Unix is composed of two main layers, a kernel and a shell. The kernel is the internal layer or core program that manages the computer system. Few users ever communicate directly with the kernel. Instead, users send instructions to the shell, an outer layer of software that interprets and executes commands, passing on to the kernel any further processing that may be required. For example, a user may give the lpr command to print a file, which the shell interprets and executes. However, the actual connection and transmission to the printer itself is handled by the kernel.

When you login, the operating system starts a C shell for you. The characters you type are entered into a command-line buffer. When you press the RETURN key at the end of a line, the operating system accepts the contents of the buffer as the command and then interprets it. If it cannot tell what the command means, then it delivers an error message back to the user. For example, if you type copy instead of cp when you try to copy a file, you will get the message

    % copy: Command not found

A command to the shell, or shell command, actually tells the shell to run a program. The cp command tells the shell to run the copy program.

The user enters instructions and commands to the shell at the shell prompt. The percent character (%) in the prompt is not an arbitrary character. It stands for the particular kind of shell the system is running, in this case, a C shell, or tsch, the enhanced version of the Berkeley Unix C shell. If the symbol were a dollar sign ($), then it would mean that a Bourne shell is running.

More than one shell can run at a time. Each time the user opens an Xterm terminal window, a new shell is created to accept user input.

Your environment is determined by the behavior of the shell and other programs that interact with it. The shell has a facility for storing data in variables. There are two categories of variables: environment variables and shell variables. Environment variables are “global,” that is, the values or settings for these variables are the same throughout all the shells you create. Shell variables are specific to each shell; in other words, each window has its own shell variables that can be set.
**Environment Variables (printenv, setenv)**

Environment variables contain information about your working environment. These are set by system administrators but can be changed by the individual user. Environment variables determine the default operation of all shells and are also passed on to application programs. The list of environment variables is fairly lengthy and includes settings for a default printer, a default editor, a default path, etc. To list them, type the command `printenv`.

The format for setting the values of variables is the command `setenv` followed by the *variable* name in caps and then the *value* the variable should receive. Some values must be placed in quotation marks, particularly those that have embedded spaces.

**Shell Variables (set)**

Shell variables work like environment variables, except for the following differences.

- A set of shell variables is used by a single C shell only.
- Shell variables do not propagate to new shells or other programs.
- Shell variable names are written in lowercase.
- Shell variables are displayed, set, and changed with the `set` command.

Generally, shell and environment variables are independent, although they may look and behave similarly, both in name and purpose. Shell variables may also obtain their initial values from environment variables. However, once values are set for a shell, they will not be affected by changes in environment variables. Setting shell variables is a way of overriding environment settings in a shell.

To change a value for a variable, use the `set` command followed by the *variable* name (in lowercase), an equals sign (=), and then the *value* to be assigned to the variable.

For example, to replace `eos%` or `unity%` with `eos/unity%`

```bash
set prompt=eos/unity%
```

Or, to show the directory where you are working as a prompt:

```bash
set prompt="~/`
```

Place the new `set prompt` command in your `.mycshrc` file to have it come up with every window (see *Customizing*).