Lesson 10: Writing Scripts

Objectives covered

- 105.2 Customize or write simple scripts (weight: 4)
- 107.2 Automate system administration tasks by scheduling jobs (weight: 4)

Customize or write simple scripts

Global environment variables

```
$ printenv
HOSTNAME=testbox.localdomain
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=192.168.1.2 1358 22
OLDPWD=/home/rich/test/test1
SSH_TTY=/dev/pts/0
USER=rich
LS_COLORS=no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:
bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=00;32:
*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=00;32:*.bat=00;32:
*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:*.arj=00;31:
*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:
*.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=00;31:
*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=00;35:*.xbm=00;35:
*.xpm=00;35:*.png=00;35:*.tif=00;35:
MAIL=/var/spool/mail/rich
PATH=/usr/kerberos/bin:/usr/lib/ccache:/usr/local/bin:/bin:/usr/bin:
/home/rich/bin
INPUTRC=/etc/inputrc
PWD=/home/rich
LANG=en_US.UTF-8
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
SHLVL=1
HOME=/home/rich
LOGNAME=rich
CVS_RSH=ssh
SSH_CONNECTION=192.168.1.2 1358 192.168.1.4 22
LESSOPEN=|/usr/bin/lesspipe.sh %s G_BROKEN_FILENAMES=1
_=/usr/bin/printenv
```

Local environment variables

```
$ set
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="3" [1]="2" [2]="9" [3]="1" [4]="release"
[5]="i686-redhat-linux-gnu")
BASH_VERSION='3.2.9(1)-release'
COLORS=/etc/DIR_COLORS.xterm
COLUMNS=80
CVS_RSH=ssh
DIRSTACK=()
EUID=500
GROUPS=()
G_BROKEN_FILENAMES=1
HISTFILE=/home/rich/.bash_history
HISTFILESIZE=1000
HISTSIZE=1000
HOME=/home/rich
HOSTNAME=testbox.localdomain
HOSTTYPE=i686
IFS=$' \t\n'
INPUTRC=/etc/inputrc
LANG=en_US.UTF-8
LESSOPEN='|/usr/bin/lesspipe.sh %s'
LINES=24
LOGNAME=rich
LS_COLORS='no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;
01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=00;32:*.cmd=00;32:
*.exe=00;32:*.com=00;32:*.btm=00;32:*.bat=00;32:*.sh=00;32:
*.csh=00;32:*.tar=00;31:*.tgz=00;31:*.arj=00;31:*.taz=00;31:
*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:*.bz2=00;31:
*.bz=00;31:*.tz=00;31:*.rpm=00;31:*.cpio=00;31:*.jpg=00;35:
*.gif=00;35:*.bmp=00;35:*.xbm=00;35:*.xpm=00;35:*.png=00;35:
*.tif=00;35:'
MACHTYPE=i686-redhat-linux-gnu
MAIL=/var/spool/mail/rich
MAILCHECK=60
OPTERR=1
OPTIND=1
OSTYPE=linux-gnu
PATH=/usr/kerberos/bin:/usr/lib/ccache:/usr/local/bin:/bin:/usr/bin:
/home/rich/bin
PIPESTATUS=([0]="0")
PPID=3702
PROMPT_COMMAND='echo -ne
"\033]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}"; echo -ne "\007"'
PS1='[\u@\h \W]\$ '
PS2='> '
PS4='+ '
```

Setting local and global variables

```
$ test=testing
$ echo $test
testing
```

```
$ test=testing a long string
 -bash: a: command not found
 $ test='testing a long string'
 $ echo $test
 testing a long string
```

```
$ bash
$ test=testing
$ echo $test
testing
$ exit
exit
$ echo $test
```

Local variables

```
$ echo $test
testing a long string
$ export test
$ bash
$ echo $test
testing a long string
```

Global variables

Locating System Environment Variables

Login shell	Interactive shell	Noninteractive shell
1. /etc/profile 2. $HOME/.bash_profile 3. $HOME/.bash_login 4. $HOME/.profile	$HOME/.bashrc	$BASH_ENV variable

Using Command Aliases

```
$ alias -p
alias l.='ls -d .* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls --color=tty'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-
alias--show-dot --show-tilde'
```

Using Command Aliases

```
$ alias li='ls -il'
$ li
total 52
  4508 drwxr-xr-x 2 rich rich 4096 Jun 12 11:21 Desktop
  4512 drwxr-xr-x 2 rich rich 4096 Jun 12 11:21 Documents
  4509 drwxr-xr-x 2 rich rich 4096 Jun 12 11:21 Downloads
```

```
$ alias li='ls -il'
$ bash
$ li
bash: li: command not found
```

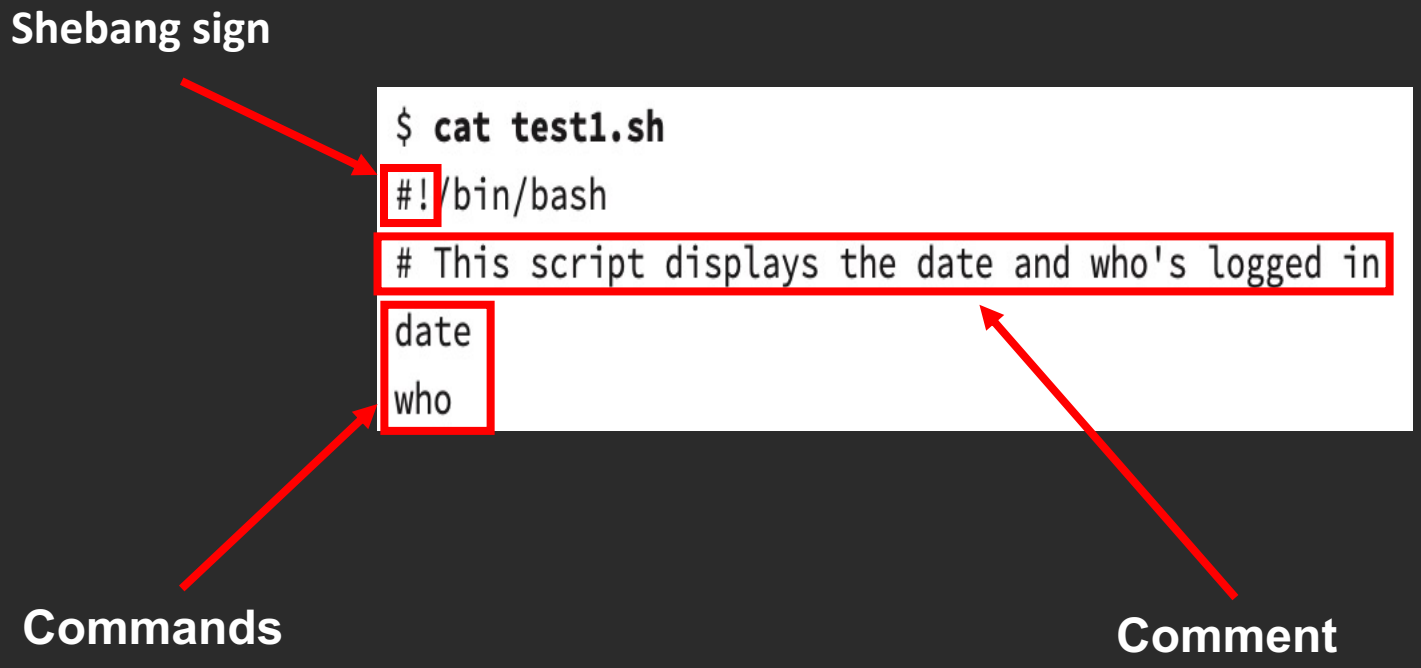
Shell features

```
$ date ; who
Thu Feb 20 19:20:06 EST 2019
rich     :0           2019-02-20 19:15 (:0)
```

```
$ ls | sort
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
test.txt
today.txt
Videos
```

```
$ who >> today.txt
$ cat today.txt
Thu Feb 20 19:21:12 EST 2019
rich     :0           2019-02-20 19:15 (:0)
```

Shell script format



Running shell scripts

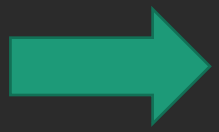
```
$ ls -l test1.sh
-rw-r--r-- 1 rich rich 73 Feb 20 19:37 test1.sh
```

```
$ ./test1.sh
Thu Feb 20 19:48:27 EST 2019
rich     :0           2019-02-20 19:15 (:0)
```

Shell variables

Global environment variables

```
$ cat test2.sh
#!/bin/bash
# display user information from the system.
echo User info for userid: $USER
echo UID: $UID
echo HOME: $HOME
```



```
$ ./test2.sh
User info for userid: rich
UID: 1000
HOME: /home/rich
```

Local variables

```
$ cat test3.sh
#!/bin/bash
# testing variables
days=10
guest=Katie
echo $guest checked in $days days ago
```

```
$ ./test3.sh
Katie checked in 10 days ago
```

Shell script arguments

```
$ cat para_script
echo First Parameter entered was $1
echo Second Parameter entered was $2
echo Third Parameter entered was $3

      $0          $1   $2    $3
$ para_script Good Day Sydney
First Parameter entered was Good
Second Parameter entered was Day
Third Parameter entered was Sydney
```

Time for labs

Execise 20: Writing scripts – Part 1

Shell script – Accept input from user

```
$ cat test5.sh
#!/bin/bash
# testing the read command

echo -n "Enter your name: "
read name
echo "Hello $name, welcome to my program."
$ ./test5.sh
Enter your name: Rich Blum
Hello Rich Blum, welcome to my program.
```

```
$ cat test6.sh
#!/bin/bash
# testing the read -p option

read -p "Please enter your age:" age
days=$[ $age * 365 ]
echo "That makes you over $days days old!"
$ ./test6.sh
Please enter your age:10
That makes you over 3650 days old!
```

```
$ cat test7.sh
#!/bin/bash
# entering multiple variables

read -p "Enter your name: " first last
echo "Checking data for $last, $first…"
$ ./test7.sh
Enter your name: Rich Blum
Checking data for Blum, Rich…
```

```
$ cat test8.sh
#!/bin/bash
# testing the REPLY environment variable

read -p "Enter a number: "
factorial=1
for (( count=1; count <= $REPLY; count++ ))
do
   factorial=$[ $factorial * $count ]
done
echo "The factorial of $REPLY is $factorial"
$ ./test8.sh
Enter a number: 5
The factorial of 5 is 120
```

Shell script – Accept input from user

Timing out reading

```
$ cat test9.sh
#!/bin/bash
# timing the data entry

if read -t 5 -p "Please enter your name: " name
then
   echo "Hello $name, welcome to my script"
else
   echo
   echo "Sorry, too slow!"
fi
$ ./test9.sh
Please enter your name: Rich
Hello Rich, welcome to my script
$ ./test9.sh
Please enter your name:
Sorry, too slow!
```

Limit the character for reading

```
$ cat test10.sh
#!/bin/bash
# getting just one character of input

read -n1 -p "Do you want to continue [Y/N]? " answer
case $answer in
Y | y) echo
       echo "fine, continue on…";;
N | n) echo
       echo OK, goodbye
       exit;;
esac
echo "This is the end of the script"
$ ./test10.sh
Do you want to continue [Y/N]? Y
fine, continue on…
This is the end of the script
$ ./test10.sh
Do you want to continue [Y/N]? n
OK, goodbye
```

Shell script – Accept input from user

Silent reading

```
$ cat test11.sh
#!/bin/bash
# hiding input data from the monitor

read -s -p "Enter your password: " pass
echo
echo "Is your password really $pass?"
$ ./test11.sh
Enter your password:
Is your password really T3st1ng?
```

Shell script - Exit status

Command completed successfully have the exit value 0

```
$ who
rich     :0           2019-02-20 23:16 (:0)
$ echo $?
0
```

Control the exit value of shell script with exit command

```
$ /bin/bash
$ exit 120
exit
$ echo $?
120
```

Shell script programming

Command subtitution

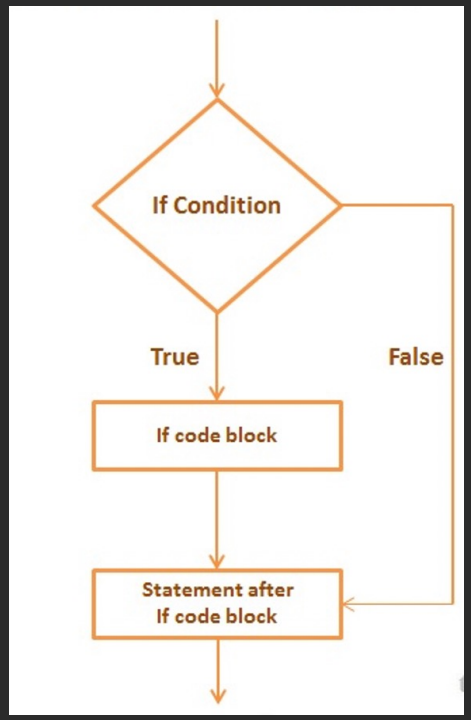
```
$ var1=`date`
$ echo $var1
Fri Feb 21 18:05:38 EST 2019
$ var2=$(who)
$ echo $var2
rich :0 2019-02-21 17:56 (:0)
```

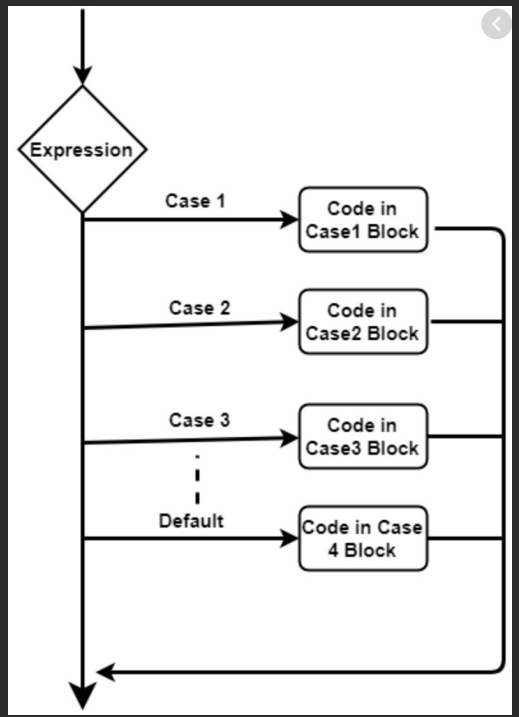
Performing math

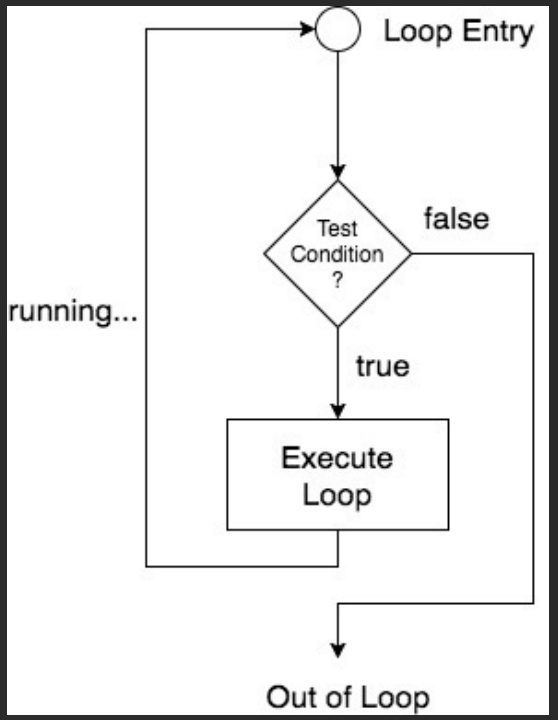
```
result=$[ 25 * 5 ]
```

```
$ var1=$(echo "scale=4; 3.44 / 5" | bc)
$ echo $var1
.6880
```

Shell script programming







Shell script programming

```
if [ condition ]
then
    commands
fi
```

Test	Type	Description
n1 -eq *n2*	Numeric	Checks if *n1* is equal to *n2*
n1 -ge *n2*	Numeric	Checks if *n1* is greater than or equal to *n2*
n1 -gt *n2*	Numeric	Checks if *n1* is greater than *n2*
n1 -le *n2*	Numeric	Checks if *n1* is less than or equal to *n2*
n1 -lt *n2*	Numeric	Checks if *n1* is less than *n2*
n1 -ne *n2*	Numeric	Checks if *n1* is not equal to *n2*
str1 = *str2*	String	Checks if *str1* is the same as *str2*
str1 != *str2*	String	Checks if *str1* is not the same as *str2*
str1 < *str2*	String	Checks if *str1* is less than *str2*
str1 > *str2*	String	Checks if *str1* is greater than *str2*
-n *str1*	String	Checks if *str1* has a length greater than zero
-z *str1*	String	Checks if *str1* has a length of zero
-d *file*	File	Check if *file* exists and is a directory
-e *file*	File	Checks if *file* exists
-f *file*	File	Checks if *file* exists and is a file
-r *file*	File	Checks if *file* exists and is readable
-s *file*	File	Checks if *file* exists and is not empty
-w *file*	File	Checks if *file* exists and is writable
-x *file*	File	Checks if *file* exists and is executable
-O *file*	File	Checks if *file* exists and is owned by the current user
-G *file*	File	Checks if *file* exists and the default group is the same as the current user
file1 -nt *file2*	File	Checks if *file1* is newer than *file2*
file1 -ot *file2*	File	Checks if *file1* is older than *file2*

Shell script programming

```
if [ condition ]
then
    commands
fi
```

```
$ cat test12.sh
#!/bin/bash
# testing the if condition
if [ $1 -eq $2 ]
then
   echo "Both values are equal!"
   exit
fi

if [ $1 -gt $2 ]
then
   echo "The first value is greater than the second"
   exit
fi

if [ $1 -lt $2 ]
then
   echo "The first value is less than the second"
   exit
fi
$ ./test12.sh 10 5
The first value is greater than the second
```

Shell script programming

case variable in
pattern1) commands1;;
pattern2 | pattern3) commands2;;
*) default commands;;
esac

```
$ cat test13.sh
#!/bin/bash
# using the case statement

case $USER in
rich | barbara)
   echo "Welcome, $USER"
   echo "Please enjoy your visit";;
testing)
   echo "Special testing account";;
jessica)
   echo "Don't forget to log off when you're done";;
*)
   echo "Sorry, you're not allowed here";;
esac
$ ./test13.sh

Welcome, rich
Please enjoy your visit
```

Time for labs

Execise 21: Writing scripts – Part 2

Shell script programming

for variable in series ; do
    commands
done

```
$ cat test14.sh
#!/bin/bash
# iterate through the files in the Home folder
for file in $(ls | sort) ; do
   if [ -d $file ]
   then
      echo "$file is a directory"
   fi
   if [ -f $file ]
   then
      echo "$file is a file"
   fi
done
$ ./test14.sh
Desktop is a directory
Documents is a directory
Downloads is a directory
Music is a directory
Pictures is a directory
```

Shell script programming

```
while [ condition ] ; do
    commands
done
```

```
$ cat test15.sh
#!/bin/bash
number=$1
factorial=1
while [ $number -gt 0 ] ; do
   factorial=$[ $factorial * $number ]
   number=$[ $number - 1 ]
done
echo The factorial of $1 is $factorial
$ ./test15.sh 5
The factorial of 5 is 120
$ ./test15.sh 6
The factorial of 6 is 720
```

Shell script programming

function name {
    commands
}

Or

name() {
    commands
}

```
$ cat test16.sh
#!/bin/bash
# using a function in a script

function func1 {
   echo "This is an example of a function"
}

count=1
while [ $count -le 5 ]
do
   func1
   count=$[ $count + 1 ]
done

echo "This is the end of the loop"
func1
echo "Now this is the end of the script"
$ ./test16.sh
This is an example of a function
This is an example of a function
This is an example of a function
This is an example of a function
This is an example of a function
This is the end of the loop
This is an example of a function
Now this is the end of the script
```

```
$ cat test17.sh
#!/bin/bash
# using the return command in a function

function dbl {
   read -p "Enter a value: " value
   echo "doubling the value"
   return $[ $value * 2 ]
}

dbl
echo "The new value is $?"
```

Running script in background

```
$ ./test18.sh &
[1] 19555
$ This is test program
Loop #1
Loop #2
```

```
$ nohup ./test18.sh &
[1] 19831
$ nohup: appending output to  'nohup.out'
```

Automate system administration tasks by scheduling jobs

Scheduling script to run – at

at [-f filename] time

Time format

- A standard hour and minute, such as 10:15
- An AM/PM indicator, such as 10:15PM
- A specific named time, such as now, noon, midnight, or teatime (4PM)
- A standard date format, such as MMDDYY, MM/DD/YY, or DD.MM.YY
- A text date, such as Jul 4 or Dec 25, with or without the year
- You can also specify a time increment:
  - Now + 25 minutes
  - 10:15PM tomorrow
  - 10:15 + 7 days

Scheduling script to run – at

at send output of the script via mail

```
$ date
Thu Feb 28 18:48:20 EST 2019
$ at -f test3.sh 18:49
job 2 at Thu Feb 28 18:49:00 2019
$ mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/rich": 1 message 1 new
>N  1 Rich                  Thu Feb 28 18:49  15/568   "Output from your job "
&
Message  1:
From rich@localhost.localdomain  Thu Feb 28 18:49:00 2019
Return-Path: <rich@localhost.localdomain>

X-Original-To: rich
Delivered-To: rich@localhost.localdomain
Subject: Output from your job        2
To: rich@localhost.localdomain
Date: Thu, 28 Feb 2019 18:49:00 -0500 (EST)
From: rich@localhost.localdomain (Rich)
Status: R

"This script ran at 18:49:00"
"This is the end of the script"
```

Scheduling script to run – at

List pending jobs

```
$ at -f test21.sh 19:15
warning: commands will be executed using /bin/sh
job 7 at 2007-11-04 10:15
$ at -f test21.sh 4PM
warning: commands will be executed using /bin/sh
job 8 at 2007-11-03 16:00
$ at -f test21.sh 1PM tomorrow
warning: commands will be executed using /bin/sh
job 9 at 2007-11-04 13:00
$ atq
7       2007-11-04 10:15 a
8       2007-11-03 16:00 a
9       2007-11-04 13:00 a
```

Remove job

```
$ atrm 8
$ atq
7       2007-11-04 10:15 a
9       2007-11-04 13:00 a
```

Scheduling script to run – cron table

min hour dayofmonth month dayofweek command

```
15 10 * * * /home/rich/test21.sh > test21out
```

Command	Description
$ crontab –l	List all the entries on the cron table of your user
$ export EDITOR=vi	Set the EDITOR variable value to vi editor for using when edit the crontab
$ crontab –e	Edit the crontab with the EDITOR pre set

Time for labs

Execise 22: Writing scripts – Part 3

Question...