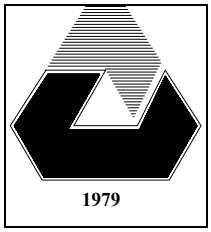

Eastern Mediterranean University

Department of Computer Engineering

CMPE-242: OPERATING SYSTEMS

EXPERIMENTAL WORK

Part A: *Interfacing with UNIX (Linux) and understanding the important Shell commands*

1. Login to your system, get the current date and time. Check the active users of your machine.
2. Using the Shell command "`man`" get and write down in your notebook the short description of the following Shell commands: `man, cal, cat, cd, cmp, cp, gcc, ls, kill, mkdir, mv, more, pr, ps, pwd, rm, sort, wc, who, write.`
3. Display the directory structure of your system and write down in your notebook.
4. Traverse the directory using the command "`cd`" and localise yourself using the command "`pwd`".
5. Select some text file in some directory and display its content.
6. Create a subdirectory in your home directory, copy some text file from any other directory into your subdirectory, and display the file from the subdirectory. Then remove the copied file and the respective subdirectory.
7. Check the command "`ls`" with different options. Consider the effect of this command with the redirection of its output into your home directory (with the subsequent printing redirected output).
8. Execute a text editor, and experiment some basic commands of general text editor (i.e. `open, save, cut, paste exit`). Then create a text files as *first.txt and second.txt* save them and exit.
9. Display the contents of text file *first.txt* which is created in step 8.
10. Count the number of words in *first.txt* using shell command only.
11. Merge two files which are created in Step 8 using "`cat`" command (name new file as *firstsecond.txt*).
12. What are the permissions of files? How can we modify the permissions of a file with command "`chmod`"?
13. Check the effect of symbol "&" at the end of any command. What will be displayed?

Part B: *Running a C program in UNIX (Linux)*

1. Connect to UNIX (Linux) system. Then, using a text editor input the given C program below.
2. Compile and link the program, using the command line below:

```
gcc  -o  date  date.c
```

```
/* date.c*/
#include  <stdio.h>
#include  <stdlib.h>
int main ()
{
     int day, year;
     day = 18;
     year = 2018;
     printf("day=%d year=%d\n", day, year);
     exit(0);
}
```

3. Run this program using the given command line below and write the result.

```
./date
```

Part C: *Running a C program in UNIX (Linux) by entering parameters from the command line*

1. Connect to UNIX (Linux) system. Then, using a text editor input the given C program below.
2. Compile and link the program, using the command line below:

gcc -o date1 date1.c

```
/* date1.c*/
#include  <stdio.h>
#include  <stdlib.h>

int main (int argc, char *argv[])
{
     int day, year;
     /* Program usage */
     if (argc != 3)
     {
         printf("Usage: %s <day> <year>\n", argv[0]);
         exit(1);
     }
     day = atoi(argv[1]); /*  <stdlib.h> is necessary*/
     year = atoi(argv[2]);

     printf("day=%d year=%d\n", day, year);
     exit(0);
}
```

3. Run this program for the given command lines below and write down results of each run.

```
a) ./date1
b) ./date1 26
c) ./date1 26 2012
```

Assignment 1: *Writing a C program in UNIX (Linux)*

Write a C program for UNIX that will be executed with the given command line below. Your program will find summation of three parameters as integer and display all these integer numbers and summation. Do not forget to write usage of the program.

./sum 10 20 30

Part D: *Usage of fork() system call in a C program*

1. Connect to UNIX (Linux) system. Then, using a text editor input the given C program below.
2. Compile and link the program, using the command line below:

```
gcc  -o  fork1  fork1.c
```

```
/* fork1.c*/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int i;
    pid_t pid;
    printf("Message A by %d \n", getpid());
    fork();
    if (pid == -1)
       {
         perror("fork");
         exit(1);
       }
    printf("Message B by %d \n", getpid());

    pid=fork();
    if (pid == -1)
       {
         perror("fork");
         exit(1);
       }
    printf("Message C by %d \n", getpid());

    exit(0);
 }
```

3. Run this program using the given command line below 3 times and write the results for each run. Is there any difference in the results? Explain your answer.

```
./fork1
```

4. Using ***man*** command get information about ***fork(), getpid(), perror()*** and ***exit()*** in Linux system. (e.g. **./ man fork**)
5. Draw a timing diagram (like a tree) to show fork(), exit() and printed messages by each process (assume that all system calls are successful). How many processes are there? How many messages are printed?
6. Logout from the system.

NOTE: The results of all steps of this experiment should be written in a white A4 paper. And it must be in report format (possibly, with additional explanations).

Assignment 2: *Usage of fork(), getpid() and getppid() system calls in a C program*

1. Connect to UNIX (Linux) system. Then, using a text editor input the given C program below.
2. Compile and link the program, using the command line below:

```
gcc  -o  fork2  fork2.c
```

```
/* fork2.c*/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main ()
{
    int i;
    pid_t pid;
    printf("Message A by %d \n", getpid());
    pid=fork();
    if (pid == -1)
       {
         perror("fork");
         exit(1);
       }
    if (pid == 0)
       { /*Child code*/
         printf("Child process ID:%d \n", getpid());
         printf("Parent process ID:%d \n", getppid());

         for (i=0; i<5; i++)
         {
          printf("Child: Message B\n");
          sleep(5);
```

```
        }
        printf("Child process terminating.\n");
        exit(0);
       }
    /*Parent code*/
    printf("My ID: %d \n", getpid());
    for (i=0;i<5;i++)
        {
         printf("Parent: Message C\n");
         sleep(5);
        }
    printf("Parent process waiting for child.\n");
    wait();
    printf("Parent process terminating.\n");
    exit(0);
}
```

3. Run this program using the given command line below 3 times and write the results for each run. Is there any difference in the results? Explain your answer.

   `./fork2`

4. Using ***man*** command get information about ***getppid(), sleep(), wait() and exec()*** in Linux system. (e.g. **./ man fork**)
5. Draw a timing diagram (like a tree) to show fork(), wait(), exit() and printed messages by each process (assume that all system calls are successful). How many processes are there?

NOTE: The results of all steps of this assignment should be written in a white A4 paper. And it must be in report format.

BASIC UNIX COMMANDS

Description of Some Basic UNIX command

COMMAND	DESCRIPTION
alias	change command names or options
cal	display a calendar
cat	list or concatenate files
cd	change working directory
chmod	change modes (permissions) of a file
chsh	change between shells
cmp	compare two files
cp	copy files
date	print date and time
exit	leave shell (normally leave UNIX)
file	attempt to interpret contents of a file
grep	search a file for a limited regular expression
groups	list groups user belongs to
logout	leave UNIX
lpq	show which files are queued for printer
lpr	print a file
lprm	remove file from printer queue
ls	list filenames
man	display manual pages
mkdir	make a new directory
more	Display contents of a file
mv	move contents elsewhere (change name of file)
pr	format text, suitable for printing
printenv	print environment variables
pwd	print working directory
rm	remove (distroy) a file
rmdir	remove (destroy) a directory
sort	sort or merge files
wc	count words, lines, and characters in a file
who	give information about current logged on users
write	send messages to another user's terminal

man command.

This is help command, and you can use man in conjunction with any command to learn more about that command for example.

- man ls will explain about the ls command and how you can use it.
- man -k pattern command will search for the pattern in given command.

cal command

cal command will print the calander on current month by default. If you want to print calander of august of 1965. That's eightht month of 1965.
cal 8 1965 will print following results.

```
    August 1965
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

clear command

clear command clears the screen and puts cursor at beginning of first line.

pwd command.

pwd command will print your home directory on screen, pwd means print working directory.

```
/home/stdt/c067890
```

ls command

ls command is most widely used command and it displays the contents of directory.

- ls will list all the files in your home directory, this command has many options.
- ls -l will list all the file names, permissions, group, etc in long format.
- ls –d will list directories.
- ls *[1,3,5] will list files that end with 1 or 3 or 5.
- ls *[1-5] will list files that end with numbers between 1 and 5 {1,2,3,4,5}

The * wildcard

The character * is called a wildcard, and will match against none or more character(s) in a file (or directory) name. For example, in your unixstuff directory, type

% ls list*

This will list all files in the current directory starting with list....

Try typing

% ls *list

This will list all files in the current directory ending with ....list

The ? wildcard

The character ? will match exactly one character.

So ?ouse will match files like house and mouse, but not grouse.

Try typing

% ls ?list

who command

who command displays information about the current status of system.
Who as default prints login names of users currently logged in.

date command.

Date displays todays date, to use it type date at prompt.

```
Sun Dec  7 14:23:08 EST 2008
```

is similar to what you should see on screen.

write command will initiate an interactive conversation with user. Syntax is

write *username*

telnet command.

Telnet command invokes a telnet protocol which lets you log on to unix machine connected over TCP/IP protocol, IPx protocol or otherwise. Syntax is
telnet hostname

rmdir command.

rmdir command will remove directory or directories if a directory is empty.

Options:

- rm -r directory_name will remove all files even if directory is not empty.
- rmdir direc1 is how you use it to remove direc1 directory.
- rmdir -p will remove directories and any parent directories that are empty.
- rmdir -s will suppress standard error messages caused by -p.

rm command.

To delete files use rm command.

Options:

- rm oldfile will delete file named oldfile.
- rm -f option will remove write-protected files without prompting.
- rm -r option will delete the entire directory as well as all the subdirectories, very dangerous command.

mv command.
mv command is used to move a file from one directory to another directory or to rename a file.

Some examples:

- mv oldfile newfile will rename oldfile to newfile.
- mv -i oldfile newfile for confirmation prompt.
- mv -f oldfile newfile will force the rename even if target file exists.
- mv * ~/bag will move all the files in current directory to ~/bag directory.

cp command.
cp command copies a file. If I want to copy a file named oldfile in a current directory to a file named newfile in a current directory.
cp oldfile newfile
If I want to copy oldfile to other directory for example tmp then
cp oldfile tmp/newfile.

cp -r directory1 directory2 copy directory1 (and its contents) to directory2

cp vol/example/ex1.txt . copy ex1.txt to the current directory.

wc command
wc command counts the characters, words or lines in a file depending upon the option.

Options

- wc -l filename will print total number of lines in a file.
- wc -w filename will print total number of words in a file.
- wc -c filename will print total number of characters in a file.

head command.
head filename by default will display the first 10 lines of a file.
If you want first 50 lines you can use head -50 filename or for 37 lines head -37 filename and so forth.

tail command.

tail filename by default will display the last 10 lines of a file.
If you want last 50 lines then you can use tail -50 filename.

cd command.

cd bag will change directory from current directory to bag directory.
Use pwd to check your current directory and ls to see if bag directory is there or not.
You can then use cd bag to change the directory to this new directory.

cd ~ will take us to our home directory.

cd.. will take us to the upper directory.

cd../.. : will take us two directories higher.

cat command

cat cal.txt cat command displays the contents of a file here **cal.txt** on screen (or standard out).

This is one of the most flexible Unix commands. We can use to create, view and concatenate files. For our first example we create a three-item English-Turkish dictionary in a file called "dict."

```
$ cat >dict
  red kirmizi
  green yesil
  blue mavi
<control-D>
```

<control-D> stands for "hold the control key down, then tap 'd'". The symbol > tells the computer that what is typed is to be put `into` the file `dict`. To view a file we use `cat` in a different way:

```
$ cat dict
  red kirmizi
  green yesil
  blue mavi
$
```

If we wish to add text to an existing file we do this:

```
$ cat >>dict
  White beyaz
  Black siyah
  <control-D>
$
```

Now suppose that we have another file `tmp` that looks like this:

```
$ cat tmp
  cat kedi
  dog kopek
$
```

Then we can join `dict` and `tmp` like this:

```
$ cat dict tmp >dict2
```

mkdir command.
mkdir bag will create new directory, i.e. here bag directory is created.

whoami --- returns your username. Sounds useless, but isn't. You may need to find out who it is who forgot to log out somewhere, and make sure you have logged out.

passwd --- lets you change your password

logout or **exit** **---** ends terminal session.

chmod -- is used to change permissions on a file.

for example if I have a text file with calender in it called cal.txt. initially when this file will be created the permissions for this file depends upon ***umask*** set in your profile files. As you can see this file has **666** or **-rw-rw-rw** attributes.

ls -l cal.txt

-rw-rw-rw- 1 ssb dxidev 135 Dec 3 16:14 cal.txt

In this line above I have -rw-rw-rw- meaning respectively that **owner can read and write file, member of the owner's group can read and write this file and anyone else connected to this system can read and write this file., next ssb is owner of this file dxidev is the group of this file, there are 135 bytes in this file, this file was created on December 3 at time16:14 and at the end there is name of this file**.
Learn to read these permissions in binary, like this for example Decimal 644 which is 110 100 100 in binary meand rw-r--r-- or user can read,write this file, group can read only, everyone else can read only. Similarly, if permissions are 755 or 111 101 101 that means rwxr-xr-x or user can read, write and execute, group can read and execute, everyone else can read and execute. All directories have d in front of permissions. So if you don't want anyone to see your files or to do anything with it use chmod command and make permissions so that only you can read and write to that file, i.e.
chmod 600 filename

u: user
g: group
o: other
a: all
r: read
w: write
x: execute
+: add permission
-: take away permission
i.e.
chmod go-rwx biglist

diff -- is used to show the differences between files. Lines beginning with a < denotes file1, while lines beginning with a > denotes file2.

diff file1 file2

du -- The du command outputs the number of kilobyes used by each subdirectory.

gzip -- This reduces the size of a file, thus freeing valuable disk space.

gzip file1.txt

gunzip--To uncompress the compressed file.

gunzip file1.txt.gz

zcat --Reads gzipped files without needing to uncompress them first

zcat file1.txt.gz

find --This searches through the directories for files and directories with a given name.

To search for all fies with the extention .txt, starting at the current directory (.) and working through all sub-directories, then printing the name of the file to the screen, type

% find . -name "*2" : Finds the files that end with 2.

ln -- Creates a link to files.

ln a.txt b.txt : Creates a link to a.txt with the name of b.txt

sort -- sorts files.

sort file1.txt : Puts the sorted output of file 1 to the screen.

sort –c file1.txt : Checks whether the file is sorted.

sort –r –o file2.txt file1.txt : Sorted output of file1 (in decreasing sequence from Z to A) is sent to file2. file 1 remains the same.

sort file1.txt > file2.txt : Sorted output of file1 (in increasing sequence from A to Z) is sent to file2. file 1 remains the same.

; -- more than one command can be executed on the same line using semicolon.

ls; pwd; cat file1.txt;

| -- The vertical bar or pipe makes the output of one command the input of other.

cat file1.txt | grep 783456

grep -- To perform a grep search that ignores case, use the -i option The grep command is case sensitive by default. It searches for specific pattern in a file.

grep -i John file1.txt