Startup and Shutdown [3]

This chapter includes procedures to do the following:

- Start the CRAY IOS-V and CRAY J90 mainframe and bring up the UNICOS system to a multiuser run state mode (startup; also called *booting*).
- Bring the UNICOS system back to single-user mode (shutdown).

This chapter also briefly describes several start-up scripts, configuration scripts and files, the aspects of the start-up process that can be customized for your site, and run-level configuration information.

If you have access to a windowing environment, the UNICOS operating system provides a point-and-click, X Window System based interface to the UNICOS Installation / Configuration Menu System. For more information, see the *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412.

To start and stop UNICOS system daemons, see Chapter 4, page 45.

3.1 Related startup and shutdown documentation

The following documentation contains more detailed information about the material presented in this section:

- *UNICOS Installation Guide for CRAY J90 Model V based Systems*, Cray Research publication SG–5271
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `bcheckrc`(8), `brc`(8), `dmdstop`(8), `fuser`(8), `init`(8), `msgdstop`(8), `rc`(8), `sdaemon`(8), and `shutdown`(8) man pages
- *CRAY IOS-V Commands Reference Manual*, Cray Research publication SR–2170
- *CRAY IOS-V Messages*, Cray Research publication SQ–2172

Procedure 1: Starting up the system

Note: The CRAY J90 IOS-V is case sensitive; enter all lowercase characters on the system console.

To boot the IOS and UNICOS software, enter the following commands at the system console:

1. To invoke the CRAY J90 console, press the right mouse button in the OpenWindows root window and select the `J90 Console` menu item. This invokes the `jcon` command, which will log on remotely to `sn`*xxxx*`-ios0` (`sn`*xxxx* is the mainframe serial number, and `ios0` is used for the initial boot and load of the IOS and the UNICOS software to the system).

2. Start the IOS by loading the appropriate device strategies and drivers and by loading and executing the IOS kernel. To do this, enter the `load` command at the IOS boot prompt, which is `BOOT[sn`*xxxx*`-ios0]>` on CRAY J90 systems.

```
BOOT[snxxxx-ios0]> load
```

   The IOS load command produces output on your terminal and returns the IOS prompt when complete:

```
snxxxx-ios0>
```

   Note: When using the system console, press `CONTROL-a` to toggle from the UNICOS prompt to the IOS prompt. To toggle from the IOS prompt to the UNICOS prompt, press `CONTROL-a RETURN`.

3. Start the UNICOS system by entering the `/bin/boot` command at the IOS prompt:

```
snxxxx-iosx> /bin/boot
```

   The `/bin/boot` script contains IOS commands that clear the mainframe memory, load the UNICOS kernel and the IOS configuration parameter file, initiate communication between the IOS and the UNICOS system, and begin executing the UNICOS system. The prompt on your system console terminal will be the `root` user prompt (#). It may be preceded by `sn` and your system's serial number, as follows:

```
sn1234#
```

   After executing the initial `/bin/boot` command, the UNICOS system is in single-user mode.

After booting the UNICOS system to single-user mode, you should run the `mfsck`(1) command to check the file systems for inconsistencies as follows:

```
snxxxx-iosx> CONTROL-aRETURN
(toggles to the UNICOS console)
# /etc/mfsck
```

Only a few processes are running: `init`, `swapper`, `idle`, and `sh`. The `root` (`/`) file system is the only file system available.

When you are in single-user mode with only the `root` (`/`) file system available, you must do all editing by using the `ed` editor, because the `vi` editor is located in the `/usr` file system. If you want to use the `vi` editor before going to multiuser mode, you must mount the `/usr` file system. Before going to multiuser mode, or if you intend to work in single-user mode, you should check the `root` (`/`) file system by using `fsck`(8); for the procedure, see Procedure 7, step 3, page 125.

The first time you use the `vi` editor, you may see the following error message:

```
I don't know what kind of terminal you are on - all I have is
'unknown'.[Using open mode]
```

If you are using a WYSE terminal, type the following command lines to solve this problem. To backspace, use the `DELETE` key.

```
:wq
# export TERM=vt100
# resize
# echo $TERM
```

If the console does not respond , it may help to power cycle the WYSE terminal by turning the power off, and then on again.

If you are using the CRAY J90 IOS-V system console (CRAY J90 console), type the following command lines:

```
:wq
# TERM=xterm

(or sun-cmd, if you are using the command tool)

# export TERM
# resize
# echo $TERM
# echo $SHELL
```

Note: Before going to multiuser mode, or if you intend to work in single-user mode, you should run `fsck` on the `root` (`/`) file system (for more information, see Section 3.4.4, page 34).

4. Bring the system to multiuser mode by signaling the `/etc/init` process to change to a new run level by entering the following command:

```
# /etc/init 2
```

Multiuser mode is usually run-level 2. Although you can configure a system to run in multiuser mode at any level between 0 and 6, you may want to reserve some states for the future. For additional information about run-level configuration, see Section 3.4.1, page 30, and Section 3.5, page 36.

As the system boots into multiuser mode, output is produced on your terminal. You will be asked whether you want to run `mkfs /tmp (y/n)`, which you must respond to for the process to proceed. At approximately midpoint in the process, the `Administrative cleanup` message appears. This message indicates that the system is moving into multiuser mode properly. You will be prompted for the system date, which is an optional entry. When the system boot is complete, you will see the following prompt:

```
Console Login:
```

5. Log in as user `root` and use the password `initial0`.

Caution: Change the `root` password by using the `/bin/passwd` command. To guard against intentional or inadvertent damage caused by unauthorized use of super-user privileges, you should change the password now.

6. Finish setting up the basic system environment for your site, such as user accounts, file systems, networking, and so on.

Procedure 2: Shutting down the UNICOS system and the IOS

To shut down the UNICOS system and the IOS, follow these steps:

1. Make sure that you are logged in as `root` and that you are in the `root` (`/`), `/etc`, or `/ce` directory; to change to the `root` (`/`) directory, enter the following command:

```
# cd /
```

2. You may want to send active users a special message about when the system will be shut down. The `/etc/shutdown` script is designed to return the UNICOS system to single-user run state in a clean, orderly manner. The `/etc/shutdown` script prompts you for a message that will be sent to all users; if you want to include a message, use the `wall`(8) command to provide the message (see Chapter 8, page 217). Before executing `/etc/shutdown`, you can use the `ps -eaf` command to see processes that are running, and the `who -u` command to see whether people are actively using the system. The `shutdown`(8) command uses the following format:

   `/etc/shutdown` *grace-period-in-seconds*

   The following command instructs the system to wait 5 minutes (300 seconds) before terminating all processes and shutting down the system:

```
# /etc/shutdown 300
Do you want to send your own message?  (y or n):  y
Type your message followed by a <Return> and then ctrl d....
System shutting down in 5 minutes for test time-Please log out now.
CONTROL-d
```

   The time it takes for the shutdown to complete depends on the number of processes that must terminate and file systems that must be unmounted; however, the shutdown process may take 3 to 5 minutes.

   When the shutdown program is complete, the following message is displayed, and you should type the following highlighted commands:

```
Message: INIT: SINGLE-USER MODE.# /bin/sync
# /bin/sync
# /bin/sync
# /etc/ldsync
```

(if you are using ldcache)

```
# df
```

(to verify that all file systems have been unmounted cleanly)

```
# /bin/sync
```

At this point, you are in single-user mode but the UNICOS system is still running. You can perform any system administration work as necessary.

3. Optional step. If you want to stop the UNICOS system from running, toggle to the IOS and enter the `mc`(8) (master clear) command, as follows:

```
# CONTROL-a
snxxxx-ios0> mc
```

Note: The CRAY J90 IOS-V is case sensitive; enter all lowercase characters on the system console.

Note: You should not reboot the UNICOS system without reloading the IOS.

4. Optional step. At this point, you can stop the IOS software by entering the `reset`(8) command, which returns the IOS boot prompt and puts the system as close as possible to the state it was in after being powered up.

```
snxxxx-ios0> reset
BOOT[snxxxx-ios0]>
```

5. Optional step. Power off your system if you choose to do so (for procedures to power off your system, see Appendix F, page 335, or see your hardware installation manual).

3.2 Shutdown information

The `/etc/shutdown` script terminates all user processes and system daemons, releases all logical device cache, and unmounts all UNICOS file systems (except for `root`). Unlike the `/etc/rc` start-up script, the operation of the `/etc/shutdown` script is not altered by any UNICOS control files. For CRAY J90 series systems, you do not have to modify the `/etc/shutdown` script directly.

3.2.1 User exits

The `/etc/shutdown` script provides three user exits (`shutdown.pre`, `shutdown.mid`, and `shutdown.pst`) that allow you to modify the shutdown process.

3.2.1.1 `shutdown.pre`

The `shutdown.pre` script is the first user exit of the `shutdown` script. If an executable named `/etc/shutdown.pre` exists, it will be executed during shutdown. At this point, nothing has been done toward shutting down the system. All daemons are still running, all file systems are mounted, and all users are still active and unaware that this script is running.

A possible use of this exit would be to verify the user's permission to run the `shutdown` script or to run some system cleanup routines. The `shutdown` script will check the return status from the `shutdown.pre` program. If the return status is nonzero, the user will be queried as to whether or not to continue the shutdown processing. At this point, the shutdown can be stopped without any effect on the system.

3.2.1.2 `shutdown.mid`

The `shutdown.mid` script is the second user exit of the `shutdown` script. If an executable named `/etc/shutdown.mid` exists, it will be executed during shutdown.

At this time, all processes (users and daemons) have been terminated, all the disk cache (ldcache or pcache) has been released, but the network interfaces are still configured, and all of the file systems are still mounted.

A possible use of this exit would be to allow NFS file systems to be unmounted before the networks are stopped.

The `shutdown` script will check the return status from the `shutdown.mid` program. If the return status is nonzero, the user will be queried whether to continue the shutdown processing or not. This exit is given to address any possible problem that may exist with the file systems still mounted and the networks that are still running.

3.2.1.3 `shutdown.pst`

The `shutdown.pst` user exit is the third (and last) user exit of the `shutdown` script. If an executable named `/etc/shutdown.pst` exists, it will be executed during shutdown. At this point, all processes (users and daemons) have been terminated, but the file systems are still mounted. This is virtually single-user mode, except for the file systems.

After this point, the file systems are unmounted and `/etc/init` is invoked to go to single-user mode. The `/etc/init s` command will kill all remaining processes (including the process running the `shutdown` script), so there is no place to put a user exit beyond this point.

Because the system is virtually shut down by this point, there is no reason to halt the script if the user exit return status is not zero. The status returned from `/etc/shutdown.pst` is checked, but `shutdown` will only issue a warning message and then go to single-user mode.

> **Note:** Be careful in what you allow `shutdown.pst` to execute. Because the various logging daemons (such as `syslogd` are not available to free up space, `shutdown.pst` could potentially fill up the file system(s) that contain the log files.

3.2.2 Shutdown process

If you install and start any local processes or daemons during the execution of the `/etc/rc` script, to stop them within the `/etc/shutdown.sh` script, you can add control information into the `/etc/config/daemons` file and insert `/etc/sdaemon` commands into the `/etc/shutdown.sh` script. You should make any changes either in `/etc/config/rcoptions` or in one of the user exits; **do not** modify `/etc/rc`. For information on starting and stopping UNICOS system daemons, see Chapter 4, page 45.

The process of a UNICOS system shutdown is as follows:

1. Executes the user exit `/etc/shutdown.pre`, if it exists. If a nonzero return status is returned from the user exit, `shutdown` will prompt the user for confirmation before continuing.

2. Sends a message, using `wall`(8), warning the users who are currently logged in to the system that the system is being shut down.
3. Shuts down the NQE subsystem to allow batch jobs to be checkpointed before they are terminated.
4. Sends a `SIGSHUTDN` signal to all currently running processes.
5. Stops the DM daemon, Tape daemon, and error logging.
6. Shuts down daemons in the `SYS1` and `SYS2` groups (defined in the `/etc/config/daemons` file), using the `sdaemon`(8) command.
7. Sends a `SIGHUP` signal to all currently running processes.
8. Sends a `SIGKILL` signal to all currently running processes.
9. Shuts down system accounting, using the following command (see `acctsh`(8)):

   ```
   /usr/lib/acct/shutacct
   ```

   `shutacct`(8) records the action of shutting down system accounting in the `/etc/wtmp` file.
10. Releases partition cache, using the `ldcache`(8) command. This ensures that all partition cache buffers are flushed.
11. Executes the user exit `/etc/shutdown.mid`, if it exists. If a nonzero return status is returned from the user exit, `shutdown` will prompt the user for confirmation before continuing.
12. Shuts down all configured network interfaces (defined in the `/etc/config/interfaces` file), using the `ifconfig`(8) command.
13. Executes the user exit `/etc/shutdown.pst`, if it exists. If a nonzero return status is returned from the user exit, a warning message will be printed (the shutdown cannot be stopped at this point, because all daemons and processes have been terminated).
14. Unmounts all file systems. If any local file systems cannot be unmounted, the `shutdown` script will issue a warning message.
15. Brings the system to single-user mode, using the `init`(8) command with an `s` argument, as follows:

    ```
    /etc/init s
    ```

3.3 Startup, shutdown, and configuration files and scripts for IOS and the UNICOS system

This section lists the files and scripts that are used for starting up, shutting down, and configuring your system.

Note: The CRAY J90 IOS-V is case sensitive; therefore, you must enter the following file names in all lowercase characters on the system console.

The following are IOS-resident configuration and start-up files:

File	Description
`/autoboot`	If the file exists, the IOS automatically tries to load itself and to boot the UNICOS system after any reset or power cycle on CRAY J90 systems. This file may contain the absolute path to an alternative IOS kernel to be loaded; otherwise, `/ios/ios` will be loaded, followed by `/bin/boot`.
`/bin/boot`	UNICOS boot script.
`/config`	IOS-V configuration file.
`/sys/*.cfg`	ASIC configuration files created by the `jconfig`(8) command for CRAY J90 systems. For details on `.cfg` files, see the `jconfig`(8) man page.
`/sys/param`	Default IOS parameter file that contains configuration specification language (CSL) statements defining physical, striped, and logical disk devices, system disk devices, and kernel parameters.
`/sys/unicos.ymp`	Default UNICOS kernel.

The following are UNICOS-resident configuration files and start-up scripts:

File	Description
`/etc/config/daemons`	File listing and daemons to be started during multiuser startup; used by `/etc/sdaemon`. See Chapter 4, page 45.

`/etc/config/rcoptions`	Sets environment variables that control `/etc/rc`.
`/etc/inittab`	Read by `/etc/init` at system boot.
`/usr/src/uts/cf.sn`*xxxx*`/config.h`	Parameter file that defines the UNICOS kernel. You should not change these parameters manually.
`/usr/src/uts/cf.sn`*xxxx*`/sn.h`	Parameter file that defines machine-specific characteristics of your mainframe.

The following are UNICOS shell scripts:

Script	Description
`/etc/bcheckrc`	Checks the system date and time, and verifies the integrity of the UNICOS file systems before being mounted.
`/etc/brc`	Detects presence of a UNICOS system dump.
`/etc/rc`	UNICOS multiuser start-up script.
`/etc/shutdown`	UNICOS shutdown script.

At boot time, the following files are created in the UNICOS `root` (`/`) file system (the root file system is chosen by the `ROOTDEV` line in the IOS `/sys/param` file):

File	Description
`/CONFIGURATION`	Contains processed CSL definitions. This file matches the IOS `/sys/param` file that was used to boot the system.
`/unicos`	A copy of the running UNICOS kernel. This file is **not** an exact copy of the bootable image that resides on the IOS disk in `/sys/unicos.ymp`.

3.4 Start-up scripts

This section describes the `/etc/init` command and start-up scripts.

3.4.1 The /etc/init command

The /etc/init command is the process control initialization command and is invoked as the last step in the UNICOS system boot procedure. init is the process from which all other processes are spawned, either directly or indirectly. The process ID (PID) of init is always 1.

At any moment, init considers the system to be in one of eight different run levels: through 6, or S (s) (a run level of S or s refers to single-user mode). When you specify S, init operates in single-user mode with the additional result that /dev/syscon is linked to the user's terminal line, thus making it the virtual system console). For more information about run-level configuration, see Section 3.5, page 36.

By default, init considers the system to be in run level S at the end of the normal system boot procedure.

For further details about /etc/init, see the init(8) man page.

3.4.2 The /etc/inittab file

The /etc/inittab file contains directions for actions when changing run levels. Each entry within the /etc/inittab file contains four fields, separated by colons.

These fields identify and provide the "when," "how," and "what" to the /etc/init process, which starts all processes as specified in the /etc/inittab file.

Field	Description
ID	A label that uniquely identifies the entry. The label can consist of a maximum of 4 characters.
run state	Run level in which an entry should be processed. A null entry (two colons) indicates that the entry must be executed when changing to any numbered (0 through 6) run state. Numbered run states signify varying levels of UNICOS system functionality and rely on the /etc/init process starting or stopping system processes as required. See the init(8) man page. The run state field is the "when" portion for the entry.

`action`	The action field specifies "how" to start the command or program specified in the process field for this entry.
`process`	The command or the name of the program to execute. This action field is the "what" portion of the entry. To insert comments into this field, prefix a line with a # symbol.

The following values are action field values for `/etc/inittab`:

Value	Description
`boot`	Starts process at multiuser boot time if the specified run level matches the `init` run level at boot time. `init` does not wait for the process to terminate. When the process dies, `init` does not restart it.
`bootwait`	Starts process at multiuser boot time if the specified run level matches the `init` run level at boot time. `init` waits for the process to terminate. When the process dies, `init` does not restart it.
`generic`	Instructs `init` to accept login requests from privileged daemons through the `/etc/initreq` FIFO special file (named pipe).
`initdefault`	Specifies run level to enter when `init` is initially invoked. If no `initdefault` action field entry exists in `inittab`, `init` requests an initial run level from the user at boot time.
`ldsynctm`	Sets the ldsync rate (the frequency with which the `init` daemon causes the data in ldcache to be flushed to disk). The default is 120 seconds.
`off`	If process is running, sends it a `SIGTERM` signal, waits 20 seconds, and then sends it a `SIGKILL` signal if it is still running. If process is not running, it does not restart it.
`once`	Starts process, does not wait and does not restart it.
`respawn`	Starts process and restarts it when it dies.

`sysinit`	Starts process at system boot time, before accessing system console, and waits for its termination before proceeding.
`timezone`	Establishes the systemwide value of the time zone; this value is exported to all processes spawned by `init`.
`wait`	Starts process and waits for its termination before proceeding.

The `/etc/inittab` file should have the following attributes:

- The initial run level (specified by an entry with the action field `initdefault`) should be single-user mode (specified by the letter `s` in the run state field).
- Following the `initdefault` entry, an entry with the action field `timezone` should exist to set the `TZ` environment variable to the appropriate value for the time zone in which the system is located.
- Following the `timezone` entry, calls to shell scripts should actually initialize the system's state for the run level being entered. By convention, the `bcheckrc` (see `brc`(8)) program is called by an entry with the action field `bootwait` to perform boot-time-only actions, and the `rc` (see `brc`(8)) program is called by an entry with the action field `wait` to perform actions for switching from one run level to another (including switching from the initial single-user mode to multiuser mode).
- An entry with the action field `wait` should link the special file `/dev/systty` to `/dev/syscon`.
- An entry must exist for all run levels, with an action field of `respawn`, which executes the following command (see `consoled`(8)) to allow logins on the system console:

  ```
  /etc/consoled
  ```

- Any run levels that accept logins from users on front-end systems need an entry with an action of `generic`. This entry instructs `init`(8) to accept login requests from daemons through the `/etc/initreq` FIFO special file (named pipe). This is true even when the run level is intended for use by just one dedicated user; you should restrict access to the system by using the `rc`(8) script (see Section 3.4.6, page 35), rather than limiting logins to specific devices, as is often done on traditional UNIX systems.

A sample `/etc/inittab` file follows:

```
#  more /etc/inittab
is:S:initdefault:
tz::timezone:TZ=CST6CDT
sd::sysinit:/etc/setdate 1>/dev/console 2>&1 #setdate from iop
bl::bootwait:/etc/bcheckrc </dev/console >/dev/console 2>&1 #bootlog
bc::bootwait:/etc/brc </dev/console >/dev/console 2>&1 #bootrun command
rc:2:wait:/etc/rc </dev/console 1>/dev/console 2>&1 #run com norm not just 2
pf::powerfail:/etc/powerfail 1>/dev/console 2>&1 #powerfail routines
fe:2:generic:#no command to execute
co::respawn:/etc/getty console console
lt::ldsynctm:300
```

3.4.3 Interaction between `/etc/init` and `/etc/inittab`

When you boot the UNICOS system, or signal `/etc/init` to change to another run level, `/etc/init` reads the `/etc/inittab` file for directions. Command lines whose run state fields match the desired new run level are executed sequentially.

> **Note:** The `/etc/init` command reads and processes entries in your `/etc/inittab` file sequentially. The order of the entries is important and determines the sequence followed when booting your UNICOS system. Except for the `timezone` entry, you should not have to modify your `/etc/inittab` file. For details on the structure of the `/etc/inittab` file, see the `inittab`(5) man page.

If you enter a digit from 0 to 6, `/etc/init` enters that multiuser run level. If you have signaled `init` to change from a single-user run level (S or s) to a multiuser run level (0 to 6), `init` scans the `/etc/inittab` file for any entry that has a `bootwait` or `boot` action field type. Any `bootwait` or `boot` type entries are executed before any normal processing of the `inittab` file occurs. This step ensures that any system initialization happens before anyone (including the system administrator) gains access to the system.

The following single-character arguments are used to signal the actions of `init`:

- 0 through 6 places the UNICOS system in one of the multiuser run levels.
- S or s places the UNICOS system in single-user mode.

While the system is running, a system administrator can use the `init q` command to force `init` to reread `inittab`. In this way, `init` can be made aware of changes to `inittab` without changing run states.

By default, the UNICOS system has a standard `/etc/inittab` file that is designed to activate the `/etc/rc` script when you execute an `/etc/init 2` command.

3.4.4 `/etc/bcheckrc` script

The `/etc/bcheckrc` script is one of the start-up scripts that `/etc/init` invokes when it reads through the `/etc/inittab` file. The `/etc/bcheckrc` script performs two major activities. It resets the system date, if necessary, and checks all file systems that will be mounted during the start-up process.

This script is invoked only the first time you change the system from single-user to multiuser mode after a reboot.

> **Note: Do not** change or reset the system date and time when the system is running in multiuser mode. For more details, see the `date`(1) man page. You should set or change the date and time only when starting multiuser mode.

After checking and, if needed, setting the system date, the `/etc/bcheckrc` script invokes the `/etc/mfsck` utility. The `/etc/mfsck` command runs several copies of `/etc/fsck`(8) in parallel, which can speed up system startup. Usually, the `/etc/mfsck` command runs several passes, checking all file systems; only the `root` (`/`) file system is checked during the first pass. The `/etc/fstab` file (see `fstab`(5)) determines when other file systems are checked.

The only UNICOS start-up scripts you should modify are the `/etc/rc.pre`, `/etc/rc.mid`, and `/etc/rc.post` scripts. To change the behavior and actions of other UNICOS start-up scripts easily, modify the `/etc/config/rcoptions` file and the `/etc/config/daemons` configuration file. For additional information, see Section 3.4.6, page 35, Section 3.4.7, page 35, and Section 3.4.8, page 36.

3.4.5 `/etc/brc` script

The `/etc/brc` script is invoked by `/etc/init` through `/etc/inittab` and is intended for use in initializing hardware devices. It also copies system dumps into a separate file system by executing the `/etc/coredd`(8) script. The `/etc/brc` script should not be used to start processes because those processes will be killed and not restarted during the subsequent system shutdown or startup.

Like the `/etc/bcheckrc` script, the `/etc/brc` script is invoked only the first time you change from single-user to any multiuser (numeric) run level.

3.4.6 The multiuser start-up script /etc/rc

The /etc/rc script is invoked by /etc/init when the UNICOS system goes from a single-user to multiuser run level.

Note: Do not modify the /etc/rc start-up script; to alter the behavior of your script, make any needed changes to the /etc/config/rcoptions file.

The /etc/rc.log file collects messages generated during execution of /etc/rc. The /etc/rc.log file is cleared during execution of /etc/rc and messages written to the /etc/rc.log file during an earlier execution of /etc/rc are lost. Not all start-up output is written into the /etc/rc.log file.

When finished, /etc/rc returns control to /etc/init, which then continues reading and processing subsequent lines from /etc/inittab.

For more information about the /etc/rc script, see *General UNICOS System Administration*, Cray Research publication SG–2301.

3.4.7 Using rcoptions to modify the actions of /etc/bcheckrc, /etc/brc, and /etc/rc

You should not modify the start-up scripts manually to alter their behavior. Instead, you must manually edit the control file that the UNICOS system uses for configuration and installation.

The control file used to alter the actions of the various start-up scripts is /etc/config/rcoptions.

The RC_LOG parameter changes the name of the log file (/etc/rc.log) used to capture output messages generated during execution of the /etc/rc script.

You may use rcoptions to do the following:

- Set the device name for the /usr, /usr/tmp, and /tmp file systems.
- Set the path name of the /etc/rc log file.
- Specify whether to run mkfs on /tmp or /usr/tmp at boot time.
- Mount the usr file system.
- Determine whether to activate ldcache.
- Determine whether to start accounting, sadc, or network.

For additional information, see *General UNICOS System Administration*, Cray Research publication SG–2301.

3.4.8 To add site-specific code to the start-up process

To add your site-specific code to the start-up process, create the following executable files; each file will be executed at a specific point during the start-up process:

File	Description
`/etc/rc.pre`	If you must do some local work before the file systems are mounted, create this executable file. The `/etc/rc` script executes the `/etc/rc.pre` file after some initial preparatory work (checking whether or not security is configured, initializing start-up logging, and so on) but before any file systems are mounted.
`/etc/rc.mid`	If you must do some local work after the user file systems are mounted, but before any daemons (except the security log daemon, `/etc/slogdemon`) are started, create this executable file. It will be executed after mounting (and optionally caching) the user file systems, starting up `/etc/slogdemon`, and preserving interrupted `vi` or `ex` sessions.
`/etc/rc.pst`	If you must do some local work after everything has been started, create this executable file. It will be executed before the `/etc/rc` script exits and returns a `Console Login:` prompt to the system console. Given that networks will already have been configured and networking daemons will have been started, `/etc/rc.pst` is not necessarily executed before any users have logged in to the system from the network. To start local daemons, configure them into the `/etc/config/daemons` file and call `/etc/sdaemons` by using the daemon name.

3.5 Run-level configuration

A *run level* is a software configuration of the system. Each run level allows only a selected group of processes to exist. Although run levels are most commonly used to configure the system in single-user or multiuser operation modes, thoughtful management of the run-level configuration on the system is a

convenient method of tailoring the system's resources to accommodate users' needs.

Two main modes of operation exist for the UNICOS system: single-user and multiuser. Single-user mode is always indicated by run level s or S. Multiuser mode is typically run level 2, although it may be level 0 through 6.

One common use of the `/etc/inittab` file is to set up a run level so that certain procedures are followed automatically only the first time a run level is entered. For example, usually you are asked to verify the date and to check the file systems the first time you change your system to multiuser mode. These actions are caused by an entry in the `inittab` file. Subsequent changes in run level do not result in this procedure automatically unless you specifically change the `inittab` file.

3.5.1 Changing run level

As system administrator, you can change the run level by issuing the following command; *level* is the run level you want to initiate:

`/etc/init` *level*

The `/etc/inittab` file controls the specific actions that occur when a run level is initiated. The following sections discuss the strategies for using run levels for various purposes.

3.5.2 Strategies for using run levels

Successful use of run levels requires that you think through the requirements for the system and tailor the initializations of the various run levels to provide for convenient transitions from one run level to another.

All systems have a single-user mode (for system work that must be performed unencumbered by the presence of other users on the system) and at least one multiuser mode. If the system is restricted at various times to dedicated use by one or more users, you should devote one or more run levels to initializing the system for this dedicated use. In all cases except for single-user mode (which requires little or no initialization), the `rc` (see the `brc`(8) man page) script performs initialization.

3.5.2.1 Single-user mode

Many system maintenance, modification, testing, configuration, and repair procedures are performed while the system is in single-user mode to protect

system users from potential instability and to ensure that user processes do not interfere with the system's work while it is in progress. Therefore, the purpose of performing any initialization before the system is in single-user mode is to ensure that the system is known to be in an idle state.

When the UNICOS system is in single-user mode, all network connections and hard-wired terminals are disabled, and only the console terminal can interact with the system. This mode of operation lets you make necessary changes to the system without doing any other processing. When the UNICOS system is in single-user mode, the # symbol (or `sn`*xxx*#) is the system prompt.

Typically, the system is brought into single-user mode either following a system boot or by using the `shutdown`(8) command. In neither case should any user processes be running after the system is in single-user mode (no user processes will have started following a boot, and `shutdown` kills all user processes before entering single-user mode). Thus, there should be no need for initialization related to user processes when the system enters single-user mode.

As an extra measure of protection against inadvertent damage done to a mounted file system by single-user mode development work or testing, you should unmount all file systems except the current `root` file system. Traditionally, users doing the system work or testing while in single-user mode mount only the partitions they require. To help with this aspect of system work, you can provide a script in `/etc` that mounts the file systems that contain system commands not usually found on the root partition (the `/usr` file system) and the `home` user file system directories of the system staff.

3.5.2.2 Multiuser mode

Traditionally, run level 2 is the system's primary run level for multiuser mode. Among the initializations generally performed for multiuser mode are the following:

- Recording system start-up time in `/etc/wtmp`.
- Mounting all file systems required for normal system operation. This includes the regular system file systems (`/usr` and `/tmp`), the file system or systems that contain the home directories' `/tmp` file system of the system's users, and other file systems that contain files to which the users must have access.
- Removing any lock files that may interfere with normal system operation (for example, a lock file for a system daemon).

- Running daemons that provide various system services. The list may include, but is not restricted to, the following:
  - `errdemon`
  - `slogdemon` (for the UNICOS multilevel security (MLS) feature)
  - `cron`
  - `tapestart` (for online tapes)
  - `syslogd`
  - `nqsdaemon` (for NQS)
- Running the `netstart` script to initialize the system's TCP/IP network connections.
- Starting system accounting.
- Moving or truncating log files (for example, `/usr/lib/cron/log` or `/usr/spool/nqs/log`) to prevent them from growing without limits.
- Allowing users to log in.

3.5.2.3 Typical tasks you can perform while in multiuser mode

The following are some typical system administration tasks that you can perform while the UNICOS system is running in multiuser mode. The most important areas to monitor include how efficiently the system is performing and the rate at which system resources are being consumed.

- Checking which file systems are mounted by using the `/etc/mount` command (see Chapter 5, page 51).
- Checking all mounted file systems to ensure that no mounted file system consumes all available free disk blocks by using the `/bin/df` command or the `/etc/fsmon` file system monitor.
- Checking the number of system users by using the `who` command. To identify idle users, enter `who -u`. To determine the number of users, enter `who | wc -l`. To generate the number of users and a list of their names, enter `who -q`.
- Informing users of system changes by using `/etc/wall` (see Chapter 8, page 217).

- Monitoring how your UNICOS system is running by using the `/usr/bin/sar` utility. The `/usr/bin/sar`(1) utility has many options used to gain information about disk performance, character list buffers, CPU performance, and IOS throughput. The most useful options for a system administrator include `-d` (disk), `-x` (IOS), and `-v` (critical internal system table sizes). For more information, see the `/usr/bin/sar`(1) man page.
- Checking all running processes by using the `ps`(1) command to determine whether a process is using an abnormally large amount of CPU time. The `-eaf` options generate a full listing for all running processes.
- Checking the contents and size of your UNICOS error logs. Usually, error logs are found in the `/usr/adm` directory. Also, ensure that the error logging daemon is executing and that IOS disk errors are being logged into the `/usr/adm/errfile` file. For log information, see Chapter 9, page 225. For details on disk error reporting, see the `/etc/errpt`(8) man page.
- Checking `mail` by using the `/bin/mail` command while logged on to `root`, or the login that receives requests to restore files. If a problem occurs, the system itself sometimes sends mail to `root`.

3.5.2.4 Dedicated system

It is sometimes necessary to provide dedicated system time so that a particularly large or time-critical job can run unencumbered by other user processes. There also will be times at which system development work requires that the system be brought up as though it were running in multiuser mode, when access to the machine is actually restricted to the system staff. To lock out all users except yourself, use `/etc/udbrstrict -r -L` *your_userid*. Do not use just `/etc/udbrstrict -r`, because this limits logins to only root, which can then be done only on the console device. For more information about the UDB `ue_permbits` field, see *General UNICOS System Administration*, Cray Research publication SG–2301.

3.6 IOS prompts, and permissible actions

You can toggle the system's console screen and keyboard between an interface to the software operating on the IOS and the UNICOS software operating on the mainframe. To toggle between the IOS and the UNICOS console interfaces, use the `CONTROL-a` two-key sequence. You may toggle between the two consoles at any time. If you toggle from one to the other, and get no response, the system to which you toggled may no longer be responding to the console interface. This could happen if that system (either the IOS or UNICOS system) has hung or

panicked. In this case, you should be able to toggle back to the original console. This section describes when you will see specific IOS prompts, what the condition(s) of the system may be at that time, and the actions that you can take.

Note: When using the CRAY J90 IOS master console, `CONTROL-a` toggles between the IOS and UNICOS prompts.

When going from the UNICOS prompt, after you press `CONTROL-a`, the prompt changes to `sn`*xxxx*`-ios0>`.

When going from the IOS prompt, the UNICOS prompt is not displayed until you press `RETURN`.

3.6.1 IOS boot prompt

The IOS boot prompt is as follows:

`BOOT[sn`*xxxx*`-ios`*x*`]>`

When you see this prompt, the following are possible system conditions:

- The IOS is down; it is running in PROM; no strategies or drivers are loaded.
- The CRAY J90 mainframe is down; the UNICOS system is not running.

When the power is turned on and after typing `reset`, you will always see the IOS boot prompt.

From this state, you can perform only the following actions:

- Take an IOS dump (not a UNICOS dump) by typing `iosdump`.
- By using the `tar` command, transfer files between the CRAY J90 system console disk and the IOS DAT (rpd03) tape drive.
- Load the IOS kernel, strategies, and drivers into memory by typing `load`. This command also starts the execution of all IOSs defined in the `/config` file.
  - The IOS kernel resides on the CRAY J90 system console disk and has the path name `/ios/ios`.
  - The IOS strategies and drivers reside on the CRAY J90 system console disk in the `/dev` directory.
  - The IOS `load` command uses the IOS configuration file `/config` to determine which strategies and drivers to load into IOS memory.

To start the IOS by loading the appropriate device strategies and drivers and loading and executing the IOS kernel, enter the `load` command at the IOS boot prompt:

```
BOOT[snxxxx-ios0]> load
```

After loading is complete, the prompt changes to the IOS prompt, which signifies that the IOS software loaded in the IOS memory is now executing instead of the PROM code.

3.6.2 IOS prompt

The IOS prompt is as follows:

sn*xxxx*-ios*x*>

When you see this prompt, the following are possible system conditions:

- The IOS is up; it is running the IOS kernel, strategies, and drivers. Any slave IOP in the IOS may or may not be running. Check the `/adm/syslog` file on the CRAY J90 system console disk for messages indicating that a slave IOS has panicked if this is suspected.
- The IOS and CRAY J90 mainframe are both up; `CONTROL-a` was pressed to change from the mainframe prompt to the IOS prompt.
- The CRAY J90 mainframe is down; the UNICOS system is not running. A mainframe system panic has occurred. The IOS is still running, however.

If a mainframe system panic occurs, the IOS may still be running, but it will be in an undefined state. Taking an IOS dump at this point may be helpful; use the `iosdump`(8) command. See the *CRAY IOS-V Messages*, Cray Research publication SQ–2172.

From this state, you can perform the following actions:

- Run diagnostics.

Note: Diagnostics should complete successfully and cause no load problems. However, if you have run diagnostics and a failure was detected or the diagnostic did not exit cleanly (for example, if you entered a `CONTROL-c` to exit a diagnostic), the system may have been left in an undefined state. This could cause the system to hang during the boot process. If you experience this problem, enter the `reload`(8) command after the IOS prompt to set the system to a known state, and then start the UNICOS system by entering the `/bin/boot` command after the IOS prompt.

- Take a UNICOS dump by entering the IOS `mfdump`(8) command.
- Flush buffers to disk, reset the VMEbus, and return the IOS to PROM (the IOS boot prompt) by entering the IOS `reset`(8) command.
- Master clear the mainframe CPUs, which stops all CPU activity, by entering the `mc` command.
- Clear central memory, as well as load and start the UNICOS system, by entering `boot`.
- Initiate a reboot of the IOS from PROM, and reload the IOS by entering `reload`.