**Signals**

**Lab no# 11**

****

**Fall 2021**

**CSE-302 System Programming Lab**

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr: Ma’am Madeha sheer**

**March** 8, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Task: Implement wait() Function.**

**A)** **By changing the default behavior of SIGCHLD(without using pause or sigsuspend or sigwait).**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <signal.h>

int x=0;

void sighandler()

{

x=1;

}

void mywait()

{

while(x==0); //infinite loop untill x become 1.

}

int main()

{

struct sigaction action; //it specifies how to handle a signal

action.sa\_handler=sighandler; //sa\_handler is used to function address. and function address is function name.

int y=sigaction(SIGCHLD,&action,NULL);

if(y==-1)

{

printf("Sorry! sigaction function can't executed successfully\n");

perror("Reason");

return -1;

}

int x=fork();

if(x==-1)

{

printf("Sorry! child can't created\n");

perror("Reason");

return -1;

}

else if(x==0)

{

printf("Hi! child process:\n");

}

else if(x>0)

{

mywait();

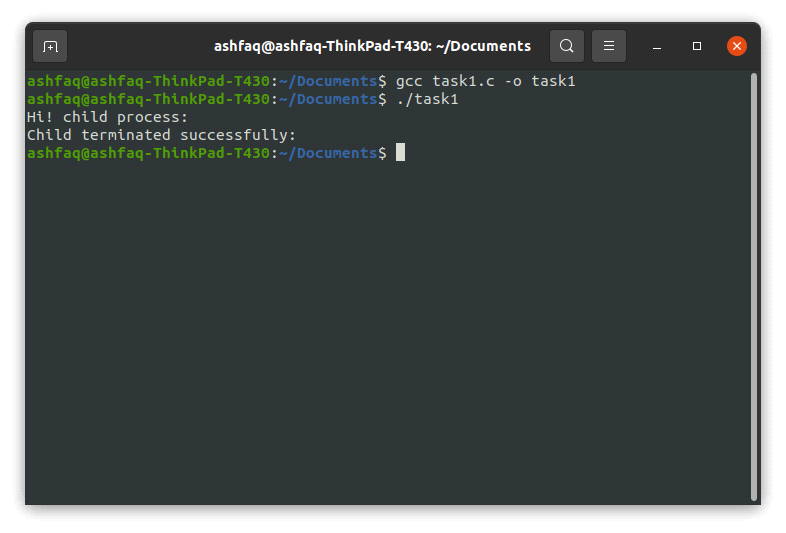
printf("Child terminated successfully:\n");

}

return 0;

}

**Output:**



**B) Using pause() Function.**

**Source Code:**

**Method 1:**

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

void sighandler()

{

printf("Inside handler\n");

return; //this function does nothing but we need a signalhandler for this task to work

}

void mywait()

{

pause();

}

int main()

{

struct sigaction action;

action.sa\_handler=sighandler;

int y=sigaction(SIGCHLD,&action,NULL);

if(y==-1)

{

printf("Sorry! sigaction function can't executed successfully\n");

perror("Reason");

return -1;

}

sigset\_t set;

int f=sigfillset(&set);

if(f==-1)

{

perror("Error Using sigfill");

return -1;

}

int d=sigdelset(&set,SIGCHLD);

if(d==-1)

{

perror("Error Using sigdel");

return -1;

}

int s=sigprocmask(SIG\_BLOCK,&set,NULL);

if(s==-1)

{

perror("Error Occured with sigprocmask");

return -1;

}

int x=fork();

if(x==-1)

{

printf("Sorry! child can't created\n");

perror("Reason");

return -1;

}

else if(x==0)

{

printf("Hi! child process:\n");

}

else if(x>0)

{

mywait();

printf("Child terminated successfully:\n");

}

return 0;

}

**Method 2:**

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

void sighandler()

{

printf("Inside handler\n");

return; //this function does nothing but we need a signalhandler for this task to work

}

void mywait()

{

pause();

}

int main()

{

struct sigaction action;

action.sa\_handler=sighandler;

int y=sigaction(SIGCHLD,&action,NULL);

if(y==-1)

{

printf("Sorry! sigaction function can't executed successfully\n");

perror("Reason");

return -1;

}

sigset\_t set;

int f=sigfillset(&set);

if(f==-1)

{

perror("Error Using sigfill");

return -1;

}

int d=sigdelset(&set,SIGCHLD);

if(d==-1)

{

perror("Error Using sigdel");

return -1;

}

int s=sigprocmask(SIG\_BLOCK,&set,NULL);

if(s==-1)

{

perror("Error Occured with sigprocmask");

return -1;

}

int x=fork();

if(x==-1)

{

printf("Sorry! child can't created\n");

perror("Reason");

return -1;

}

else if(x==0)

{

printf("Hi! child process:\n");

}

else if(x>0)

{

mywait();

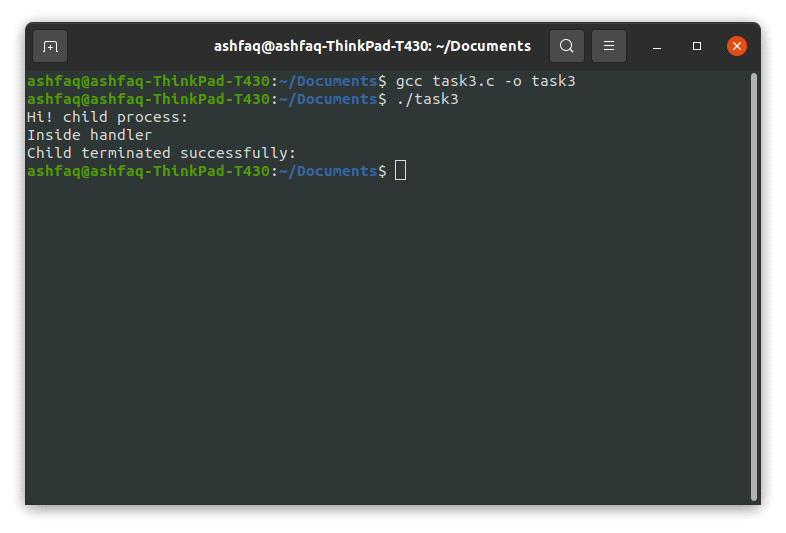
printf("Child terminated successfully:\n");

}

return 0;

}

**Output:**



**C) Using sigsuspend() Function**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

sigset\_t set;

void sighandler()

{

printf("Inside handler\n");

return; //this function does nothing but we need a signalhandler for this task to work

}

void mywait()

{

sigsuspend(&set);

}

int main()

{

struct sigaction action;

action.sa\_handler=sighandler;

int y=sigaction(SIGCHLD,&action,NULL);

if(y==-1)

{

printf("Sorry! sigaction function can't executed successfully\n");

perror("Reason");

return -1;

}

int f=sigfillset(&set);

if(f==-1)

{

perror("Error Using sigfill");

return -1;

}

int d=sigdelset(&set,SIGCHLD);

if(d==-1)

{

perror("Error Using sigdel");

return -1;

}

int x=fork();

if(x==-1)

{

printf("Sorry! child can't created\n");

perror("Reason");

return -1;

}

else if(x==0)

{

printf("Hi! child process:\n");

}

else if(x>0)

{

mywait();

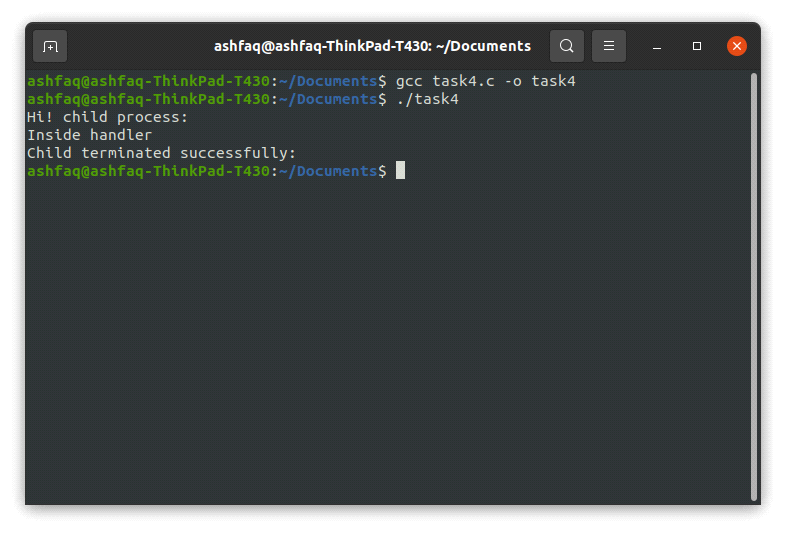
printf("Child terminated successfully:\n");

}

return 0;

}

**Output:**



**D) Using sigwait() Function**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

#include <stdlib.h>

sigset\_t set;

void sighandler()

{

printf("Inside handler\n");

return; //this function does nothing but we need a signalhandler for this task to work

}

void mywait()

{

int signo;

int w=sigwait(&set,&signo);

if(w==-1)

{

perror("Sorry an Error Occured with sigwait");

exit(0);

}

printf("The Signal No that caused sigwait to return : %d\n",signo);

}

int main()

{

struct sigaction action;

action.sa\_handler=sighandler;

int y=sigaction(SIGCHLD,&action,NULL);

if(y==-1)

{

printf("Sorry! sigaction function can't executed successfully\n");

perror("Reason");

return -1;

}

int e=sigemptyset(&set);

if(e==-1)

{

perror("Error Using sigfill");

return -1;

}

int d=sigaddset(&set,SIGCHLD);

if(d==-1)

{

perror("Error Using sigdel");

return -1;

}

int x=fork();

if(x==-1)

{

printf("Sorry! child can't created\n");

perror("Reason");

return -1;

}

else if(x==0)

{

printf("Hi! child process:\n");

}

else if(x>0)

{

mywait();

printf("Child terminated successfully:\n");

}

return 0;

}

**Output:**

