UNIX FUNdamentals

Student Workbook

Revision 1.2

CONTENTS

EXERCISE 1 - ACCESS TO UNIX SYSTEMS

Exercise 1 - Instructions

1. Log in to the system with the user name and password provided by your instructor. It should be a user name such as teamxx where xx is a double digit number like 01, 02, and so forth.
   The very first time you log in with your new user name, you will be prompted to change your password. Keep the password the same as your login name. The passwords you supply on the command line will not be displayed.

2. Change your current password to a password that you would like to use for the remainder of the class. Memorize the new password. Remember, writing down a password could lead to a security problem.

3. Verify that the password has been set by logging out and back in.

4. There are two commands that will display information about all users currently on the local system. Display who is currently logged in on your system.
   Check to see when they logged in.

5. Display just your login name.

END Of Exercise 1

Exercise 1 - Instructions With Hints

1. Log in to the system with the user name and password provided by your instructor. It should be a user name such as teamxx where xx is a double digit number like 01, 02, and so forth.
   The very first time you log in with your new user name, you will be prompted to change your password. Keep the password the same as your login name. The passwords you supply on the command line will not be displayed.

   login: teamxx (at the login prompt)

   Password: teamxx (default password same as user name)

   You are required to change your password. Please choose a new one. teamxx's New

   password: teamxx (keep it the same for now). Enter new password again: teamxx

2. Change your current password to a password that you would like to use for the remainder of the class. Memorize the new password. Remember, writing down a password could lead to a security problem.

   $ passwd

   Changing password for teamxx teamxx's

   Old password: teamxx's New password:

   Enter the new password again:

3. Verify that the password has been set by logging out and back in.

   $ exit

   login: teamxx

   Password: (key in your new password)

4. There are two commands that will display information about all users currently on the local system. Display who is currently logged in on your system.
   Check to see when they logged in.

```
$ who
```

-OR

```
$ finger <username>
```

5. Display just your login name.

```
$ whoami
```

END Of Exercise 1

EXERCISE 2 - UNIX COMMANDS & SHELL BASICS

Exercise 2 - Instructions

UNIX Command Basics

1. Display the system's date:

2. Display a count of just the number of lines in a file named /etc/passwd:

3. Display the whole calendar for the year 2000:

4. Display the month of September for the year 1752. Notice anything peculiar about September?

5. Display the month of January for the year 1999 and 99. Are 1999 and 99 the same?

6. Use banner to display Out to Lunch:

7. Use the echo command to write the character string Out to Lunch to your display:

8. Use the clear command to clear your screen:

Variable Substitution

9. Display the shell built in variables

10. Set a Variable called LUNCH to soup and a variable called DINNER to sandwich. Display these two new variables using the echo command. Locate them in the list of shell variables.

11. Ensure you are in your home directory. Create a new directory called quoting.

12. Change to the quoting directory. Create a zero length file in the quoting directory named fileq. Create a shell variable named Q set the value of Hello. Test your work above by listing the contents of the directory quoting and the value for Q.

13. From the quoting directory, execute the following commands. Record the output. Check the "with hints section" for the expected output.

$echo '* $Q `ls` $(ls)

$echo "* $Q `ls` $(ls)"

$echo * \$Q \`ls\`\$\(ls\)

$echo * $Q `ls` $(ls)

$echo * $Q ls

END Of Exercise 2

Exercise 2 - Instructions With Hints

UNIX Command Basics

1. Display the system's date:

   $ date

2. Display a count of just the number of lines in a file named /etc/passwd:

   $ wc –l /etc/passwd

3. Display the whole calendar for the year 2000:

   $ cal 2000

4. Display the month of September for the year 1752. Notice anything peculiar about September?

   $ cal 9 1752

5. Display the month of January for the year 1999 and 99. Are 1999 and 99 the same?

   cal1 1999

   cal1 99

6. Use the banner command to display Out to Lunch:

$ banner Out to Lunch

7. Use the echo command to write the character string Out to Lunch to your display:

$ echo “Out to Lunch”

8. Use the clear command to clear your screen:

$ clear

Variable Substitution

9. Display the shell built in variables

$ set OR env

10. Set a Variable called LUNCH to soup and a variable called DINNER to sandwich. Display these two new variables using the echo command. Locate them in the list of shell variables.

$ LUNCH=soup

$ DINNER=sandwich

$ echo $LUNCH ; echo $DINNER

$ echo set

11. Ensure you are in your home directory. Create a new directory called quoting.

$ cd

$ pwd

$ mkdir quoting

12. Change to the quoting directory. Create a zero length file in the quoting directory named fileq. Create a shell variable named Q set the value of Hello. Test your work above by listing the contents of the directory quoting and the value for Q.

$ cd quoting

$ touch fileq

$ Q=Hello

$ ls

$ echo $Q

13. From the quoting directory, execute the following commands. Record the output. Check the "with hints section" for the expected output.

$echo '* $Q `ls` $(ls)

Single quotes suppresses everything between them.

$echo "* $Q `ls` $(ls)"

Double quotes do command and variable substitution only.

$echo * \$Q \`ls\`\$\(ls\)

Backlash negates the character following it. NB the use of backslash in front of each backquote.

$echo * $Q `ls` $(ls)

fileq Hello fileq fileq

$echo * $Q ls

Fileq Hello ls

END Of Exercise 2

EXERCISE 3 UNIX Processes & Job Control

Exercise 3 - Instructions

UNIX Processes

1. Display just your processes:

2. What is the PID of the current shell you are running:

3. Show all the processes running on the host:

4. Count the number of processes in use by the user root:

5. From question 10 what is the process id of your current shell?

Write down your Process id here =

6. Create a subshell by typing ksh. What is the process id of this new subshell? Are the process id's for the two shells different?

Job Control

7. Ensure your in your home directory. Execute the command ls –R / 1> outfile 2> errfile in the foreground.

8. Suspend the job you just started and using cat or pg view the contents of the two redirected files.

9. List all the jobs that you are running on the system and restart the above job in the background:

10. A file called jobproc can be found in your home directory. It contains the following lines :

    ls –R / 1> outfile 2> errfile&

    find / -name "*.log" –print 1> outfile1 2> errfile1&

    Modify the permissions on the file so that ONLY the owner can read, write, and execute the file (hint: use chmod 700 jobproc). Start the script, reference it using an explicit path and put it in the background. Check and make a note of the process id of this background job. Attempt to exit the subshell (this will take two attempts) and note any messages. Now check to see if the process (pid) is still running?

11. Start the script with the nohup command, reference it using an explicit path and put it in the background. Note the process id and exit the subshell (this may take two attempts) and note any messages.

Process id =

12. Check your own shell processes and then check for the existence of the process id from question 11 above using the grep command.

13. Start the script, reference it using an explicit path and put it in the background. Don't forget to redirect the output to a file (e.g. jobproc.log). Note the process id.

14. If you did not note the process id in question 13 above, how could you find it? Once you know the process id kill the process. Checked to see if the process was killed.

End Of Exercise 3

Exercise 3 – Instructions With Hints

UNIX Processes

1. Display just your processes:

   $ ps

2. What is the PID of the current shell you are running:

   $ echo $$

3. Show all the processes running on the host:

   $ ps -ef

4. Count the number of processes in use by the user root:

   $ ps –ef | grep root | wc -l

5. From question 10 what is the process id of your current shell?

   echo $$

   Write down your Process id here =

6. Create a subshell by typing ksh. What is the process id of this new subshell? Are the process id's for the two shells different?

ksh

echo $$

Yes they should be different.

Job Control

7. Ensure your in your home directory. Execute the command ls –R / 1> outfile 2> errfile in the foreground.

ls –R / 1> outfile 2> errfile

8. Suspend the job you just started and using cat or pg view the contents of the two redirected files.

$<ctrl-z>

9. List all the jobs that you are running on the system and restart the above job in the background:

$ jobs

$ bg %<jobno.>

10. A file called jobproc can be found in your home directory. It contains the following lines:

ls –R / 1> outfile 2> errfile&

find / -name "*.log" –print 1> outfile1 2> errfile1 &

Start a new new korn shell and goto your home directory. Modify the permissions on the file so that ONLY the owner can read, write, and execute the file (hint: use chmod 700 jobproc). Start the script, reference it using an explicit path and put it in the background. Check and make a note of the process id of this background job. Attempt to exit the subshell (this will take two attempts) and note any messages. Now check to see if the process (pid) is still running?

$ ksh

$ cd

$ chmod 700 jobproc

$ ./jobproc &

$ ps

$exit (You should get a message saying "There are running jobs")

$ exit

$ ps –ef | grep <process id>

The process will not be present in the process table as the shell it was running in has been closed.

11. Start the script with the nohup command, reference it using an explicit path and put it in the background. Note the process id and exit the subshell (this may take two attempts) and note any messages.

$ nohup ./jobproc &

Process id =

$ exit A message is displayed “There are running processes”

$ exit No message is displayed.

A message something like “Sending nohup output to nohup.out.” should be displayed. Press Enter to clear the message.

12. Check your own shell processes and then check for the existence of the process id from question 11 above using the grep command.

$ ps

$ ps –ef | grep <process id>

You should see three processes, the first should be the ./jobproc script running, the second should be the sleep command running and finally there should be the grep process you just ran. Note the ./jobproc now has a ppid of 1.

13. Start the script, reference it using an explicit path and put it in the background. Don’t forget to redirect the output to a file (e.g. jobproc.log). Note the process id.

$ ./jobproc &

14. If you did not note the process id in question 13 above, how could you find it? Once you know the process id kill the process. Checked to see if the process was killed.

$ ps -f

$ kill <process id>

$ ps -f

END Of Exercise 3

EXERCISE 4 - FILESYSTEMS & DIRECTORIES

Exercise 4 - Instructions

1. Change from your current directory to the root directory:

2. Change current directory to your home directory (There are 3 ways to do this):

3. Using the pwd command verify you are in your home directory:

4. List your home directory (current) contents:

5. Execute a long listing of the current directory (your home directory):

6. Issue the ls command with the –a and –ltr options:

7. Ensuring you are in your home directory, make a new directory called newdir and list its contents:

8. Move into the new directory you created above in question 7 and create a new file called myfile1 (hint: do not use vi, try the touch command inside)

9. Redirect (append) the output of ls –l into the new file you created above (myfile1):

10. Rename the file myfile1 to myfile2

11. Copy myfile2 to a new file called myfile1

12. Change the permissions on the file myfile2 so that the owner, group and other (world) have read only access. Long list the contents of the directory to check this is correct:

13. Change the group ownership of the file myfile1 to the lp group. What happens?

14. Delete the two new files in newdir, what happens and why?

15. Delete the directory (and its contents) newdir:

END Of Exercise 4

Exercise 4 Instructions With Hints

1. Change from your current directory to the root directory:

   $ cd /

2. Change current directory to your home directory (There are 3 ways to do this):

   $ cd

   $ cd $HOME

   $ cd /home/teamxx

3. Using the pwd command verify you are in your home directory:

   $ pwd

4. List your home directory (current) contents:

   $ ls

5. Execute a long listing of the current directory (your home directory):

   $ ls -l

6. Issue the ls command with the –a and –ltr options:

   $ ls –a

   $l-ltr

7. Ensuring you are in your home directory, make a new directory called newdir and list its contents:

$ pwd

$ mkdir newdir

$ls –l newdir

8. Move into the new directory you created above in question 7 and create a new file called myfile1 (hint: do not use vi, try the touch command inside)

$ cd newdir

touch myfile1

9. Redirect (append) the output of ls –l into the new file you created above (myfile1):

$ ls –l >> myfile1

10. Rename the file myfile1 to myfile2

$ mv myfile1 myfile2

11. Copy myfile2 to a new file called myfile1

$ cp myfile2 myfile1

12. Change the permissions on the file myfile2 so that the owner, group and other (world) have read only access. Long list the contents of the directory to check this is correct:

$ chmod u-wx g-wx o-wx myfile2

$ chmod u=r g=r o=r myfile2

$chmod 444 myfile2

$ls -l

13. Change the group ownership of the file myfile1 to the lp group. What happens?

$ chgrp lp myfile1

You cannot change the group permissions to lp because you do not have permission to do so.

14. Delete the two new files in newdir, what happens and why?

$ rm my*

The two files are deleted because you have permission to do this as the file permissions are inherited from the directory.

15. Delete the directory (and its contents) newdir:

$ cd -

pwd

rm –R newdir

END Of Exercise 4

EXERCISE 5 - Using vi

Exercise 5 - Instructions

1. Check you are in your home directory. Create a file called vitest.

2. When you open a vi file you are automatically placed in command mode. Press the <i> key (insert) to switch to text mode. You can also press the <a> (append) key. The use of i key or a key simply determines if typing starts before or after the cursor. There is currently no indication that you are in input mode.
   To switch from input mode to command mode press the <esc> key. Press <esc> a second time and you will/should hear an audible beep from the terminal. The beep indicates that you are in command mode already.

3. Input the text below, exactly as you see it, line by line. Then key in the alphabet one character per line. The following will show a-d, but continue through to z.

   This is a training session about using the vi editor. So we will need a few lines of text to learn the most common commands of this editor. We are in entry mode and switch to command mode when we finish the text
   a
   b
   c
   d
   .
   .
   z

4. Now save and quit the file:

5. Open the file vitest using vi. Using the cursor movement keys h, j, k, l practice moving the cursor down one line, up one line, right a couple of characters and back (left) a couple of characters.

6. Next try the following methods of moving around the document (in command mode):

- Move forward one page
- Move back one page
- Scroll the screen up ½ the window size
- Move the cursor to the last line in the file.
- Move the cursor to the first line in the file
- Move the cursor to line 4 of the file
- Move the cursor to the end of the line
- Move the cursor to the beginning of the line.

7. Quit the file without saving:

Customising your .profile & .kshrc files

8. The following exercise will customise your environment and have it take affect every time you login. The changes will be to the .profile file which is read every time you login. Open your .profile (found in your home directory) and make the following changes:

a) Change the primary prompt string to reflect the current directory

b) A message that will display your login name and the time you logged in each time you login into the host.
c) Define an alias named dir that invokes the command ls –l.

9. Exit and log back in, did you notice any differences?

10. Most settings, with the exception of system variables, only apply to the current environment and are not passed to subshells (child processes). To pass customised settings down to subshells, the ENV variable must be set in your .profile along with the existence of a customised .kshrc file.
    Customise your .profile to include the appropriate .kshrc file

11. Now edit or create the .kshrc file

12. Now log out, log back in, open a new subshell and create a new file using vi to see the differences made with .kshrc

END Of Exercise 5

Exercise 5 - Instructions With Hints

1. Check you are in your home directory. Create a file called vitest.

$ cd

$ pwd

$vi vitest

2. When you open a vi file you are automatically placed in command mode. Press the <i> key (insert) to switch to text mode. You can also press the <a> (append) key. The use of i key or a key simply determines if typing starts before or after the cursor. There is currently no indication that you are in input mode.
   To switch from input mode to command mode press the <esc> key. Press <esc> a second time and you will/should hear an audible beep from the terminal. The beep indicates that you are in command mode already.

<i>

<esc>

<esc> Did you hear the beep?

3. Input the text below, exactly as you see it, line by line. Then key in the alphabet one character per line. The following will show a-d, but continue through to z.

<i>

This is a training session about using the vi editor. So we will need a few
lines of text to learn the most common commands of this editor. We are in
entry mode and switch to command mode when we finish the text
a
b
c
d
.
.
Z

<esc>

4. Now save and quit the file:

:wq

OR

<shift-ZZ>

5. Open the file vitest using vi. Using the cursor movement keys h, j, k, l practice moving the cursor down one line, up one line, right a couple of characters and back (left) a couple of characters.

vi vitest

j (down a line)

k (up a line)

l (right a character)

h (left a character)

6. Next try the following methods of moving around the document (in command mode):

- Move forward one page
- Move back one page
- Scroll the screen up ½ the window size
- Move the cursor to the last line in the file.
- Move the cursor to the first line in the file
- Move the cursor to line 4 of the file
- Move the cursor to the end of the line
- Move the cursor to the beginning of the line.

<ctrl-f> Move forward one page

<ctrl-b> Move back one page

<ctrl-u> Scroll the screen up ½ the window size

<shift-G>	Move the cursor to the last line in the file
1<shift-G> OR :1	Move the cursor to the first line in the file
4<shift-G> OR :4	Move the cursor to line 4 of the file
$	Move the cursor to the end of the line
<shift-^> OR 0 (zero)	Move the cursor to the beginning of the line

7. Quit the file without saving:

:q!

Customising your .profile & .kshrc files

8. The following exercise will customise your environment and have it take affect every time you login. The changes will be to the .profile file which is read every time you login. Open your .profile (found in your home directory) and make the following changes:

d) Change the primary prompt string to reflect the current directory
e) A message that will display your login name and the time you logged in each time you login into the host.
f) Define an alias named dir that invokes the command ls –l.

$ cd

$ pwd

$ vi. profile

PS1='$PWD => '

echo User $LOGNAME logged in at `$date`

alias dir='ls –l'

:wq

9. Exit and log back in, did you notice any differences?

```
$ exit
```

10. Most settings, with the exception of system variables, only apply to the current environment and are not passed to subshells (child processes). To pass customised settings down to subshells, the ENV variable must be set in your .profile along with the existence of a customised .kshrc file.
    Customise your .profile to include the appropriate .kshrc file

```
$ cd
$ pwd
$ vi. profile
PS1='$PWD => '
ENV=/home/teamxx/.kshrc
export PATH PS1 ENV
echo User $LOGNAME logged in at $date
:wq
```

11. Now edit or create the .kshrc file

```
$ vi .kshrc
set –o vi
alias dir='ls –l'
:wq
```

12. Now log out, log back in, open a new subshell and create a new file using vi to see the differences made with .kshrc

END Of Exercise 5

EXERCISE 6 - REDIRECTION & REGULAR EXPRESSIONS

Exercise 6 - Instructions

Wildcards

1. Change directories to /usr/bin and check the working directory. Now list all the files beginning with the letter a.

2. Now list all the two character file names:

3. Next list all the files starting with the letters a, b, c or d:

4. List all the files except those beginning with c through to t. BEWARE this will be a long list, so you may want to pipe the output to pg or more.

Redirection

5. Redirect the output from the command issued in question 3 to a file in your home directory called junk.

6. Now move to your home directory and append some text to the file junk – anything you like! To close the file goto a new line and type <ctrl-d>.

7. Next view the contents of the file using the more command and when complete remove the file from your home directory:

Pipes, Tees & Filters

8. Using the ls command list the files in your home directory and make a note of the number. Next list the files again, but redirect the output to a file called templst

9. Use the appropriate command to count the number of words and then the number of lines in the file templst. Remove the file templst.

10. Use the wc and ls commands to count the number of files in your home directory. Was the result what you expected? Was it the same as in question 9?

11. Use the command from question 10, but this time insert a tee in the middle to trap the result in a file called junk2. Check the contents of junk2 to make sure it is what you expected.

Command Grouping & Line Continuation

12. On the command line display the date, who is logged in, the name of your current directory and the names of the files in your current directory. Do these commands have any relationship to each other?

13. The main purpose of this instruction is use the line continuation (\) with a command that is too long to fit onto a single command line.

a) Do a long listing of files in your home directory including hidden files.
b) Capture the output to a file called listing
c) Count only the number of words in the file
d) Capture the number of words in a file called numwords
e) Count the number of lines in the file numwords and redirect the output to a file called pointless.log

END Of Exercise 6

Exercise 6 - Instructions With Hints

Wildcards

1. Change directories to /usr/bin and check the working directory. Now list all the files beginning with the letter a.

```
$cd /usr/bin
$ pwd
$ ls a*
```

2. Now list all the two character file names:

```
$ ls ??
```

3. Next list all the files starting with the letters a, b, c or d:

```
$ ls [abcd]*
OR
$ ls [a-d]*
```

4. List all the files except those beginning with c through to t. BEWARE this will be a long list, so you may want to pipe the output to pg or more.

```
$ ls /usr/bin | grep ^[^c-t] | pg
```

Redirection

5. Redirect the output from the command issued in question 3 to a file in your home directory called junk.

$ ls | grep [a-d]* > /home/teamxx/junk

6. Now move to your home directory and append some text to the file junk – anything you like! To close the file goto a new line and type <ctrl-d>.

$ cd

$ cat >> junk

Any text you like

<ctrl-d> on a new line to return to the shell prompt

7. Next view the contents of the file using the more command and when complete remove the file from your home directory:

$ more /home/teamxx/junk

$ rm /home/teamxx/junk

Pipes, Tees & Filters

8. Using the ls command list the files in your home directory and make a note of the number. Next list the files again, but redirect the output to a file called templst

$ ls /home/teamxx/

Number of files =

$ ls /home/teamxx/ > /home/teamxx/templst

9. Use the appropriate command to count the number of words and then the number of lines in the file templst. Remove the file templst.

$ wc –w templst

$ wc –l templst

$ rm templst

10. Use the wc and ls commands to count the number of files in your home directory. Was the result what you expected? Was it the same as in question 9?

$ cd

$ ls | wc –w

$ ls | wc –l

11. Use the command from question 10, but this time insert a tee in the middle to trap the result in a file called junk2. Check the contents of junk2 to make sure it is what you expected.

$ ls | tee junk2 | wc –w

Was the result as you expected?

$ cat junk2

Command Grouping & Line Continuation

12. On the command line display the date, who is logged in, the name of your current directory and the names of the files in your current directory. Do these commands have any relationship to each other?

```
$ date ; who ; pwd ; ls
```

Command grouping is just a shortcut for executing non-related commands from the same command line. They have NO relationship to each other.

13. The main purpose of this instruction is use the line continuation (\) with a command that is too long to fit onto a single command line.

f) Do a long listing of files in your home directory including hidden files.
g) Capture the output to a file called listing
h) Count only the number of words in the file
i) Capture the number of words in a file called numwords
j) Count the number of lines in the file numwords and redirect the output to a file called pointless.log

```
$ ls –al | tee listing | wc –w | tee numwords \
| wc –l >pointless.log
```

```
$ cat numwords
```

```
$ cat pointless.log
```

END Of Exercise 6

EXERCISE 7 – DISK & LOGICAL VOLUME MANAGEMENT

Exercise 7 - Instructions

1. List all the mounted filesystems on the host

2. List the disk utilisation for your current directory.

3. List the disk usage of all subdirectories and files (include hidden) within the /usr directory, sorted by filesize

4. List the total and free space in ALL the filesystems on the host, in kilobytes.

5. List the total and free space in current filesystems on the host, in kilobytes.

END of Exercise 7

Exercise 7 - Instructions With Hints

1. List all the mounted filesystems on the host

   $ mount

2. List the disk utilisation for your current directory.

   $ du -sk

3. List the disk usage of all subdirectories and files (include hidden) within the /usr directory, sorted by filesize

   $ mount

4. List the total and free space in ALL the filesystems on the host, in kilobytes.

   $ df –k

5. List the total and free space in current filesystems on the host, in kilobytes.

   $ df –k .

END of Exercise 7

EXERCISE 8 - UNIX TOOLS & UTILITIES

Exercise 8 - Instructions

The grep command

1. Find all the instances of team in the file /etc//passwd

2. Find all the lines in /etc/passwd that contain a digit 0-9.

3. Use the ps and grep commands to display the processes belonging to users other than yourself.

The sort command

4. Display the contents of the /etc/passwd file in alphabetic order and then in reverse order.

The find command

5. Find and display all the files in the /tmp directory

6. Find all the files in the /etc directory that end in .conf.

7. Find all the files in your home directory that begin with the letter s and have ls –l automatically execute on each file name found as a result of the search.

END of Exercise 8

Exercise 8 - Instructions With Hints

The grep command

1. Find all the instances of team in the file /etc//passwd

   $ grep '^team' /etc/passwd

2. Find all the lines in /etc/passwd that contain a digit 0-9.

   $ grep [0-9] /etc/passwd

3. Use the ps and grep commands to display the processes belonging to users other than yourself.

   $ ps –ef | grep –v teamxx

The sort command

4. Display the contents of the /etc/passwd file in alphabetic order and then in reverse order.

   $ sort /etc/passwd

   $ sort –r /etc/passwd

The find command

5. Find and display all the files in the /tmp directory

```
$ cd
```

```
$ find /tmp
```

6. Find all the files in the /etc directory that end in .conf.

```
$ find /etc –name “*.conf”
```

7. Find all the files in your home directory that begin with the letter s and have ls –l automatically execute on each file name found as a result of the search.

```
$ cd
```

```
$ find . –name ‘s’ –exec ls –l {} \;
```

END of Exercise 8

EXERCISE 9 - UNIX MANUAL PAGES

Exercise 9 - Instructions

1. Bring up the man pages for the man command. Read the text to gain a better understanding and functionality of the man command.

2. Using the man command search on the keyword calendar. From the list produced find the command that displays a calendar.

3. Having found the cal command from the previous step, use the man command without any options to obtain the correct syntax of the command

END of Exercise 9

Exercise 9 - Instructions With Hints

1. Bring up the man pages foir the man commad. Read the text to gain a better understanding and functionality of the man command.

   $ man man

   <ctrl-c> OR q to quit

2. Using the man command search on the keyword calendar. From the list produced find the command that displays a calendar.

   $ man –k calendar

3. Having found the cal command from the previous step, use the man command without any options to obtain the correct syntax of the command

   $ man cal

END of Exercise 9