

# Introduction to the shell – Part II

Graham Markall

<http://www.doc.ic.ac.uk/~grm08>  
[grm08@doc.ic.ac.uk](mailto:grm08@doc.ic.ac.uk)

Civil Engineering Tech Talks  
16<sup>th</sup> November, 1pm

# Last week

- Covered applications and Windows compatibility
- Basic usage of the shell, a couple of scripts
- Left off with a “for loop” example
- This week:
  - Wildcards
  - Streams, pipes and redirects
  - Environment variable
  - Some shell utilities

# Last week's for loop

```
#!/bin/bash
```

```
for i in `seq 1 10`; do
```

```
    echo Count: ${i}
```

```
done
```

```
$ ./script2.sh
```

```
Count: 1
```

```
Count: 2
```

```
Count: 3
```

```
Count: 4
```

```
Count: 5
```

```
Count: 6
```

```
Count: 7
```

```
Count: 8
```

```
Count: 9
```

```
Count: 10
```

# Wildcards - \* and ?

- Can be used in place of characters
- Avoid specifying an exact filename
  - Any number of characters: \*
  - A single character: ?
- 
- Examples...

# Wildcards - \* and ?

```
$ ls
ascript.sh  IPDPS_submission.pdf  script1.sh  script2.sh
Ultimate_Guide_ver100.pdf
```

```
$ echo *.pdf
IPDPS_submission.pdf  Ultimate_Guide_ver100.pdf
```

```
$ echo *.sh
ascript.sh  script1.sh  script2.sh
```

```
$ echo script?.sh
script1.sh  script2.sh
```

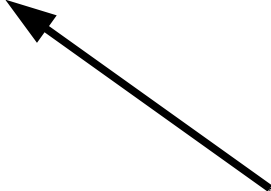
```
$ echo *
ascript.sh  IPDPS_submission.pdf  script1.sh  script2.sh
Ultimate_Guide_ver100.pdf
```

# \* - wildcard in for loop

- Perform some operation on every file in the directory:

```
#!/bin/bash
```

```
for file in *.pdf; do  
    pdftohtml ${file}  
done
```



pdftohtml is a command that converts a pdf file to html

# Streams, pipes, and redirects

- **What if you want to do something with the output from a script?**
- Programs have three streams:
  - 0: stdin, usually piped to your keyboard
  - 1: stdout, usually piped to your screen
  - 2: stderr, also usually piped to the screen
- Pipes connect streams together:
  - `ls | less`

# Streams, pipes and redirects (cont.)

- Stream from stdin to stdout:
  - `cat`
- Stream a file to stdout:
  - `cat <file>`
- Stream a file to stdout, then search for lines containing <string>:
  - `cat <file> | grep <string>`

# Streams, pipes and redirects (cont.)

- What about redirecting streams back to a file?
- Redirect stdout to a file:
  - `ls > my_directory_listing.txt`
- Redirecting the stderr to a file:
  - `<some program> 2> program_errors.txt`
- Redirecting all output to a file:
  - `<some program> &> all_output.txt`
- What if we want to pipe stdout and stderr to stdin of another process?
  - `<some program> 2>&1 | <another_program>`

# Environment variables

- An environment variable stores a value for the duration of your session in bash
- Some are special, and used by the system:
- `echo $PATH`
  - `/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin`
- Used by the system to find your programs.
- `which gcc:`
  - `/usr/bin/gcc`
- Setting an environment variable:
  - `export PATH=/home/graham/programs:$PATH`

# Part 3

A few shell utilities

# Read the fantastic manual

- Most commands have manual pages, that explain how to use them
- They can be viewed with
  - `man <command name>`
- Up and down scroll, q to quit

# SSH and SCP

- Secure access to a remote computer, as if you were sitting at the terminal of that computer:
  - `ssh <username>@<computername> {-X}`
- Securely copy a file to a remote computer:
  - `scp <sourcefile> \  
<username>@<computername>:<directory>/<targetfile>`

# Example ssh session

- Connect to shell1.doc.ic.ac.uk with username grm08:

```
$ ssh grm08@shell1.doc.ic.ac.uk
```

```
grm08@shell1.doc.ic.ac.uk's password:
```

```
0 grm08@shell1 ~
```

```
$ ...
```

```
<do some stuff on shell1>
```

```
$ exit
```

```
logout
```

```
Connection to shell1.doc.ic.ac.uk closed.
```

```
0 graham@grahamspc ~/scripts
```

```
$
```

# Installing packages from the command line

- Recall Synaptic for installing packages from earlier
- Package installation can also be done via the command line:
  - `apt-get install <package name>`
  - `apt-get remove <package name>`
  - `aptitude search <search string>`

# Acting as the superuser

- Sometime's it's not enough to run a command as a normal user:

```
$ apt-get install python
```

```
E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission denied)
```

```
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
```

```
$ sudo apt-get install python
```

- This will succeed, running as root using sudo

# Zipping and unzipping

- Many files are bundled up using the tar and gzip programs:
  - Tar bundles up all the files into one big file
  - Gzip compresses the one big file.
- Extracting a “tarball”:
  - `tar xvzf archive.tar.gz`
- Creating a tarball:
  - `tar cvzf archive.tar.gz <list of files>`

c.f. `gunzip <file> | tar xv,`  
`tar cv <file> | gzip`

# Command-line editors

- Nano: reasonably straightforward, the default on recent Ubuntu installations
- Vi & emacs: More complicated, but more powerful - extensible

# What's not been covered?

- Many things, but other useful shell tools include:
  - sed – Stream editor. Parses text and replaces selected strings
  - Awk – for processing text-based data
  - Grep – searching for strings
  - Xargs – build input to a new command from stdin
  - Split – split a file into pieces
  - Head – print out the first lines of a file
  - Tail – print out the last lines of a file
  - Ps – list running processes
  - Kill – kill a running process
  - Top – interactively view running processes
  - Touch – update the last-modified time of a file to the current time
  - Dd – copy data from one place to another
  - Df – show disk usage
  - Ln - create symbolic links
  - True – returns 0 (true)
  - False – returns 1 (false)
  - Sleep – pauses for a given time
  - Pwd – print the current working directory
  - Mkisofs – make a CD image file
  - Wodim – burn a CD image file to a CD
  - Mount – make a new filesystem available at a “mount point”
  - Shutdown – shutdown the system
  - Diff – show the difference between two files
  - Wget – download files from a web site

# Further resources

- Ubuntu bash scripting beginners: <https://help.ubuntu.com/community/Beginners/BashScripting>
- Bash programming howto: <http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>
- Advanced bash scripting guide: <http://tldp.org/LDP/abs/html/>
- Awk tutorial: <http://www.softpanorama.org/Tools/awk.shtml>
- Streams, pipes and redirects:  
<http://www.ibm.com/developerworks/linux/library/l-lpic1-v3-103-4/index.html>
- Openssh manuals: <http://www.openssh.com/manual.html>
- Man pages: <http://www.linuxmanpages.com/>

# Finishing points

- The shell is a very powerful tool, that can be used for practically any task on a Linux system
- I've only touched on many points of shell scripting
  - There are many resources for learning more
  - I'll add these in the online version of the slides
- Please get involved with the IC Linux Users' Society!
  - Post questions to [ic-linux-discuss](mailto:ic-linux-discuss)!