

This lab is designed to introduce you to Linux, C language, and some simple examples on how the operating system works as a ‘resource manager.’

- a. B. Wajid, H. Iqbal and M. Jamil. "Linux programming for the faint of heart," Sabz Qalam, 2020 (ISBN #: 978-969-7941-00-1).
- b. R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau. "Operating Systems: Three Easy Pieces," Arpaci-Dusseau Books, LLC, 2019.

HW 1 – Lab 1: You are expected to read “Part I – Getting Started,” (Chapters 1 and 2) at home [1]. Don’t worry these are just 18 pages written using a large font size, on a standard small book size.

(ETC: 1.5 hours)

[illegible]

```

last login: Tue Aug 28 15:27:08 on console
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208895.
MU-IMAC-VGL01:~ bu022033$ absoedfg
-bash: absoedfg: command not found
MU-IMAC-VGL01:~ bu022033$ history
1  absoedfg
2  history
MU-IMAC-VGL01:~ bu022033$ history -c
MU-IMAC-VGL01:~ bu022033$ sddg
-bash: sddg: command not found
MU-IMAC-VGL01:~ bu022033$ raxr
-bash: raxr: command not found
MU-IMAC-VGL01:~ bu022033$ qerty
-bash: qerty: command not found
MU-IMAC-VGL01:~ bu022033$ limgp
-bash: limgp: command not found
MU-IMAC-VGL01:~ bu022033$ !l
sddg
-bash: sddg: command not found
MU-IMAC-VGL01:~ bu022033$ !w
qerty
-bash: qerty: command not found
MU-IMAC-VGL01:~ bu022033$ !-3
limgp
-bash: limgp: command not found
MU-IMAC-VGL01:~ bu022033$ !!
limgp
-bash: limgp: command not found
MU-IMAC-VGL01:~ bu022033$ date
Tue Aug 28 16:18:58 PKT 2024
MU-IMAC-VGL01:~ bu022033$ cal
      August 2024
Su Mo Tu We Th Fr Sa
                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
MU-IMAC-VGL01:~ bu022033$ whoami
bu022033
MU-IMAC-VGL01:~ bu022033$ hostname
MU-IMAC-VGL01:~ bu022033$ df -h
Filesystem      Size  Used Avail Capacity iused ifree %used  Mounted on
/dev/disk3s31  47872492 19764286 281043248   7%  392791 1522166240   0%  /dev
/dev/disk3s6    256          0  281043248   0%      0      281043248   0%  /dev
/dev/disk3s6    47872492 12731848 281043248   4%  1272 1522166240   0%  /System/Volumes/Preboot
/dev/disk3s2    47872492 128036 281043248   0%  44 1522166240   0%  /System/Volumes/Update
/dev/disk3s2    1824888 12728 788272  2%  1 3761368   0%  /System/Volumes/efi
/dev/disk3s1    1824888 131768 788272 15%  63 3761368   0%  /System/Volumes/ISOPreboot
/dev/disk3s3    1824888 148 788272  1%  38 3761368   0%  /System/Volumes/Recovery
/dev/disk3s1    47872492 168682288 281043248 32% 1177863 1522166240   0%  /System/Volumes/Data
map actfs_home  0          0  180N    0  0  -  /System/Volumes/Data/Network/Server
map -fat32
-bash: free: command not found
MU-IMAC-VGL01:~ bu022033$ df -h
Filesystem      Size  Used Avail Capacity iused ifree %used  Mounted on
/dev/disk3s31  22601  9.46G  1460G   7%  1944  1.5G   0%  /
/dev/disk3s6    19801  19801  801 100N   684  0 100N  0%  /dev
/dev/disk3s6    22601  9.46G  1460G   7%  1.3k  1.5G   0%  /System/Volumes/Preboot
/dev/disk3s2    22601  9.46G  1460G   7%  44 1.5G   0%  /System/Volumes/Update
/dev/disk3s2    19801  19801  801 100N   2  3.8M   0%  /System/Volumes/efi
/dev/disk3s1    19801  19801  801 100N   15  3.8M   0%  /System/Volumes/ISOPreboot
/dev/disk3s3    19801  19801  801 100N   38  3.8M   0%  /System/Volumes/Recovery
/dev/disk3s1    22601  9.70G  1460G  32% 1.28 1.5G   0%  /System/Volumes/Data
map actfs_home  801  801  801 100N   0  0  -  /System/Volumes/Data/Network/Server
map -fat32
MU-IMAC-VGL01:~ bu022033$

```

3. C Coding (ETC: 0.5 hours)

Chap. 2 of the book [2] introduces a simple C code, reproduced herein below, that continues to print a string passed by the user, until the program is forcefully terminated (by pressing CTRL+C).

```

-----
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <assert.h>

int main (int argc, char *argv[])
{
    if (argc != 2) {

        fprintf( stderr, "Usage: cpu <string> \n" );
        fprintf( stderr, "Example: cpu \"University\" \n" );

        exit(1);
    }

    char *str = argv[1];
    int sleep_seconds = 2;

    while (1) {
        sleep( sleep_seconds );
        printf("%s\n", str);
    }

    return 0;
}
-----

```

Lab 01: Introduction to Bash

Fall 2024

Write the above code using a text editor. Save the code as "cpu.c". Once saved, open the terminal, and navigate yourself to the folder where the file is present. Once there, type the following:

```
> gcc -o cpu cpu.c -Wall
```

This will generate an executable file called "cpu." Now run the file by typing the following on the terminal:

```
> ./cpu
```

This will give an error. Now, run the code again, this time giving a string as an input. This string could be anything you want to repeat. For example:

```
> ./cpu "Habib University"
```

The code will continue printing the string every 2 seconds. To "kill" the program, press CTRL+C. Now run multiple instances of the same code by typing the following on the terminal. Note an instance like "<Your name>" means that you are expected to type your name here.

```
> ./cpu "Habib University" & ./cpu "<Your name>" & ./cpu "<Your thoughts>"
```

Note, even though we have only one processor, all three instances are running concurrently.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <assert.h>
int main (int argc, char *argv[])
{
    if (argc != 2) {
        fprintf( stderr, "Usage: cpu <string>
\n" );
        fprintf( stderr, "Example: cpu
\n\"University\" \" \"\n" );
        exit(1);
    }
    char *str = argv[1];

    while (1) {
        for (int i=0;i<=1000;i++){

            printf("%s\n", str);

        }
        return 0;
    }
}
```

Q1: The main function here, had two arguments (`int argc`, `char *argv[]`), explain their use?

Answer: The first parameter tells us about the number of command line arguments passed and first argument is always the program name while the second parameter is an array of strings pointers which will hold actual arguments passed content

Q2: How do we increase the delay in printing strings?

Answer: To increase the delay we will need to change the value of variable sleep seconds, it is initialized with 2 value we can increase it to 6 or so to increase delay in printing values. It will then wait for that many seconds to print next.

Q3: What is a shell, and which shell are you in?

A shell is a command interpreter that receives command, that translates them into binary instructions and then sends them to OS's kernel. The shell we are using is Bash shell.

Q4: What is a “Home Directory,” and what is your Home Directory?
A home directory is personal directory assigned to user on UNIX and it lets user to store files, scripts etc. By entering Shift and ~ we will get name of our directory on terminal, my directory is /Users/bq08283:

Q5: What's a Working Directory and which directory are you in?
Working Directory is directory we are currently inside and working in. Current working directory can be checked by using pwd (print working directory) command. My directory is /Users/bq08283

Q6: Differentiate between an 'Absolute Path' and a 'Relative Path'?

An absolute path specifies location to file or directory in relation to root directory and it can be easily identified by command `'/'` while Relative Path specifies location to a file or directory, in relation to directory in which user is currently working on.

Q7: What's the largest file inside the directory `"/usr/bin"`?
By using command `"ls -l"` we get list of files in long format then to check the largest file I did it manually and what I get is

0	drwxr-xr-x	32	itadmin	staff	1088	May 31, 2023	itadmin
---	------------	----	---------	-------	------	--------------	---------

Q8: What's the most recently created file inside the directory /usr/bin?
The most recent file created is

```
drwxr-xr-x+ 11 hm09237      HUF\Domain Users      374 Jan 25 2024 hm09237
```

Q9: List all the hidden files and directories in your home directory.
Using “ls -a” we can get all hidden directories and files.

[illegible]

Q10: What does the command 'file' do?
The command file is to determine file type. It examines the content of the file and then provide information about file's format and type.

- Q11:** Search for the “-h” option of “ls.” What do they do? Use them.
The -h option displays file in human readable form.

```
ff88889  na08754  sa04367
[HU-iMAC-LIB08:users bq08283$ ls -lh
Shared  fr04020  mb05087  sa06152
aa03422  na04048  nl04092  sa07310
aa06184  gh05177  nj05516  sf08188
aa07729  ha07239  nk02318  sf09294
af00714  ha05468  na04181  sg06468
af08271  ha08895  na07888  sh02929
ah02087  ha10464  nm02396  sh04400
ak02413  hg02084  mm05534  sh04412
all:iauddin  hm05793  mm04888  sh04404
ap06711  h05466  mohammad.faijan  sh07304
aq03452  ha02514  np07476  sh05872
at01778  ha04465  nq04524  sh07806
at02265  ha07088  nq04497  sj01596
at04077  hm07237  nq04618  sj02738
at05980  ha05098  na05382  sj03641
at06174  ha05526  na05767  sm05723
as06526  ha09041  na06373  sm03692
as07982  hu  nt04491  sm01969
as08818  la06721  nt07121  sq07254
asaf.rahmed  it0011  na04161  sr04461
at05439  ja02548  m08861  sr04368
at08816  ka03725  muhammad.danish  sr03700
au08843  ka05088  mq07277  sr05562
az01852  ka05762  ny06280  sr07804
az09314  libadmin (Deleted)  n090914  st07112
bq08283  libadmin  na08716  sw09045
b05446  na04712  na02399  uc07509
ea02993  na05127  n090978  wp06088
ea03876  na05747  of03638  yf06196
ea05018  na04418  of05554  za05339
ff88889  na06939  rh06882  za04611
fn03441  na08754  rh04364  zu05748
fn05828  na08765  sa04367
```

- Q12:** Make the directory “mine/subdir/subsubdir” using one command only.

```
mkculr: cd: File exists
mkdir: subdir: File exists
mkdir: mkdir: File exists
[3]+  Done                  mkdir mine/subdir
[HU-iMAC-LIB08:desktop bq08283$ mkdir mine cd mine mkdir subdir cd mine/subdir mkdir subsubdir ]
mkdir: mine: File exists
mkdir: cd: File exists
mkdir: mkdir: File exists
```

- Q13:** While staying in your home directory, create an empty file dummy.txt in mine/subdir/subsubdir.
We can use touch command specifying the path where we want to create file then specify file type to .txt
“touch mine/subdir/subsubdir/dummy.txt”
- Q14:** While staying in your home directory, copy the files zip, zipgrep, zipinfo from /usr/bin to mine/subdir/subsubdir
To copy items, a cp command is used. It will copy the files zip, zipgrep, zipinfo from /usr/bin to mine/subdir/subsubdir
“cp /usr/bin/zip /usr/bin/zipgrep /usr/bin/zipinfo mine/subdir/subsubdir/”
- Q15:** Move all files from mine/subdir/subsubdir to mine/subdir/
To move all files from mine/subdir/subsubdir to mine/subdir/ we will use mv command.
“mv mine/subdir/subsubdir/* mine/subdir/”
- Q16:** List all the files in /etc whose second letter is c.
To list all the files /etc whose second letter is c we will use find command.
“find /etc-type f-name '?c*'”
- Q17:** Copy all of them to mine/subdir. Then delete all files that contain a digit.
[HU-iMAC-LIB08:subdir bq08283\$ find /etc -type f -name '?c*' -exec cp {} mine/subdir/ \;
[HU-iMAC-LIB08:subdir bq08283\$ find mine/subdir -type f -name '*[0-9]*' -exec rm i l;
- Q18:** Delete the mine/subdir/ directory
[HU-iMAC-LIB08:subdir bq08283\$ rm -r mine/subdir/

References

- [1] B. Wajid, H. Iqbal and M. Jamil. “Linux programming for the faint of heart,” Sabz Qalam, 2020 (ISBN #: 978-9697941-00-1).
- [2] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau. “Operating Systems: Three Easy Pieces,” Arpaci-Dusseau Books, LLC, 2019.