CS 35L-5

Week 2 Lec 1

Linux Commands continued..

- pipeline: |
- redirection: >, >>, <
- ls, cat, head, tail
- grep: selects input lines that match given pattern
  - grep ‘ERROR’ file.txt
- wc: word, line, character and byte count
  - wc -l file.txt

sort, comm and tr

sort: sorts lines of text files
  Usage: sort [OPTION]…[FILE]…

comm: compare two sorted files line by line, select/reject lines **comm**on to 2 files
  Usage: comm [OPTION]…FILE1 FILE2
  comm -23 file1 file2

tr: **tr**anslate or delete characters
  Usage: tr [OPTION]…SET1 [SET2]
  echo "password a1b2c3" | tr -d [:digit:] -> password abc
  echo “abc” | tr [:lower:] [:upper:] -> ABC

sed, test and expr

sed: stream editor, modifies the input as specified by the command(s)
  substitution – s/regex/replacement/flags
  sed s/day/night/g < oldFile > newFile
  echo $PATH | sed s/:.*//
  sed 's/<[^>]*>//g' a.html

test: evaluates an expression; exit status = 0(true), 1(false), >1(error)
  test 4 -gt 3; equivalent to [4 -gt 3]   **!!Spaces around [and]!!**

expr: evaluates the expression and returns the result
  a=$(expr $a + 1)

more sed examples

- sed 12,18d file.txt// delete 12-18 lines
- sed -n 12,18p file.txt
- sed '1~3d' file.txt
- sed '1,20 s/Johnson/White/g' file.txt
- sed '/pattern/d' file.txt
- sed -e ‘1p’ -e ‘3p’ file.txt
- sed -n -e '/BEGIN/,/END/p' file.txt
- sed '/regexp/!d' file.txt
- sed '/./!d' file.txt

Regular Expressions

- Quantification

- How many times of previous expression?
- Most common quantifiers: ?(0 or 1), *(0 or more), +(1 or more)

- Alternation

- Which choices?
- Operators: [] and |
  Hello|World [A B C]

- Anchors

- Where?
- Characters: ^ (beginning) and $ (end)

- **^** start of line
- **$** end of line
- \ turn off special meaning of next character
- **[ ]** match any of enclosed characters, use **-** for range
- **[^]** match any characters except those enclosed in []
- **.** match a single character of any value
- ***** match 0 or more occurrences of preceding character/expression
- **+** match 1 or more occurrences of preceding
- **\\{x,y\\}** match x to y occurrences of preceding
- **\\{x\\}** match exactly x occurrences of preceding
- **\\{x,\\}** match x or more occurrences of preceding

http://www.robelle.com/smugbook/regexpr.html

Quoting - To preserve literal meaning of special characters

- Escape Character \ - Literal value of following character
  echo \\|

- Single Quote - Literal Meaning of all within ‘’
  $hello=1
  $str=‘$hello’
  echo $str -> $hello

- Double Quote - Literal meaning except for $, ` and \\.
  $hello=1
  $str=“abc$hello”
  echo $str -> abc1

  Backquote - execute the command
  echo \`ls\` -> prints result after running ls

Shell Scripting

- **Shell**: The shell provides you with an interface to the UNIX system.
  -It gathers input from you and executes programs based on that input.
  -When a program finishes executing, it displays that program's output.
- **Shell-script**: A file containing shell commands (and comments - preceded by #) to execute
  - The #! First Line (shebang): a way to tell the kernel which shell to use for a script
  #!/bin/sh
  - Make it executable: chmod +x scriptFile
  - Execute: path_to_script/scriptFile or
    sh path_to_script/scriptFile

Shell Programming Constructs

Variables

- Valid character string [a-zA-Z0-9_] to which a value is assigned
  var_name=var_value **!!No spaces around =!!**
- Access using $: echo $var_name
- Special Variables: certain characters reserved as special variables
  $: PID of current shell
  #: number of arguments the script was invoked with
  n: nth argument to the script
  ?: exit status of the last command executed
  echo $$; echo $#; echo $2; echo $?;
- scalar variable vs array variable:
  array_name[index]=value; echo ${array_name[index]}

Accessing Shell Script Arguments

Example:

```
$ who | grep betsy                         Where is betsy?
betsy pts/3 Dec 27 11:07
```

Script:

```
#! /bin/sh
# finduser --- see if user named by first argument is logged in
who | grep $1
```

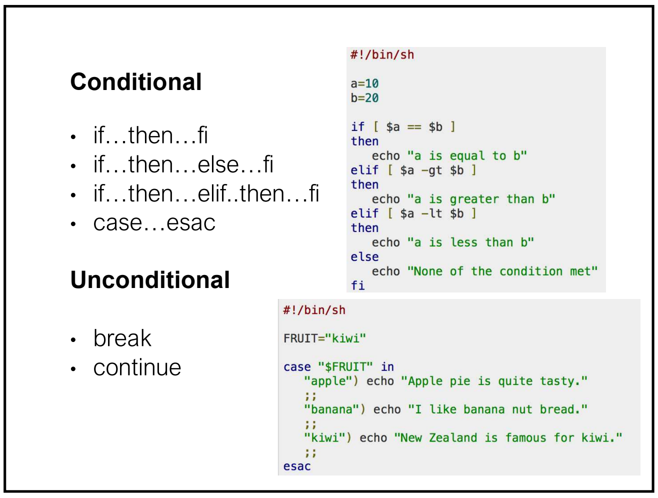
Run it:

```
$ chmod +x finduser                   Make it executable
$ ./finduser betsy                    Test it: find betsy
betsy pts/3 Dec 27 11:07
$ ./finduser benjamin                 Now look for Ben
benjamin dtlocal Dec 27 17:55
```

Looping

```
• for var in list_values
  do
    command 1
    ..
    command n
  done

• while condition
  do
    command 1
    ..
    command n
  done
```



Conditional

- if…then…fi
- if…then…else…fi
- if…then…elif..then…fi
- case…esac

Unconditional

- break
- continue

```
#!/bin/sh

a=10
b=20

if [ $a == $b ]
then
   echo "a is equal to b"
elif [ $a -gt $b ]
then
   echo "a is greater than b"
elif [ $a -lt $b ]
then
   echo "a is less than b"
else
   echo "None of the condition met"
fi
```

```
#!/bin/sh

FRUIT="kiwi"

case "$FRUIT" in
   "apple") echo "Apple pie is quite tasty."
   ;;
   "banana") echo "I like banana nut bread."
   ;;
   "kiwi") echo "New Zealand is famous for kiwi."
   ;;
esac
```