|  |
| --- |
| [What Are They?](http://www.injunea.demon.co.uk/pages/page207.htm#10-1) [Why Use Them](http://www.injunea.demon.co.uk/pages/page207.htm#10-2) [Example Useful Profiles](http://www.injunea.demon.co.uk/pages/page207.htm#10-2-1) [Example Cron Test Function](http://www.injunea.demon.co.uk/pages/page207.htm#10-2-2) [Example Raw Cron Test](http://www.injunea.demon.co.uk/pages/page207.htm#10-2-3) [Example Setting User Environment](http://www.injunea.demon.co.uk/pages/page207.htm#10-2-4) [Environment](http://www.injunea.demon.co.uk/pages/page207.htm#10-3) [Example Environment Variables](http://www.injunea.demon.co.uk/pages/page207.htm#10-3-1) |
| **Startup Files & Environment** |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Now lets just take a timeout from the syntax rules and have a quick look at startup files.  **[What Are They?](http://www.injunea.demon.co.uk/pages/page207.htm" \l "top)**  Start-up Files are files that are read by the script at the start of execution. There is a set of editable default files supplied with UNIX, or you can create you own and force the shell to read these instead, or indeed, as well as. Again there is a difference between C Shell and the rest. The C Shell will read two files by default at login, then one each time the C Shell is invoked. The files are **.login** and **.cshrc** (note the leading *dot*) and both files do not require their execution bit to be set. They are only *read* by the shell and the commands contained within the files are executed within the current shells' environment. It is in these files that environment variables usually get set. If you want to force another file to be read at shell start up, you can use the source command within a script thus:  **source .my\_environment**  Which will read the file **.my\_environment** into the current shell and then execute the commands found just as if they had existed in the executing script.  For the **sh** & **ksh** there is only one default file called **.profile** and it is only run at login time. If you want a start-up file to be run at every execution of these shells, or at the start of any shell script, you must force it with the dot (.) command as shown below:  **. .my\_profile**  Now this is what I call an abbreviation! It can be very easy to overlook such an innocent looking command, which is one reason I always include it as the first thing after the banner and give the read file such an obvious name. As my default environment is the C Shell, I just use .profile for the **sh** & **ksh** start-up files (yes they can and do share a common file - just put the fancy **ksh** specific stuff in a file that **sh** will not see). In all cases, nesting the start-up files is allowed and I am not aware of any nesting limits within reason.  **[Why Use Them?](http://www.injunea.demon.co.uk/pages/page207.htm" \l "top)**  Well, it's nice to know that when you call on previously written utilities from within a script, that they will be found. Paths therefor are a common feature of the start-up files. There are other bits of information that you may initially think are not so useful but are indeed called upon regularly from within scripts. The Example 10.2.1 shown below is a chunk from my own .profile to show some useful features which most of my other scripts rely on.  **[Example useful profiles](http://www.injunea.demon.co.uk/pages/page207.htm" \l "top)**  **export COMPANY='ACME'**  **export THISHOST=`uname -n`**  **export MASTER\_DB='mdb\_\_001'**  **export SCRIPTS=/$COMPANY/$THISHOST/unix/cen/scripts**  **export SQL=$SCRIPTS/sql**  **export PRINTER='hplj5`**  **if [ `tty | grep -ci not` -eq 0 ]**  **then**  **stty ERASE ^H**  **TERM=SUN**  **fi**  Here you see the export statement used before the setting of the variable, which has the effect of exporting the variable into the parent environment. On the second line there is a command called **uname**which is executed (by enclosing in *back-quotes*) with an option**-n** to give the name of the executing host machine. The output from this command is passed to the environment variable THISHOST. Then on line 4 you can see this same variable supplying its value in the setting of the SCRIPTS variable. This is then used to help set the SQL variable on the next line. There is a literal string 'ACME' on line 1, and another on lines 3 and 6 - indicated by the single forward facing quotes.  The actual .profile that I use is now in several parts for ease of editing. The parts are:   * .profile Always called * .profile\_functions Always called (now in 6 sub-files) * .profile\_oracle Always called * .profile\_paths Always called * .profile\_cron Called from cron jobs only   The first file is called from within the script using the obvious call:  **. $HOME/.profile**  The .profile itself ends with three calls to **.profile\_functions**, **.profile\_oracle**, and **.profile\_paths**. All cron job scripts contain a further call to **.profile\_cron** which contains paths specific for the cron jobs and some additional security features that stop the cron scripts from being executed by a user.  A word of warning about start-up files. Don't put commands in here which will output things to the screen (**echo, cat,**etc.) unless it is for debugging purposes only. Output from the start-up files will interfere with **rsh**,**ftp** and most other remote access procedures. Output will also get sent to USER mail from a script run from the **crontab**(the UNIX batch queue). Now this is a useful feature because it can be used as an alert mechanism. But overloading the facility with all kinds of garbage messages from start-up files soon gets very annoying and eventually you will start deleting the mail before actually reading it, which is no good at all.  Also, if you are one of those for whom the backspace key should deliver a <CTRL-H> character (as I am), then you no doubt have an **stty erase ^H** statement hanging around in your start-up files somewhere. Beware of **tty** and **stty** statements if there's a chance that a cron job will pick them up. The cron processor has no terminal, so references to terminal set-up statements can create disruptive messages in the USER mailbox from cron. The way I cope with this is to have a user defined function in my **.profile\_functions** called **s\_is\_cron()** which contains the line:  **tty | grep -ci not**  This function will return the value 0 for a script running with a terminal, and 1 for a cron script running in background. This can then be used in an **if** statement to protect several terminal commands as follows:  **[Example cron test function](http://www.injunea.demon.co.uk/pages/page207.htm" \l "top)**  **if [ `s\_is\_cron` -eq 0 ]**  **then**  **stty erase ^H**  **set TERM=SUN**  **fi**  This same example is tagged onto the end of Example 10.2.1 above but without relying on a function call. For the C Shell (where there is no function definition statement - another good reason not to use it) the equivalent syntax would be as shown in Example 10.2.3 below.  **[Example raw cron test](http://www.injunea.demon.co.uk/pages/page207.htm" \l "top)**  **if ( `tty | grep -ci not` != 0 ) then**  **stty erase ^H**  **setenv TERM SUN**  **endif**  Of course, if you are going to use functions, then you have to make sure that you load the definition of the function before you call it for use. Usually these personalised sections are placed last, and a good way to do this is to put it all in a separate file and call it last in the start-up sequence. I have my own personal file which contains my own set of aliases and other short-cuts. They are all in the file **.my\_defaults** in my home directory. The first line of this file says source **$PROJECT/.cshrc** which picks up the departments defaults for this project, the rest of the file adds my own C Shell tweeks. Remembering of course that all the C Shell stuff will be forgotten as soon as a **sh** or**ksh** script starts in its own sub-shell.  Another trick when connecting to a shared account (Several DBA's may have shared access to the dba account for creating databases etc.) is to set your own personalised environment from the following test (shown in C Shell as it would appear in a shared **.cshrc**file):  **[Example setting user environment](http://www.injunea.demon.co.uk/pages/page207.htm" \l "top)**  **#======================================**  **# Set Up Dave's Environment**  **#======================================**  **set dave\_1 = `who am i | grep -c 'dwsun3'`**  **set dave\_2 = `who am i | grep -c '19.123.123.123'`**  **set dave = `expr $dave\_1 + $dave\_2`**  **if ( $dave == 1 ) then**  **stty erase ^H**  **setenv EDITOR emacs**  **setenv DISPLAY 19.123.123.123:0.0**  **setenv PRINTER hplj2**  **endif**  Depending on how Dave logged in, the **who am i** command will return either a machine name where Dave connected from, or the IP address if the local **/etc/hosts** file has no record of Dave's terminal to look up the name mapping. So one of the two variables will be set to one. Add them together and the next variable will always be set to one whenever Dave logs in. From here the test is simple and all Dave's personal settings can be set-up. For some real world examples of **.profile** and **.cshrc** files, see Appendix A which contains listings from my own system.  **[Environment:](http://www.injunea.demon.co.uk/pages/page207.htm" \l "top)**  As well as the variables you set yourself or export into the environment using the **export** command, there are some *environment variables* which get set when you launch or execute a **sh** script. We covered the special variables like *$$* and the *positional parameters* (*$digit*) in [Parameters](http://www.injunea.demon.co.uk/pages/page204.htm#7-2-3), but there are some more. There is a set that you collect at login time from the system defaults and another set supplied by the shell, as it is invoked. Some of these can be updated by the shell, some cannot. The following list is the default set showing where they are set and their default values.  **[Example environment variables](http://www.injunea.demon.co.uk/pages/page207.htm" \l "top)**   |  |  | | --- | --- | |  HOME | Is set at login to your *home directory*and is used as the default argument for the **cd** command. This can be re-set. | |  PATH | The *command search path* is set at login and updated when a shell is invoked to reflect the path for that shell. This is volatile and can be updated unless the shell is restricted (usr/bin/rsh), in which case it is fixed. | |  CDPATH | The *path* of the **cd** command. Needed in case you screw up the PATH! | |  PS1, PS2 | The primary and secondary *prompt* strings. Initialised by the shell. Can be revised if required (usually: root = #, user = $). | |  IFS | Internal Field Separators (normally *space, tab and newline*). The **cut -d** option temporarily changes the IFS for the duration of the **cut** command. | |  SHELL | If set and if the contents include the string **rsh**, shell will become the restricted shell (usr/bin/rsh). | |  SHACCT | If set to a filename writable by the user, the shell will write an account record in the file for each shell procedure executed. | |  MAIL | + | |  MAILPATH | + | |  MAILCHECK | Are all related to the mail system you are using. See the man pages for the mail command you are using for details. | |  LC\_TYPE | + | |  LC\_MESSAGE | Are related to the local language and character sets in use. They allow messages and codes to be displayed in the local language and character set. |   The shell provides defaults for PATH, PS1, PS2, IFS, and MAILCHECK. While the user login service provides defaults for HOME and MAIL. The SHELL variable is not changeable at any time and is set at shell invocation time. The PATH variable becomes immutable if the SHELL contains a reference to the restricted shell. In the restricted shell certain other commands will also become unavailable, but which ones rather depends on your system administrator! Be nice and you may get a few more. | |
| [Home](http://www.injunea.demon.co.uk/index.htm) [Next](http://www.injunea.demon.co.uk/pages/page208.htm) [Preface](http://www.injunea.demon.co.uk/pages/page201.htm) [Introduction](http://www.injunea.demon.co.uk/pages/page202.htm) [Basic Shells](http://www.injunea.demon.co.uk/pages/page203.htm) [Shell Syntax](http://www.injunea.demon.co.uk/pages/page204.htm) [Built-In Commands](http://www.injunea.demon.co.uk/pages/page205.htm) [Command Substitution](http://www.injunea.demon.co.uk/pages/page206.htm) [Startup & Environment](http://www.injunea.demon.co.uk/pages/page207.htm) [Pipes, Lists & Redirection](http://www.injunea.demon.co.uk/pages/page208.htm) [Input & Output](http://www.injunea.demon.co.uk/pages/page209.htm) [Using Files](http://www.injunea.demon.co.uk/pages/page210.htm) [Design Considerations](http://www.injunea.demon.co.uk/pages/page211.htm) [Functions](http://www.injunea.demon.co.uk/pages/page212.htm)[Debugging](http://www.injunea.demon.co.uk/pages/page213.htm) [Putting It All Together](http://www.injunea.demon.co.uk/pages/page214.htm) [Appendix](http://www.injunea.demon.co.uk/pages/page215.htm) [Code Examples](http://www.injunea.demon.co.uk/pages/page201.htm#bottom) |
| [*Page 207*](http://www.injunea.demon.co.uk/pages/page207.htm#top) |
| This page was brought to you by [rhreepe@injunea.demon.co.uk](mailto:rhreepe@injunea.demon.co.uk) |