Lab Manual

Submitted to: Sir Tahir Khalil

Submitted by: Jamal Siddique Qadri

Subject: Operating System (LAB)

Registration no: FA21-BCS-130 (5c)

COMSATS University Islamabad Sahiwal Campus

Operating System

Linux Lab Command

Lab#1

For installation of Linux in CMD as administrative

wsl –install

wsl –install-d ubuntu

WSL stand for windows system linux

1) MNT

```
jamal@DESKTOP-HSR25JE: /mnt/d
jamal@DESKTOP-HSR25JE:~$ cd /mnt/d
jamal@DESKTOP-HSR25JE:/mnt/d$
```

Description: This command is used for entering into the windows from linux.

2) CD

```
jamal@DESKTOP-HSR25JE:/mnt/d$ cd Lectures/news
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$
```

Description: This command is used for changing the directry.

3) LS

```
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$ ls
innerfolder  new.txt.txt
```

Description: For list down the files and folder of the specific folder

4) PWD

```
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$ pwd
/mnt/d/Lectures/news
```

Description: for checking the current path

5) MKDIR

```
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$ mkdir lectures1
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$ ls
innerfolder  lectures1  new.txt.txt
```

Description: for making folder in that dir/folder

6) TOUCH

```
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$  touch news1.txt
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$ ls
innerfolder  lectures1  new.txt.txt  news1.txt
```

Description: for making filein that dir/folder

7) CAT

```
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$ cat news1.txt
hello, this is the file.jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$
```

Description: for reading data into the file

Lab#2

1)

For DELETION the folder

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ rm news
rm: cannot remove 'news': Is a directory
jamal@DESKTOP-HSR25JE:/mnt/d/source$ rm -rf news
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  news.txt
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

if folder is empty then => rm folder

if folder having some file then => rm-rf folername

2)

For DELETION the FILE

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ rm news.txt
rm: remove write-protected regular file 'news.txt'? y
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  jamal.txt  new.txt  news.txt.gz
```

file then => rm new.txt

3)

for CUT and past FILE(not delete)

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ mv new.txt ../desti/
jamal@DESKTOP-HSR25JE:/mnt/d/source$ cd ..
jamal@DESKTOP-HSR25JE:/mnt/d$ cd desti
jamal@DESKTOP-HSR25JE:/mnt/d/desti$ ls
new.txt  news.txt
jamal@DESKTOP-HSR25JE:/mnt/d/desti$
```

for moving the file

command	source	destination
mv	d.txt	newfolder

4)

For COPY the FILE

```
jamal@DESKTOP-HSR25JE:/mnt/d/desti$ cp new.txt ../source/
jamal@DESKTOP-HSR25JE:/mnt/d/desti$ cd ..
jamal@DESKTOP-HSR25JE:/mnt/d$ cd source
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  jamal.txt  new.txt  news.txt.gz
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

command	source	destination
cp	d.txt	newfolder

5)

For COPY the FOLDER

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ cp -r news ../desti/
jamal@DESKTOP-HSR25JE:/mnt/d/source$ cd ..
jamal@DESKTOP-HSR25JE:/mnt/d$ cd desti
jamal@DESKTOP-HSR25JE:/mnt/d/desti$ ls
new.txt  news  news.txt
jamal@DESKTOP-HSR25JE:/mnt/d/desti$
```

command	source	destination
cp -r	snews	newfolder

6)

cd => for going back to root in one step

cd.. => for going back to root step by step

7)

For finding the text FILE base on extension

* For all files

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ find -name "*txt"
./h.txt
./jamal.txt
./new.txt
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

find-name "*.txt"

8)

For creating the links

SOFT LINK:

Discription:

if original file delete then the link also delete.(become red)

Command:

command		file_name	link_name
ln	-s	file.txt	f.txt

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ln new.txt n.txt
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  j.txt  n.txt  new.txt  news  news.txt.gz
jamal@DESKTOP-HSR25JE:/mnt/d/source$ rm new.txt
rm: remove write-protected regular file 'new.txt'? y
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  j.txt  n.txt  news  news.txt.gz
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

HARD LINK:

Discription:

if the original file delete then the lnk will not remove/delete.

if we write in the original file then the link file also updated by default.

Command:

command		file_name	link_name
ln		file.txt	f.txt

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ln -s jamal.txt j.txt
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  j.txt  jamal.txt  new.txt  news  news.txt.gz
jamal@DESKTOP-HSR25JE:/mnt/d/source$ rm jamal.txt
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  j.txt  new.txt  news  news.txt.gz
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

Lab#3

1)wc file.txt

* lines *words *size(bytes)

// for checking the file info

wc -l file.txt // for lines

wc -w file.txt // for words checking

wc -c file.txt // for checking the size

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ wc n.txt
1 1 6 n.txt
jamal@DESKTOP-HSR25JE:/mnt/d/source$ wc -w n.txt
1 n.txt
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

2) nano // linux editor

program extension .c

ctrl+x //exit

ctrl+s //save

NOTE: we always write a C program in Nano editor

3) sort file.txt

//for sorting the file

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ sort n.txt
hello
```

4) rwx

r // read

w //write

x //execute

rwx	//owner	//read, write, execute
rw-	//group	//read, write
r--	//all	//read

5) Change the permission of the file

command:

chmod 555 file.txt

first	5	//for owner
second	5	//for group
third	5	//for all/everyone else

Commands list for changing permission

0	no permission
1	execute
2	write
3	write, execute
4	read
5	read, execute
6	read, write
7	read, write, execute

6) For changing the name of the command

alias os="ls-al"

7) For printing in linux (on screen)

echo "hello world"

```
U}jamal@DESKTOP-HSR25JE:/mnt/d/source$ echo "hello how are you"
hello how are you
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

8)For writing into file without opening nano or going to window

echo "text which you want to write">>file.txt

9) For compressng the file

gzip	file.txt		//for compressing the file
gzip	-k	file.txt	//compressed the original file also generate a new copy of the file
gzip	-d	file.txt	//for decompressing the original file

10) ls-al // for cheking the permission of rwx

LAB#4,5,6

System calls:

when we write the program in gcc then we have to call it through system call.

syntax ?

command:

=> man(manual) write

=> we write code in "C" language

=> For write command we use header file

```
 #include<snistd.h>
int main(){

}
```

=> we write on screen or on file

=> ubuntu take everything/ read everything as a file.

```
child.c: command not found
jamal@DESKTOP-HSR25JE:/mnt/d/source$ nano child.c
jamal@DESKTOP-HSR25JE:/mnt/d/source$ gcc child.c
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ./a.out
the parent process id  596
this is child process id  597
this is child process id 597
this is parent process id  596
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

Write Function:

=>write(int fd,buf,size)

Parameters explanations:

=>file discriptes(fd)

[0=>"keyboard",1=>"for screen/showing output",2=>"error"]

(built-in define) //where read and where write ,where to write

=>buf(buffer)

small part of data of RAM in buffer that give data in a short time.

// what is data , what to write

=>size

length of words, characters which we are going to read //size of write

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ nano system.c
jamal@DESKTOP-HSR25JE:/mnt/d/source$ gcc system.c
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ./a.out
jamal@DESKTOP-HSR25JE:/mnt/d/source$ cat new.txt
what is you name
hello
how are you.
what is your name.
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

**********For writing and execution of the program**********

(1) nano news.c

(2) write code

PROGRAMM

```
#include<snistd.h>
int main(){
write(1,"hello world",11);                    //we are going to write on screen(1),msg,  character of the msg
}
```

(3) gcc new.c

(4) ./a.out

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ echo "hello world"
hello world
```

**********For reading and execution of the program**********

(1) nano read.c

(2) read code

PROGRAMM

```
#include<snistd.h>
int main(){
char buf[50];
read(0,buf,20);              //we are going to write on screen(1),msg,  character of the msg
write(1,buf,20);
}
```

(3) gcc new.c

(4) ./a.out

```
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$ cat news1.txt
hello, this is the file.jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$
```

=>Read and Write through file is called open system call.

Open system call parameters

open(int fd, flags,mode)

FD>

File discriptor->write file name

```
jamal@DESKTOP-HSR25JE:/mnt/d/source$ rm news.txt
rm: remove write-protected regular file 'news.txt'? y
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  jamal.txt  new.txt  news.txt.gz
```

FLAGS=>

wnat to read or write

Read--> O_RDONLY

Write--> O_WRONLY

Read,Write-->O_RDWR

O_CREATE | O_WRONLY , 0642 => When we want to create a file and then wanna write it.

MOOD=>

0642

we use mode when we do not have file laready created

We have to add these header file while writing the OPEN SYSTEM CALLS.

```
#include <unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

int main(){
```

```
U}jamal@DESKTOP-HSR25JE:/mnt/d/source$ echo "hello how are you"
hello how are you
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

```
char buf[30];

int fd,n;


n=read(0,buf,20);

fd=open("file.txt",O_WRONLY);

write(fd,buf,n);

}
```

```
jamal@DESKTOP-HSR25JE:/mnt/d$ cd Lectures/news
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$
```

**********READ AND WRITE BOTH IN FILE FROM FILE*********

```
#include <unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>
```

```
jamal@DESKTOP-HSR25JE:/mnt/d/desti$ cp new.txt ../source/
jamal@DESKTOP-HSR25JE:/mnt/d/desti$ cd ..
jamal@DESKTOP-HSR25JE:/mnt/d$ cd source
jamal@DESKTOP-HSR25JE:/mnt/d/source$ ls
h.txt  jamal.txt  new.txt  news.txt.gz
jamal@DESKTOP-HSR25JE:/mnt/d/source$
```

```
int main(){


char buf[30];

int fd,fd1,n;


fd=open("file.txt",O_RDONLY);
```

```
read(fd,buf,20);

fd1=open("new.txt",O_CREAT|O_WRONLY,0642);

write(fd1,buf,n);

}
```

```
jamal@DESKTOP-HSR25JE:/mnt/d$ cd Lectures/news
jamal@DESKTOP-HSR25JE:/mnt/d/Lectures/news$
```

Linux Lab Command *After Mid*

CODE#1

Creating Threads Without Parameters

```
jamal@DESKTOP-HSR25JE: ~
  GNU nano 6.2                                news.c *
#include<unistd.h>
#include<stdio.h>
#include<pthread.h>
void *fun1();
void *fun2();

int main(){
        pthread_t th1,th2;
        pthread_create(&th1,NULL,fun1,NULL);
        pthread_create(&th2,NULL,fun2,NULL);

//use join to for witing the termination of threads
        pthread_join(th1,NULL);
        pthread_join(th2,NULL);
}

void *fun1(){

for(int a=0; a<5; a++){
printf("%d\t",a);
}

}
void *fun2(){
for(int b=0; b<5; b++){

printf("%d\t",b);
}

}
```

```
jamal@DESKTOP-HSR25JE: ~
jamal@DESKTOP-HSR25JE:~$ gcc news.c
jamal@DESKTOP-HSR25JE:~$ ./a.out
0       1       2       3       4       0       1       2       3       4       jam
al@DESKTOP-HSR25JE:~$
```

Explanation:

it is just executing the two threads of one process and they are simply running the for loops till 5.

CODE#2

Creating Threads With Parameters

jamal@DESKTOP-HSR25JE: ~

GNU nano 6.2 news.c

```c
#include <unistd.h>
#include <stdio.h>
#include <pthread.h>

void *fun1(void *arg);
void *fun2(void *arg);

int main() {
    pthread_t th1, th2;
    int a = 0, b = 0;

    pthread_create(&th1, NULL, fun1, (void *)&a);
    pthread_create(&th2, NULL, fun2, (void *)&b);

    pthread_join(th1, NULL);
    pthread_join(th2, NULL);

    return 0;
}

void *fun1(void *arg) {
    int *a = (int *)arg;

    for (*a = 0; *a < 5; (*a)++) {
        printf("Fun1 With parameter %d\n", *a);
    }

}

void *fun2(void *arg) {
    int *b = (int *)arg;

    for (*b = 0; *b < 5; (*b)++) {
        printf("Fun2 With parameter %d\n", *b);
    }

}
```

```
jamal@DESKTOP-HSR25JE: ~
jamal@DESKTOP-HSR25JE:~$ gcc news.c
jamal@DESKTOP-HSR25JE:~$ ./a.out
Fun1 With parameter 0
Fun1 With parameter 1
Fun1 With parameter 2
Fun1 With parameter 3
Fun1 With parameter 4
Fun2 With parameter 0
Fun2 With parameter 1
Fun2 With parameter 2
Fun2 With parameter 3
Fun2 With parameter 4
jamal@DESKTOP-HSR25JE:~$
```

Explanation:

Why use void and how to pass parameters:

In C, when using **pthread_create** to create threads, the fourth argument is of type **void ***. This allows you to pass a pointer to any data type by casting it to **void ***. The reason for using **void *** is to provide a generic way of passing parameters to the thread function.

When you pass parameters to a thread function using **pthread_create,** you need to cast the arguments to **void *** and then cast them back to the appropriate type within the thread function. This is because **pthread_create** accepts a **void *** argument, and you can't directly pass other types of arguments.

For example, if you want to pass an integer to a thread function, you'd need to:

- Cast the integer to **void *** when passing it to **pthread_create**.
- Cast the **void *** back to an integer type within the thread function.

CODE#3

Creating Race Condition

jamal@DESKTOP-HSR25JE: ~

GNU nano 6.2 news.c *

```
#include<unistd.h>
#include<stdio.h>
#include<pthread.h>
void *fun1();
void *fun2();
int v_1=1;
int main(){
        pthread_t th1,th2;
        pthread_create(&th1,NULL,fun1,NULL);
        pthread_create(&th2,NULL,fun2,NULL);

        //use join to for witing the termination of threads
        pthread_join(th1,NULL);
        pthread_join(th2,NULL);
        printf("The last value of  v_1 %d\n",v_1);
}

void *fun1(){

int a;
a=v_1;
a++;
printf("The value of a %d \n",a);
sleep(1);
v_1=a;
printf("The value of thread 1 is  %d \n",v_1); //2

}
void *fun2(){

int b;
b=v_1;
b--;
printf("The value of b %d \n",b);
sleep(1);
v_1=b;
printf("The value of thread 2 is  %d \n",v_1); //0

}
```

```
jamal@DESKTOP-HSR25JE: ~
jamal@DESKTOP-HSR25JE:~$ nano news.c
jamal@DESKTOP-HSR25JE:~$ gcc news.c
jamal@DESKTOP-HSR25JE:~$ ./a.out
The value of a 2
The value of b 0
The value of thread 1 is  2
The value of thread 2 is  0
The last value of  v_1 0
jamal@DESKTOP-HSR25JE:~$
```

Explanation:

- In the above example we are using creating the race condition using threads.
- Two threads are using the shared resources of process.

Main function:

We are creating the two threads using the “pthread.h” library and creating the two threads using the type ”pthread_t” th1 and th2 then using the pthread_create() function we are making a thread which is creating the thread th1 with NULL parameter and then calling the func1 with the NULL parameter which do the work in first thread. Same for thread th2 but this call the fun2 function with NULL parameter.

Why use pthread_join function:

It will not allow the main process to terminate before terminating the threads to that process. Like:

pthread_join(th1,NULL);

pthread_join(th2,NULL);

Working of threads:

- The shared resource is v_1 which is global variable with the int data type and having a value 1.
- The thread func1 increment the value by 1 it will show the message of 2
- When thread func2 is calling it decrementing the value by 1 and should give the result 1 but it is giving the result 0;

Problem:

The problem is that when two thread use the same resources the showing result is false.

CODE#4

Solution Of Race Condition Using Semaphore

jamal@DESKTOP-HSR25JE: ~

GNU nano 6.2 news.c *

```
#include<unistd.h>
#include<stdio.h>
#include<pthread.h>
#include<semaphore.h>
void *fun1();
void *fun2();
int v_1=1;
sem_t s;
int main(){
        sem_init(&s,0,1);                       // 0 for thread, 1 for process
        pthread_t th1,th2;
        pthread_create(&th1,NULL,fun1,NULL);
        pthread_create(&th2,NULL,fun2,NULL);
        pthread_join(th1,NULL);
        pthread_join(th2,NULL);
        printf("The last value of  v_1 %d\n",v_1);
}
void *fun1(){
int a;
sem_wait(&s);
a=v_1;
a++;
printf("The value of a %d \n",a);
sleep(1);
v_1=a;
printf("The value of thread 1 is  %d \n",v_1); //2
sem_post(&s);
}
void *fun2(){
int b;
sem_wait(&s);
b=v_1;
b--;
printf("The value of b %d \n",b);
sleep(1);
v_1=b;
printf("The value of thread 2 is  %d \n",v_1); //0
sem_post(&s);
}
```

```
jamal@DESKTOP-HSR25JE:~$ nano news.c
jamal@DESKTOP-HSR25JE:~$ nano news.c
jamal@DESKTOP-HSR25JE:~$ gcc news.c
jamal@DESKTOP-HSR25JE:~$ ./a.out
The value of a 2
The value of thread 1 is  2
The value of b 1
The value of thread 2 is  1
The last value of  v_1 1
jamal@DESKTOP-HSR25JE:~$
```

Explanation:

- The problem mentioned above race condition is solved by semaphore using the semaphore.h libraray and the sem_init function.

- We enter the sem_wait() function with parameter of “s” it only enter to this section if it is having the value of 1. This section is called critical section. When a critical section is in use the next thread can not enter into the critical section. So the control does not go to the fun2 until the value of “s” become 1 again.

- When a fun1 enter into the critical section the sem_wait() having the variable “s” with the value of 1 will decrement to 0 for all other threads and processes. When it execute the sem_post() function with parameter of “s” it will increment the value by one and make the value of “s” to 1. Now other processes are enable to enter into critical section.

CODE#5

```
jamal@DESKTOP-HSR25JE: ~
  GNU nano 6.2
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
int main(){
int fd[2];
int n;
char buffer[20];
pid_t p;

p = fork();
if(p>0){
printf("Parent will write on process\n");
}else
{
printf("\nChild will read from process\n");
}

write(fd[1], "data", 4);

n = read(fd[0],buffer,20);

write(1,buffer,n);
}
```

```
jamal@DESKTOP-HSR25JE:~$ gcc newss.c
jamal@DESKTOP-HSR25JE:~$ ./a.out
Parent will write on process
data
Child will read from process
datajamal@DESKTOP-HSR25JE:~$ nano newss.c
jamal@DESKTOP-HSR25JE:~$
```

CODE#6

jamal@DESKTOP-HSR25JE: ~

GNU nano 6.2

```c
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
void *fun1();
void *fun2();
pthread_mutex_t first_mutex, second_mutex;
int main(){
        pthread_mutex_init(&first_mutex,NULL);
        pthread_mutex_init(&second_mutex,NULL);
        pthread_t T1, T2;
        pthread_create(&T1, NULL, fun1, NULL);
        pthread_create(&T2, NULL, fun2, NULL);
        pthread_join(T1, NULL);
        pthread_join(T2, NULL);
        printf("Thread joined\n");
}
void *fun1()
{
        printf("T1 Trying to acquire first_mutex\n");
        pthread_mutex_lock(&first_mutex);
        printf("T1 acquired first_mutex\n");
        sleep(1);
        printf("T1 Trying to acquire second_mutex\n");
        pthread_mutex_lock(&second_mutex);
        printf("T1 acquired second_mutex\n");
        pthread_mutex_unlock(&first_mutex);

}
void *fun2()
{
        printf("T2 Trying to acquire second_mutex\n");
        pthread_mutex_lock(&second_mutex);
        printf("T2 acquired second_mutex\n");
        sleep(1);
        printf("T2 Trying to acquire first_mutex\n");
        pthread_mutex_lock(&first_mutex);
        printf("T2 acquired first_mutex\n");
        pthread_mutex_unlock(&second_mutex);
}
```

```
jamal@DESKTOP-HSR25JE:~$ nano jamal.c
jamal@DESKTOP-HSR25JE:~$ gcc jamal.c
jamal@DESKTOP-HSR25JE:~$ ./a.out
T1 Trying to acquire first_mutex
T1 acquired first_mutex
T2 Trying to acquire second_mutex
T2 acquired second_mutex
T1 Trying to acquire second_mutex
T2 Trying to acquire first_mutex
```