# CS5103: Computing Lab-1

## Shell scripting


**Arijit Mondal**

**Dept of CSE**

arijit@iitp.ac.in

https://www.iitp.ac.in/~arijit/

# Purpose

- Putting commands in a file
- Special purpose code
- Runs in interpreted mode
- Rarely used when speed is concerned
- Mostly used for automation

# Basic Commands

- ls
- cat
- cp
- mv
- date
- who
- more
- echo
- expr
- diff
- cd
- head
- tail
- rm
- wc
- pwd
- mkdir
- ps
- chmod
- man
- sed
- awk
- test
- grep
- sort
- touch
- find
- nohup
- stty
- sleep
- seq
- getopts
- bc
- set

# Basic Commands

- `ls`
- `cat`
- `cp`
- `mv`
- `date`
- `who`
- `more`
- `echo`
- `expr`
- `diff`
- `cd`
- `head`
- `tail`
- `rm`
- `wc`
- `pwd`
- `mkdir`
- `ps`
- `chmod`
- `man`
- `sed`
- `awk`
- `test`
- `grep`
- `sort`
- `touch`
- `find`
- `nohup`
- `stty`
- `sleep`
- `seq`
- `getopts`
- `bc`
- `set`

- *command optional-arguments optional-filenames*

# Shell scripts

- Program starts with one of the followings
  - `#!/bin/bash` (Other options depending on shells – `#!/bin/sh`, `#!/bin/tcsh`)
- Structure of a program

```
#!/bin/bash
# this is comment
Body of program
```

- Execution of the program
  - `bash scriptfile`
  - `chmod +x scriptfile`
  - `./scriptfile`

# Example: Hello world

```
$> cat firstprog
  #!/bin/bash
  echo "Hello world!"

$> bash firstprog
  Hello world!
```

```
$> ls -l firstprog
  -rw-rw-r-- ... firstprog
$> chmod +x firstprog
  -rwxrwxr-x ... firstprog
$> ./firstprog
  Hello world!
```

# Example: Variable

```
$> cat secondprog
  #!/bin/bash
  name="Shivi"
  echo "Hello $name"

$> bash secondprog
  Hello Shivi
```

# Example: Command line argument

```
$> cat scriptfile
  #!/bin/bash
  name=$1
  echo "Hello $name"

$> bash scriptfile Vikram
  Hello Vikram
```

# Example: Command line argument (cont)

```
$> cat scriptfile
  #!/bin/bash
  echo "There are $# parameters."
  echo "The parameters are $@"
  echo "The script name is $0"
  echo "The first parameter is $1"
  echo "The second parameter is $2"
$> bash scriptfile a b c
  There are 3 parameters.
  The parameters are a b c
  The script name is scriptfile
  The first parameter is a
  The second parameter is b
```

# Example: I/O

`$> prog > outfile`

- Output of `prog` will be stored in `outfile`

`$> prog >> outfile`

- Output of `prog` will be appended to `outfile`

`$> prog < infile`

- `prog` will read input from file `infile`

`$> prog1 | prog2`

- Output of `prog1` will be used as input for `prog2`
  - `who | wc -l`
  - `ls -l | grep ".txt$"`

# Reading input

- Example:

```
echo "Please, enter your firstname and lastname"
read FN LN
echo "Hi! $LN, $FN !"
```

# Conditional statement: if-then-else

- Example: if-then

```
if [ "xyz" = "abc" ]; then
   statements
fi
```

- Example: if-then-else

```
var1="xyz"; var2="abc";
if [ "$var1" = "$var2" ]; then
   statements
else
   statements
fi
```

# Loop - for

- Syntax of 'for' loop

```
for var in list
do
   statements
done
```

- Example

```
list="1 2 3 4"
for var in $list
do
   echo $var
done
```

Output:

```
1
2
3
4
```

# Loop - while

- Syntax of 'while' loop

```
while condition
do
    statements
done
```

- Example

```
count=1
while [ $count -le 5 ]
do
    echo "Count: $count"
    ((count++))
done
```

Output:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

# Local variable & function

- Example:

```
HELLO=Hello
function myfunc() {
    local HELLO=World
    echo $HELLO
}
echo $HELLO
myfunc
echo $HELLO
```

Output:

```
Hello
World
Hello
```

# Function with parameters

- Example:

```
function myfunc2(){
   exit
}
function myfunc1() {
   echo $1
}
myfunc1 Hello
myfunc1 World
myfunc2
echo "Here"
```

Output:

Hello
World

# Arithmetic operation

- Example
  - `echo $[45*10]`

    450

  - `echo $((55*10))`

    550

  - `xyz=echo`$((55*10))`; echo $xyz`

    550

# sed

- `$> cat infile`

```
abcd efgh
IIT Patna
Mtech CSE students
Bihta
Bihar
```

- `$> sed '/IIT/p' infile`

- Output:

```
abcd efgh
IIT Patna
IIT Patna
Mtech CSE students
Bihta
Bihar
```

# sed (contd)

- `$> cat infile`

```
abcd efgh
IIT Patna
Mtech CSE students
Bihta
Bihar
```

- `$> sed -n '/IIT/p' infile`

- Output:

```
IIT Patna
```

- Same as grep

# sed (contd)

- `$> cat infile`

```
abcd efgh
IIT Patna
Mtech CSE students
Bihta
Bihar
```

- `$> sed '/IIT/d' infile`

- Output:

```
abcd efgh
Mtech CSE students
Bihta
Bihar
```

- Deleted the line containing the pattern

# sed (contd)

- `$> cat infile`

```
abcd efgh
IIT Patna
Mtech CSE students
Bihta
Bihar
```

- `$> sed 's/IIT/I_I_T/g' infile`

- Output:

```
abcd efgh
I_I_T Patna
Mtech CSE students
Bihta
Bihar
```

- Original file remains the same

# sed (contd)

- `$> cat infile`

  ```
  abcd efgh
  IIT Patna
  Mtech CSE students
  Bihta
  Bihar
  ```

- `$> sed -i 's/IIT/I_I_T/g' infile`

- There will be no output in the terminal
- `$> cat infile`

  ```
  abcd efgh
  I_I_T Patna
  Mtech CSE students
  Bihta
  Bihar
  ```

# awk

- `$> cat infile`

```
name S1 S2 S3
abc 45 90 89
cde 33 47 98
lkm 97 76 52
pqr 76 20 02
jkl 78 88 88
```

- `$> awk '{print $1,$2}' infile`

- Output

```
name S1
abc 45
cde 33
lkm 97
pqr 76
jkl 78
```

# awk (contd)

- $> cat infile

```
name S1 S2 S3
abc 45 90 89
cde 33 47 98
lkm 97 76 52
pqr 76 20 02
jkl 78 88 88
```

- $> awk '{print $1","$2","$3","$4}' infile

- Output

```
name,S1,S2,S3
abc,45,90,89
cde,33,47,98
lkm,97,76,52
pqr,76,20,02
jkl,78,88,88
```

# awk (contd)

- $> cat infile

```
name S1 S2 S3
abc 45 90 89
cde 33 47 98
lkm 97 76 52
pqr 76 20 02
jkl 78 88 88
```

- $> awk 'BEGIN{s=0} {s=s+$2}
  END{print s}' infile

- Output

```
329
```

# awk (contd)

- $> cat infile

```
name S1 S2 S3
abc 45 90 89
cde 33 47 98
lkm 97 76 52
pqr 76 20 02
jkl 78 88 88
```

- $> awk 'BEGIN{s=0}
  {if(NR!=1){s=s+$2}}
  END{print s}' infile

- Output

  329

# awk (contd)

- $> cat infile

```
name S1 S2 S3
abc 45 90 89
cde 33 47 98
lkm 97 76 52
pqr 76 20 02
jkl 78 88 88
```

- $> awk '

```
{if(NR==1){print $0,"Total"}
 else{print $0,$2+$3+$4}}
' infile
```

- Output

```
name S1 S2 S3 Total
abc 45 90 89 224
cde 33 47 98 178
lkm 97 76 52 225
pqr 76 20 02 98
jkl 78 88 88 254
```

# getopts

```
#!/bin/bash
filename=""; myarg=""; minsize="" ; # Initialize variables
while getopts ":f:t:i:" opt; do
   echo "Option: $opt Argument: $OPTARG"
   case $opt in
   i) minsize="$OPTARG" ;;
   f) filename="$OPTARG" ;;
   t) myarg="$OPTARG" ;;
   \?) echo "Invalid option: -$OPTARG" >&2 ; exit 1 ;;
   :) echo "Option -$OPTARG requires argument." >&2; exit 1 ;;
   esac
done
shift $((OPTIND - 1)) # Shift past the processed options
echo "minsize: $minsize File: $filename Remaining-arguments: $@"
```