Shell Commands and Scripting

Arvind K. Saibaba

North Carolina State University

What is the shell?

- "The shell is a program that takes keyboard commands and passes them to the operating system to carry out." [1]
- Most Linux/Unix distributions supply a shell program from the GNU Project called `bash`.
- MacOSX is based on Unix and also contains a bash shell.
- The name "bash' is an acronym for "Bourne Again SHell." The original Unix shell program written by Steve Bourne.

[1] from the book "The Linux Command Line" by William Shotts"

Why use commandline/shell scripting?

- Work with files and folders.
- Automate painful tasks
  - Examples: batch processing, such as change all png files in a folder to eps, add a specific file recursively to each subfolder.
- Use in computer clusters to submit jobs.
- Developing/installing software packages.

Basic shell commands

You are here

`pwd` - present working directory.

```
$ pwd
/Users/arvindks/Dropbox/Courses/
```

Help!

- '`man`' is the equivalent of `help` in MATLAB.
  Use '`h`' for help and `q` to quit.
- '`whatis`' gives a one-line description of the command.

Change Directory

Use the command `cd` to change the directory. Two ways to use it:

1. Absolute path

```
$ cd /Users/asaibab/Documents
```

2. Relative path (`~` is an alias for home directory)

```
$ cd ~/Documents/
```

Change Directory

Use the command `cd` to change the directory. Two ways to use it:

1. Absolute path

```
$ cd /Users/asaibab/Documents
```

2. Relative path (`~` is an alias for home directory)

```
$ cd ~/Documents/
```

Some shortcuts

`cd`	Goes to the home directory
`cd ../`	Goes to the directory one level higher
`cd -`	Goes to the previous directory (from where you came)
`cd .`	Stay in present directory (alt. `cd ./`

List

`ls` - lists all files and folders

Usage examples :

```
$ ls
$ ls ../
$ ls ~/Documents/
```

One can use the same shortcuts as that for `cd`

List

`ls` - lists all files and folders

Usage examples :

```
$ ls
$ ls ../
$ ls ~/Documents/
```

One can use the same shortcuts as that for `cd`

Some other options

`ls -l`	long format
`ls -lh`	long format with human readable file sizes
`ls -S`	sort by filesize

See `man ls` for more options.

Wildcards

The symbol `*` is a wildcard matches and can be used in several ways:

`ls *.tex`	List all possible TeX files
`ls a*.tex`	List all TeX files starting with ‘a’
`ls *verb*.tex`	List all files containing the word ‘verb’ anywhere in the filename.
`ls filename.*`	list all filenames with any possible extension
`ls *.*`	List all files with any extensions

Wildcards

The symbol `*` is a wildcard matches and can be used in several ways:

`ls *.tex`	List all possible TeX files
`ls a*.tex`	List all TeX files starting with ‘a’
`ls *verb*.tex`	List all files containing the word ‘verb’ anywhere in the filename.
`ls filename.*`	list all filenames with any possible extension
`ls *.*`	List all files with any extensions

By contrast, the wildcard `?` only matches a single character and not a string of characters. Example

```
$ ls file????.tex
```

Any TeX file beginning with ‘file’ with exactly 4 additional characters.

Structure of shell commands

Possible forms of shell command

```
command
command arguments
command -options
command -options arguments
```

Combine multiple command in one line using ‘;’

Structure of shell commands

Possible forms of shell command

```
command
command arguments
command -options
command -options arguments
```

Combine multiple command in one line using ‘;’ Examples

- List all contents (in long human-readable format) of directory dir1

  ```
  ls -lh dir1
  ls -l -h dir1
  ```

- Go to directory dir1 and list all its contents

  ```
  cd dir1; ls
  ```

Investigating files

`file <filename>`	lists the filetype
`less <filename>`	list the file contents
`head <filename>`	list the top part of the file
`tail <filename>`	list the bottom part of the file
`wc <filename>`	lists the number of lines, words, and bytes in files

MacOSX has another useful command `open`, which opens any file using the default program set for that file.

Remove, Copy, and move files

Overview

- `rm` - remove files and directories
- `cp` - copy files and directories
- `mv` - move/rename files and directories

Remove, Copy, and move files

Overview

- `rm` - remove files and directories
- `cp` - copy files and directories
- `mv` - move/rename files and directories

Examples: Removing files

`rm file1 *.tex`	Remove file1 and all TeX files
`rm file2 dir1`	Remove file2 and directory 1
`rm -rf dir2`	Recursively delete contents of the directories

A warning: `rm` can cause a lot of (unintentional) damage when used with wildcards.

Remove, Copy, and move files

- Copy file1 to file 2, and delete file2 if it exists

  `cp <filename1> <filename2>`

  Use `-i` option if you want to check with user if they want to overwrite

Remove, Copy, and move files

- Copy file1 to file 2, and delete file2 if it exists

  `cp <filename1> <filename2>`

  Use `-i` option if you want to check with user if they want to overwrite

- Copy multiple files instead of a single file

  `cp file1.tex file2.png dir1/`

  Directory dir1 has to exist

Remove, Copy, and move files

- Copy file1 to file 2, and delete file2 if it exists

  `cp <filename1> <filename2>`

  Use `-i` option if you want to check with user if they want to overwrite
- Copy multiple files instead of a single file

  `cp file1.tex file2.png dir1/`

  Directory dir1 has to exist
- Wildcards are permissible

  `cp *.tex dir2/`

Remove, Copy, and move files

- Copy file1 to file 2, and delete file2 if it exists

  `cp <filename1> <filename2>`

  Use `-i` option if you want to check with user if they want to overwrite
- Copy multiple files instead of a single file

  `cp file1.tex file2.png dir1/`

  Directory dir1 has to exist
- Wildcards are permissible

  `cp *.tex dir2/`
- Use `-r` for recursively copying all the folders, subfolders, and files

  `cp -r dir1 dir2`

  Directory dir2 will be created, if it doesn't exist.

Remove, Copy, and move files

- The command `mv` works similar to `cp`, except it removes the files rather than retaining a copy.

  ```
  mv file1.tex file2.png dir1/
  ```

  This deletes the files file1.tex and file2.png in the current directory. Think of '`mv`' as a combination of '`cp`' and '`rm`'

Remove, Copy, and move files

- The command `mv` works similar to `cp`, except it removes the files rather than retaining a copy.

  ```
  mv file1.tex file2.png dir1/
  ```

  This deletes the files file1.tex and file2.png in the current directory. Think of '`mv`' as a combination of '`cp`' and '`rm`'

- You would also use `mv` to rename a file/folder.

Remove, Copy, and move files

- The command `mv` works similar to `cp`, except it removes the files rather than retaining a copy.

  ```
  mv file1.tex file2.png dir1/
  ```

  This deletes the files file1.tex and file2.png in the current directory. Think of '`mv`' as a combination of '`cp`' and '`rm`'

- You would also use `mv` to rename a file/folder.

- Use '`-r`' for folders (recursive) and '`-i`' (interactive) to prompt the user for confirmation.

Search and redirect

- Search using `grep`

  `grep <pattern> *.tex`

Search and redirect

- Search using `grep`

  ```
  grep <pattern> *.tex
  ```

- Redirect the result of a command to a file
  - Creates a file called 'listoftexfiles.txt' with the output of the command.

    ```
    ls -l *.tex > listoftexfiles.txt
    ```

  - Does not rewrite the file 'listoftexfiles.txt' but appends at the end of file.

    ```
    ls -l *.tex >> listoftexfiles.txt
    ```

Search and redirect

- Search using `grep`

  `grep <pattern> *.tex`
- Redirect the result of a command to a file
  - Creates a file called 'listoftexfiles.txt' with the output of the command.

    `ls -l *.tex > listoftexfiles.txt`
  - Does not rewrite the file 'listoftexfiles.txt' but appends at the end of file.

    `ls -l *.tex >> listoftexfiles.txt`
- Redirect the result of a command to a different command

  `ls *.tex | sort`

  Lists all the files and sorts the list. (Alt: `ls -S *.tex`)

Check your understanding

Write down the shell code to

1. Delete all TeX files in the current folder and subfolders
2. Copy all PNG files containing 'project' in its title from present folder to `dir1`
3. Recursively copy all files and folders from present folder to `dir1`
4. In the present directory are files photo001.png to photo999.png. List all the photos corresponding to numbers 600-699.

Check your understanding

Write down the shell code to

1. Delete all TeX files in the current folder and subfolders

   ```
   rm -r *.tex
   ```

2. Copy all PNG files containing 'project' in its title from present folder to dir1

   ```
   cp *project*.png dir1
   ```

3. Recursively copy all files and folders from present folder to dir1

   ```
   cp -r . dir1
   ```

4. In the present directory are files photo001.png to photo999.png. List photos 600-699
   Either `ls photo6*.png` or `photo6??.png`

Processes

- `top` - displays the list of processes that are running. Type `q` to quit.

```
Processes: 432 total, 3 running, 429 sleeping, 2865 threads
Load Avg: 2.12, 2.24, 2.09  CPU usage: 3.91% user, 4.62% sys, 91.45% idle  SharedLibs: 242M resident, 38M
MemRegions: 172495 total, 5027M resident, 86M private, 2314M shared. PhysMem: 15G used (2894M wired), 845
VM: 1214G vsize, 628M framework vsize, 9193639(0) swapins, 10376670(0) swapouts.
Networks: packets: 13169731/16G in, 4580663/1239M out. Disks: 8261825/130G read, 3346547/147G written.

PID    COMMAND      %CPU TIME     #TH    #WQ  #PORTS MEM    PURG   CMPRS  PGRP  PPID  STATE    BOOSTS
95543  com.apple.ap 0.0  00:00.03 2      2    23     12K    0B     928K   95543 1     sleeping  0[1]
95542  com.apple.cm 0.0  00:00.01 2      1    22     8192B  0B     880K   95542 1     sleeping  0[1]
95541  storeuid     0.0  00:01.81 3      1    131    1704K  0B     3672K  95541 1     sleeping  0[457]
95536  MTLCompilerS 0.0  00:00.11 2      2    31     40K    0B     4948K  95536 1     sleeping  0[1]
94568  netbiosd     0.0  00:00.97 2      2    40     464K   0B     2808K  94568 1     sleeping *0[1]
```

Useful to see the status of a program that has been running for a while.

Processes

- `top` - displays the list of processes that are running. Type `q` to quit.

```
Processes: 432 total, 3 running, 429 sleeping, 2865 threads
Load Avg: 2.12, 2.24, 2.09  CPU usage: 3.91% user, 4.62% sys, 91.45% idle  SharedLibs: 242M resident, 38M
MemRegions: 172495 total, 5027M resident, 86M private, 2314M shared. PhysMem: 15G used (2894M wired), 845
VM: 1214G vsize, 628M framework vsize, 9193639(0) swapins, 10376670(0) swapouts.
Networks: packets: 13169731/16G in, 4580663/1239M out. Disks: 8261825/130G read, 3346547/147G written.

PID    COMMAND      %CPU TIME     #TH    #WQ  #PORTS MEM    PURG   CMPRS  PGRP  PPID  STATE    BOOSTS
95543  com.apple.ap 0.0  00:00.03 2      2    23     12K    0B     928K   95543 1     sleeping  0[1]
95542  com.apple.cm 0.0  00:00.01 2      1    22     8192B  0B     880K   95542 1     sleeping  0[1]
95541  storeuid     0.0  00:01.81 3      1    131    1704K  0B     3672K  95541 1     sleeping  0[457]
95536  MTLCompilerS 0.0  00:00.11 2      2    31     40K    0B     4948K  95536 1     sleeping  0[1]
94568  netbiosd     0.0  00:00.97 2      2    40     464K   0B     2808K  94568 1     sleeping *0[1]
```

Useful to see the status of a program that has been running for a while.

- To kill a process: `$ kill -9 pid`
  where pid is the process id (see figure).

Shutting down and restarting

These process require special authorization and you may be prompted for your authentication.

`sudo reboot`	Restart the system
`sudo shutdown`	Shutdown

- `su` means super user.

- On NC state math laptops, you may not have `su` access and may have to request it.

Alias

Suppose you have a sequence of commands you are tired of repeating

```
$ cd /usr; ls; cd -
X11 bin lib libexec local sbin share standalone
/Users/asaibab/
```

Aliases are user-defined commands built from other commands, without having to write an entire script.

```
$ alias mycommand="cd /usr; ls; cd -"
$ mycommand
X11 bin lib libexec local sbin share standalone
/Users/asaibab/
```

Note that once you close the shell terminal, or open a different one, this command will no longer be active.

Alias

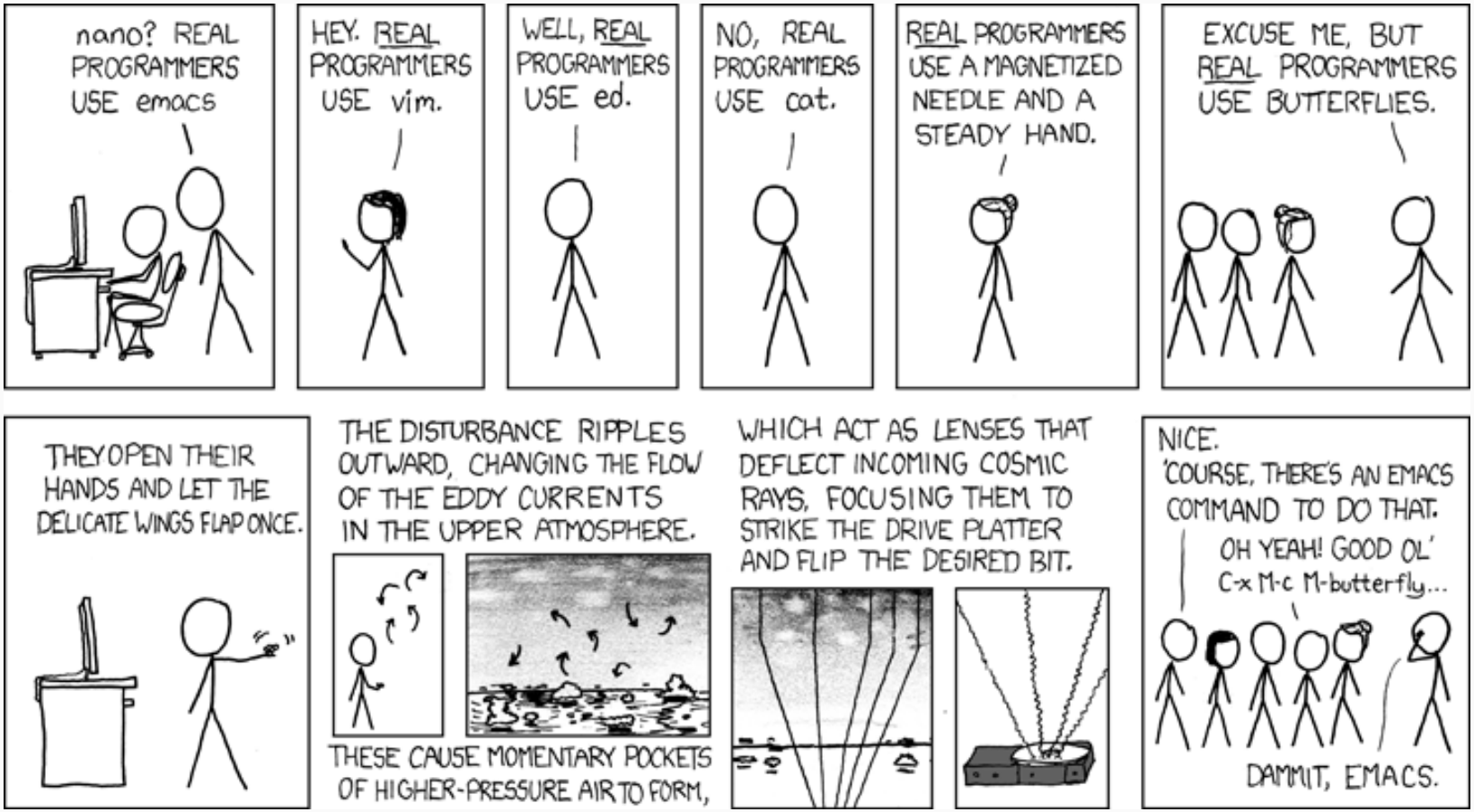
To list the active aliases

```
$ alias
alias mvim='/Applications/MacVim.app/Contents/bin/mvim'
alias mycommand='cd /usr; ls; cd -'
alias skim='/Applications/Skim.app/Contents/MacOS/Skim'
```

To enter an alias permanently, add the `alias` commands to the `~/.bash_profile` file. Note: files starting with `.` are not visible on a browser.

Writing shell scripts

Picking an editor



I personally use vi (actually gvim, which has a graphical interface).
But I recommend nano (especially on computer clusters).

Writing and executing your first shell script

Save the file as 'first.sh' (the extension is unimportant but tells the user that it is a shell script)

```
#!/bin/bash
#This is a comment
echo "Hello world!" #This is also a comment.
```

- The first line tells the OS which program to use to execute; Alternatively, use `#/bin/sh`.
- Other executables are found in `/bin/`, `/usr/bin/` and `/usr/local/bin`.
- To execute this script type '`./first.sh`'. Don't forget the '`./`'!
- Sometimes you may not have permission to execute the script. Type `chmod a+x first.sh` to give permission to execute the script.

Symbolic links

Suppose your code `first.sh` is really useful and you want it to be usable everywhere.

```
$ sudo cp first.sh /usr/local/bin/<name your command>
```

Move it to the folder `/usr/local/bin/`. You will be prompted for authentication. You can now use the new command name since it is now in your search path.

Symbolic links

Suppose your code `first.sh` is really useful and you want it to be usable everywhere.

```
$ sudo cp first.sh /usr/local/bin/<name your command>
```

Move it to the folder `/usr/local/bin/`. You will be prompted for authentication. You can now use the new command name since it is now in your search path.

Alternative option: Symbolic link

```
ln -s <target path/filename> <link path/desiredname>
```

Creates a second file in `<link path>` that links to the file in the `<first path>`.

Variables

Shell variables

```
$ var=5
$ str="This is a string"
$ dir="/usr/local/bin"
```

We can use these variables in the following ways:

- `echo $var` (not `echo var`)
- `ls $dir`
- `cp $file1 $file2`

for loops

Two examples:

- ```
  $ for i in A B C D; do echo $i; done
  A
  B
  C
  D
  ```

- ```
  for i in shellscripting.*; do echo "$i"; done
  shellscripting.aux
  shellscripting.log
  shellscripting.nav
  shellscripting.out
  shellscripting.pdf
  ...
  ```

if and while

- If

```
x=5
if [ "$x" -eq 5 ]; then
    echo "x equals 5."
else
    echo "x does not equal 5."
fi
```

if and while

- If

```
x=5
if [ "$x" -eq 5 ]; then
    echo "x equals 5."
else
    echo "x does not equal 5."
fi
```

- While

```
count=1
while [[ "$count" -le 5 ]]; do
    echo "$count"
    count=$((count + 1))
done
echo "Finished."
```

Makefiles

From the `man` page: "The make utility will determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them."

Here is a simple makefile that I use while writing papers with LaTeX. Save this file with the name `makefile`.

```
all:
        pdflatex rdeim
        bibtex rdeim
        pdflatex rdeim
        pdflatex rdeim

paper:
        pdflatex rdeim

clean:
        rm *.aux *.bbl *.blg *.log *.thm
~
```

Makefiles - continued

The makefile defines three different commands all, paper, and clean

- `make all`: executes the four commands. This is needed to correctly obtain the bibliography references.
- `make paper`: only executes pdflatex but not bibtex.
- `make clean`: deletes all the intermediary files generated during LaTeX compilation.

If the files haven't changed since the last execution, then `make` will not do anything. Alternatively, if the files to be deleted are missing, then also `make` will complain.

Topics not discussed

- Permissions
  - `chmod`, `chown`, `su`, and `sudo`
- Connecting to a remote computer/cluster
  - `scp/sftp`, `ssh`, `wget`.
- Searching for files
  - `find`,`locate`.
- Arxiving
  - `tar`, `zip`
- Regular expressions

  ```
  perl -pe 's/(^|[^\\])%.*/\1%/' < old.tex > new.tex
  ```

  Removes all comments from the LaTeX files.

Regular expressions