Advanced Shell Usage I

- **Built-ins**
- **Different Shell Versions**
- **Ways a Shell Can Be Created**
- **Your Login Shell**
- **Interactive Non-login Shells**
- **Non-interactive Shells**
- **Creating Startup Files**
- **Running a Startup File *after* a Change has been Made**
- **Commands that are Symbols**

Built-ins

- Not all commands can be found on disk as executable files; some are actually contained i*n the shell itself*
- Such commands are called **built-ins**
- When you run a built-in, the shell does not have to create a new process
  - Instead, the shell calls a procedure in its own code to perform the task, so no sub-process is created
  - This makes execution faster

Built-ins

- If there is an executable file with the same name as a built-in, then the shell will run the ***built-in*** instead of the file; for example -- ***echo***
- ***echo*** is a built-in
- There is also an executable version of ***echo*** on the disk, that you can see with the **which** command

```
$ which echo
/bin/echo
```

Built-ins

- If you want to run the *disk* version of **`echo`** you have to specify the *pathname*

```
$ /bin/echo foo
foo
```

- Most built-ins have no executable counterpart on disk
- Running **`which`** on such a command will find *nothing*

```
$ which bg
$
```

Built-ins

- The ***type*** command will confirm this

```
$ type bg
bg is a shell builtin
```

- ***type*** is also a built-in

```
$ type type
type is a shell builtin
```

Different Shell Versions

- The shell we have been using is ***Bash***
- Bash stands for **B**ourne **a**gain **sh**ell
  - The *original* Bourne shell was written by Steve Bourne at AT&T's Bell Laboratories
  - The original Bourne shell -- the ***sh*** shell – is still with us.
- Many scripts are needed to set up and maintain Linux and Unix

Different Shell Versions

- Many of these scripts are quite old and were written before the Bash shell
  - Those scripts are still in use
  - There are subtle differences between different shells
  - It's best to run a script in the shell *for which it was written*
- Debian Linux and its offshoots use a stripped-down version of Bash called ***Dash***
  - Dash is much smaller than Bash

Different Shell Versions

- Dash is designed only to run scripts and has no interactive features
- Its memory footprint is small, so it loads and executes scripts faster than Bash

- *System V Unix* introduced the ***Korn*** shell, which was written by David Korn
  - It introduced *aliases* and **command line editing**
  - It also introduced other features that are now found in Bash

Different Shell Versions

- A standard exists for how shells should run on Unix that specifies how they must work
  - It was created by Portable Application Standards Committee of the ***IEEE*** (**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers)
  - It is called ***POSIX*** (Portable Operating System Interface) ***1003.2***
- The GNU community is working on making Bash fully compliant with POSIX
  - Until then you can run ***bash*** with the **`--posix`** option
  - This will make ***bash*** more compatible with the POSIX standard

Ways a Shell Can Be Created

- There are three ways a user can run a shell
  - Login shells
  - Interactive non-login shells
  - Non-interactive shells
- There are subtle differences between these three types
- We'll concentrate on ***login*** shells in this course, but you should *be aware* of the existence of the other shell types

Your Login Shell

- When you first login to Unix, you are r*unning a shell*
- This shell is your *login shell*
- Which shell version you run is determined by the **SHELL** system variable

```
$ echo $SHELL
/bin/bash
```

- In Ubuntu, the default shell version is ***Bash***

Your Login Shell

- When your login shell starts up, it runs the commands found in `/etc/profile`
- This is a file customized by the sys-admin for all users
- You can create your *own* customizations in a **startup file**, in your home directory
- That file must have one of these names

  `.bash_profile`
  `.bash_login`
  `.profile`

Your Login Shell

- If there is more than one of these files in your home directory, then ***bash*** will each execute them in the order given above
- We will use `.bash_profile`

Interactive Non-login Shells

- The shell is a program, just like ***cat*** or ***ls***
- You can run *another* shell as a ***sub-shell*** of your current shell by typing the name of the shell at the command line

```
$ ps
  PID TTY          TIME CMD
12778 pts/1    00:00:00 bash
12969 pts/1    00:00:00 ps

$ bash

$ ps
  PID TTY          TIME CMD
12778 pts/1    00:00:00 bash
12970 pts/1    00:00:00 bash
12973 pts/1    00:00:00 ps
$
```

Interactive Non-login Shells

- Notice that there are **two** ***bash*** processes
- When you run ***script*** to generate a typescript file you are working inside an interactive non-login shell
- Your login ***bash*** shell is still running, but…
  - You are now running a second ***bash*** …
  - Which is running inside your login ***bash*** shell as a sub-shell
  - This sub-shell is the second type of shell…

Interactive Non-login Shells

- It is **not** a login shell because you ran this shell from the command line when you were *already* logged in
- Rather, it is an *interactive* non-login shell

- It is *interactive* because you can type commands to it through the keyboard, but it is ***not*** the shell you got when you logged in
- A non-login interactive shell is a shell that you create *without having to enter a password*

Interactive Non-login Shells

- The commands in the startup files named above

  `.bash_profile`
  `.bash_login`
  `.profile`

  are **NOT** run before starting this kind of shell
- Instead, the commands found in `.bashrc` are run for a non-login interactive shell
- You are not limited to running Bash...

Interactive Non-login Shells

- You can also run other shells, such as ***sh***, in a sub-shell

```
$ ps
  PID TTY          TIME CMD
19874 pts/27   00:00:00 bash
20500 pts/27   00:00:00 ps

$ sh

$ ps
  PID TTY          TIME CMD
19874 pts/27   00:00:00 bash
20510 pts/27   00:00:00 sh
20526 pts/27   00:00:00 ps
```

Interactive Non-login Shells

- You leave an interactive login shell by typing ***exit***

```
$ ps
  PID TTY          TIME CMD
19874 pts/27   00:00:00 bash
20737 pts/27   00:00:00 ps

$ sh

$ ps
  PID TTY          TIME CMD
19874 pts/27   00:00:00 bash
20743 pts/27   00:00:00 sh
...
```

```
...
20751 pts/27   00:00:00 ps

$ exit
exit

$ ps
  PID TTY          TIME CMD
19874 pts/27   00:00:00 bash
20771 pts/27   00:00:00 ps
```

Non-interactive Shells

- When you create a file of Linux commands, you have created a shell script
- *This* is what you have been doing in the Class Exercises
- A shell script contains Unix commands which only a shell can understand.
  - However, your current shell *goes to sleep* when you run a program.
  - For that reason, your shell has to create a *sub-shell* to run the commands

Non-interactive Shells

- Such a shell is called a *non-interactive* shell
- There is no standard startup file for such a shell
- You can create a startup file for non-interactive shells if you put name of the file in the shell variable **BASH_ENV**

Creating Startup Files

- A startup file contains Unix commands that are run *just before* you get a prompt
- Bash normally uses two startup files

  `.bash_profile`

  `.bashrc`

- `.bash_profile` commands are run before you get a prompt in a *login* shell

Creating Startup Files

- `.bashrc` commands are run before you get a prompt in an *interactive, non-login* shell
- `.bash_profile` is where you define variables
- We will not be talking much about `.bashrc`, which most Ubuntu installations only use `.bashrc` when running a GUI
- Every time you open a window in a Linux GUI, you are creating an interactive non-login shell, which can be customized in `.bashrc`

Running a Startup File after a Change has been Made

- Usually, when you change a startup file you want the changes to take place immediately
- But, if you made a change to **`.bash_profile`** , the changes won't take effect until the next time you login
- Unix has a way to make the changes take effect immediately – by running the ***`source`*** command.

```
source .bash_profile
```

Running a Startup File after a Change has been Made

- ***source*** is a built-in

```
$ which source
$
```

- Another you may see is the character . , which is also often used to make the changes immediately

```
$ . .bash_profile
$
```

- But, ***source*** is fine for now!

Commands that are Symbols

- Unix has some commands that are *symbols*, rather than words
- I'll just mention them now – and go into greater detail in future classes

`( )`	Runs whatever commands are enclosed in the parentheses ***in a sub-shell***
`$( )`	***Command substitution:*** Runs the commands enclosed in the parentheses in a subshell and ***returns their value*** to the command line, replacing the dollar sign, the parentheses and everything in them with this value.
`(( ))`	***Evaluates an arithmetic expression:*** By default, Unix treats everything as text, but this command evaluates whatever it encloses as a **numerical**, rather than a string, expression
`$(( ))`	***Arithmetic expansion:*** Evaluates an *arithmetic* expression and returns its value at that place on the command line
`[ ]`	***The test command:*** Used to evaluate a *boolean* expression in constructs like *if* clauses
`[[ ]]`	***The conditional expression:*** Similar to `[ ]` but adds *string comparisons*