MAKE
SCHOOL

UNIX UTILITIES

More powerful than the spiky blue shell

MAKE SCHOOL

WHAT IS UNIX?

Family of operating systems – not one OS

Common foundation for Unix-*compatible* or Unix-*like* (less compatible) modern OSes

GNU/Linux family: Ubuntu, Debian, etc.

Mac OS X, Solaris, MinGW, etc.

UNIX PHILOSOPHY

“Do *one* thing, and do it *well*.”

Avoid monolithic programs that do everything

Provide many small utilities (often called tools) that perform specific functions

Utilities can be easily chained together

SHELLS

A *shell* provides a command-line interface to files, utilities, + basic programming constructs

sh: “Bourne **sh**ell” – common functionality

bash: “**B**ourne **a**gain **sh**ell” – very popular

csh, **zsh** – similar features, different syntax

More info: Unix shells, bash scripting tutorial, shell feature comparison table

MAKE SCHOOL

BASIC COMMANDS

pwd – print **p**ath of current **w**orking **d**irectory

ls – **l**i**s**t all files in current working directory

cd – **c**hange **d**irectory (navigate into)

```
cd code/project/source
```

```
cd code; cd project; cd source
```

MAKE SCHOOL

SHELL SHORTCUTS

Shells provide helpful shortcuts:

`~` – user home directory (e.g., `/Users/Alan`)

`cd ~/code` (short for `/Users/Alan/code`)

`*` – wildcard expansion, matches any text

`ls ~/code/*.py` (lists Python source files)

MANUAL (MAN)

`man` – show manual of shell command/tool

```
man ls
man chmod
man find
man man
```
– Yes, you can do that!

WORD COUNT (WC)

Counts number of lines, words, characters

```
wc ~/Make/Code/CallRouting/*.py
 182    927   7319 CalculateRates.py
  42    156   1212 GenerateRoutes.py
 224   1083   8531 total
```

FILE PERMISSIONS

Read, write, and execute file permissions apply to 3 user sets: *user* (owner), *group*, and *others*

Metadata bits used to represent permission in compact way (`rwx` all set = `111`, all unset = `000`)

View permissions with long listing: `ls -l`

```
-rwxr-x---@  Alan  staff  script.sh
```

MODIFY FILE PERMISSIONS

File permission can be modified with **chmod**

Simple syntax to **+**add/**−**remove **r**ead/**w**rite/ e**x**ecute permissions for **u**ser/**g**roup/**o**thers

To revoke permissions from all other users:

```
chmod go-rwx file ~/directory/
```

TEXT STREAMS

Command-line tools + processes use *text streams* – like files, but not necessarily saved on disk

All processes have 3 standard streams:

- `stdin` – input (reads keyboard input by default)
- `stdout` – output (prints to terminal by default)
- `stderr` – *error* output (prints to terminal or file)

REDIRECTING STREAMS

Input and output streams can be redirected to/from files, or even into other commands

`cmd < file` – read input from `file`

`cmd > file` – write output to `file`

`cmd1 | cmd2` – *pipe* output of `cmd1` into input of `cmd2` (allows command chaining!)

SLICE STREAMS

`head` – the beginning of a stream

```
head -n5 ~/code/script.sh
```

`tail` – the end of a stream

```
tail -n20 packages/install.log
```

SIFT STREAMS

`sort` – sort all lines in a stream

```
sort phone-numbers.txt
```

`uniq` – remove duplicate lines in a stream

```
sort phone-numbers.txt | uniq
```

CUT LINES

`cut` – cut characters from lines in a stream

Show all unique area codes:

```
cut -c1-5 phone-numbers.txt |
sort | uniq
```

TRANSLATE (TR)

Replaces characters or bytes in a stream

```
echo "Make School" | tr a-z A-Z
```

STREAM EDITOR (SED)

Modifies stream using regular expressions

```
sed –e 's/abc/xyz/' ~/foo.txt
```

FIND FILES

`find` – locate files based on attributes

```
find ~/code -name script.sh

find ~/code -name '*.py'

find ~/movies -size 100M -ls
```

FILTER LINES (GREP)

grep: “**g**lobally search a **r**egular **e**xpression and **p**rint” – filters by matching stream lines

```
grep 'function' ~/code/script.sh

grep '[0-9]{5}' ~/addresses.txt

find ~/code -name '*.py' | grep
```

PASS ARGUMENTS (XARGS)

`xargs` – pass stream lines as arguments for next command

```
find | xargs | wc
```

```
find | xargs | grep
```

SECURE SHELL (SSH)

Protocol for secure network services

Authentication with *public + private key pair*

Command-line login to remote servers

Git, Mercurial, and SVN communicate via SSH

Can be used for network *tunneling*

SECURE SHELL (SSH)

Unix provides SSH client named `ssh`

```
ssh username@host
```

Host can be domain name or IP address

Commands are executed on remote server

Keys and config are saved in `~/.ssh/`

SECURE COPY (SCP)

Protocol + tool for secure file transfer (uses SSH)

Can copy files from local machine to remote server and vice-versa

```
scp localdirectory/localfile
user@host:directory/remotefile
```

Simple copy mechanism (reads + writes bytes)

REMOTE SYNC (RSYNC)

Utility, also uses SSH for authentication + encryption

Can synchronize/backup files + directories between local machine and remote server (simple Dropbox)

```
rsync user@host:directory/remotefile
localdirectory/localfile
```

Sophisticated copy mechanism (uses delta encoding to only copy file changes, not always entire contents)

CRON

Utility to schedule command-line tasks

Very useful for scheduled backups, server maintenance, home automation, etc

Scheduled tasks are called *cron jobs* and are held in a file called **`crontab`**, typically located in **`/etc`** + user specific one in home directory

CRON JOBS

Cron jobs can be any command or script

Crontab entries consist of timing frequency, command, and user to run command under

Every day at 10:30pm, run `backup.sh` as `root`:

```
30 22 * * * root ~/scripts/backup.sh
```

Mac OS X provides `launchd` as a scheduler

More info: Cron, crontab syntax tutorial, all about cron, Mac scheduled jobs

RESOURCES

Linux Shell Scripting Tutorial by Vivek Gite

ARTICLES

Useful Unix commands for data science by Greg Reda

Command-line tools can be 235x faster than your Hadoop cluster by Adam Drake

MAKE
SCHOOL