Shell Programming
Lec-05

Use of Semicolons

- Instead of being on separate lines, statements can be separated by a semicolon (;)
  - For example:

```
if grep "UNIX" myfile; then echo "Got it"; fi
```

Use of Colon

- Sometimes it is useful to have a command which does “nothing”.
- The : (colon) command in Unix does nothing

```
#!/bin/sh
if grep unix myfile
then
  :
else
   echo "Sorry, unix was not found"
fi
```

The test Command – File Tests

- test –f file does file exist and is not a directory?
- test -d file does file exist and is a directory?
- test –x file does file exist and is executable?
- test –s file does file exist and is longer than 0 bytes?

```
#!/bin/sh
count=0
for i in *; do
   if test –x $i; then
    count=`expr $count + 1`
   fi
done
echo Total of $count files executable.
```

The test Command – Integer Tests

- Integers can also be compared:
  - Use -eq, -ne, -lt, -le, -gt, -ge
- For example:

```
#!/bin/sh
smallest=10000
for i in 5 8 19 8 7 3; do
  if test $i -lt $smallest; then
      smallest=$i
   fi
done
echo $smallest
```

Use of [ ]

- The test program has an alias as [ ]
  - Each bracket must be surrounded by spaces!
  - This is supposed to be a bit easier to read.
- For example:

```
#!/bin/sh
smallest=10000
for i in 5 8 19 8 7 3; do
   if [ $i -lt $smallest ] ; then
      smallest=$i
   fi
done
echo $smallest
```

The while Loop

- While loops repeat statements as long as the next Unix command is successful.
- For example:

```
#!/bin/sh
i=1
sum=0
while [ $i -le 100 ]; do
  sum=`expr $sum + $i`
  i=`expr $i + 1`
done
echo The sum is $sum.
```

Command Line Arguments (1)

- Shell scripts would not be very useful if we could not pass arguments to them on the command line
- Shell script positional arguments are "numbered" from left to right
  - $1 - first argument after command
  - $2 - second argument after command
  - ... up to $9
  - They are called "positional parameters".

Command Line Arguments (2)

- Example: get a particular line of a file
  - Write a command with the format:

    getlineno *linenumber filename*

    ```
    #!/bin/sh
    head -$1 $2 | tail -1
    ```

- Other variables related to arguments:
  - $0 name of the command running
  - $* All the arguments
  - $# the number of arguments

Reading Variables From Standard Input (1)

- The read command reads one line of input from the terminal and assigns it to variables given as arguments

- Syntax: read var1 var2 var3 ...
  - Action: reads a line of input from standard input
  - Assign first word to var1, second word to var2, ...
  - The last variable gets any excess words on the line.

Reading Variables from Standard Input (2)

- Example:

```
% read X Y Z
Here are some words as input
% echo $X
Here
% echo $Y
are
% echo $Z
some words as input
```

Assignment-1 Additional Questions

1. Write a shell script to find the largest among the 3 given numbers.
2. Write a shell script to ask your name, program name and enrolment number and print it on the screen.