CS 246 -

Cameron Roopnarine

Last updated: January 9, 2020

Contents

1 2020-01-07

1.1 Linux Shell

Shell: interface to an OS

Graphical shell: click/touch, intuitive

Command line shell: type commands, not intuitive, more powerful

Stephen Bourne (70s): original UNIX shell Bourne Again Shell: bash

This course uses bash, a command line shell.

# is a comment within the shell.

```
# check what command line shell
echo $0
# go into bash
bash
```

1.2 Linux File System

Directories: files that contain files (called folders in Windows), e.g. usr, share, dict are all directories

Root (a backslash) /: top directory

Path: location of a file in a file system, e.g. /usr/share/dict/words

Absolute path: path that starts at the root directory

Relative path: path relative to a directory. The path dict/words relative to /user/share is /usr/share/dict/-words

1.3 Examples of Commands

```
# view non-hidden files in the current directory
ls
# list all files (including hidden ones), the -a is an argument
ls -a
# print working directory (prints absolute path of current directory)
pwd
# change directory
cd -argument
# possible arguments for cd : explanation
# .. : parent directory (back one, up one directory)
# . : current directory
# - : previous directory
# ~ : home directory
# ~userid : userid's directory
```

2 2020-01-09

It's strongly recommend that you **do not** memorize these commands presented, you should try them out on your own to see what the output is.

CTRL + C : kill signal

```
# print out all text on console
cat FILENAME
```

CTRL + D : EOF (end-of-file)

> output redirection, overwrites files, e.g.

```
# Redirects output produced by cat to the file out.txt
cat > out.txt
# Redirects all text from t1.txt to t2.txt
cat t1.txt > t2.txt
```

>> appends at the end of the file instead of overwriting like >

```
# Input redirection, the shell handles this
cat < sample.txt
# The cat (program) handles this, in this case sample.txt is an argument
cat sample.txt
# cat: command
# -n: argument, adds a line number to all lines
# < in: input redirection
# > out: output redirection
cat -n < in > out
```

2.1 Linux Streams

1. Standard input (stdin): keyboard, use < to change to file

2. Standard output (stdout): screen, use 1> to change to file, side note: the 1 is not needed before the >, buffered

3. Standard error (stderr): screen, use 2> to change to file, non-buffered

We use the non-buffered stream when we immediately want to output an error so that it does not take extra CPU cycles (extra material).

Within the stream,

stdin $\rightarrow$ program $\rightarrow$

1) stdout

2) stderr

```
# &1 is the location (address in C like CS 136) of stdout
prog arg1 < in > out 2>&1
```

2.2 Wildcard Matching

```
# *.txt is a globbing pattern, it will match anything that
# ends with .txt, the shell performs this operation
ls *.txt
# concatenate
cat
```

single/double quotes will suppress globbing patterns

\ is the escape character

2.3 Example

Count the number of words in the first 15 lines of sample.txt.

Solution.

```
# print number of words in entire text
wc –w
# get only the first 15 lines of sample.txt
head –15 sample.txt
# putting it all together
head –15 sample.txt > temp.txt wc –w temp.txt
```

What if we didn't want temp.txt to be produced?

2.4 Linux Pipe

Connect stdout of program 1 to stdin of program 2.

```
head –15 sample.txt | wc –w
```

Suppose words*.txt that contain words one per line produce a duplicate list of words in words*.txt.

```
cat words*.txt | sort –u
cat words*.txt | sort | uniq
# $(date) is embedding a command date
echo Today is $(date)
```

Double quotes: not supressing embedded commands

Single quotes: suppressing embedded commands