	7.8125
150	15	3.75	3	30	15	15	18.75	4.6875
			4	40	20			
G1 Total:		25%	G2 Total:		50%	G3 Total:		25%

a11426

Figure 4. Share normalizing

Normalizing is accomplished as follows:

1. Calculate the number of shares active, SA_g, within each group g for each member m:

$$SA_g \sum_{m=1}^{m=max} share_m \tag{4.1}$$

2. Calculate machine share, MS_m, for each member m of group g:

$$MS_m = Group\_proportion_g \times (share_m \div SA_g)$$

This example considers only one hierarchy level. When more than one level is being supported on a given system, the method shown above is recursively applied down the hierarchy tree until the MS_m of each group and user node has been calculated. As more levels are added, however, there is a corresponding increase in the system overhead required to perform all the calculations for each cycle. See "Tuning the fair-share scheduler," Section 4.6, page 227, for more information on this overhead.

4.3.7 Process Scheduling

The purpose of process scheduling is to ensure that all system processes are running and to assign the remaining CPU resource to user processes. As discussed previously, this means that the CPU is assigned to the process at the top of the run queue. As resources are utilized, processes move down the queue, and as processes age, they move up the queue. This change in queue position is also influenced by the user's share of the system, the interaction between shares, resource consumption, and the passage of time. This process scheduling is the essence of the fair-share scheduling process.

4.3.8 Fair-share Limits Node (Lnode)

When a user enters the system (through NQS, `cron`, `rsh`, or interactive access) or exits the system, the system tracks that user's usage as follows:

1. The system creates a user limits structure called an *lnode* (limits node) in kernel memory. In Share by User mode, there is an lnode for each active user on the system; in Share by Account mode, there is an lnode for each active account ID (shareholder). System usage from the relevant UDB entry is passed to the kernel by the `setshares` () routine, which calls the `limits`(2) system call.

2. The time elapsed since the user last exited the system is determined, and the user's or shareholder's usage is aged by this last exit time. The fair-share scheduler uses this calculated decayed usage in the adjustment of process priorities. The decay rates can be configured by the system administrator; see `shradmin`(8) for more detail.

3. If necessary, the system creates lnode entries representing the ancestors (controlling levels in the fair-share hierarchy) of the user's lnode.

4. Once the lnode chain has been created and the user's login shell has been spawned (for both interactive and batch usage), all subsequent processes of the session reference (*are attached to*) the terminal lnode. However, in Share by Account mode the `newacct`(1) command can be used to attach a process to a different lnode.

5. The fair-share daemon, `shrdaemon`(8), updates the user database every 60 seconds (by default). Information from any lnode structures that have no attached process table entries is recorded; these lnodes are subsequently released. When the lnode structure is removed (if a user or group has no running processes), `shrdaemon` records the exit time in the `shextime` field of the user's UDB entry, and records the usage in the `shusage` field.

6. To prevent loss of current usage information stored in the lnode (for example, because of a system interrupt), `shrdaemon` writes all the active lnode structures to the checkpoint file `/etc/lnodes.chkpt` every 5 minutes.
7. In user-level fair-share mode, the scheduling policy algorithms are applied to lnode data by the fair-share daemon rather than by the kernel.

4.3.9 Fair-share and NQS

Fair-share data can also be used by the Network Queuing System (NQS) to select batch jobs for execution. This makes it possible for the fair-share scheduler to influence both queued and running jobs. The amount of influence fair-share has on NQS job scheduling is established by NQS scheduling parameters.

Sites running fair-share NQS should read the following section on fair-share setup and administration ("Using the fair-share scheduler," Section 4.4, page 202). For information on start-up and administration of fair-share NQS, see the `qmgr`(8) man page and the UNICOS NQS and NQE Administrator's Guide, publication SG-2305.

4.3.10 Fair-share and URM

On a system using the fair-share scheduler, the Unified Resource Manager (URM) uses share values in its priority calculations for NQS job initiation recommendations. During the batch job ranking phase, if the fair-share weighting factors for machine share, usage, and share entitlement are nonzero values, URM computes a fair-share priority value based on effective share and past usage. This priority is combined with other scheduler weighting factors to determine the job selection order.

For more information on URM setup and administration, see "Unified Resource Manager (URM)," Chapter 8, page 353.

4.4 Using the Fair-share Scheduler

The following sections describe how to set up, activate, and monitor the fair-share scheduler, including the following tasks:

- Setting up a fair-share hierarchy
- Creating resource groups

- Allocating shares to users
- Setting up Share by Account mode
- Setting up system UDB entries
- Activating the fair-share scheduler
- Modifying fair-share scheduler settings
- Enabling resource group administrators
- Disabling the fair-share scheduler
- Monitoring the fair-share scheduler

The fair-share scheduler has two modes of operation: Share by User and Share by Account. In the default Share by User mode, the fair-share scheduler calculates priorities and costs for each active user. In the optional Share by Account mode, fair-share calculates priorities and costs for each active account ID. If a user works on different projects, Share by Account mode allows that user to switch between projects, which are set up as accounts, by using the `newacct`(1) command. This allows the user to work under a different set of fair-share priorities for each project.

The steps to set up and activate Share by Account mode are similar to those for Share by User mode, but there are some significant differences. The hierarchy of resource groups can be set up the same way. In addition, you must create UDB entries for each valid shareholder, or account ID, and assign share allocations to them. The following sections describe the procedures for setting up Share by User mode. See "Setting up Share by Account mode," Section 4.4.5, page 211, for specific information on setting up this optional share allocation method.

4.4.1 Setting up a Fair-share Hierarchy

This section describes how to establish a fair-share hierarchy. An example hierarchy is used throughout the section for demonstration purposes; note that actual division of resources on Cray Research systems will differ for each installation.

Although using a multilevel fair-share hierarchy is optional, the process of share allocation is simplified by organizing users in resource groups. In Share by User mode, the fair-share hierarchy has users at the end of the resource group chain; in Share by Account mode, the hierarchy has projects or groups, also known as *account IDs*, at the end of the chain.

Resource group chains are created for important divisions of users at the site (for example, different divisions, projects, or physical locations). As an example, site ACME divides UNICOS resources into three categories for three different projects: Projects 1, 2, and 3. Each of these projects must be assigned the appropriate proportion (share) of the system. In addition, system maintenance functions must receive enough resources to accomplish their tasks. The following figure shows the desired division of resources:

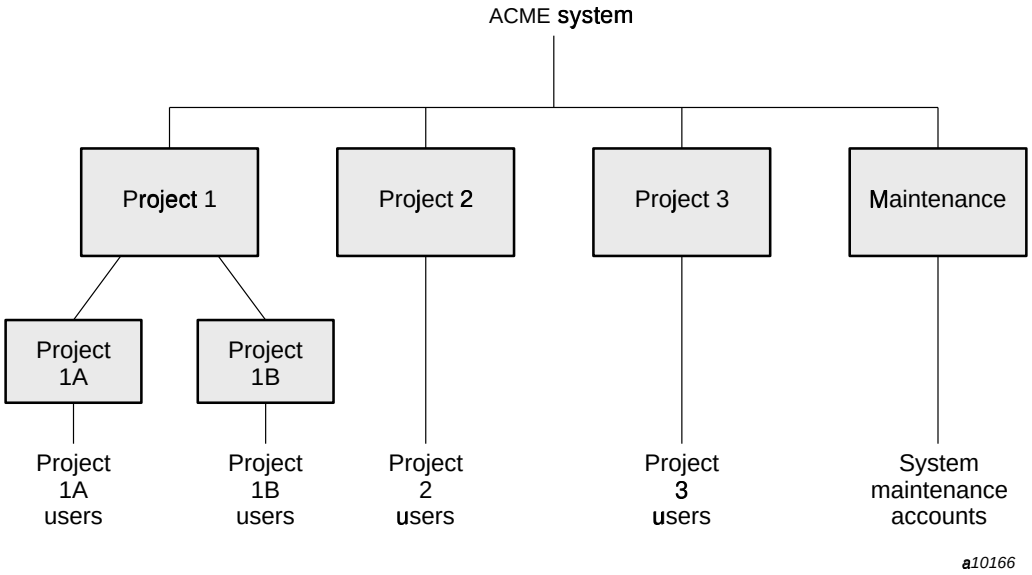

Figure 5. Example of system resource division for fair-share

Four levels of hierarchy are available by default (the number of levels may be changed by using `shradmin -G`). The first level is permanently assigned to `root`. Additional levels of resource groups can be defined by creating a resource group that references its parent resource group in the UDB.

Note: Users or shareholders must not be connected directly to the root lnode. There must be at least one level of resource groups above the user or shareholder level. When this rule is not followed, the `shrtree`(8) command issues a warning message about referencing `ROOT` directly.

When designing a fair-share hierarchy, keep it as simple as possible. You can make later refinements to the hierarchy as needed without having to change other fair-share parameters. For information on how to make UDB or hierarchy

changes on a running system, see "Modifying fair-share scheduler settings," Section 4.4.7, page 215. For information on how to display the fair-share hierarchy, see "Using the `shrtree`(8) command," Section 4.4.11.3, page 222.

4.4.2 Creating Resource Groups

The first step in creating a usable fair-share hierarchy is to create UDB entries for the resource groups. All resource groups must have an entry in the UDB. (Use the `udbgen`(8) command to create these entries.) The following guidelines apply to resource groups:

- Each resource group must have a unique user ID (UID). It is helpful to use a unique range of UIDs so the resource groups are easy to distinguish. To prevent accidental logins for the resource group's UID, do not specify a password (the default encrypted password is the character "*").
- Each resource group must be assigned shares relative to the shares allocated to other resource groups at the same level of the hierarchy. A common technique would be to apportion the shares among resource groups at the same level such that the total number of shares adds up to 100 or 1000 (for example, 50-50 or 700-200-100); however, any numbers can be used.
- Resource groups must have their `shflags` field in the UDB initialized to 040000. (The fair-share scheduler now enforces correct usage of the fair-share fields in the UDB.)
- For each resource group, the `resgrp` field indicates the UID of the controlling resource group (that is, the resource group above it in the hierarchy chain). Resource groups can belong to other resource groups, as explained on Section 4.3.1, page 194. Top-level resource groups (chain leaders) should have this field set to the `root` UID, which is always 0.

Because resource groups are defined in the UDB with the ordinary user entries and shareholders (account IDs), it is recommended that you establish a naming scheme to help distinguish resource group entries from user and shareholder entries and to avoid use of identical names. One convention that works well is to capitalize the first character of the names of the resource groups.

> **Note:** A resource group entry for a system group is required for the system UDB entries such as `root`, `cron`, and system daemons. This group is often called `system` or `admin` (group `Maint` is used at example site ACME).

For the example site, the following project names are used as resource group chain leaders: `Proj1`, `Proj2`, and `Proj3`. Table 30 contains the project names, their desired proportion of the system, and their equivalent share value. A

separate share of the system has also been reserved for maintenance (`Maint`). A total share value of 1000 is distributed among the projects in the desired proportion.

Table 30. Share division among resource groups

Project	Proportion	Share value
`Maint`	10%	100
`Proj1`	50%	500
`Proj2`	25%	250
`Proj3`	15%	150
Total	100%	1000

The UDB entries for these resource group chain leaders can be created by using `udbgen`(8) with the following directives:

```
create:Maint:uid:999:gid:10:passwd:*:shflags:040000:shares:100:resgrp:0:
create:Proj1:uid:111:gid:20:passwd:*:shflags:040000:shares:500:resgrp:0:
create:Proj2:uid:222:gid:30:passwd:*:shflags:040000:shares:250:resgrp:0:
create:Proj3:uid:333:gid:40:passwd:*:shflags:040000:shares:150:resgrp:0:
```

A `comment` field can be used to include an explanation for the UDB entries, but that was not done in this example. The value used in the `gid` field is a site-dependent group ID (GID); at least one GID must be specified, or warning messages will occur. The `passwd` field has been set to the character "*" to ensure that no one is able to log in to a resource group account. Notice the `shflags` value of 040000. This value marks these entries as resource groups. Also note that the `shares` field contains the share value numbers from Table 30.

Additional levels of resource groups can be defined by creating a resource group, as in the first example, including the name or user ID of the parent resource group in the `resgrp` field. (Four levels are available by default; use the `shradmin -G` command to increase this amount as desired.) For example, site ACME would subdivide resource group `Proj1` into two resource groups, `Proj1A` and `Proj1B`, using the following `udbgen` directives:

```
create:Proj1A:uid:444:gid:20:passwd:*:shflags:040000:shares:500:resgrp:111:
create:Proj1B:uid:555:gid:20:passwd:*:shflags:040000:shares:500:resgrp:111:
```

Each new resource group now has 500 shares, or, half the resources of `Proj1` (1000 was used as the total share value for this level). The `resgrp` field is set to the UID of `Proj1` (111) to mark these resource groups as shareholders of `Proj1`.

4.4.3 Allocating Shares to Users

The next step in setting up fair-share allocations is to decide which users belong to each resource group and assign them to the appropriate group with an appropriate share value. Within each resource group, the share values are relative to each other. A convenient total share value, such as 100 or 1000, will make the job of assigning proportional shares easier. However, any number can be used for the total share value; only the relative values are important.

The allocation of shares in the UDB should occur as follows:

1. If Share by User mode will be enabled, set the resource group of each user (the `resgrp` field) to the name of that user's controlling resource group.
2. If Share by Account mode will be enabled, ensure that each user has a list of valid account IDs in the acids list (the `acids` field) of the UDB; at least one entry is required in this list. Fair-share uses the first account ID in the list as the user's default, or initial, account. (See "Setting up Share by Account mode," Section 4.4.5, page 211, for more information.)
3. Allocate each user and shareholder entry in the UDB a number of shares (in the `shares` field). The exact number is not critical, but it is convenient to pick a value that could later be adjusted up or down. (For example, each user could be allocated 100 shares.)

You can use the `udbgen` command to analyze share resource assignments in the UDB and report any problems. For Share by User mode, use the following command to analyze the default UDB in the `/etc` directory:

```
udbgen -a -R
```

For Share by Account mode, use the following command to analyze share resource groups based on the `acids` field instead of the `resgrp` field:

```
udbgen -a -A
```

Table 31 shows the division of shares for the example site.

Table 31. Share division within resource groups

Group	User	Proportion	Share value
Maint	u1	25%	25
	u2	25%	25
	u3	25%	25
	u4	25%	25
Proj1			
Proj1A	u5	25%	50
	u6	25%	50
Proj1B	u7	25%	50
	u8	25%	50
Proj2	u9	50%	50
	u10	30%	30
	u11	10%	10
	u12	10%	10
Proj3	u13	25%	25
	u14	25%	25
	u15	25%	25
	u16	25%	25

These entries show, for each resource group, the users in the group ("User" column), the percentage of resources each user is to have within the group ("Proportion" column), and the individual share value equivalent to the proportion ("Share value" column). For each group, a total share value of 100 was used.

The following `udbgen` directives update the UDB to reflect this division of resources (only Share by User mode will be enabled):

```
update:u1:resgrp:999:shares:25:
update:u2:resgrp:999:shares:25:
update:u3:resgrp:999:shares:25:
update:u4:resgrp:999:shares:25:
update:u5:resgrp:444:shares:50:
```

```
update:u6:resgrp:444:shares:50:
update:u7:resgrp:555:shares:50:
update:u8:resgrp:555:shares:50:
update:u9:resgrp:222:shares:50:
update:u10:resgrp:222:shares:30:
update:u11:resgrp:222:shares:10:
update:u12:resgrp:222:shares:10:
update:u13:resgrp:333:shares:25:
update:u14:resgrp:333:shares:25:
update:u15:resgrp:333:shares:25:
update:u16:resgrp:333:shares:25:
```

The `resgrp` field for each entry is set to the UID of the resource group (`Maint` is `999`, `Proj1A` is `444`, `Proj1B` is `555`, `Proj2` is `222`, and `Proj3` is `333`). The specified resource group and user entries must already exist in the UDB before these `update` directives can be executed.

4.4.4 Setting up System UDB Entries

System UDB entries such as `Idle`, `root`, and system daemons have special requirements for the resource group and shares fields. This section describes the following accounts:

- `Idle` account
- `UnKnown` or `unknown` account
- Other system accounts (`root`, `cron`, system daemons, and so on)

4.4.4.1 Idle Account

A special account, called `Idle`, is required for the fair-share scheduler to work correctly. (During initial installation of the UNICOS operating system, the UDB initialization process sets up a correct `Idle` entry by default.) The following guidelines exist for the `Idle` account:

- The UID for the `Idle` account (the `uid` field in the UDB) must be 11.
- The `Idle` account must be assigned no shares (the `shares` field must be omitted or set to 0).
- The controlling resource group (the `resgrp` field) must be 0, which is the UID of `root`.
- The `acids` field must be set to a null value (that is, `acids::`).

- The `shflags` field must be 0. (The fair-share scheduler now enforces correct usage of the fair-share fields in the UDB.)

The following example shows a sample `Idle` account for site ACME:

```
Idle:uid:11:comment:System Idle:passwd:*:gids:0:acids::resgrp:0:shflags:0:
```

Note: It is critical to the proper operation of the system that only the `Idle` entry be allocated zero shares in the UDB.

4.4.4.2 UnKnown or Unknown Account

The UDB must contain an entry called `UnKnown` or `unknown`; this entry is used when no valid UDB entry for a user can be found. You must add this entry to the UDB before enabling the fair-share scheduler.

The following example shows a sample `unknown` account for site ACME:

```
unknown:uid:12:passwd:*:gids:0:acids::resgrp:999:shflags:0:shares:1:
```

4.4.4.3 Other System Resource Accounts

Each system account requires a resource group entry in the UDB. This includes entries for `root`, `cron`, the operator, NQS, system daemons, and other system user IDs. These system UDB entries must be assigned to the system or administrator resource group (often called `system` or `admin`; group `Maint` is used at example site ACME). In addition, if Share by Account mode will be enabled, the `acids` field of each system entry should be set to the account ID of the system or administrator shareholder.

For initial installations of the UNICOS operating system the UDB skeleton file creates a default list of resource group and user accounts in the UDB, and sets the `resgrp` field to 0 (`root`).

The UDB skeleton file is located in the file `/usr/src/skl/c1/etc/initudb`. It sets up the following accounts: `root`, `sync`, `bin`, `sys`, `adm`, `cron`, `nqs`, `daemon`, `operator`, `ce`, `Idle`, `unknown`, `osi`, and `nobody`.

Note: Upgrading sites should verify their UDB system account entries against the UDB skeleton file.

To enable a fair-share hierarchy, you must add the UDB entry for a system or administrator resource group and set the `resgrp` field of the system accounts to the UID of this resource group. The `resgrp` field is used to link the resource tree of the hierarchy.

If Share by Account mode will be enabled, the `acids` field of the `root` UDB entry should be set to the UID of the system maintenance or administration resource group (for example, 999 at site ACME specifies the `Maint` resource group).

4.4.5 Setting up Share by Account Mode

By default, the fair-share scheduler runs in Share by User mode, in which a user's relative priority is determined by the `resgrp` field in the UDB. However, you can activate the optional Share by Account mode by using the `shradmin`(8) command to set the `SHAREBYACCT` scheduling flag. In this mode, fair-share associates and sets priorities based on the shareholder, or account ID (`acids` field in the UDB). The initial association is based on the default account ID. This is the first ID in the list of valid account IDs for a user in the UDB entry.

Administrators have the option of assigning shares to users (Share by User mode) or accounts (Share by Account mode). If a user works on different projects, Share by Account mode allows that user to switch between projects, which are set up as accounts, by using the `newacct`(1) command. This allows the user to work under a different set of fair-share priorities for each project.

Enabling Share by Account mode is similar to enabling Share by User mode. The hierarchy of resource groups is set up the same way. However, the following additional administration tasks are necessary to set up Share by Account mode:

- Create a UDB entry for each valid account. It is helpful to use a unique range of UIDs so the account IDs are easy to distinguish. The UIDs for account ID entries (shareholders) must not overlap with resource group or user entries in the UDB.
- Set the `shflags` field for each account ID entry to 01000000. (The fair-share scheduler now enforces correct usage of the fair-share fields in the UDB.)
- Assign share allocations to each valid account ID entry.
- Assign a passwd of "*" to each account ID entry so no user can log in as a resource group.
- Verify that each account ID entry has been added to the `/etc/acid` file before `udbgen`(8) is executed.
- Set the `resgrp` field of each account ID entry in the UDB to the appropriate resource group entry. The `resgrp` field is used to link the resource tree of the hierarchy.

- Specify the first account ID in the `acids` field of each user entry as the name of the user's default account ID. Each user must have at least one account ID in this field; enter a comma-separated list of account IDs for users who can change to different accounts.
- Verify that the `Idle` UDB entry specifies a null account ID field; that is, the `acids` field should be omitted or set to `acids::`.
- Set the `acids` field for the `root` UDB entry to the UID of the system maintenance or administration resource group (for example, `999` at site ACME specifies the `Maint` resource group).
- If there are more than four levels in the fair-share hierarchy, including account ID entries, use the `shradmin -G` command to increase the number of hierarchy levels.

4.4.6 Activating the Fair-share Scheduler

To activate the fair-share scheduler during system startup, the system configuration script must be modified to execute the `shradmin`(8) command with the appropriate options and start up the fair-share daemon, `shrdaemon`(8). (To change and reactivate fair-share on a running system, see "Modifying fair-share scheduler settings," Section 4.4.7, page 215.)

There are two methods to modify the system configuration script: using the UNICOS Installation and Configuration Menu System, or editing the file `/etc/config/daemons`.

If you are using the menu system, access the following menu to make this change:

```
Configure System ->
        System daemons configuration ->
                System daemons table
```

If you are not using the menu system, access the file `/etc/config/daemons` to make this change. An example of `/etc/config/daemons` is as follows:

```
# /etc/config/daemons excerpt
# group tag        start   kill      pathname      arguments
  SYS1  share      YES     *        /etc/shradmin   options
  SYS1  share      YES     *        /etc/shrdaemon
```

These lines control initiation of `/etc/shrdaemon` (the fair-share daemon) and the administrator command `/etc/shradmin`, which activates the fair-share scheduler using the options described in the following paragraphs.

Note: The `/etc/shradmin` command must be run before the `/etc/shrdaemon` command.

4.4.6.1 Setting Scheduling Options and Flags

The following `shradmin` options are important for enabling fair-share functions. (See the `shradmin`(8) man page for a complete description of all options; see "Tuning the fair-share scheduler," Section 4.6, page 227, for information on performance impact of `shradmin` values.)

Option	Description	
`-F`	(*flags*) Sets the fair-share control flags in the kernel `sh_consts` structure. The following flags are available:	
	`NOSHARE`	(001) Turns off the fair-share scheduler. Leaves accumulated charges in the UDB unless they are cleared by the system administrator.
	`ADJGROUPS`	(002) Specifies adjustments by group IDs (group share allocation).
	`LIMSHARE`	(004) Specifies limits on maximum share.
	`SHAREBYACCT`	(010) Specifies Share by Account mode.
	`NOSCHED`	(020) Gathers fair-share charges and usage information, but does not use these values for CPU scheduling.
	`USRLEVLFSS`	(0100) Specifies *user-level fair-share mode*; in this mode, the fair-share daemon (`shrdaemon`(8)) performs the share calculations and updates the lnodes. This will be the default mode in future releases of the UNICOS operating system.
`-K`	(*half-life*) Establishes the length of time that past usage of resources are remembered. Longer decay rates cause the	

scheduler to distribute resources fairly over a longer period of time. This option can have a significant impact on interactive responsiveness; see "Tuning the fair-share scheduler," Section 4.6, page 227, for more information.

Note: Never set the `-K` option to a value lower than 1 hour (3600 seconds).

`-R` (*delta*) Sets the major fair-share adjustment cycle to the specified value. This means that once every cycle, the resource usage of all users and groups active on the system is evaluated, and, if the `ADJGROUPS` flag is set, the group hierarchy is evaluated to determine what adjustments are necessary to achieve proper group sharing.

`-t` (*tick*) Sets the cost per tick. The usage field of the user owning that process is increased by this amount.

`-X` (*maxushare*) Limits how much past usage can be accrued; it is used only when the `LIMSHARE` flag is set. See "Tuning the fair-share scheduler," Section 4.6, page 227, for more information.

`-Y` (*mingshare*) Specifies the deviation between group share allocation and the actual rate of resource consumption allowed before the scheduling algorithm tries to compensate. It has an effect only when the `ADJGROUPS` flag is set. See "Tuning the fair-share scheduler," Section 4.6, page 227, for more information.

`-Z` (*sharemin*) Sets a minimum allocation of machine shares for any active lnode (user or account ID). See "Tuning the fair-share scheduler," Section 4.6, page 227, for more information.

An example of ACME's `/etc/config/daemons` is as follows:

```
# /etc/config/daemons excerpt
# group tag    start  kill  pathname        arguments
  SYS1  share  YES    *     /etc/shradmin   -t100 -F06 -K3600s -X3.0 -Y.85 -Z .002
  SYS1  share  YES    *     /etc/shrdaemon
```

The `-t` option sets the cost per tick to 100. The `-F` option enables Share by User mode by setting the fair-share control flags; that is, the `ADJGROUPS` and `LIMSHARE` flags are set, while the `NOSHARE` flag is cleared. The `-K` option sets the usage decay rate to a half-life of 3600 seconds. The `-X` option sets *maxushare* to 3.0, which decreases the possibility that users with very low past usage can monopolize the CPU. The `-Y` option sets *mingshare* to .85 to decrease the deviation between group share allocation and the actual rate of resource

consumption. The `-Z` option sets *sharemin* to .002, which guarantees each user or shareholder a minimum machine share of 0.2%.

4.4.7 Modifying Fair-share Scheduler Settings

Elements of fair-share configuration that can be changed on a running system include the following:

- Maximum levels in the fair-share hierarchy
- Distribution of shares within resource groups and account ID shareholders
- Existing UDB entries, such as the `acids`, `resgrp`, or `shares` fields
- Addition or deletion of UDB entries
- All `shradmin`(8) options (tuning parameters)

Use the `shradmin -G` command to change the maximum number of levels for the fair-share hierarchy (there are four levels by default). See the `shradmin`(8) man page for more information.

For information on allowing other users to administer share distribution within a resource group, see "Enabling resource group administrators," Section 4.4.8, page 216.

Use the `udbgen`(8) command to analyze share resource assignments in the UDB and report any problems. For Share by User mode, use the following command to analyze the default UDB in the `/etc` directory:

```
udbgen -a -R
```

For Share by Account mode, use the following command to analyze share resource groups based on the `acids` field instead of the `resgrp` field:

```
udbgen -a -A
```

See the `udbgen`(8) man page for more information.

To make and enable share allocation changes on a running system, use the `shrdist`(8) command to make changes. Users do not have to log out in order for these changes to take effect. (To make substantial changes, you can also use the `udbsee`(1), `udbgen`(8), and `shrdist`(8) commands; see the man pages for more information.)

Note: In order to change from Share by User mode to Share by Account mode (or vice versa), it is recommended that the system be brought to single-user mode, then restarted with the appropriate `shradmin` options. Changing between fair-share modes on a running system can cause unpredictable results for priority calculations.

If your site will alternate between fair-share modes, ensure that each user entry in the UDB sets both the `resgrp` field and the `acids` field to the appropriate values. See "Setting up Share by Account mode," Section 4.4.5, page 211, for more information on setting these fields.

4.4.8 Enabling Resource Group Administrators

System administrators may find it useful to set up separate administrators for each resource group. This capability allows the owner of a resource group (for example, the project manager) to modify share allocations within the resource group without requiring system administrator intervention to do so. For example, if project A changes priority over time, or project B needs more shares towards the end of the month, a resource group administrator could make the necessary changes without system administrator intervention. This is done by setting up the `shrdist.auth` file, which enables the listed resource group administrators to reallocate shares within the group.

To redistribute shares within a resource group, use the `shrdist`(8) command. For this command to work properly, you must create an authorization file called `/etc/shrdist.auth`. The `shrdist.auth` file consists of two fields: the login name of the resource group owner and the name of the resource group or groups.

The following lines from a `shrdist.auth` file allow user `mlb` to administer shares for two resource groups, `Mktg` and `tng`. If a resource group is not specified (with the `-g` option), the first match (in this case, `Mktg`) is used.

```
mlb     Mktg
mlb     tng
```

The `shrdist`(8) command can be used to adjust shares. The `shrdist` command operates in both batch and interactive mode. To reallocate share values in interactive mode, perform these steps:

1. Use the following command to access the share allocation database:

   ```
   /etc/shrdist
   ```

2. Position the curser at the rightmost digit of the share allocation field for the desired account and enter the new values.
3. Enter `u` to update the share allocation database.
4. Enter `q` to exit the `shrdist` command.

You can also perform updates in batch mode by using the `shrdist -b` command. However, this method only allows you to update one account for every execution of the `shrdist` command. Test mode is also supported (`shrdist -p`). This capability is similar to test mode for the `udbgen`(8) and `udbsee`(1) commands.

4.4.9 Disabling the Fair-share Scheduler

To turn off the fair-share scheduler after it has been enabled, perform the following steps:

1. Execute the following `shradmin` command:

   ```
   shradmin -F 021
   ```

   This sets the `NOSHARE` flag, which specifies that no fair-share scheduling is done. It also sets the `NOSCHED` flag, which prevents the marooning of running processes by ignoring the `shcharge` and `shusage` information in the CPU scheduling algorithm.
2. Wait for at least 1 minute. This delay allows `shrdaemon` enough time to update the fair-share usage and charge information in the UDB.
3. Find the process ID (PID) of the fair-share daemon, `shrdaemon`, as follows:

   ```
   ps -efl | grep shrdaemo
   ```

   (The `ps` command truncates command names at 8 characters.)
4. Disable updating of fair-share information in the UDB by stopping the `shrdaemon` process with the `SIGTERM` signal, as follows:

   `kill -27` *PID_of_shrdaemon*

Turning off the fair-share scheduler in this manner leaves all fair-share usages and charges in the UDB. If a clean UDB is desired (for example, at the beginning of a new fiscal year), perform the following steps:

1. Use the udbsee(1) command to create an ASCII version of the UDB fair-share usage and charge fields, shusage and shcharge, and save this output to a file, as follows:

   ```
   udbsee -a -fupdate,uid,resgrp,shares,shusage,shcharge > tempfile
   ```

   The uid, resgrp, and shares fields will not be changed, but they may be useful as a reference during this operation.

2. Clear the fair-share information by editing *tempfile* as follows: replace the values in the shrusage and shcharge fields with 0.

3. Bring the system to single-user mode. It is important that the next step be performed when no other process is updating the UDB.

4. Use the udbgen command with the edited file, tempfile, to update the UDB with the changes removing the fair-share information.

5. Return the system to multiuser mode.

4.4.10 Costs, Usage, and Background Users

Watch the relationship between costs, usage, and the MAXUSAGE value. (MAXUSAGE is set by the shradmin -U command to the upper bound for reasonable usages; the default is a very large number.) If the cost factors are set high enough that a user's usage accumulation rises more quickly than the decay rate can bring it down, and the usage accumulation exceeds MAXUSAGE, then that user is put into the background user category by the fair-share scheduler. This situation can be controlled by either raising MAXUSAGE with the shradmin -U command, or by adjusting the following cost factors in a relatively uniform downward direction: bio (block I/O operations), tio (stream I/O operations), click (memory ticks), syscall (system calls), and tick (CPU ticks).

The tick cost, which is charged to the user up to 60 times per second, is usually the cost that causes users' usages to rise very rapidly. A background user can be recognized as running at priority 996 in a ps(1) display.

Processes that have been assigned a nice value of 39 are treated as special by the fair-share scheduler. They run at priority 997 and are not charged for CPU use. Processes attached to lnodes with 0 shares run at priority 998.

See "Tuning the fair-share scheduler," Section 4.6, page 227, for more information on fair-share costs and nice values, or for information on increasing the default MAXUSAGE value.

4.4.11 Monitoring the Fair-share Scheduler

Three commands display information about what is currently happening in a fair-share system: `shrview`(1), `shrtree`(8), and `shrmon`(8). The `shrview`(1) command provides the most functionality; it serves as a single interface for the functions of the other fair-share display commands. The `shrtree`(8) command displays the fair-share hierarchy. The `shrmon`(8) command is intended for use by the administrator; it is located in the `/etc` directory.

Note: The `shrmon`(8) command will not be available in future releases of the UNICOS operating system. Its functionality is replaced by the `shrview`(1) command.

All fair-share display commands can be executed in line mode, repeat mode, or continuous mode. In each mode, the time period used as a delay to collect information, or between successive display updates, is the share cycle time set by the `-R` option of the `shradmin`(8) command.

In line mode, the command collects information and prints its findings to standard output (`stdout`). In repeat mode, the command behaves as if in line mode and repeats the cycle as many times as specified with the `-r` option to `shrview`, separating the outputs with a form-feed character. In continuous mode, the command uses the `curses`(3) library capability and produces a full-screen display that it updates periodically. For displays involving more data than can be displayed (for example, 40 users on a 24-line terminal screen), only the first screen is displayed.

4.4.11.1 Using the `shriview`(1) Command

The `shrview` command is an integrated tool for displaying information about the behavior and current state of the fair-share scheduler. Many different display options and formats are available. Selection and configuration of displays may be done interactively when in curses mode.

The following sample display shows the `ADJGROUPS` display (`-da` option) of resource groups (`-so` option), organized by group name in alphabetical order (`-oi` option). This display helps quantify system use by resource group; the `Rate%`, `CPU%`, and `Rshr%` columns show that resource group `Xydev` in `SoftDev` is using the largest percentage of the system.

```
% shrview -da -so -oi

  SHRVIEW  Type:adjgroup  Select:only groups  Sort_opt:id

    Name      Rate%   CPU%  Rshr%   Nrun  Rate  Proj%    Adj_a   New%   Pri%
-----------  -----  -----  -----  -----  ----  -----  -------  -----  -----
 CCN           0.00   0.00  26.67   0.00  0.00  20.92     1.00  16.63  15.67
  SysAdm       0.00   0.00  17.78   0.00  0.00  13.82     1.00  10.99  10.35
  Syssup       0.00   0.00   8.89   0.00  0.00   7.10     1.00   5.64   5.32
 Mktg          0.31   0.23  13.33   0.00  0.00  56.65     1.00  45.05   9.99
  Country      0.09   0.00   4.44   0.00  0.00  53.91     1.00  42.87   6.66
  Intl         0.22   0.23   4.44   0.00  0.00   0.83     1.00   0.66   1.66
  TechOps      0.00   0.00   4.44   0.00  0.00   1.91     1.00   1.52   1.66
 SoftDev      97.62  99.77  60.00  65.76 21.48  22.42     1.00  38.31  74.34
  Userint     12.37  21.14  10.00  14.00 12.31   3.73     1.00   2.97   9.05
  Users        3.29   7.95  10.00  11.76  1.94  11.30     1.00  22.09  18.66
   Netdev      1.59   3.86   2.00   2.00  1.00   0.00     1.00   0.00   0.75
  Xydev       81.96  70.68  40.00  40.00  7.22   7.39     1.00  13.25  46.63
```

The following example shows the monitor display (`-dm` option) on a system with two levels of hierarchies.

```
# shrview -dm

Shrview: sampling over 11 seconds starting at Tue Jul 25 12:27:33 1995

  SHRVIEW  Type:monitor  Select:all  Sort_opt:id

    Name      Rate%  Eshr%  Rshr%   Usage(k)    Rate   Muse  Ref  Chld   Flags
-----------  -----  -----  -----  ----------  -----  -----  ---  ----  ------
 CCN          93.84  11.76  11.76  1000000000   1.00      0    0     2   50000
  SysAdm      93.84   7.84   7.84   199667501   0.84 1358396   12     0 1010000
  Syssup       0.00   3.92   3.92   570650974   0.84 489820    1     0 1010000
 Idle          0.00   0.00   0.10  1000000000  17.84     24    2     0   10000
 Mktg          6.16  11.76  11.76  1000000000   1.00      0    0     2   50000
  Country      0.00   5.88   5.88   119543509   0.84  29760    2     0 1010000
  TechOps      6.16   5.88   5.88    41308301   0.84   3652    1     0 1010000
 SoftDev       0.00  52.94  52.94  1000000000   1.11      0    0     4   50000
  Netdev       0.00  13.24  13.24    67308849   0.84  83308    6     0 1010000
  Userint      0.00  13.24  13.24   203251100   1.11 541828   23     0 1010000
  Users        0.00  13.24  13.24   200168203   0.84 1609728   8     0 1010000
 *Xydev        0.00  13.24  13.24   144301956   0.94 236500   19     0 1010000
 System        0.00  23.53  23.53  1000000000   1.00      0    0     1   50000
```

```
  Admin        0.00  23.53  23.53    10437769   0.84 428412    1     0 1010000
```

4.4.11.2 Using the shrmon(8) Command

The fair-share monitor, shrmon(8), produces a columnar formatted output that can be used to monitor most of the more important accumulators and feedback parameters found in the fair-share scheduling node within the kernel.

Note: The shrmon(8) command will not be available in future releases of the UNICOS operating system. Its functionality is replaced by the shrview(1) command.

The following display shows sample shrmon output:

```
$  shrmon -vv
Tue Sep 20 09:06:20 1988
 CPUs charge   rate nrun  %Rate %share  %CPU %rshare kids  ref
 11.2    0.0    1.3   13    0.0  100.0 100.0  100.0   12   59 System
  0.0    0.0    3.8    4    0.0    0.0   0.0    0.0    0    4  Idle
  3.4    0.2    1.3    2    0.0    8.3  30.5   33.3    2    9  Regions
  0.0    0.0    0.8    0    0.0    4.2   0.0    2.0    0    3   poulet
  3.4    0.2    1.4    2    0.0    4.2  30.5   33.3    0    6   ub
  0.0    0.0    0.0    0    0.0    8.3   0.0    2.0    2    2  LibProd
  0.0    0.0    0.8    0    0.0    4.2   0.0    2.0    0    1   saa
  0.0    0.0    0.8    0    0.0    4.2   0.0    2.0    0    1   jam
  7.7    0.3    5.0    5    0.0   16.7  69.4   66.7    3   22  OpSysDev
  0.0    0.0    0.8    0    0.0    5.6   0.0    2.0    0    1   dak
  0.0    0.0    1.0    1    0.0    5.6   0.0    2.0    0    3   sen
  7.7    0.4    5.0    5    0.0    5.6  69.4   66.7    0   20   dws
  0.0    0.0    0.0    0    0.0    8.3   0.1    2.2    4   11  Operatns
  0.0    0.0    0.8    0    0.0    2.1   0.1    2.0    0    3  operator
  0.0    0.0    0.8    0    0.0    2.1   0.0    2.0    0    3   tqa
  0.0    0.0    0.8    0    0.0    2.1   0.0    2.0    0    5   backup
  0.0    0.0    0.8    0    0.0    2.1   0.0    2.2    0    0   ce
  0.0    0.0    0.8    0    0.0    2.8   0.0    2.0    0    1   mab
  0.0    0.0    0.8    0    0.0    2.8   0.0    2.0    0    4   gop
  0.0    0.0    0.8    0    0.0    2.8   0.0    2.0    0    1   aaw
  0.0    0.0    0.0    0    0.0    8.3   0.0    2.0    1    1  Appl
  0.0    0.0    0.8    0    0.0    8.3   0.0    2.0    0    1   wgh
  0.0    0.0    0.0    0    0.0    8.3   0.0    2.0    1    0  IOS
  0.0    0.0    0.8    0    0.0    8.3   0.0    2.0    0    0   stt
  0.0    0.0    0.0    0    0.0    8.3   0.0    2.0    1    2  Services
  0.0    0.0    0.8    0    0.0    8.3   0.0    2.0    0    2   hhj
  0.0    0.0    0.0    0    0.0    8.3   0.0    2.0    1    1  CompDev
```

```
0.0    0.0    0.8    0    0.0    8.3   0.0    2.0    0    1   plu
0.0    0.0    0.0    0    0.0    8.3   0.0    2.0    1    1  Diag
0.0    0.0    0.8    0    0.0    8.3   0.0    2.0    0    1   ljl
0.0    0.0    0.0    0    0.0    8.3   0.0    2.0    1    0  Netwrkng
0.0    0.0    0.8    0    0.0    8.3   0.0    1.9    0    0   jyj
```

The most important factors in this display are `%rshare` and `%CPU`. The `%rshare` factor is the machine share of the system, expressed as a percentage, to which this particular user or group is entitled based on the current active users and the fair-share hierarchy defined in the UDB. The `%CPU` factor is the percentage of CPU resources allocated to the lnode over the sampling interval.

Interactive users, who are relatively idle at the time of the sample, each hold a 2.0% `rshare`.

Note that the fair-share hierarchy in this example involves two levels: group levels have names beginning with a capital letter; and users are identified by lowercase login names.

4.4.11.3 Using the `shrtree`(8) Command

The `shrtree`(8) command displays the share tree, or fair-share hierarchy, that is defined in the UDB, and highlights problems that could prevent logging in to the system or submitting jobs. Any problems in the UDB that would affect the fair-share scheduler are marked in the `shrtree` output. (Most problems can prevent logging in and submitting jobs for the affected users.)

The `shrtree` command displays such useful statistics as the group share and usage value from the UDB. This command also displays a general approximation of share distribution by displaying a static analysis of what the relative entitlements would be if all the users in the UDB were logged on at the same time.

The following display shows sample output for `shrtree` with no options (short form report). The system is running Share by User mode, with a maximum of four levels in the fair-share hierarchy. In this example, the `Serv` entry has warning flag `Nc`; `Unknown` has warning flag `Zs`; and `Country`, `Region`, and `TechOps` have flags `Nc` and `Zs`.

```
$ shrtree

DISPLAY OF SHARE TREE

_________________________
UDB path:      DEFAULT
```

```
Analyzed:      By UID
Format:        Groups only
Maxgroups:     4
Node:          ALL
Group Count:   15
Account Count: 0
User Count:    1370
Warnings:      9
Errors:        0

Warning Count: 4     (Nc) Group has no references
Warning Count: 1     (Zs) User has zero shares
Warning Count: 4     (Zs) Group has zero shares
```

Lv	Name	ID	System Share	Group Share	System Usage	Status	Flags
0	_ROOT_	0	100.0%	100.0%	100.0%	G	40000
1	Demos	8367	100.0%	100.0%	0.0%	G	40000
2	Serv	8001	100.0%	100.0%	0.0%	G|Nc	40000
1	System	8389	0.0%	0.0%	0.0%	G	40000
2	Admin	8306	0.0%	2.9%	0.0%	G	40000
1	Unknown	8393	0.0%	0.0%	0.0%	G|Zs	40000
1	Users	8395	0.0%	0.0%	28.8%	G	40000
2	CCN	8354	0.0%	4.8%	0.2%	G	40000
2	Country	8359	0.0%	0.0%	0.0%	G|Nc|Zs	40000
2	HardDev	8372	0.0%	4.8%	0.0%	G	40000
2	Intl	8374	0.0%	14.3%	0.0%	G	40000
2	Mktg	8381	0.0%	28.6%	0.1%	G	40000
2	Region	8385	0.0%	0.0%	0.0%	G|Nc|Zs	40000
2	SoftDev	8386	0.0%	47.6%	28.5%	G	40000
2	TechOps	8390	0.0%	0.0%	0.0%	G|Nc|Zs	40000

The following example shows the error display for the same system (`shrtree` with the `-e` option). Only the entries with errors are displayed.

```
$ shrtree -e

DISPLAY OF SHARE TREE
_____________________
UDB path:      DEFAULT
Analyzed:      By UID
Format:        Groups only
```

```
Maxgroups:     4
Node:          ALL
Group Count:   15
Account Count: 0
User Count:    1370
Warnings:      9
Errors:        0

Warning Count: 4     (Nc) Group has no references
Warning Count: 1     (Zs) User has zero shares
Warning Count: 4     (Zs) Group has zero shares

  Type      Name    ID    Status  Description
_________ ________ _____ ________ ________________________________________
**WARN*** Serv      8001       10 Nc: Group has no references
**WARN*** Unknown   8393     1000 Zs: Group has zero shares
**WARN*** unknown     12     1001 Zs: User has zero shares
**WARN*** Country   8359     1010 Nc: Group has no references
**WARN*** Country   8359     1010 Zs: Group has zero shares
**WARN*** Region    8385     1010 Nc: Group has no references
**WARN*** Region    8385     1010 Zs: Group has zero shares
**WARN*** TechOps   8390     1010 Nc: Group has no references
**WARN*** TechOps   8390     1010 Zs: Group has zero shares
```

4.5 Customizing the Fair-share Scheduler (User Exit)

You can customize the fair-share scheduler's CPU scheduling policy at your site. This allows use of a different scheduling algorithm without modifying the UNICOS kernel.

To customize the scheduling policy, you provide a user exit routine and set the `USRLEVLFSS` flag with the `shradmin`(8) command. The fair-share daemon, `shrdaemon`(8), includes a stub routine (in the module `shrsiteux.c`) that has an entry point named `site_adjust_lnodes`. After reading the lnodes from the kernel and performing the calculations indicated by the share flags, this user exit is invoked before writing the lnodes back to the kernel.

The following calling sequence is defined in the include file `sys/lnode.h`:

```
site_adjust_lnodes( &Lnodes[0], count, Traceflag)
```

The arguments of `site_adjust_lnodes` have the following meanings:

Argument	Description
`Lnodes`	This argument is a pointer to the lnode table in the `shrdaemon`(8) memory; it is of type `struct kern_lnode *Lnodes`. All the address fields in the lnode table can be used for scanning, because `shrdaemon` converts them to `shrdaemon` memory addresses before performing any analysis.
`count`	This parameter specifies the number of lnodes in the lnode array; it is of type `int count`.

`Traceflag`	This parameter is of type `int Traceflag`; it can be used to specify logging, as in the following example:
	`if (Traceflag)`
	`fprintf(stderr, "Adjusting lnodes\n").`

4.5.1 Fair-share User Exit Example

As a simple example of a user exit, the following routine allows UID `104` to use as many CPU resources as necessary. This capability allows a system daemon to run under a different UID than `root` (some sites prefer to do this for accounting reasons), but removes the resource limitations that are normally enforced for all non- `root` UIDs.

```
static char USMID[] = "@(#)man/2302/04.fair.share       92.1    12/11/95 16:48:26";

/*
 *      (C) COPYRIGHT CRAY RESEARCH, INC.
 *      UNPUBLISHED PROPRIETARY INFORMATION.
 *      ALL RIGHTS RESERVED.
 */

#include <stdio.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/param.h>
#include <sys/lnode.h>
#include <sys/share.h>
/*
```

```
 *      site_adjust_lnodes()
 *
 *      User exit stub to allow sites to perform any calculations
 *      desired upon lnode data before the shrdaemon writes them
 *      back to kernel memory.
 *
 *      A ptr to the lnode table in the shrdaemon memory is passed
 *      in; all the address linkages have been converted to shrdaemon
 *      memory addresses.
 */

void
site_adjust_lnodes(struct kern_lnode *Lnodes, int count, int Traceflag)
{
        int i;
        struct kern_lnode *lp;

        /*
         * Site can now perform any calculations desired on lnode data.
         */

        for (lp = Lnodes, i = 0; i < count; i++, lp++) {
            if (lp->kl.l_uid == 104) {
                /* This user is special, must never starve for cpu. */
                lp->kl.l_usage = 2.0;
                lp->kl_usage = 2.0;

            }
                if (Traceflag) {
                        fprintf(stderr, "Adjusting lnode %s.0, lp->kl.l_name);
                        fprintf(stderr, "l_uid %d0, lp->kl.l_uid);
                        fprintf(stderr, "l_usage %g0, lp->kl.l_usage);
                        fprintf(stderr, "l_charge %g0, lp->kl.l_charge);
                        fprintf(stderr, "kl_usage %g0, lp->kl_usage);
                        fprintf(stderr, "kl_totuse %g0, lp->kl_totuse);
                        fprintf(stderr, "kl_adj %g0, lp->kl_adj);
                        fprintf(stderr, "kl_rate %g0, lp->kl_rate);
                        fprintf(stderr, "kl_cost %d0, lp->kl_cost);
                        fprintf(stderr, "0);
                        fflush(stdout);
                }
        }
```

```
        return;
}
```

4.6 Tuning the Fair-share Scheduler

This section discusses tuning issues relating to the fair-share scheduler. If you run fair-share, it is important to consider other factors such as memory scheduling and NQS configuration to achieve your CPU scheduling goals.

This section discusses the following:

- Fair-share parameters
- Memory scheduling parameters and fair-share
- Priority-based scheduling with nice values (this item is not strictly about fair-share; but, it is a related topic)

4.6.1 Fair-share Parameters

Fair-share parameters provide you with considerable control over the behavior of the fair-share scheduler but do not allow you to directly affect system throughput. Before attempting to tune fair-share, it is important to establish a clear set of CPU scheduling goals. Tuning fair-share usually involves a compromise between accurate distribution of resources by share allocations and interactive responsiveness.

This section describes the `shradmin`(8) options most useful for tuning the fair-share scheduler and provides an example of possible parameter settings. For more information on fair-share parameters, see the `shradmin`(8) man page.

4.6.1.1 `shradmin`(8) Options Affecting Cost

The following options for the `shradmin`(8) command allow you to set the cost for various system resources:

Option	Description
`-b`	Cost for logical I/O request
`-m`	Cost for each memory click
`-s`	Cost for each system call
`-t`	Cost for a CPU tick

`-y` Cost for a tty read/write operation

The `shradmin` command allows you to apply a percent multiplier to all of these cost factors. For example, the command `shradmin 20` specifies that all cost values should be multiplied by the factor 0.2. By applying the multiplier, you can uniformly scale the cost parameters.

Because fair-share schedules only the CPU resources, it is recommended that costs be weighted heavily, or exclusively, for CPU use.

4.6.1.2 `-U` Option (MAXUSAGE)

If the usage value for a process reaches `MAXUSAGE`, that process hangs if there is no idle time (the priority of the process is set to 998, which prevents the process from getting the CPU). Therefore, `MAXUSAGE` should be set such that it will only take effect under extreme circumstances.

The default setting of `MAXUSAGE` is $1.0e^{+12}$. This is too low for systems with 8 or more CPUs and fast cycle times. For a system with 8 CPUs, a better starting value is $1.0e^{+15}$; for a system with 16 CPUs, a better starting value is $1.0e^{+18}$. For any site, if a very long usage decay rate (greater than 24 hours) or large cost values are used (greater than 1000), it is recommended that this value be increased.

Use the `shrview`(1) command with the `-ds` option to obtain statistics on process usage values. This command samples the average number of users logged on at one time, as in the following example:

```
% shrview -ds
Shrview: sampling over 5 seconds starting at Thu Aug  5 16:55:14 1993

  SHRVIEW  Type:statistics  Select:all  Sort_opt:id

                                          High         Max
                                       ----------  ----------
Active id's:   62/300            Usage:     5.9689e+09  1.0000e+12
Active groups:  6                Share_pri: 3.8694e+03  1.0000e+28

          syscall       bio         sio         tick       click
        -----------  ----------  ----------  ----------  ----------
Charge:          0%          0%          0%        100%          0%
Costs:            0           0           0         100           0
Counts:   501753913     6866002           0    15649624           0
```

If *clipping* is occurring on MAXUSAGE (that is, if the actual usage is consistently close to the MAXUSAGE setting), increase the value for MAXUSAGE in the /etc/config/daemons file.

4.6.1.3 -D Option (*priority Decay Rates*)

The -D option sets *priority decay rates* for processes in the following areas and alters the effect of nice:

- Normal nice values (20)
- Maximum nice values (39)

These two values are used to build a table of priority decay rates for each nice value (0 through 39). As the difference between the two rates increases, the effect of nice is more pronounced. This effect is not related directly to fair-share; it occurs even if fair-share is not enabled. Shorter decay rates cause the scheduler to act in more of a round-robin fashion. Longer values cause the scheduler to be more influenced by fair-share information.

The effect of this option is subtle and might have undesirable side effects. Long decay rates can cause *marooning*, where a compute-bound process can control the CPU(s) for long periods of time. The default values for the -D option (2,60) are recommended, and any changes should be carefully considered. Do not use values of less than (1,30) because this generates negative decay rates for some of the lower nice values.

The shrinfo (1) command used with the -v option shows the decay rates for the nice values of 0, 20, and 39, as in the following example:

```
Scheduling flags    = ADJGROUPS, LIMSHARE
Charging percentage = 100%
Usage decay rate    = 0.95484160 (half-life of  60.0 seconds)
Active users = 59/300, active groups = 6.

     ---syscall-- ----bio---- ----tio--------tick--- ---click--- (NULL)
Charge:         0%          0%          0%        100%          0%
Costs:           0           0           0         100           0
Counts:  494584243     6698323           0    15173200           0

Process priority decay rate biased by "nice":-
high priority (nice -20) 0.4044 (half-life of   0.8 seconds),
avg  priority (nice   0) 0.7039 (half-life of   2.0 seconds),
low  priority (nice  19) 0.9885 (half-life of  60.0 seconds).
Run rate decay rate      0.8409 (half-life of   4.0 seconds).
```

```
Max. value for normal usage        = 1.000000e+12,
Max. value for normal p_sharepri   = 1.000000e+28,
Max. value for idle   p_sharepri   = 1.000000e+38.
High value of current normal usage = 3.369134e+10,
High value of current p_sharepri   = 9.236291e+03.

User <anne> owns 100 shares.
```

The `shrview -dn` command shows a table of decay rates for each nice value, as in the following example:

```
Shrview: sampling over 5 seconds starting at Thu Aug5 16:41:36 1993

  SHRVIEW  Type:nice tables  Select:all  Sort_opt:id

             NiceDecays       Nice   Nice
 N  Nice   Decay  1/2life   Rates  Ticks
--  ----  ------  -------  ------  -----
 0   -20  0.4044    0.766  0.1591    150
 1   -19  0.4194    0.798  0.1591    147
 2   -18  0.4343    0.831  0.1591    145
 3   -17  0.4493    0.866  0.1591    142
 4   -16  0.4643    0.903  0.1591    140
 5   -15  0.4793    0.942  0.1591    137
 6   -14  0.4943    0.984  0.1591    135
 7   -13  0.5092    1.027  0.1591    132
 8   -12  0.5242    1.073  0.1591    130
 9   -11  0.5392    1.122  0.1591    127
10   -10  0.5542    1.174  0.1591    125
                .
                .
                .
17    -3  0.6590    1.662  0.1591    107
18    -2  0.6740    1.757  0.1591    105
19    -1  0.6890    1.860  0.1591    102
20     0  0.7039    1.974  0.1591    100
21     1  0.7189    2.100  0.1515     97
22     2  0.7339    2.240  0.1446     95
23     3  0.7489    2.397  0.1384     92
                .
                .
                .
```

```
37    17  0.9586   16.377  0.0860     57
38    18  0.9735   25.844  0.0837     55
39    19  0.9885   60.000  0.0000      1
```

4.6.1.4 -X Option (*maxushare*)

The -X option (*maxushare*) defines the maximum effective share for an individual user, when the global scheduling flag LIMSHARE is set (with the shradmin -F command). This option can be used to minimize marooning by preventing users with very low past usage from monopolizing the CPU or by preventing users with very high past usage from being locked out.

Values close to 1 ensure good interactive response regardless of past usage and provide a leveling effect so that all users will have scheduling priorities that correlate closely to their share allocations. Increasing this value allows scheduling priorities to deviate more from share allocations as determined by past usage.

The default value of 2.0 is a good initial value if interactive use is important at your site. Values of 8.0 and greater are good for a batch environment. If interactive response is not important, and consistent distribution of resources has the highest priority, you should turn off this option by setting the LIMSHARE flag equal to 0. This option must be set to a value greater than 1.0.

You can view the effect of the *maxushare* option with the shrview -dl command.

4.6.1.5 -Y Option (*mingshare*)

The -Y option (*mingshare*) specifies the deviation between group share allocation and the actual rate of resource consumption allowed before the scheduling algorithm tries to compensate. It has an effect only when the global scheduling flag ADJGROUPS is set (with the shradmin -F command). This option can diminish to some degree the effect of the -X (*maxushare*) option.

The default setting of .75 allows a 25% or less deviation. Values closer to 1.0 allow less deviation, but may interfere with *maxushare*. This option must be set to less than 1.0.

When accurate distribution of resources based on group share allocations is a high priority, this option is very important and should be set near to 1.0 (.90 to .95).

The effect of the *mingshare* option can be viewed with the shrview -da command, as in the following example:

```
% shrview -da

Shrview: sampling over 5 seconds starting at Thu Aug  5 16:47:12 1993

  SHRVIEW  Type:adjgroup  Select:all  Sort_opt:id

    Name      Rate%   CPU%  Rshr%   Nrun  Rate  Proj%    Adj_a   New%   Pri%
------------  -----  -----  -----  -----  ----  -----  -------  -----  -----
 Idle          0.00   3.11   0.02   0.00  7.62   0.00     1.00   0.00   0.00
 System        4.52   6.14  10.00   0.00  0.00   0.00     1.00   0.00   8.12
  operator     0.00   0.00   5.00   0.00  0.84   0.00     1.00   0.00   6.50
  jkl          4.52   6.14   5.00   5.00  1.00   0.00     1.00   0.00   1.62
 Users        63.65  45.37  90.00   0.00 19.48   0.00     1.00   0.00  29.24
  CCN          0.00   0.00   5.00   0.00  0.84   0.00     1.00   0.00   1.62
   abc         0.00   0.00   1.64   0.00  0.84   0.00     1.00   0.00   0.53
   zyx         0.00   0.00   1.64   0.00  0.84   0.00     1.00   0.00   2.13
   c1234       0.00   0.00   1.64   0.00  0.84   0.00     1.00   0.00   2.13
  HardDev      5.41   2.43   5.00   0.00  0.84   0.00     1.00   0.00   1.62
   paul        2.71   2.43   4.76   0.00  0.87   0.00     1.00   0.00   1.55
  Mktg         0.17   0.05  30.00   0.00  0.84   0.00     1.00   0.00   9.75
  *anne        0.08   0.05   3.61   0.00  0.84   0.00     1.00   0.00   1.17
   steven      0.00   0.00   3.61   0.00  0.84   0.00     1.00   0.00   1.17
   c0987       0.00   0.00   3.61   0.00  0.84   0.00     1.00   0.00   1.17
   gregj       0.00   0.00   3.61   0.00  0.84   0.00     1.00   0.00   4.70
  SoftDev     58.07  42.90  50.00   0.00 13.77   0.00     1.00   0.00  16.25
   ali         0.00   0.00   1.20   0.00  0.84   0.00     1.00   0.00   1.57
   birk        0.00   0.00   1.20   0.00  0.84 100.00     1.00 100.00   1.57
   bobo        0.17   0.18   1.20   1.20  1.19   0.00     1.00   0.00   0.39
   cde         0.00   0.00   1.20   0.00  0.84   0.00     1.00   0.00   1.57
                .
                .
                .
   zzzyzzy     0.08   0.60   1.20   0.00  0.84   0.00     1.00   0.00   0.39
```

4.6.1.6 -Z Option (*sharemin*)

The -Z option (*sharemin*) specifies the minimum machine share allocated to a user. This option, when combined with the *maxushare* option, sets a range for individual priorities to allow reasonable interactive response for all users. The default value of 0 sets no minimum share allocation for users. It is recommended that you select a nonzero value for this option to provide reasonable response for all users. Select a value based on the reciprocal of the average number of users logged on (1 *average_users*), rounded to three digits.

For example, use the value 0.013 on a system with an average of 80 users logged on at any one time.

The effect of the `-Z` option can be seen by comparing the `Eshr` and `Rshr` columns from the `shrview -dm` output, as in the following example. The values in these columns should be as close as possible.

```
Shrview: sampling over 5 seconds starting at Thu Aug  5 16:57:21 1993

  SHRVIEW  Type:monitor  Select:all  Sort_opt:id

    Name      Rate%  Eshr%  Rshr%   Usage(k)    Rate   Muse  Ref  Chld   Flags
------------  -----  -----  -----  ----------  -----  -----  ---  ----  ------
 Idle          0.00   0.00   0.02  1000000000   7.62     56    8     0   10000
 System        0.00  10.00  10.00  1000000000   0.00      0    6     2   50000
  operator     0.00   5.00   5.00           0   0.84 1210410   1     0   10000
  jkl          0.00   5.00   5.00           0   0.84  48068    5     0   10000
 Users        66.67  90.00  90.00  1000000000  15.93      0 1598474  4   50000
  CCN          0.00   5.00   5.00           0   0.84      0 60666    3   10000
   abc         0.00   1.64   1.64           0   0.84  32836    4     0   10000
   zyx         0.00   1.64   1.64           0   0.84   3475    3     0   10000
   c1234       0.00   1.64   1.64           0   0.84  14100    3     0   10000
  HardDev      1.44   5.00   5.00           0   0.84      0 14813    1   10000
   paul        0.72   4.76   4.76           0   0.94  81902    9     0   10000
  Mktg        10.41  30.00  30.00           0   1.02      0 211197   7   10000
  *anne        0.06   4.11   4.11           0   0.84   3536    3     0   10000
   steven      0.00   4.11   4.11           0   0.84   2626    2     0   10000
   c0987       0.00   4.11   4.11           0   0.84   3414    1     0   10000
   gregj       0.00   4.11   4.11           0   0.84   2848    1     0   10000
  SoftDev     54.81  50.00  50.00           0   8.46      0 1311798  41   10000
   ali         0.00   1.20   1.20           0   0.84   5720    1     0   10000
   birk        0.00   1.20   1.20           0   0.84   6638    1     0   10000
   bobo        0.23   1.20   1.20           0   1.82 131743   11     0   10000
   cde         0.00   1.20   1.20           0   0.84  14404    1     0   10000
                .
                .
                .
   zzzyzzy     0.00   1.20   1.20           0   0.84  89449    4     0   10000
```

4.6.1.7 -R Option (*delta*)

The -R option (*delta*) determines how often usage values are updated and is directly related to the overhead for fair-share processing. Shorter values increase the scheduler's responsiveness to changing conditions, but they increase overhead longer values decrease overhead but also decrease responsiveness.

The default of 4 seconds is a good compromise for interactive environments. For batch environments, the value can be increased; however, overhead is increased by the number of active lnodes and levels in the fair-share hierarchy.

4.6.1.8 -K Option (*usage Decay Rate*)

The -K option (*usage decay rate*) establishes the length of time that past usage of resources are remembered. Longer decay rates cause the scheduler to distribute resources fairly over a longer period of time. This option can have a significant impact on interactive responsiveness. On a heavily loaded machine with a large number of users, short decay rates provide uniformly poor response for all users with small share allocations. Longer rates will provide good interactive response for users that have used less than their allocated share and worse response for those who have used more. Very short usage decay rates (less than 3600 seconds, or 60 minutes) limit the effect of the *maxushare* and *mingshare* options. For these reasons, decay rates greater than 60 minutes will generally provide more accurate distribution of resources and better interactive response times.

Note: Do not set the decay rate to less than 1 hour (the default) for general fair-share operation. Decay rates of minutes or seconds are recommended only for debugging purposes.

4.6.1.9 Example Parameter Settings

The following parameter settings show sample values at a site presently running fair-share. The goals for choosing these settings were to provide the following:

- Good interactive response during prime time regardless of the share allocations or past usage
- Close tracking of share allocations and usage during nonprime time
- Incentive for nonprime time usage
- Usage decay rate that would be similar in effect to the one-week refreshing of "bank point" (a local concept for this site)

- Limit on the impact of long-running batch jobs on interactive performance

The following option settings are used during prime time:

Option setting	Description
`-D 2,60`	Priority decay rate
`-F 006`	`LIMSHARE` and `ADJGROUPS` flags
`-K 120`	Usage decay rate of 5 days
`-R 4`	*delta*
`-X 2.0`	*maxushare*
`-Y 0.7`	*mingshare*
`-Z .02`	*sharemin* (2%; default)
`-b 2`	Cost of block I/O
`-m 2`	Cost of memory click
`-t 200`	Cost per CPU tick
`-s 2`	Cost of system calls
`100`	Percent multiplier (100%)

The following option settings are used during nonprime time:

Option setting	Description
`-D 2,60`	Priority decay rate
`-F 006`	`LIMSHARE` and `ADJGROUPS` flags
`-K 120`	Usage decay rate of 5 days
`-R 8`	*delta*
`-X 8.0`	*maxushare*
`-Y 0.9`	*mingshare*
`-Z .01`	*sharemin* (1%)
`-b 2`	Cost of block I/O
`-m 2`	Cost of memory click
`-t 200`	Cost per CPU tick
`-s 2`	Cost of system calls

`60,40`	Percent multiplier (60% night, 40% weekend)

4.6.2 Fair-share and the Memory Scheduler

When using the fair-share scheduler, it is important to consider how fair-share interacts with the memory scheduler. If you are using fair-share, the memory scheduler parameters should be set to achieve the CPU scheduling goals without excessive swapping overhead. If your goals are to base scheduling completely on share allocations, it is possible that the system could become idle as large memory processes dominate the system. Using both the memory scheduler and the fair-share scheduler provides your site with a compromise between improving system throughput and meeting scheduling objectives.

Deciding what values to use in order to achieve the desired results is more difficult than if you were using only the memory factors or only the priority factors. It is important to investigate the following areas:

- Memory oversubscription
- Fair-share priorities of jobs in relation to memory size (in other words, does a user with low fair-share priority run big memory jobs and a user with high fair-share priorities run small memory jobs?)

Using this type of scheduling ensures that processes with high fair-share priorities are more likely to be in memory without causing excessive swapping and system overhead.

For more information on the memory scheduler, see the `nschedv`(8) man page in the UNICOS Administrator Commands Reference Manual, publication SR-2022.

4.6.2.1 Priority-based Scheduling and I/O Resources

When you use both fair-share and priority memory scheduling, it is important to consider how the system I/O resources are used. If a job with a low fair-share priority uses a large amount of disk or SDS resources, significant problems could result for the entire system.

If you can determine that I/O-intensive jobs are a problem for your system, consider changing the memory scheduling parameters so that either this type of job is favored or has equal access to memory. Changing fair-share parameters might also be necessary. Getting this job out of the system as soon as possible helps the total system throughput.

4.7 Using CPU Quotas

The CPU quota feature allows you to control the total CPU time used by each user login or account on the system in increments of tenths of seconds. CPU quotas function only when your site is running the fair-share scheduler.

The CPU quota feature is similar to the CPU limits feature, but resource consumption information is accumulated for the user rather than the job or process. When a user reaches the quota, a `SIGCPULIM` signal is sent to the user's processes. (The `SIGCPULIM` signal is ignored by default.) When the user reaches a specified threshold above the quota (the default is 3 seconds), the kernel sends a `SIGKILL` signal to all the user's processes and terminates them.

The UDB contains a quota field (`cpuquota`) and a time-used field (`cpuquotaused`) that set a user's CPU quota in seconds and the amount of CPU time used in seconds, respectively. You can use the `udbgen`(8) command to update these fields. For example, to set a quota of 10 seconds for user `xyz`, you would enter the following command:

```
udbgen -c 'update:xyz: cpuquota:10:'
```

To cancel the accumulated time for user `xyz`, you would enter the following command:

```
udbgen -c 'update:xyz: cpuquotaused:0:'
```

The amount of time that a user accumulates while running jobs (the *accumulated CPU time*) is calculated by the kernel. After a user's last session exits the system, the fair-share daemon updates the `cpuquotaused` field in the UDB (as well as other fields) with the accumulated CPU time. During this process, any changes that have been made to an active user's `cpuquotaused` field in the UDB are overwritten with the old value from the UDB (plus the latest accumulated CPU time). Therefore, you must take extra steps to ensure that changes made to an active user's `cpuquotaused` field, such as clearing the field, remain in effect after the user has exited the system.

To ensure that changes to an active user's `cpuquotaused` field are not lost, use the `shrsync`(8) command. This command synchronizes various fields in the UDB, including the `cpuquotaused` field, with the corresponding data in the kernel.

The following procedure shows the `shrsync` commands used to update the `cpuquotaused` fields in the UDB while in multiuser mode:

1. Use the `-u` option of the `shrsync` command to update the UDB with the information from the active system. Because the active system will be

updated with information from the UDB, the information in the UDB must first be brought up to date (in particular, the CPU-quota-used information of the running sessions).

```
/etc/shrsync -u
```

2. Update the `cpuquotaused` fields in the UDB, as in the following example for user `xyz`:

```
/etc/udbgen -c 'update:xyzfP: cpuquotaused:0:' ; ...
```

3. Use the `-q` option of the `shrsync` command to indicate that all active users will have their `cpuquotaused` information updated on the system from the UDB, as follows:

```
/etc/shrsync -q
```

See the `udbgen`(8) man page for more information on the CPU quota fields; see the `shrsync`(8) man page for more information on synchronizing UDB and kernel information.

4.8 Additional Reference Material

The paper, "A Fair Share Scheduler," by J. Kay and P. Lauder, was published in the January 1988 (volume 31, number 1) issue of *Communications of the ACM* (pages 44-55). Although the UNICOS implementation does not exactly parallel the description as published, it is substantially similar, and this paper is recommended to anyone interested in the theory and philosophy of this scheduling mechanism.

The paper "The Fair Share Scheduler," by G.J. Henry, was published in the October 1984 *Bell Labs Technical Journal*, LVIII-8b. This paper also contains design information about the fair-share scheduler.