Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**6**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 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:

Date: 18/04/2022

**Task 1: Write what you have learned in few lines on each of the three programs that were using the *fork()* system call.**

**Program 1:**

In this program we are using the fork() system call after printing “before forking” which is why it does not duplicate the command. However, we call the fork() system call right before printing “after forking” which then duplicates the command and prints “after forking” twice, one for parent process and the other for child process.

**Solution :**

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

void main(){

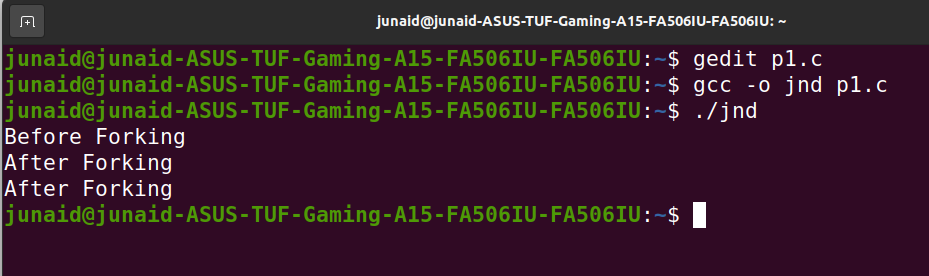
printf("Before Forking\n");

fork();

printf("After Forking\n");

}

**Output :**

****

**Program 2:** In this program, we created two methods namely a parent and child process. Since fork duplicates the program we can execute both the conditions using fork() system call. It runs the parent process first, with i = 5 and then when the fork runs the program the value of i has already been incremented to 15 which is why when the child process runs it executes the else condition and prints the corresponding statement.

**Solution :**

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

int i = 5;

void parent\_process();

void child\_process();

void main(){

pid\_t pid;

pid = fork();

if(pid==0){

i+=10;

child\_process();

}

else{

parent\_process();

}

}

void parent\_process(){

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

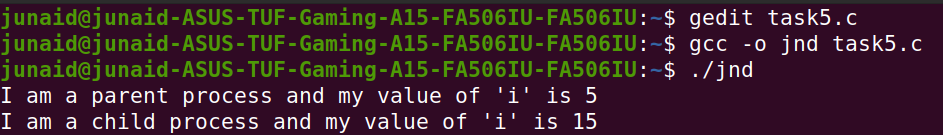
}

void child\_process(){

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

}

**Output :**

****

**Program 3:**

fork() system call’s formula is as follows 2n, so if another fork call is encapsulated in a fork then the execution runs 2n times.

**Solution :**

#include<stdio.h>

#include<unistd.h>

void main(){

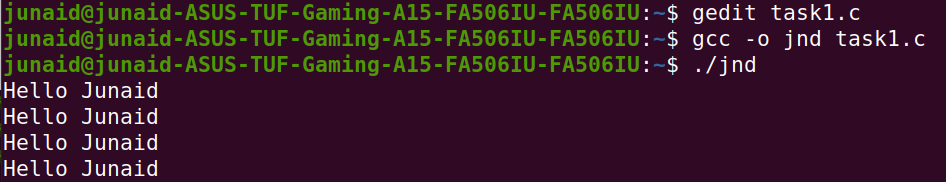
fork();

fork();

printf("Hello Junaid\n");

}

**Output :**

****

**Task 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.**

**SOLUTION :**

#include<stdio.h>

#include<unistd.h>

void main(){

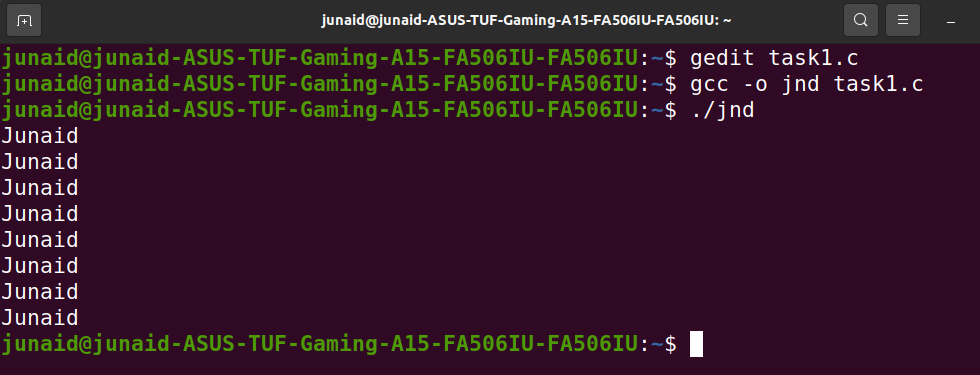
fork();

fork();

fork();

printf("Junaid\n");

}

**OUTPUT :**

**Task 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.**

**SOLUTION :**

#include<stdio.h>

#include<string.h>

#include<sys/types.h>

#include<unistd.h>

#include<stdlib.h>

int main(){

int pid;

pid = fork();

if (pid < 0){

fprintf(stderr,"Fork Failed!\n");

exit(-1);

}

else if (pid == 0){

printf("I am Child which return from Fork = %d\n",pid);

execlp("/bin/ls","ls",NULL);

}

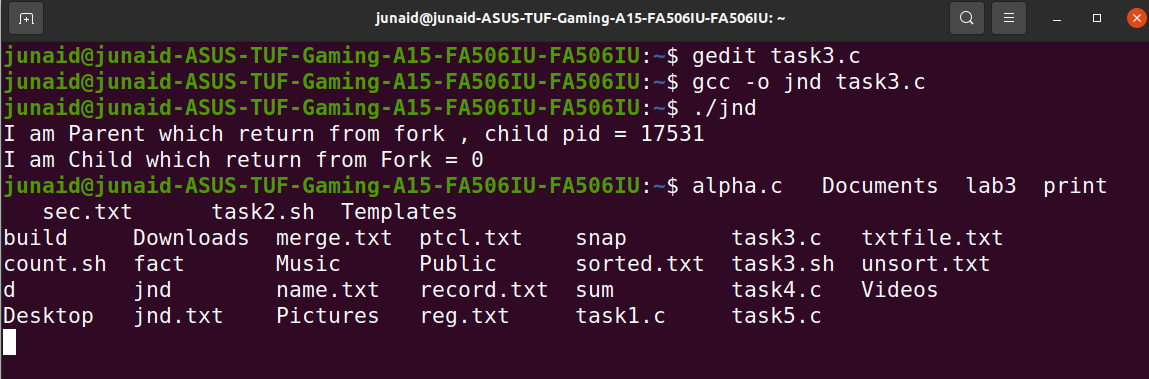
else{

printf("I am Parent which return from fork , child pid = %d\n",pid);

}

}

**OUTPUT :**



**Explanation:**

In this program, we are creating two processes (i.e A parent and child using fork()). We are checking if it is a parent process, if it satisfies the condition, then we print the parent’s statement. Else, we will print child process’ statement and display all the folders and files present in the directory. Whereas execlp system call creates a new process and executes the path of the file given in the first parameter.

**Task 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.**

**Solution :**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

int counter\_parent=0,counter\_child=0;

void main(){

pid\_t pid=fork();

if(pid>0){

printf("Parent process is start...\n");

for(int i=1;i<6;i++){

counter\_parent+=5;

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

}else if(pid==0){

printf("Child process is start...\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 :**

