Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

6

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTI VE |
| 1 | Write what you have learned in few lines on each of the three programs that were using the fork() system call. |
| 2 | Write a C program that uses fork() system call to print a single line eight times without using for loop and repeated printf command. |
| 3 | Code the C program given below and explain what it does along with providing a snapshot of the output. Investigate and write about the usage of execlp() system call |
| 4 | Write a program to declare a counter variable initialized by zero. After fork() system call two processes will run in parallel both incrementing their own version of counter and print numbers 1 -5 . After printing numbers child process will sleep for three second, then print process id of its grandparent and terminates by invoking a gedit editor. Meanwhile, its parent waits for its termination. |

Submitted On:

17/4/2022

( Date: DD/MM/YY )

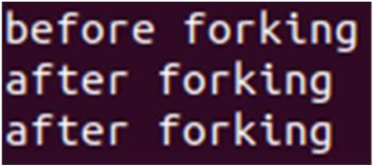
Task# 01:- Write what you have learned in few lines on each of the three programs that were using the fork() system call.

Program 1:-

int main() { printf(“before forking \n”); fork();

printf(“after forking \n”); return 0;

}



In this program,We are printing a line before doing fork system call.After the system call is made,The After forking line is printed two times once for the parent process and other for the child process.

Program 2:-

int i = 5;

void parent\_process(); void child\_process();

int main() { pid\_t pid; pid = fork(); if(pid == 0) { i += 10;

child\_process();

} else {

parent\_process();

}

return 0;

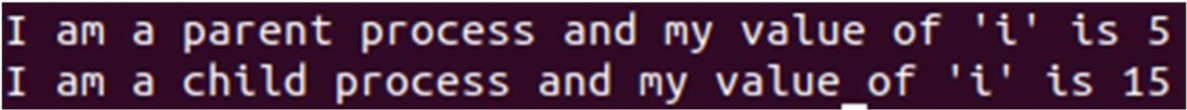
}

void parent\_process() { printf("I am a parent process and my value of 'i' is %d \n",i);

}

void child\_process() { printf("I am a child process and my value of 'i' is %d \n",i);

}



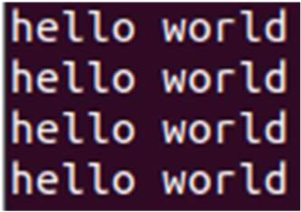
In this program,We have made two functions parent\_process() and child\_process().Both of them are printing that i am a parent/child process.Then,A fork() is called which makes two processes:- parent and child.We are checking that if a child process is made,We are calling the child\_process() method while the parent\_process() is being called if the child process is not made.

Program 3:-

int main ()

{ fork(); fork(); printf("hello world \n"); return 0;

}



In this program,Two fork system calls are called which are printing hello world 4 times because n fork() calls = 2^n process calls.

Program 4:-

int main()

{ pid\_t pid; pid = fork(); if(pid == 0)

{

printf("I am child and my parent is %d and my own PID is %d\n", getppid(), getpid());

}

else if(pid > 0)

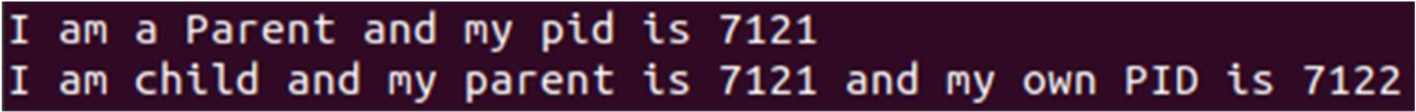
{

printf("I am a Parent and my pid is %d\n", getpid());

}

return 0;

}



In this program,Two processes are made:-one is parent process and other is child process.We are checking that if the child process is not created successfully,We are printing the parent line with parent process id otherwise we are printing the child line with child process id and parent process id.

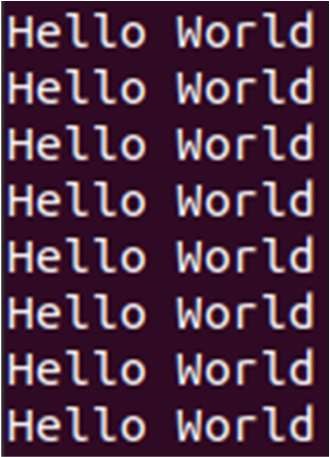
Task # 02: Write a C program that uses fork() system call to print a single line eight times without using for loop and repeated printf command.

Solution:- int main()

{ fork(); fork(); fork(); printf("Hello World\n");

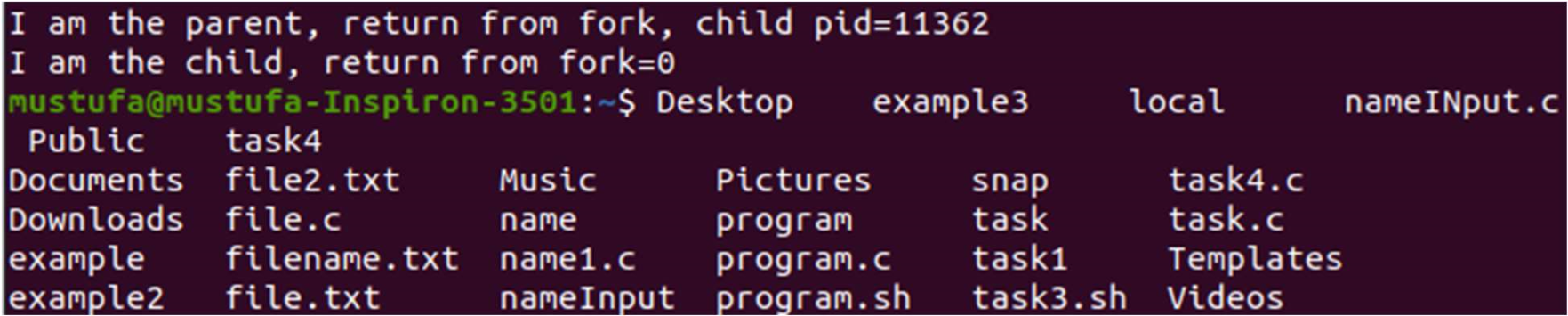
}

Output:-



Task#03:- Code the C program given below and explain what it does along with providing a snapshot of the output. Investigate and write about the usage of execlp() system call.

Output:-



Explanation:-

In this program,We are creating two processes:- parent and child.We are checking if it is a parent process,Then,We are printing i am parent statement.Else,We are printing i am child statement and displaying all the folders and files present in the directory.

execlp system call:- execlp system call creates a new process and executes the path of the file given in the first parameter.

Task# 04:- Write a program to declare a counter variable initialized by zero. After fork() system call two processes will run in parallel both incrementing their own version of counter and print numbers 1 -5 . After printing numbers child process will sleep for three second, then print process id of its grandparent and terminates by invoking a gedit editor. Meanwhile, its parent waits for its termination.

Solution:-

int counter\_parent = 0,counter\_child = 0; int main()

{ pid\_t pid = fork(); if (pid > 0) {

printf("Parent Process starting\n"); for(int i = 1;i<6;i++) { counter\_parent += 5;

printf("%d\n",i);

}

}

else if(pid == 0) { printf("Child Process starting\n"); for(int i = 1;i<6;i++) { counter\_child += 10;

printf("%d\n",i);

} sleep(3); printf("The process id of the parent process is %d",getppid()); execlp("/bin/gedit","gedit",NULL);

}

}

Output:-

