LAB-3 Operating System

Name : Sagnik Chatterjee

SEC : D

ROLL No : 61

Sem : 5

Q1. C program to block a parent process until the child completes using a wait system call .

Code :

/\*

AUTHOR: SAGNIK CHATTERJEE

DATE : DEC 11,2020

Usage : ./q1

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <unistd.h>

int main(){

int status;

pid\_t pid;

pid=fork();

//printf("The process id of the parent process is %d",getpid());

//printf("The process id of the child process is %d",pid);

if(pid==-1){

printf("[ERROR] Couldn't create child process !!\n");

}

else if(pid==0){

printf("[STATUS] This is the child process,with process id :- %d\n,and the process id of its parent process is :- %d\n",getpid(),getppid());

exit(0);

}

else {

wait(&status);

printf("[STATUS] This is the parent process ,with process id :- %d\n",getpid());

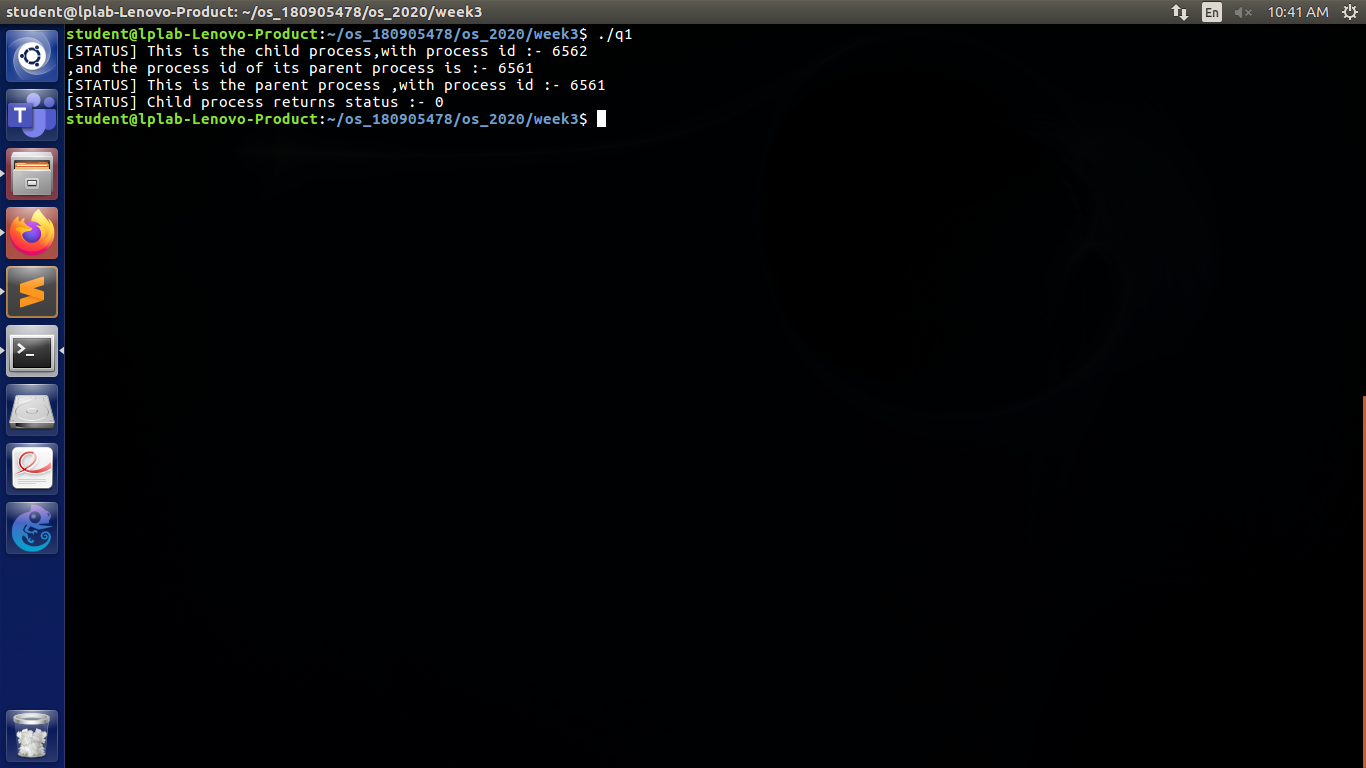
printf("[STATUS] Child process returns status :- %d\n",status);

}

return 0;

}

Screenshot :



Q2. C program to load the binary executable of the previous program in a child process using exec system call.

Code:

/\*

AUTHOR :SAGNIK CHATTERJEE

DATE : DEC 11 ,2020

USAGE : ./q2

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

int main(){

int status;

pid\_t pid;

pid=fork();

if(pid<0){

printf("[ERROR] Child could not be created!\n ");

exit(-1);

}

else if(pid==0){

printf("[STATUS] Executing the command ...\n");

execlp("./q1","q1",NULL);

}

else{

wait(NULL);

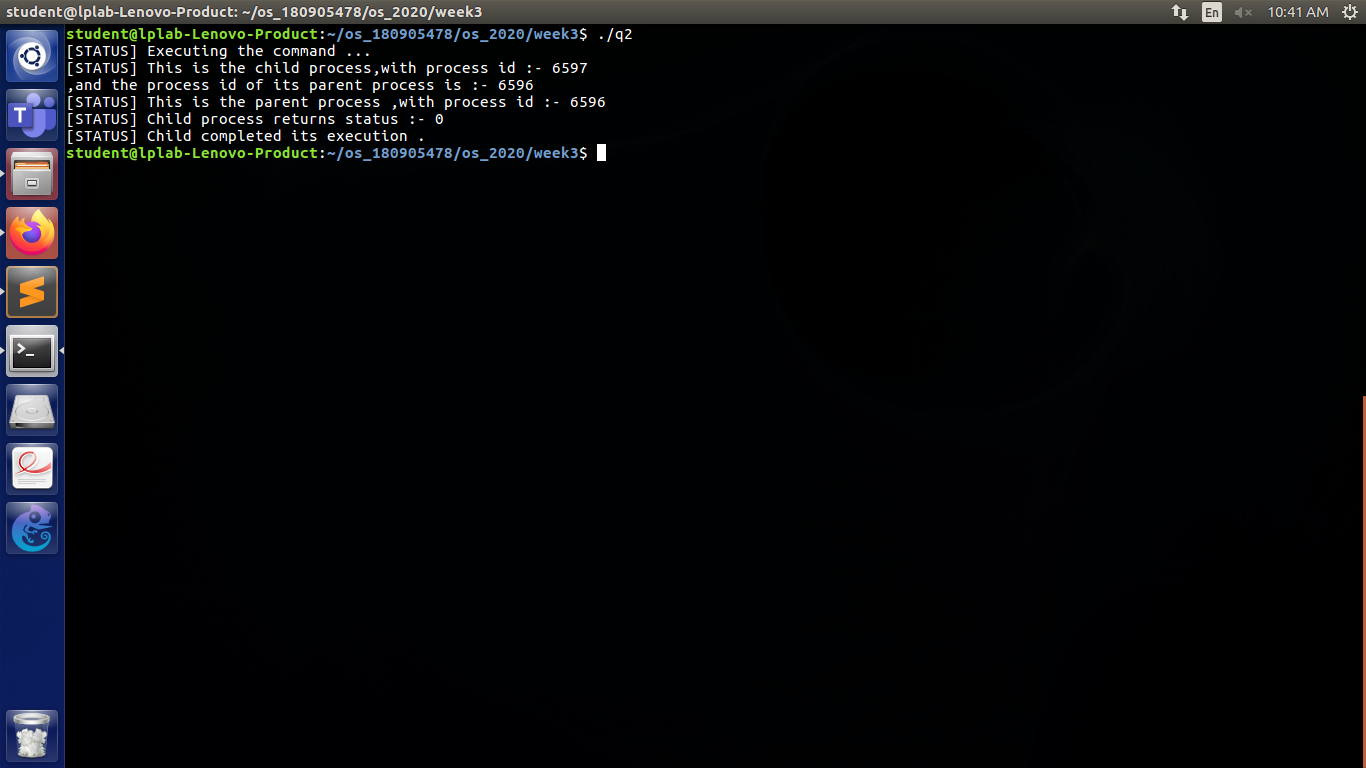
printf("[STATUS] Child completed its execution .\n");

exit(0);

}

}

Screenshot:



Q3. Program to create a child process.Display the process ids of the process , the parent and the child in both the parent and the child processes.

Code:

/\*

AUTHOR : SAGNIK CHATTERJEE

DATE : DEC 11,2020

USAGE : ./q3

\*/

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

int main(){

pid\_t pid;

pid=fork();

if(pid==-1){

printf("[ERROR] Child could not be created!\n");

exit(-1);

}

else if(pid==0){

//process ids of both parent and child when pid==0 i.e child process

printf("[STATUS] Inside the child process ...\n");

printf("Process id of the child process :- %d\n",getpid());

printf("Process id of the parent process :- %d\n",getppid());

}

else {

wait(NULL);//wait for child process to finish

//process ids of both parent and child when pid >0 i.e parent process

printf("[STATUS] Inside the parent process ...\n");

printf("Process id of the child process :- %d\n",getpid());

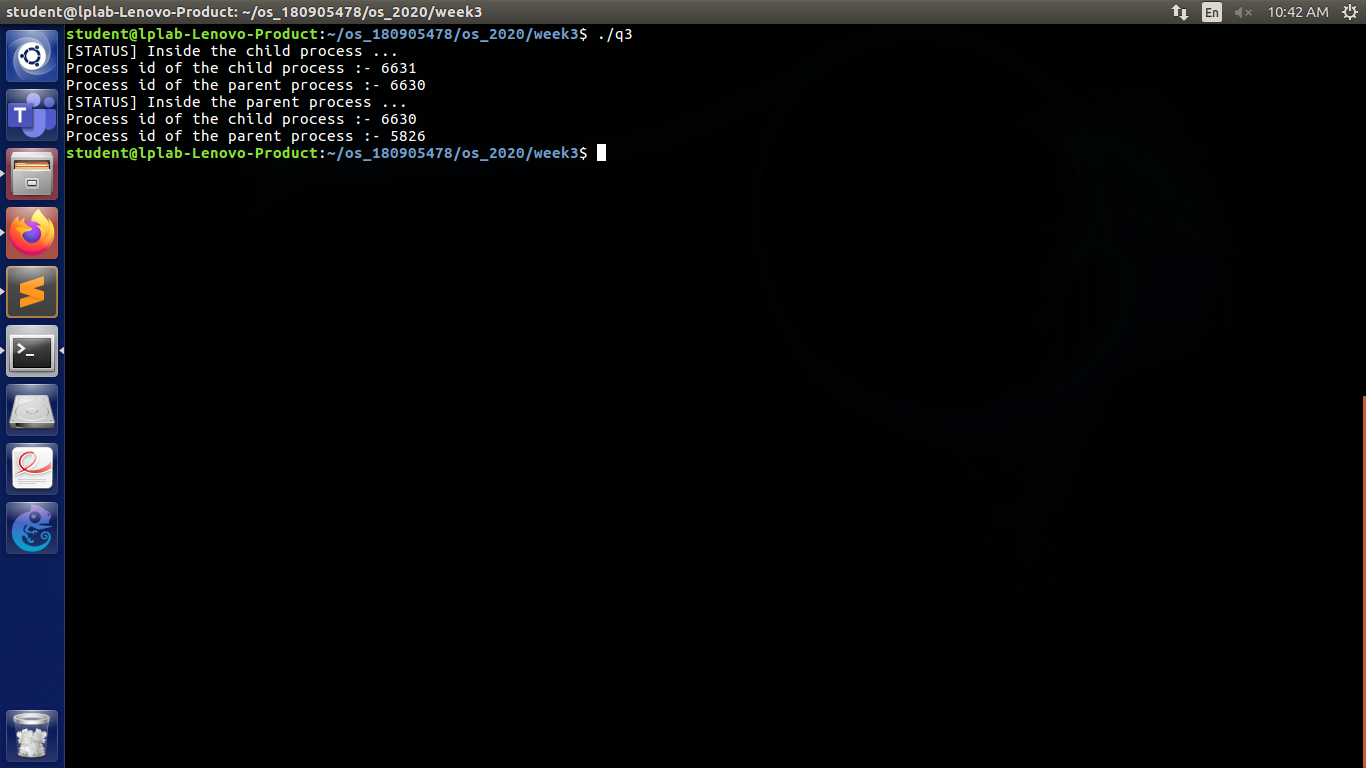
printf("Process id of the parent process :- %d\n",getppid());

}

return 0;

}

Screenshot:



Q4. Create a zombie process and then show the output using ps command .(for showing the defunct process)

Code :

/\*

AUTHOR : SAGNIK CHATTERJEE

DATE : DEC 11,2020

USAGE : ./q4 & in one terminal to start the defunct process in the background

and in anohter terminal check using

ps aux | grep 'Z'

\*/

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

//creating a zombie process

int main(){

pid\_t pid;

int status;

pid=fork();

if(pid<0){

perror("[STATUS] Child process could not be created .\n");

exit(-1);

}

//creating a child process

if(pid==0){

printf("[STATUS] Child process created.\n");

exit(0);

}

//since for this pid >0 so parent process

sleep(100);

//parent sleeps for 70s hence cant call the wait process

//child already exited but its process id is still in the process table

//pid=wait(&status);

if(WIFEXITED(status)){//to know the exit status of the child

fprintf(stderr,"[STATUS] [%d] Process %d exited with status %d.\n",

getpid(),pid,WEXITSTATUS(status));

//wexitstatus is fot obtaining the exit status of the child process.

}

return 0;

}

Screenshot :

