Part One

grep, sort, uniq, wc

Part 1: Outline

1. Review of wild cards and file expansion
2. Review of pipeline and redirections
3. Newlines and TABS
4. Quotations and variables
5. **grep**, **sort**, **uniq**, and **wc**

File Expansion (1)

Filename expansion. When bash sees

```
$ ls *.txt
```

It expands the command to, for example:

```
$ ls a.txt b.txt
```

This is the simplest type of file expansion

File expansion (2)

A fairly complete list of expansion terms

*	Zero or more characters
?	One of any character
\	Escape following special chararacter
[xyz]	Matches any of the enclosed chararacters
{a,b,c}	Matches any of the enclosed strings

File expansion: examples

```
*.faa
*.{faa,fna}
$ ls ?[frs]??[b-f]*.{txt,py}
Africa-stuff.txt  psyceval.py
$ rm ??_[0-9]*.gff
```

Escaping special chars

```
$ rm Harry Potter.pdf  # two files
$ rm Harry\ Potter.pdf # escape space
$ rm \*\&\?.txt # removes file ‘*&?.txt’
```

Never put spaces in filenames. Generally only use letters, ‘.’, ‘_’, and ‘-’

Redirection

Given A and B are programs and f is a file

```
A | B    Pipe STDOUT from A to STDIN of B
A > f    Overwrite f with A's STDOUT
A >> f   Append A's STDOUT to end of f
A < f    Send contents of f to A's STDIN
```

Anatomy of a pipeline

```
cat a.txt | sort | uniq > b.txt
```

Variables

Declaring variables:

```
x='/path/to/some/file.txt'
head $x
```

Environmental variables:

```
echo $PATH
```

Shell uses of ‘$’

1. Variables

```
x=’asdf’ # define variable
echo $x  # interpret as variable name
```

2. Special strings

```
cut --delimiter $’\a’ a.txt # interpret string
```

3. Retrieving output of a subshell

```
grep $(grep AT x.txt | head -1) y.txt
```

Types of Quotes: Single

Returns the inside string exactly

```
# $x will not be interpreted as a variable, not \t as a TAB
$ echo ‘\t$x\n’
\t$x\n

No funny business with single quotes
```

Types of Quotes: Double

- No file expansion within quotes
- Special characters interpretation
- Variable interpolation

Quotes Example

```
$ ls
a.txt b.txt c.txt d.pdf
$ x=*.txt
$ ls $x
a.txt b.txt c.txt
$ ls “$x”
ls: cannot access “*.txt”: No such file or directory
$ ls ‘$x’
ls: cannot access “$x”: No such file or directory
```

Newline and TAB

\n - commonly represents a newline

\t - commonly represents a TAB

These are two very widely used special characters.

ANSI C Quotes ($’ ‘)

```
# Interprets \t as TAB
$ echo $’\ta’
    a
# Yay! Unicode emoticons
$ echo $’\U1F608\u2661 Windows’
□♡ Windows
# Chinese characters
$ echo $’\u7535\u8111’
电脑
```

Quotes Overview

quote type	File Expansion	Variable Expansion	Special characters
None	YES	YES	NO
“ ”	NO	YES	NO
‘ ’	NO	NO	NO
$‘ ’	NO	NO	YES

Command Substitution

Evaluates as shell commands, replace with output of command

```
echo `date`
rm `ls *.log | grep -L ‘error’`
rm $(ls *.log | grep -L ‘error’) # preferred
```

New Commands: grep

grep - a general, line-by-line search tool

```
$ grep zebra masters-of-sed.txt
zebrazial's arcane sed abilities were
he kept a zebrafish in his pocket whenever
```

some grep options

`--help` list of options and brief explanations

`-E, --extended-regexp`
`-i, --ignore-case`
`-v, --invert-match`
`-h, --no-filename`
`-r, --recursive`
`-c, --count`

`-B, --before-context`
`-A, --after-context`
`-C, --context`
`-L, --files-without-match`
`-l, --files-with-match`

```
grep --help | grep context
```

New Commands: sort

sorts data line-by-line in various ways

```
$ echo ‘a\nc\nb’ | sort
a
b
c
```

some sort options

`--help` list of options and brief explanations

`-g, --general-numeric-sort` (scientific notation)

`-n, --numeric-sort` `-R, --random-sort`

`-r, --reverse` `-f, --ignore-case`

Sorting by column:

`-k, --key=POS1[,POS2]`

`-t, --field-separator=SEP`

New Commands: uniq

deals with unique lines in various ways

INPUT MUST ALREADY BE SORTED

So sort ALWAYS appears upstream of uniq

uniq options

`--help` list of options and brief explanations

`-c, --count` count occurences of each line

`-d, --repeated` print only duplicated lines

`-u, --unique` print only uniq lines

`-i, --ignore-case`

```
sort first-names.txt | uniq -c | sort -rnk 1
```

New Commands: wc

word **c**ount - counts words, characters, lines

```
$ wc -l a.txt
84  # number of lines in file
$ sort a.txt | uniq -d | wc -l
3   # number of duplicated lines
```

Examples

```
# Sort the names by frequency
sort firstnames.txt | uniq -c | sort -rnk 1,1
# Count the number of uniq names
sort first-names.txt | uniq | wc -l
# Count the names that occur only once
sort first-names.txt | uniq -u | wc -l
```

Your Turn

First download the material from github

```
git clone https://github.com/zbwrnz/adv-unix-workshop
```

Navigate to `1st` folder, read the note