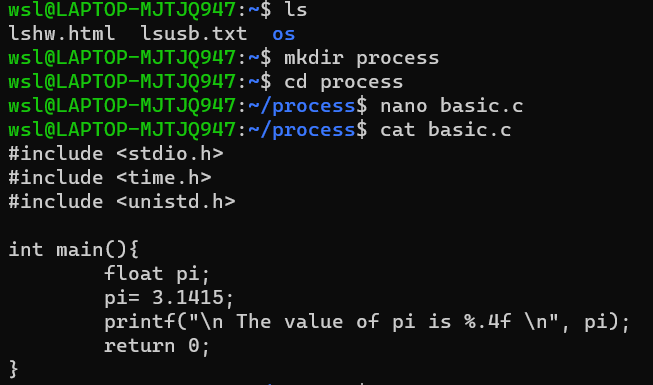
**Lab 3: System Calls for Basic Process Management**

1. ls

mkdir process

cd process

nano basic.c



**A simple C program for printing a constant value:**

#include <stdio.h>

#include <time.h>

#include <unistd.h>

int main(){

float pi;

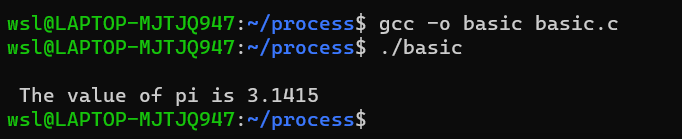
pi=3.1415;

printf("\n The value of pi is %.4f\n",pi);

return 0;

}

Output:



1. nano fork.c

**A program for creating child process from a parent process using function fork():**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

// make two process which run same

// program after this instruction

if(fork()==0){

printf("I am a child.\n");

}

else {

printf("I am the parent.\n");

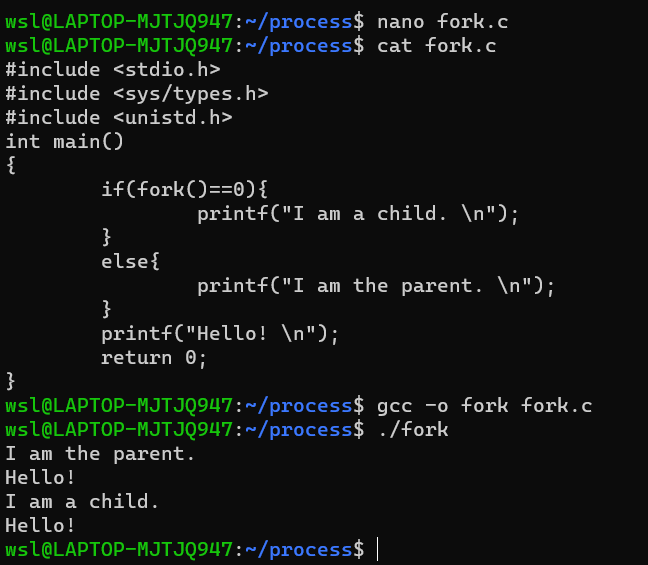
}

printf("Hello!\n");

return 0;

}

Output:



1. nano pid.c

**A simple program for demontration of fork() function which is used to create**

**a child process from parent process:**

#include <stdio.h>

#include <unistd.h> //contains fork prototype

int main()

{

int pid,pid2;

pid = fork();

if (pid == 0)

{

printf("\nChild Process id : %d",getpid());

printf("\nChild Process with parent id : %d \n",getppid());

}

pid2=fork();

if(pid2==0){

printf("\nChild Process id : %d",getpid());

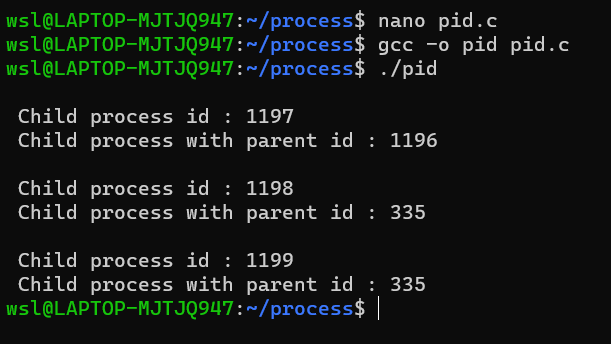
printf("\nChild Process with parent id : %d \n",getppid());

}

return 0;

}

Output:



1. nano execute.c

**A program for demontrating execve() function which causes the program that is**

**currently being run by the calling process to be replaced with a new program, with**

**newly initialized stack, heap, and (initialized and uninitialized) data segments:**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main() {

char \*temp[] = {NULL,"hello","world",NULL};

temp[0]="basic";

if(fork()==0){

printf("I am a child process.\n");

printf("My id is: %d\n",getpid());

printf("My parent is: %d\n",getppid());

printf("I am going to execute a c program\n");

execve("basic",temp,NULL);

}

else{

printf("The id of current process is %d\n",getpid());

}

return 0;

}

Output:

