Intro to shell for (big) data processing

Rohan Kekatpure

Intuit, Inc

rohan.kekatpure@gmail.com

May 11, 2015

Why care?

- Excellent toolset for processing text and columnar files
- Collection of C programs $\Rightarrow$ Fast !
- Natively present on Unix/Linux (Mac is Unix)
- Your dev environment already has it – no install required

Why care?

- Can manipulate large files (100s of GB)
  - Excel is ruined after >10 MByte files
- Glue language between applications
- Shell philosophy is copied (Hadoop)
- Create complex automation pipelines with less code
  - Data loads
  - File transformations
  - Notifications: emails, text messages, Graphite . . .
  - Monitoring

Today

- Introduction to shell
- Review common (and possibly familiar) commands
- Test formatting, filtering, and replacement
- Working with delimited files
- Case studies

Focus

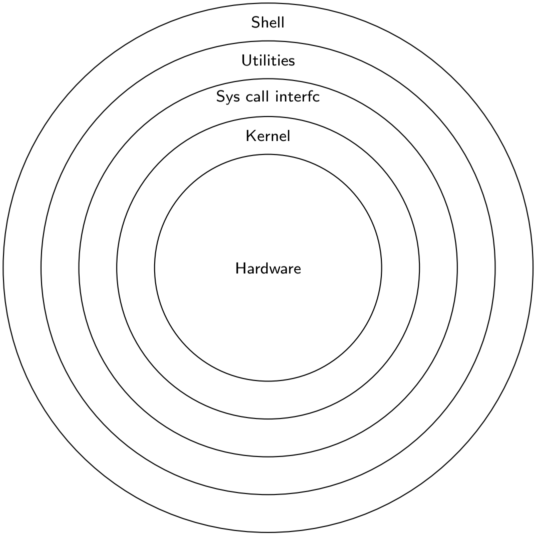
Demonstrating shell capabilities vs detailed syntax description

4 things to make your shell experience easier

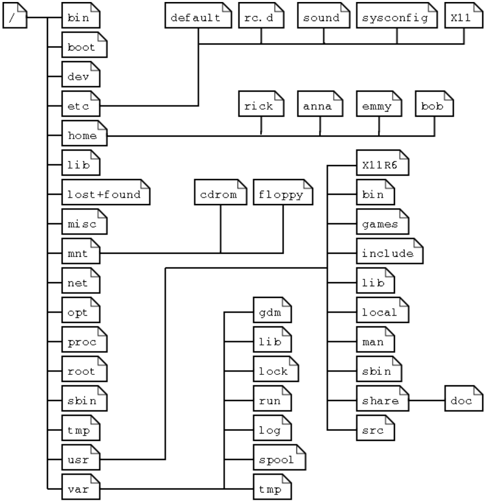
- Each tool does one thing and does it well
- No news is good news
- Everything is a file
  - Directories are files
  - Hardware devices are files
  - Sockets are files
  - Prove it!
- Most operations are file operations

```
$ cd /dev
$ ls
```

What is shell



Linux file system



File system on GW node

```
$ cd /
$ ls
```

What do you see?

Basic commands (you prob. know)

```
$ pwd # your location
$ ls # list stuff in current dir
$ ls /path/to/file/or/dir
$ ls -l # list in long format
$ ls -lR # recurse into subdirs
$ ls -lt # sort by latest modified
$ ls -la # show hidden files
$ ls *.<ext> # show files with extension
$ cd /path/to/dir
$ cd - # go 'back'
$ cd # go to your home folder
$ cd ~ # go to your home folder
```

Basic commands (you prob. know)

```
$ mkdir <dirname> # create a dir
$ mkdir -p dir0/dir1/dir2 # make a path
$ touch myfile.txt # create a file
$ cp file1 file2 # copy a file
$ cp -i file1 file2 # copy, with confirm
$ cp -n file1 file2 # copy, don't overwrite
$ cp -f file1 file2 # force copy
$ cp -r dir1 dir2 # Recursively copy

$ mv -[infr] file1 file2  # Move a file
$ rm -[infr] file # delete file
$ rm -rf file # be careful
```

- Hard to recover a file after `rm`
- Consider `shred` for secure delete

Online help

Online help is available via the `man` command

```
# help for various basic commands
$ man ls
$ man pwd
$ man cd
$ man cp
$ man mv
$ man rm
```

Quiz

Using the manual page for `ls`, list files in `/lib` directory in decreasing filesize order. The filesizes should be human readable.

Operators

```
# Write command output to file
$ ls -l > file.txt

# Append command output to file
$ ls -l >> file.txt

# Read input from a file
$ cat < file.txt

# Piping
$ cat file.txt | wc -l

# Chaining
$ cmd1 | cmd2 | cmd3 | cmd4 | ...
```

Displaying file contents

```
# Stream whole files
$ cat file.txt
$ cat file1.txt file2.sas ...
$ cat -n file.txt

# Show first and last 10 lines
$ head file.txt
$ tail file.txt

# Show first and last 50 lines
$ head -50 file.txt
$ tail -50 file.txt
```

Quiz

Display first 10 lines of file with line numbers

Printing text

```
# echo a string
$ echo "string in quotes"

# echo and interpret escape characters
$ echo -e "string1 \t string2"

# Print an integer range
# Handy for generating loop ranges
$ echo {10..20}
$ echo {20..10}

# Print formatted strings
$ printf "Today is %s, the %sth
day of the week\n" "Saturday" "5"

$ printf "Countdown %s\n" "$(echo {1..10})"
```

Printing text (continued)

```
# Without quotes printf gets a
# list and will loop through it
# E.g., this generates 10 statements
$ printf "Countdown %s...\n" $(echo {10..0})

# Convert from decimal to scientific format
$ printf "Value of G is %0.15e\n"
 .000000000066730
# prints 6.673000000000000e-11

# Convert from scientific to decimal format
$ printf "Value of G is %0.15f\n" 6.673e-11
# prints 0.000000000066730

# Convert from decimal to hex or back
$ printf "\%d (decimal) = \%x (hex)\n" 15 15
$ printf "\%d (decimal) = \%X (hex)\n" 15 15
```

Text filtering

```
$ cp /home/rkekatpure-admin/shelltut/schdist.txt .

# print header row and get delimiter
$ head -1 schdist.txt | cat -t

# Print all school districts in Alabama
$ grep "\tAL\t" schdist.txt
$ grep $'\tAL\t' schdist.txt
$ grep -P "\tAL\t" schdist.txt

# Print number of school districts in CA
$ cat schdist.txt | grep $'\tCA\t' | wc -l
$ grep -c $'\tCA\t' schdist.txt
```

Text filtering

```
# Print number of school districts in CA, AZ, WA, OR
# Have to use "extended" regex option -E
# to use OR conditions
$ grep -cE $'\t(WA|OR|CA|AZ)\t' schdist.txt

# Print all school districts in 408 and 650 area code
# To print their number, add "-c" option
$ grep -E $'\t(408|650)[0-9]{7}\t' schdist.txt

# Print count of school districts *not* in
# CA, AZ, WA, OR (invert matching)
$ grep -cEv $'\t(WA|OR|CA|AZ)\t' schdist.txt

# Print school districts in Daly City, ignore case
$ grep -iE $'\tdaly city\t' schdist.txt
```

Text replacement

```
# Change to square brackets without affecting text
$ echo "(make square brackets)" | tr "()" "[]"

# Change text to lowercase
$ echo "THIS WAS UPPERCASE" | tr "[:upper:]" "[:lower:]"

# delete 'i' from 'thisstring'
$ echo "thisstring" | tr -d 'i'

# delete all characters *except* 'i' from 'thisstring'
$ echo "thisstring" | tr -Cd 'i'

# Replace all i's in 'thisstring' with 'j'
$ echo "thisstring" | tr 'i' 'j'

# Replace all characters *other than* 'i' with 'j'
$ echo 'thisstring' | tr -C 'i' 'j'

# Delete all occurrences of '~' from file
$ cat /path/to/file | tr -d "~"
```

Text replacement

Quiz

Replace all `tab` characters in `schdist.txt` by |

Complex text replacement with sed

```
# sed format
$ sed 'pattern cmd/oldpat/newpat/flag' filename

# Basic sed usage
$ echo "sunday" | sed 's/day/night/'
$ echo "sunday monday" | sed 's/day/night/'
$ echo "sunday monday" | sed 's/day/night/g'

# sed understands regular expressions
# Replace lat/long in schdist.txt with x
$ head -2 schdist.txt | sed -E 's/[0-9]+\.[0-9]+/x/g'

# Remove fractional part in lat/long
$ head -2 schdist.txt | sed -E 's/([0-9])+\.[0-9]+/\1/g'

# Change first letter of every line to uppercase
$ cat names.txt | sed -E 's/^[a-z]/\U&/'
```

sed can perform all text replacement tasks, just search!

sed use case

Pig script template

```
-- script.pig
A = load 'tto_dwh.trans_clickstream'
using org.apache.hcatalog.pig.HCatLoader();

B = filter A by (load_date=='@START_DATE'
and reportsuite_name=='TTO');

C = foreach (group B by RANDOM()) generate FLATTEN(B);
store C into '/user/rkekatpure-admin/@OUTPUT_DIR';
```

Create a materialized script

```
$ cat script.pig | \
sed "s/\@START_DATE/$START_DATE/" | \
sed "s/\@OUTPUT_DIR/$HDFS_OUTPUT_DIR/" \
> "script_20150511.pig"
```

Working with columnar data

cut, awk, uniq, sort

```
# Select second column from a CSV file
$ cut -d"," -f2 schdist.txt

# Select second column from a tab delimited (TSV) file
# Notice the single quotes around "\t" character
$ cut -d$'\t' -f2 schdist.txt

# Select fields 1, 3 and 25 in a TSV file
$ cut -d$'\t' -f1,3,25 schdist.txt

# Select fields 3 through 18 (inclusive) in a TSV file
$ cut -d$'\t' -f3-18 schdist.txt

# Select fields 1 and 2 through 10 in a TSV file
$ cut -d$'\t' -f1,2-10 schdist.txt
```

Working with columnar data

```
# Print number of columns in a delimited file
$ head -1 schdist.txt | awk -F"\t" '{print NF}'

# Get column numbers from header
$ head -1 schdist.txt | \
awk  'BEGIN {IFS="\t"; OFS="\n"}{$1=$1; print}' | nl

# Print number of fields in each record
$ cat schdist.txt | awk -F"\t" '{print NF}'

# Print number of fields if it is not 30
$ cat schdist.txt | awk  -F"\t" 'NF!=30 {print NF}'

# Print anomalous records
$ cat schdist.txt | awk -F"\t" 'NF!=30 {print}'

# Filter clean records for database ingestion
$ cat schdist.txt | awk -F"\t" 'NF==30 {print}' \
> schdist_clean.txt
```

Simple analytics

```
# Create a alphabetical list of states and number
# of school districts in each
$ cat schdist.txt |
cut -d$'\t' -f9 | sort | uniq -c | nl

# Reprocess by removing header
$ cat schdist.txt | awk 'NR>1 {print}' |
cut -d$'\t' -f9 | sort | uniq -c | nl

# Top 10 states with school
$ cat schdist.txt | awk 'NR>1 {print}' |
cut -d$'\t' -f9 | sort | uniq -c |
sort -k1nr | head -10 | nl
```

Case studies

- Fraud repository
- Survey repository
- Benchmarking Netezza