CS2043 - Unix Tools & Scripting

Cornell University, Spring 2014[1]

Instructor: Bruno Abrahao

January 29, 2014

[1] Slides evolved from previous versions by Hussam Abu-Libdeh and David Slater

Vim: Tip of the day!

Search and Replace

Search for search_term:

`/search_term`

Replace pattern with string:

`:%s/pattern/string/[options]`

Once you find the first occurrence of search_term, hit **n** to find the next occurence. The key **N** goes to the previous occurence.

Tools: Tip of the day!

Listing files

The human option:

```
ls -lh
```

One file per line

```
ls -1
```

Shell's Again

Many shells for UNIX-like systems:

- `sh`: The Bourne Shell -
  a popular shell made by Stephen Bourne

- `bash`: The Bourne Again Shell -
  default shell for the GNU OS, most Linux distros, and OSX

- `csh`: The C Shell -
  interactive and close to C
  default shell for BSD-based systems

- `zsh`: The Z Shell -
  possibly the most fully-featured shell inspired by sh, bash, ksh, and tcsh

Shell's Again

- Since `bash` is the gold standard of shells and has more than enough features for this course, we'll stick with it.
- For more info, use Wikipedia as a starting point: http://en.wikipedia.org/wiki/Comparison_of_command_shells

If the machine do not default to Bash

- If you are already logged in to the server, just type `bash`
- More importantly we would like the server to automatically put us into bash when we login. One way to do this is by editing the file `/.login` which gets executed each time you log in to the server and csh starts up.

Start bash automatically

Add the following line to the end of `/.login`

```
if ( -f /bin/bash) exec /bin/bash --login
```

If you had root privileges you could just edit /etc/passwd and find the line corresponding to the current user.

Using Bash Efficiently

In this lecture:

- Shell shortcut keys
- Reusing history
- Aliasing
- Special character expansion
- File compression

Shell Shortcuts

Make entering commands easier:

- Tab completion
- Up-down arrow: browse through command history
  - so you do not have to retype everything
- Ctrl + e: jump cursor to end of line
- Ctrl + a: jump cursor to beginning of line
- Ctrl + u : delete everything from cursor to beginning of line
- Ctrl + k : delete everything from cursor to end of line
- Ctrl + l : clear the screen

More shortcuts at:
http://linuxhelp.blogspot.com/2005/08/bash-shell-shortcuts.html

Reusing History: Bang (!)

Use the *bang* operator (`!`) to repeat a command from your history that begins with the characters following it.

Example

```
hussam@orjwan:~$ pdflatex lecture3.tex
hussam@orjwan:~$ !p
hussam@orjwan:~$ !pdf
```

`!p` and `!pdf` will recall `pdflatex lecture3.tex`

Using `!` can save you many keystrokes when repeating tasks.

Reusing History: Search

You can search through your command history using the shortcut Ctrl + R:

- Press Ctrl + R and type a search string. Matching history entries will be shown.
- Press Ctrl + R again to see other matches.
- If you like an entry, press ENTER to re-execute it.
- Press ESC to copy the entry to the prompt without executing.
- Press Ctrl + G to exit search and go back to an empty prompt.

Example

```
(reverse-i-search)‘pd’:  pdsh -w node[0-9] -R ssh hostname -i
```

You will not be able to type if there are no matches.

Aliases

The more you use Bash the more you see what options you use all the time. For instance `ls -l` to see permissions, or `rm -i` to insure you don't accidentally delete a file. Wouldn't it be nice to be able to make shortcuts for these things?

Alias:

`alias name=command`

- The alias allows you to rename or type something simple instead of typing a long command
- You can set an alias for your current session at the command prompt
- To set an alias more permanently add it to your `.bashrc` or `.bash_profile` file in your home directory.

Alias Examples

Examples

```
alias ls=‘ls --color=auto’
alias dc=cd
alias ll="ls -l"
```

- Quotes are necessary if the string being aliased is more than one word
- To see what aliases are active simply type `alias`
- Note: If you are poking around in .bashrc you should know that any line that starts with # is commented out.

Shell Expansion

In a bash shell, if we type:

```
$ echo This is a test
This is a test
```

But if we type

```
$ echo *
Lec1.pdf Lec1.dvi Lec1.tex Lec1.aux
```

What happened?
The shell expanded * to all files in the current directory. This is an example of path expansion, one type of shell expansion.

Interpreting Special Characters

The following are special characters:

```
$ * < > & ?  { } [ ]
```

- The shell interprets them in a special way unless we escape (\$) or place them in quotes "$".
- When we first invoke a command, the shell first translates it from a string of characters to a UNIX command that it understands.
- A shell's ability to interpret and expand commands is one of the powers of shell scripting.

We will cover all those special characters later in the course.

Shell Expansion

`*` `^` `?` `{` `}` `[` `]` Are all "wildcard" characters that the shell uses to match:

- Any string
- A single character
- A phrase
- A restricted set of characters

The shell's ability to interpret and expand commands is one of the powers of shell scripting.

Shell Expansion

- `*` matches any string, including the null string (i.e. 0 or more characters).

Examples:

Input	Matched	Not Matched
`Lec*`	`Lecture1.pdf Lec.avi`	`ALecBaldwin/`
`L*ure*`	`Lecture2.pdf Lectures/`	`sure.txt`
`*.tex`	`Lecture1.tex Presentation.tex`	`tex/`

Shell Expansion

- ? matches a single character

Examples:

Input	Matched	Not Matched
`Lecture?.pdf`	`Lecture1.pdf Lecture2.pdf`	`Lecture11.pdf`
`ca?`	`cat can cap`	`ca cake`

Shell Expansion

- `[...]` matches any character inside the square brackets
  - Use a dash to indicate a range of characters
  - Can put commas between characters/ranges

Examples:

Input	Matched	Not Matched
`[SL]ec*`	`Lecture Section`	`Vector.tex`
`Day[1-4].pdf`	`Day1.pdf Day2.pdf`	`Day5.pdf`
`[A-Z,a-z][0-9].mp3`	`A9.mp3 z4.mp3`	`Bz2.mp3 9a.mp3`

Shell Expansion

- `[^...]` matches any character **not** inside the square brackets

Examples:

Input	Matched	Not Matched
`[^A-P]ec*`	`Section.pdf`	`Lecture.pdf`
`[^A-Za-z]*`	`9Days.avi .bash_profile`	`vacation.jpg`

Shell Expansion

- **Brace Expansion:** `{...,...}` matches any phrase inside the comma-separated brackets

Examples:

Input	Matched
`{Hello,Goodbye}\ World`	`Hello World Goodbye World`

NOTE

Brace expansion must have a list of patterns to choose from. (i.e. at least two options)

Shell Expansion

And of course, we can use them together:

Input	Matched	Not Matched
`*i[a-z]e*`	`gift_ideas profile.doc`	`DriVer.exe`
`[bf][ao][ro].mp?`	`bar.mp3 foo.mpg`	`foo.mpeg`

Compression & Archiving

- `zip` / `unzip`
  - Compress and archive (bundle) files into a single file.
  - A new compressed .zip file is created and the original files stay intact.
  - `zip <zipped_file_name> <files_to_compress>`
  - `unzip <zipped_file_name>`
  - Many options! E.g., add files to an existing zip, encrypt with a password ..etc

Compression & Archiving

- `gzip`
  - Compress files using Lempel-Ziv coding.
  - Does not bundle files, the compressed files will replace the original files.
  - `gzip <file_to_compress>`
  - `gunzip <compressed_file>`
- `bzip2`
  - Compress files using Burrows-Wheeler block sorting text compression algorithm and Huffman coding.
  - More efficient than `gzip` on most files, but a bit slower.
  - Like `gzip`, this is only a compression tool, and thus compressed files will replace the original files.
  - `bzip2 <file_to_compress>`
  - `bunzip2 <compressed_file>`

Tarballs!

- To archive multiple files together, we can use the "Tape Archive" utility (`tar`).
- `tar` bundles multiple files together into a single file (but does not compress them or replace them)
  - `tar -cf archive.tar foo bar`
    Create archive.tar from files foo and bar
  - `tar -xf archive.tar`
    Extract all files from archive.tar

Compressed Tarballs

- To compress a tarball we can pipe the outcome of `tar` to a tool like `gzip` or `bzip2`.
- However, `tar` has flags to automatically do this:
  - `-z` : compress using `gzip`
  - `-j` : compress using `bzip2`
  - `tar -czf archive.tar.gz foo bar`
    Creates a compressed file (archive.tar.gz) from files foo and bar
- **Naming convention:**
  - `archive.tar.gz` or `archive.tgz`: gzipped tarballs
  - `archive.tar.bz2` or `archive.tbz`: bzip2 tarballs
- Works with directories too!
  - `tar -czf cs2042.tgz cs2042/*`
    Creates a compressed file containing the directory and contents of cs2042 directory

A Backup Script

Here is something a little more practical - a simple script to back up all the files in your documents directory:

Example: backup.sh

```
#!  /bin/bash
tar -czf ~/backups/cs2042.backup.tar.gz \
~/Documents/cs2042/
```

This script makes use of the `tar` archiving command:

Making Tarballs:

```
tar -c(z/j)f <dest_archive> <source>
tar -x(z/j)f <archive>
```

- `-c` version creates a new archive from a source file/dir
- `-x` extracts an existing archive to the current dir
- pick either `-z` or `-j` options (-z $\Rightarrow$ .tar.gz , -j $\Rightarrow$ .tar.bz2)