Configuring Your Login Session

When you log into UNIX, you are running a program called a shell. The shell is the program that provides you with the prompt and that submits to the computer commands that you type on the command line. This shell is highly configurable. It has already been partially configured for you, but it is possible to change the way that the shell runs.

Many shells run under UNIX. The shell that SSCC users use by default is called the `tcsh`, pronounced "Tee-Cee-shell", or more simply, the C shell. The C shell can be configured using three files called .login, .cshrc, and .logout, which reside in your home directory. Also, many other programs can be configured using the C shell's configuration files. Below are sample configuration files for the C shell and explanations of the commands contained within these files. As you find commands that you would like to include in your configuration files, use an editor (such as EMACS or nuTPU) to add the lines to your own configuration files.

Since the first character of configuration files is a dot ("."), the files are called "dot files". They are also called "hidden files" because you cannot see them when you type the `ls` command. They can only be listed when using the `-a` option with the `ls` command. Other commands may have their own setup files. These files almost always begin with a dot and often end with the letters "rc", which stands for "run commands". See documentation for those commands to determine how the configuration files are used.

The .cshrc Configuration File

The .cshrc file is the first configuration file that the C shell executes when it is invoked. Also, the .cshrc file is executed at any other time that a shell is created. For instance, if you run the UNIX `zcat` command from within a SAS program, the .cshrc file will be executed. For this reason, follow one important rule when using .cshrc: *NEVER put any command in this file that will print anything to the screen automatically!* A common mistake is to put the `date` command in the .cshrc file. This will corrupt your data when reading compressed data into SAS, for example. In the examples below, no commands print to the screen.

Although you can put any command in the .cshrc file that you can execute on the command line, there are three commands that ordinarily only are used in the setup files. They are:

`set`	sets local variables, to configure the shell
`setenv`	sets environmental variables, to configure other programs
`alias`	creates aliases, which are alternative command names

The `set` and `setenv` commands set variables for the shell and for other programs. A variable is simply a selection among choices of ways to operate. In general, the `set` command configures the C shell itself, while the `setenv` command configures other programs. It is not important to know when to use `set` and when to use `setenv`. In the examples below, the appropriate command is used and you can copy the command from

the examples to your own configuration file. There are many predefined variables used by the shell. See the manual entry for csh(1) and for tcsh(1) for a detailed list of variables. The example .cshrc file contained later in this handout illustrates some of the commonly used predefined variables.

The `alias` command creates alternate command names. This is most commonly used when you have a command that you use often, but is long or difficult to remember. This alias that you define then becomes a command name that runs the original command. For example, the command:

```
alias ll "ls -l"
```

establishes an alias so that whenever you execute the `ll` command, the system will execute the `ls -l` command.

Below is an example of a .cshrc file. After this example is a line by line discussion of the .cshrc file.

```
 1  # @(#) ~/.cshrc: an easily customizable .cshrc file for home directories
 2  #
 3  # Written by Pete Keller on August 25, 1997.
 4  # Last modified by Ralph Rodriguez on October 20, 1997.
 5  #
 6  # Please read the descriptions of the options and comment them in and
 7  # out to customize your shell.
 8
 9  # Allows you to source your .cshrc file without getting too long a path:
10  if ( ! "$?origpath" ) then
11          setenv origpath "$path"
12  endif
13
14  # Add new elements to your path here.
15  # The dot, '.', meaning the present working directory, is always LAST
16  # for security reasons. Note that removing dot altogether from your path
17  # greatly improves your security against possible attacks by others.
18  set path = ( ~/bin $origpath . )
19
20  # If this is an interactive session, do a lot of work.
21  if ($?prompt) then
22
23          # Keep a history of 100 commands.
24          set history = 100
25
26          # Save the history list when logging out, up to 100 commands:
27          set savehist = 100
28
29          # Notify user immediately when a background process is completed
30          # or blocked.  If not set, user will be notified the next time
31          # the prompt is displayed.
32          set notify
33
34          # Screen should blink, instead of beep, when sending warning:
35          set visiblebell
36
37          # Ask user to confirm before execution of "rm *" command.
38          set rmstar
39
40          # Complete partially typed filename when Tab or Esc is hit and
```

```
41          # display choices if there are more than one possibility:
42          set autolist
43
44          # Correct incorrect path names when Tab is hit.  This is not
45          # very reliable, but can be helpful, occasionally:
46          set autocorrect
47
48          # Sets the time interval to check mail and the path to the mail
49          # directory:
50          set mail=(300 /var/spool/mail/$USER)
51
52          # Set the prompt: You may set the prompt to whatever you desire.
53          # Below is a default prompt for you. Also, included is a list of
54          # valid constructs to be used with the prompt variable.
55          #
56          #       %n      User name
57          #       %~      Current working directory (~, if home directory)
58          #       %h      Current history event number
59          #       %M      Full machine hostname.
60          #       %m      Hostname up to the first "."
61          #       %S      Start standout mode (highlighted)
62          #       %s      Stop standout mode
63          #       %U      Start underline mode
64          #       %u      Stop underline mode
65          #       %t      Current time of day, in 12-hour, am/pm format
66          #       %T      Current time of day, in 24-hour format
67          #       %d      Weekday in <Day> format
68          #       %D      Day in dd format
69          #       %w      Month in <Mon> format.
70          #       %W      Month in mm format.
71          #       %y      Year in yy format.
72          #       %Y      Year in yyyy format
73          #
74          # Example of a simple prompt:
75          #
79          #       set prompt = "%m "
77          #
78          # But a more useful prompt is:
76          set prompt = "%n@%m:%~ <%h> "
80
81          # Set default permissions for newly created files.
82          #
83          #                       Will lead to default permissions
84          #       A umask of:     for directories:        for files:
85          #
86          #       002             drwxrwxr-               -rw-rw-r–
87          #       022             drwxr-xr-x              -rw-r--r–
88          #       027             drwxr-x–              -rw-r-----
89          #       007             drwxrwx–              -rw-rw----
90          #       077             drwx------              -rw-------
91          #
92          # Make it restrictive for the user safety:
93          umask 027
94
95          # General aliases.  These are for your convenience.  Change them
96          # if you desire:
97          alias a         "alias"
98          alias c         "clear"
99          alias clean    'rm core *~ *.BAK *.bak #*# .*~ *.ixx'
100         alias h         "history 25"
101         alias lf        "ls -FC $*"
```

```
102          alias ll        "ls -l $*"
103          alias mail      Mail
104          alias pd        pushd
105          alias pop       popd
106          alias so        "source ~/.cshrc"
107          alias f         finger
108          alias vt100     "set term = vt100"
109
110          # Allow logout to work in subshells:
111          alias logout exit
112
113          # Force the rm, mv and cp commands (remove, move, and copy) to
114          # ask you before overwriting files:
115          alias rm 'rm -i'
116          alias mv 'mv -i'
117          alias cp 'cp -i'
118
119          # Aliases for printing:
120          alias double           "lpr -Puserdouble"
121          alias single           "lpr -Pusersingle"
122          alias twopage          "lpr -N2 -Puserdouble"
123          alias smallprint       "lpr -1Courier7 -Puserdouble"
124          alias landscape              "lpr -Olandscape -Puserdouble"
125
129          alias shortprompt 'set prompt = "<%h> "'
130          alias longprompt  'set prompt = "%n@%m:%~ <%h> "'
131
132          # Aliases for setting the terminal keys:
133          alias  back    'xmodmap -e "keysym 0xffff = BackSpace"'
134          alias  escape  'xmodmap -e "keysym 0xffc8 = Escape"'
135
136          # Use xrs (resize) to reset your window size after you
137          # have resized an xterm:
138          if ($TERM == xterm) then
139          alias xrs 'set noglob; eval `/usr/local/bin/resize`;unset noglob'
140              xrs
141          endif
142
143          # Sets default editor to emacs:
144          setenv EDITOR  emacs
145
146          # Sets the pager to less and sets options:
147          setenv PAGER   /usr/local/bin/less
148          setenv LESS    "-emqs"
149
150          # Sets the path for man pages to find local man pages:
151          setenv MANPATH
            /usr/man:/usr/local/man:/usr/X11R5/man:/usr/local/newsprint/man
152
153          # Some programs, such as MH, require a signature setting.
154          # To use this, replace "Jane Doe" with your own name.
155          # Note, though, that the name MUST be in quotes:
156          setenv SIGNATURE "Jane Doe"
157
158          # The Helpline product needs these two variables set:
159          setenv LD_LIBRARY_PATH /usr/lib:/usr/local/uis/helpline/exe
160          setenv UIS_DISPLAY_INTERFACE "M"
161
162          # One of the following is needed for running STATA.  Use the
163          # "pd.X" version to run STATA on an X terminal, and use the
164          # "pd.wy99" version to run STATA on any other terminal:
```

```
165          # setenv STATAPD "b /usr/local/STATA/pd.wy99"
166          setenv STATAPD "b /usr/local/STATA/pd.X"
167
168  endif
```

The line numbers provided above are not part of the .cshrc file. They are given here only so that we can easily reference particular lines. Below are the definitions of these lines and possible alternative entries.

```
Line 1: # ~/.cshrc: an easily customizable .cshrc file for home directories
Line 2: #
Line 3: # Written by Pete Keller on August 25, 1997.
Line 4: # Last modified by Ralph Rodriguez on September 17, 1997.
Line 5: #
Line 6: # Please read the descriptions of the options and comment them in and
Line 7: # out to customize your shell.
```

Lines beginning with pound signs are ignored. Blank lines are also ignored. Note that both are used throughout this document for ease of use.

```
Line  9: # Allows you to source .cshrc file without getting too long a path:
Line 10: if ( ! "$?origpath" ) then
Line 11:         setenv origpath "$path"
Line 12: endif
Line 13:
Line 14: # Add new elements to your path here.
Line 15: # The dot, '.', meaning the present working directory, is always LAST
Line 16: # for security reasons. Note:  removing dot altogether from your path
Line 17: # greatly improves your security against possible attacks by others.
Line 18: set path = ( ~/bin $origpath . )
```

The path environmental variable tells the computer where to look for commands that you might want to execute. You may add additional locations to look for commands by adding them to the `set path` command. For example, if you have put your own commands in the scripts subdirectory of your home directory, you might want to change the `set path` line to:

```
set path = ( ~/bin $origpath ~/scripts . )
```

The dot in the path means that the shell should look in the present working directory for commands. This is not needed, generally, and you can remove it if you are very concerned about security. For example, a more secure path setting would be:

```
set path = ( ~/bin $origpath ~/scripts )
```

The lines, above, setting the origpath environmental variable exist to guard against accidentally getting an absurdly long path, particularly for those editing their .cshrc file and then running it. How it does this is not important to the novice, or even moderately experienced user. Just leave the lines as they are and only make changes to the set path line and all will work properly.

```
Line  20: # If this is an interactive session, do a lot of work.
Line  21: if ($?prompt) then
Line 168: endif
```

These cause the commands between the `then` and the `endif` to be executed only if the prompt is set. This is a standard method. Do not disturb these lines. Put your commands after the `then` and before the `endif` and do not worry about the arcane meaning of these lines.

```
Line 23: # Keep a history of 100 commands.
Line 24: set history = 100
```

If the history variable is set to a number, that number of commands will be remembered by the system. The user can then reference the command by the command number. For instance, if command number 53 was:

```
lpr -Pdouble -N2 runjob.log
```

and the user wants to repeat this command without typing the whole line again, the user may just type:

```
!53
```

and command number 53 on the history list will be executed.

```
Line 26: # Save the history list when logging out, up to 100 commands:
Line 27: set savehist = 100
```

If the savehist variable is set to a number, this number of commands will be remembered by the shell when the user logs in again. In the example above, 100 commands will be remembered when the user logs out, and those 100 commands will be on the history list when the user logs back in.

```
Line 29: # Notify user immediately when a background process is completed
Line 30: # or blocked.  If not set, user will be notified the next time
Line 31: # the prompt is displayed.
Line 32: set notify
```

Ordinarily, when a job running in the background finishes running, or is suspended for any reason, the user is not notified until the user completes a command. If "notify" is set, the user will be informed immediately that the command has completed or is suspended.

```
Line  34: # Screen should blink, instead of beep, when sending warning:
Line  35: set visiblebell
```

If set, when the system has to warn you of a problem, it will blink (visually) instead of beep (aurally).

```
Line  37: # Ask user to confirm before execution of "rm *" command.
Line  38: set rmstar
```

The commands `rm *` and `rm -r *` are dangerous commands, deleting all files in a directory. If rmstar is set, and you issue one of these commands, the shell will ask you if you are sure you want to delete all files in the current directory before it performs the deletions.

```
Line  40: # Complete partially typed filename when Tab or Esc is hit and
```

```
Line  41: # display choices if there are more than one possibility:
Line  42: set autolist
```

When you hit the tab key in the middle of typing a file name, the shell will attempt to complete the file name. If it cannot do so, and autolist is set, the shell will present you with a list of choices, a list of files that match the file name fragment that you have so far typed. This is very useful and should be left set, unless it irritates you to get these file listings when you type a tab character.

```
Line 44: # Correct incorrect path names when Tab is hit. This is not
Line 45: # very reliable, but can be helpful, occasionally:
Line 46: set autocorrect
```

If set, this will attempt to correct misspelled path names when you complete a path name by typing a tab character. Be warned that the autocorrect function will only take a good guess. Great care should be taken when using this, particularly when using it with an `rm`, `mv`, or `cp` command.

```
Line 48: # Sets the time interval to check mail and the path to the mail
Line 49: # directory:
Line 50: set mail=(300 /var/spool/mail/$USER)
```

The mail variable should be set with a number and a path name. The path name is the location of your system mail file. The number is the time interval that the shell uses to check to see if you have new mail. Do not set this to a very low number (below 120 seconds), as it will cause your shell to take up a lot of system resources, constantly checking mail. The default time, shown above, is 5 minutes, which, ordinarily, is sufficient.

```
Line 52: # Set the prompt: You may set the prompt to whatever you desire.
Line 53: # Below is a default prompt for you. Also, included is a list of
Line 54: # valid constructs to be used with the prompt variable.
Line 55: #
Line 56: #       %n      User name
Line 57: #       %~      Current working directory (~, if home directory)
Line 58: #       %h      Current history event number
Line 59: #       %M      Full machine hostname.
Line 60: #       %m      Hostname up to the first "."
Line 61: #       %S      Start standout mode (highlighted)
Line 62: #       %s      Stop standout mode
Line 63: #       %U      Start underline mode
Line 64: #       %u      Stop underline mode
Line 65: #       %t      Current time of day, in 12-hour, am/pm format
Line 66: #       %T      Current time of day, in 24-hour format
Line 67: #       %d      Weekday in <Day> format
Line 68: #       %D      Day in dd format
Line 69: #       %w      Month in <Mon> format.
Line 70: #       %W      Month in mm format.
Line 71: #       %y      Year in yy format.
Line 72: #       %Y      Year in yyyy format
Line 73: #
Line 74: # Example of a simple prompt:
Line 75: #
Line 76: #       set prompt = "%m>"
Line 77: #
Line 78: # But a more useful prompt is:
Line 79: set prompt = "%n@%m:%~ <%h> "
```

The prompt is the string of characters that you see when the shell is waiting for you to enter a command. By default, the prompt is set to the host name, but you may set it to anything you wish. For instance, if you want your prompt to be the words "Enter command here>> " you would set the prompt like this:

```
set prompt  = "Enter command here>>  "
```

Instead of seeing "norman.ssc.wisc.edu " when you are entering a command, you will see the prompt set above. The T-C-shell has an extensive set of shortcuts for important bits of information that can be used for a prompt. The comment above includes some of these shortcuts and the setting, above, sets an informative prompt:

```
"%n@%m:%~ <%h> "
```

The `%n` means the user name and the `%m` means the machine name. The `%~` is the present working directory and the `%h` is the current history number. The other characters, the `@, :, <, >`, and spaces, simply represent themselves. So if user smith is working on machine NORMAN and her present working directory is /tmp, and the command is number 107, the prompt will look like:

```
smith@norman:/tmp <107>
```

This prompt is much more informative than a prompt that merely says:

```
norman.ssc.wisc.edu>
```


and it is only a few characters longer.

```
Line 81: # Set default permissions for newly created files.
Line 82: #
Line 83: #                       Will lead to default permissions
Line 84: #       A umask of:     for directories:        for files:
Line 85: #
Line 86: #       002             drwxrwxr-               -rw-rw-r--
Line 87: #       022             drwxr-xr-x              -rw-r--r--
Line 88: #       027             drwxr-x—              -rw-r-----
Line 89: #       007             drwxrwx—              -rw-rw----
Line 90: #       077             drwx------              -rw-------
Line 91: #
Line 92: # Make it restrictive for the user safety:
Line 93: umask 027
```

The `umask` determines the default permissions that files and directories will have when they are created. You can always change the permissions, but the `umask` sets the permissions by default. Likely settings for the `umask` are listed in the comment lines above the `umask` command. A `umask` of 077 is the most restrictive, allowing only the owner to read or modify a file or directory. A `umask` of 002 is the most permissive of reasonable umasks, allowing the user and members of the user's group to read or modify files and directories and allowing all others to read the files and search the directories. Select a reasonable `umask` from the list above and use the three digit code on the `umask` line.

```
Line 95: # General aliases.  These are for your convenience.  Change them
Line 96: # if you desire:
```

```
Line 97: alias a "alias"
```

Here begin a series of aliases, new names for old commands.
The `alias` command itself can be aliased. Here, the very brief command `a` is used to declare an alias. The syntax of the `alias` command is:

`alias` *newalias "UNIX commands"*

where "newalias" is the new command name that you are creating, and "commands" are the UNIX commands that you want to be executed when the alias is used. In this example, the `alias` command itself is being aliased so that you only need to type "a" in other alias commands.

```
Line 98: alias c "clear"
```

The `clear` command clears the screen. This will probably be one of your most used commands. Making an alias for it will save many key strokes.

```
Line 99: alias clean     'rm core *~ *.BAK *.bak #*# .*~ *.ixx'
```

Many commands make backup files of various sorts. Periodically, you will want to clear out these backup files. This command will perform this task for you. It will also remove core files for you. If you use this alias, never create a file or directory called "core" (this is a good practice, even if you do not use this `clean` command).


```
Line 100: alias h        "history | tail -20"
```

To print the history list, use the "history" command. The command as written above will print the last 20 items on the history list.

```
Line 101: alias lf       "ls -FC $*"
Line 102: alias ll       "ls -l $*"
```

Typical options for the `ls` command. The `-l` flag means give a long listing. The `-FC` option puts a "/" after directory names.

```
Line 103: alias mail     Mail
```

The command `Mail` is easier to use if not capitalized. This is only useful for people who actually use the `Mail` command.

```
Line 104: alias pd       pushd
Line 105: alias pop      popd
```

Two alternate forms of the `cd` command, `pushd dirname` effectively changes directories to "dirname", but remembers the directory that you were in before the change. You can then use the `popd` command to pop back to the directory that you were in before. `pushd` can keep track of many directories. These aliases are short versions of the `pushd` and `popd` commands.

```
Line 106: alias so       "source ~/.cshrc"
```

When changing the .cshrc file, changes do not take effect until either you log back in or you force the shell to reread the .cshrc file. The `source` command is the command that rereads the .cshrc file. So that you do not have to type in the entire command, we set up and alias for the full `source` command.

```
Line 107: alias f         finger
```

Another common and useful command is `finger`. This provides a one letter shortcut for the command.

```
Line 108: alias vt100     "set term = vt100"
```

The most commonly emulated terminal is the VT100. This alias will tell the computer that you are on a VT100 terminal

```
Line 110: # Allow logout to work in subshells:
Line 111: alias logout exit
```

This ensures that you can exit out of any shell by typing `logout`.

```
Line 113: # Force the rm, mv and cp commands (remove, move, and copy) to
Line 114: # ask you before overwriting files:
Line 115: alias rm 'rm -i'
Line 116: alias mv 'mv -i'
Line 117: alias cp 'cp -i'
```

The `rm`, `mv`, and `cp` commands can be destructive. If a wrong file is removed or overwritten, it may not ever be recoverable. However, when using the `-i` option, the computer will ask you if you are sure that you want to remove, move, or copy a file. This provides the user with a moment to consider whether the proper command has been executed. These commands will force the usage of the `-i` option whenever these commands are used.

```
Line 119: # Aliases for printing:
Line 120: alias double           "lpr -Puser2double"
Line 121: alias single           "lpr -Puser2single"
Line 122: alias twopage          "lpr -N2 -Puser2double"
Line 123: alias smallprint       "lpr -1Courier7 -Puser2double"
Line 124: alias landscape        "lpr -Olandscape -Puser2double"
```

It is often useful to have short aliases for complex print commands. Above are a few examples. Here, the user prefers to get printout in the second floor user room. The user also prefers to use double-sided printouts. Change printing queues, command names, and options to suit yourself.

```
Line 129: alias shortprompt 'set prompt = "<%h> "'
Line 130: alias longprompt  'set prompt = "%n@%m:%~ <%h> "'
```

Informative prompts can get to be very long, occasionally, and very annoying. These aliases allow you to temporarily change to a short prompt, consisting of only a history event number, and then to revert to the long prompt, described above.

```
Line 132: # Aliases for setting the terminal keys:
Line 133: alias   back    'xmodmap -e "keysym 0xffff = BackSpace"'
```

```
Line 134: alias   escape  'xmodmap -e "keysym 0xffc8 = Escape"'
```

When using X terminals, sometimes it is necessary to tell the computer which keys you want to use for the backspace key and the escape key. These lines allow you to use simple aliases to map the backspace key to the proper key on your keyboard and map the escape key to the F11 function key.

```
Line 136: # Use xrs (resize) to reset your window size after you
Line 137: # have resized an xterm:
Line 138: if ($TERM == xterm) then
Line 139:  alias xrs 'set noglob; eval `/usr/local/bin/resize`; unset noglob'
Line 140:      xrs
Line 141: endif
```

When you change the size of an X terminal, the system often can tell that you have done so, and will adjust itself. However, if it ever does not do so, then this `xrs` alias can inform your terminal of the new size of the window.

```
Line 143: # Sets default editor to emacs:
Line 144: setenv EDITOR   emacs
```

Some programs need to start editors. Often, they will refer to the EDITOR variable to determine which editor to start up. If you prefer to use another editor such as `pico, tpu` or `vi`, put this in place of the `emacs` command, above.

```
Line 147: # Sets the pager to less and sets options:
Line 148: setenv PAGER    /usr/local/bin/less
Line 149: setenv LESS     "-emqs"
```


Some programs need to present information one screenful at a time. Often, they will refer to the PAGER environmental variable to determine which of the several UNIX pagers to use. The pager selected here is `less`. The `less` command looks at the LESS variable to determine which options to use by default. Other pagers include `more` and `pg`.

```
Line 150: # Sets the path for man pages to find local man pages:
Line 151: setenv MANPATH
            /usr/man:/usr/local/man:/usr/X11R5/man:/usr/local/newsprint/man
```

Reference pages, commonly called manual pages, or, most colloquially, man pages, are located in several different places. The MANPATH variable tells the `man` command where to look to find these reference pages on line.

```
Line 153: # Some programs, such as MH, require a signature setting.
Line 154: # To use this, replace "Jane Doe" with your own name.
Line 155: # Note, though, that the name MUST be in quotes:
Line 156: setenv SIGNATURE "Jane Doe"
```

As the comment above states, some programs, such as MH, require a signature setting. Place this line in your .cshrc file, replacing "Jane Doe" with your name in quotes.

```
Line 158: # The Helpline product needs these two variables set:
Line 159: setenv LD_LIBRARY_PATH /usr/lib:/usr/local/uis/helpline/exe
Line 160: setenv UIS_DISPLAY_INTERFACE "M"
```

SSCC uses the Helpline product, called uhl, to report and track problems. You can use this product to report problems you find and to track problems previously reported. However, the `uhl` command requires some variables to be set. Above are the proper settings.

```
Line 162: # One of the following is needed for running STATA.  Use the
Line 163: # "pd.X" version to run STATA on an X terminal, and use the
Line 164: # "pd.wy99" version to run STATA on any other terminal:
Line 165: # setenv STATAPD "b /usr/local/STATA/pd.wy99"
Line 166: STATAPD "b /usr/local/STATA/pd.X"
```

When using graphs in STATA, the system needs to know whether you are using an X terminal or a terminal without graphics capability (or known graphics capability). The pd.X setting is used for X terminals and the pd.wy99 setting is used for non-graphics terminals (such as a PC at your home). If you work primarily from X terminals, leave these lines as they are. If you work primarily on a PC, comment out the line ending in pd.X and uncomment the line ending in pd.wy99.

The .login Configuration File

The .login file is executed when you first log in. It is executed after the .cshrc file. The difference between the .cshrc file and the .login file is that the .login file is only executed at login time, whereas the .cshrc file is executed whenever a new shell is spawned for any reason.

Generally, little is placed in a .login file. It is used primarily for commands that print to the screen. For example, stty commands, which set terminal settings, are placed in the .login file. Also typical is a `date` command. Below is a sample .login file:

```
 1  # @(#) ~/.login: an easily customizable .login file for home directories
 2  #
 3  # Written by Ralph Rodriguez on October 21, 1997.
 4  #
 5  # Please read the descriptions of the options and comment them in and
 6  # out to customize your shell.
 7
 8  # Items within this if/then/endif structure only happen if the
 9  # TERM environmental variable is set.
10  if ($?TERM) then
11
12          # Notifies you if you receive mail:
13          biff y
14
15          # These commands place the name and present working
16          # directory in the title bar of your xterm.
17          if ($TERM == xterm) then
18            alias icon_name 'set icon_name = /`echo $cwd | sed -e s-.\*/--`'
19alias mytitle 'icon_name; echo -n ^[]1;$icon_name\^G^[]2\;$HOSTNAME\: $cwd\^G'
20                  alias name 'echo -n ^[]0\;\!*\^G'
21                  alias icon 'echo -n ^[]1\;\!*\^G'
22                  alias title 'echo -n ^[]2\;\!*\^G'
23                  mytitle
24                  alias cd 'chdir \!*; mytitle'
25                  alias pushd 'pushd \!*; mytitle'
26                  alias popd 'popd \!*; mytitle'
27          endif
28  endif
```

```
29
30
31  # Do not allow creation of core files:
            32  limit coredumpsize 0
            33
            34  # Print the date and time:
            35  date
            36
            37
```

Again, line numbers are just provided for convenience. They can be ignored.

```
Line 1: # @(#) ~/.login:an easily customizable .login file for home  dirs
Line 2: #
Line 3: # Written by Ralph Rodriguez on October 21, 1997.
Line 4: #
Line 5: # Please read the descriptions of the options and comment them in and
Line 6: # out to customize your shell.
Line 7:
```

As with the .cshrc file, lines beginning with pound signs are considered comments and are ignored. Blank lines are also ignored.

```
Line  8: # Items within this if/then/endif structure only happen if the
Line  9: # TERM environmental variable is set.
Line 10: if ($?TERM) then
Line 28: endif
```

The commands within the if/then/endif structure are only executed if the TERM environmental variable is set.

```
            Line 12: # Notifies you if you receive mail:
            Line 13: biff y
```

The `biff` command notifies you if you receive mail.

```
Line 15: # These commands place the name and present working
Line 16: # directory in the title bar of your xterm.
Line 17: if ($TERM == xterm) then
Line 18:     alias icon_name 'set icon_name = /`echo $cwd | sed -e s-.\*/--`'
Line 19:     alias mytitle 'icon_name; echo -n
            ^[]1\;$icon_name\^G^[]2\;$HOSTNAME\: $cwd\^G'
Line 20:         alias name 'echo -n ^[]0\;\!*\^G'
Line 21:         alias icon 'echo -n ^[]1\;\!*\^G'
Line 22:         alias title 'echo -n ^[]2\;\!*\^G'
Line 23:         mytitle
Line 24:         alias cd 'chdir \!*; mytitle'
Line 25:         alias pushd 'pushd \!*; mytitle'
Line 26:         alias popd 'popd \!*; mytitle'
Line 27: endif
```

These lines will place your login name and your present working directory in the title bar of your xterm.

```
            Line 31: # Do not allow creation of core files:
            Line 32: limit coredumpsize 0
```

Sometimes when a command fails in a severe manner, it creates a file called "core". These files are images of the area of memory that the command was using. While extremely helpful to a developer, they are not useful to the average user. This `limit` command causes core files not to be saved, which may save a lot of disk space.

```
Line 34: # Print the date and time:
Line 35: date
```

People often like to have the current date and time displayed when they first log in. This simple `date` command will display this information.

The .logout Configuration File

The .logout file is executed when you log out. Generally, it is little used, often containing only the `date` command so that you can see the time that you are logging out. The sample .login file above could easily be a .logout file. Below is a sample .logout file.

```
 1 # @(#) ~/.logout: an easily customizable .login file for home directories
 2 #
 3 # Written by Ralph Rodriguez on October 21, 1997.
 4 #
 5 # Print the date:
 6 date
 7 #
 8 # Pause before logging out so that the date can be read:
 9 sleep 5
```

```
Line 1: # @(#) ~/.logout: an easily customizable .login file for home dirs
Line 2: #
Line 3: # Written by Ralph Rodriguez on October 21, 1997.
```

Again, blank lines and lines beginning with pounds signs are ignored.

```
Line 5: # Print the date:
Line 6: date
```

Display the date to the screen when you log out.

```
Line 8: # Pause before logging out so that the date can be read:
Line 9: sleep 5
```

Pause for five seconds after displaying the date, but before exiting. Exiting will clear the screen and you want to be able to see the date.

Changing Configuration Files

As stated earlier, if you change a configuration file, the changes will not take effect until you log in again. This can be overcome by forcing the C shell to read the configuration file. The command to do this is the `source` command. If you change the .cshrc file and want the changes to take effect immediately, execute the command:

```
source .cshrc
```

Learning More

For additional information on this topic, see the reference pages for the tcsh and the csh. The following commands will display the reference pages:

```
man tcsh
man csh
```

The example files used in this handout can be found in:

/usr/global/etc/sample.cshrc
/usr/global/etc/sample.login
/usr/global/etc/sample.logoff