`vi` editing	To enter this mode, enable the `vi` option. There are two typing modes in this option. Initially, when you enter a command, you are in the input mode. To edit, control mode by typing `Esc`, moves the cursor to the point needing correction, then inserts or deletes characters or words as needed as if the command line were a text file being edited using `vi`.
Job control	Lists information about each given process (*job*) or all active processes if the `job` argument is omitted. The `-l` flag lists process ID numbers in addition to the normal information. The `-n` flag displays only jobs that have stopped or exited since last notified. The `-p` flag causes only the process group to be listed. See the `ksh(1)` man page for a description of the format of the `job` argument. The `bg` command puts each specified process into the background. The current process is put in the background if `job` is not specified. The `fg` command brings each process specified to the foreground. Otherwise, the current process is brought into the foreground.

Displaying Windows on Remote Workstations

You can invoke a graphical utility or application on a remote networked workstation and direct the window and all input and output to your own workstation. This is convenient when you perform maintenance on remote workstations from your own desk. The program you invoke runs on the remote workstation, and the window is displayed on the specified display workstation.

You must allow the remote system access to your display. To do this, you can use the `xhost` command on the display workstation:

`xhost +`*`remote_workstation`*

Next, use `rsh(1)`, `rlogin(1)`, or `telnet(1)` to log in to the remote workstation with whatever privilege level is required to perform the maintenance on that system. This may be as simple as the guest account, or you may have your own user account on the system, or you may require `root` permission. Choose the level of access appropriate to your task. Then, for `csh` and `tcsh` users, issue the command:

`setenv DISPLAY` *`local_workstation:`***`0`**

Or, `bsh` and `ksh` users enter:

`DISPLAY=`*`local_workstation:`***`0 ; export DISPLAY`**

The name of the workstation where the window is to be displayed is substituted for *local_workstation*. The name of the local workstation must be found in the `/etc/hosts` file of the remote system, where the program is actually running.

Now, when you invoke the desired utility or application on the remote system the window displays on the local workstation. All input and output is handled through the local workstation. Remember that due to restrictions of network carrying capacity, response time in the program may be slower (in some cases, much slower) than usual.

When you are finished, exit the display program normally and log out of the remote system.

Creating a Custom Shell Window

The IRIX system allows you to create a shell window using any colors you like from the palette on a graphics workstation. You may also select any font you prefer from the font set on your system. The `xwsh` command creates the shell window, and the options to this command control the various fonts, colors, and other features available to you. The command shell used in the window is taken by default from the `/etc/passwd` file entry, or it can be specified on the command line according to the instructions in the `xwsh(1)` man page.

For a complete list of the features available with xwsh, see the `xwsh(1)` man page. The most commonly used features are described in the following examples.

To create a simple shell window with a dark gray background and yellow text, issue the following command:

```
xwsh -fg yellow -bg gray40 &
```

The above command generates a new window and a new shell using the colors specified. The window uses the default font selection and window size, since these attributes were not specified. The command that created the shell was placed in the background, so the shell does not tie up the window where you gave the command. You can always place a command in the background by adding the ampersand character (&) to the end of the command line. For more information on placing processes in the background, see the `csh(1)` man page.

Setting and Changing System Defaults

These system-wide defaults affect programs and system functions:

- System display
- Processor allocation
- Time zone
- Name of the system
- Network address
- Default system printer
- System date and time
- File and directory access permissions
- Access control lists (ACLs) and capabilities

These defaults are described more thoroughly in specific sections of this guide, but they are all presented here to provide an overview of the IRIX system.

Changing the System Display

You can make the output of programs and utilities running on one system appear on the screen of another system on the same network by changing the `DISPLAY` environment variable. This is useful if your network includes graphical systems and non-graphical servers. In order to view information from the server graphically, you reset the display to a graphics workstation.

For example, if your server has only a character-based terminal as its console and you wish to run `gr_osview` to visually inspect your CPU usage, you would issue this command on the server (for `csh` and `tcsh` shells):

```
setenv DISPLAY graphics_system:0
gr_osview
```

or this command for `ksh` and `sh` shells:

```
DISPLAY=graphics_system:0; export DISPLAY
```

Also issue an `xhost` command on the graphics system to allow the server to display on it:

`xhost +`*`server_system`*

When you invoke `gr_osview` on the server, the window with the output will appear on the graphics system name you specify. In this example, *graphics_system* was used in place of the system name. The `:0` used after the system name indicates that display monitor 0 (the graphics console) should be used to display the output. When you have finished using the graphics console, be sure to reset the display by issuing this command on the server:

`setenv DISPLAY` *local_server*`:0`

local_server is the name of your server. If you logged in from the graphics system to the server and set the `DISPLAY` variable that way, simply log out when you are finished.

Changing Processor Assignment on Multiprocessor Systems

If you have a multiprocessor system, the `mpadmin(1M)` command and the Miser suite of commands allow you to change the way programs are assigned to the various processors on your system. To determine if your system is multiprocessor, use the `hinv(1M)` command. A multiprocessor system returns information similar to the following in its `hinv` output:

```
Processor 0: 40 MHZ IP7
Processor 1: 36 MHZ IP7
Processor 2: 40 MHZ IP7
Processor 3: 40 MHZ IP7
Processor 4: 40 MHZ IP7
Processor 5: 40 MHZ IP7
Processor 6: 40 MHZ IP7
Processor 7: 40 MHZ IP7
```

Or, alternately, output similar to the following:

```
8 40 MHZ IP7 Processors
```

A single-processor system returns information similar to the following for the `hinv` command:

```
1 100 MHZ IP22 Processor
```