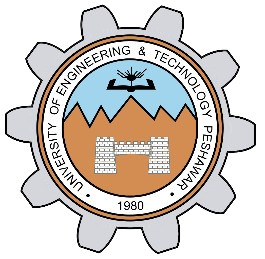
**SIGNAL**

**LAB # 11**



**Fall 2020**

**CSE302L System Programming Lab**

Submitted by: Maaz Habib

Registration No. : **20PWCSE1952**

Class Section: **C**

“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. Abdullah Hamid**

Monday, January 16, 2023

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Task 1:**

**Implement wait () Function.**

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

**Code:**

#include <stdio.h>

#include <unistd.h> #include <signal.h>

int x=0; void SigHandler()

{

x=1;

}

void mywait()

{

while(x==0);

}

int main()

{

int ret = fork(); if(ret==-1)

{

perror("Failed to create a child");

return -1;

} if(ret==0)

{

printf("Child: Hi\n");

}else{

struct sigaction act; act.sa\_handler =SigHandler;

if(sigaction(SIGCHLD,&act,NULL)==-1)

{

perror("Error using sigaction");

return -1;

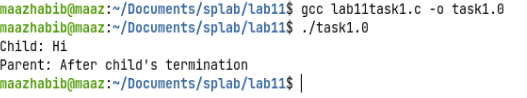
}

mywait();

printf("Parent: After child's termination\n"); }

}

**Output:**



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

**Code:**

#include <stdio.h>

#include <unistd.h> #include <signal.h>

sigset\_t set; void SigHandler()

{

//Does nothing but we need a signal handler for this task to work return;

} void mywait() {

if(sigdelset(&set,SIGCHLD)==-1) perror("Failed to remove SIGCHLD from set");

if(sigprocmask(SIG\_SETMASK,&set,NULL)==-1) perror("Error using sigprocmask");

pause();

}int main()

{

struct sigaction act; act.sa\_handler = SigHandler; if(sigaction(SIGCHLD,&act,NULL)==-1){ perror("Error using sigaction");

return -1;

}

if(sigfillset(&set)==-1)

{

perror("Error using sigfillset");

return -1;

}

if(sigprocmask(SIG\_BLOCK,&set,NULL)==-1){

perror("Error using sigprocmask");

return -1; } int ret = fork(); if(ret==-1)

{

perror("Failed to create a child");

return -1; } if(ret==0) {

printf("Child: Hi\n");

}else{ mywait();

printf("Parent: After child's termination\n"); }

}

**Output:**



**C) Using sigsuspend() Function**

**Code:**

#include <stdio.h>

#include <unistd.h> #include <signal.h>

sigset\_t set; void SigHandler()

{

//Does nothing but we need a signal handler for this task to work return;

}

void mywait() {

sigsuspend(&set); }int main()

{

struct sigaction act; act.sa\_handler = SigHandler; if(sigaction(SIGCHLD,&act,NULL)==-1){ perror("Error using sigaction");

return -1;

}

if(sigfillset(&set)==-1)

{

perror("Error using sigfillset");

return -1;

}

if(sigdelset(&set,SIGCHLD)==-1){

perror("Failed to delete SIGCHLD from set"); return -1; } int ret = fork(); if(ret==-1)

{

perror("Failed to create a child");

return -1; } if(ret==0) {

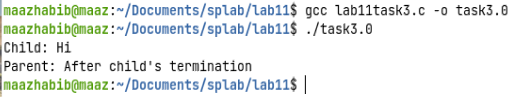
printf("Child: Hi\n");

}else{ mywait();

printf("Parent: After child's termination\n"); }

}

**Output:**



**D) Using sigwait() Function**

**Code:**

#include <stdio.h>

#include <unistd.h> #include <signal.h>

sigset\_t set; void SigHandler()

{

//Does nothing but we need a signal handler for this task to work return;

} void mywait()

{

int signo;

if(sigwait(&set,&signo)==-1)

{

perror("Error using sigwait");

}

printf("Signal number that caused sigwait to return: %d\n",signo);

}

int main()

{

struct sigaction act; act.sa\_handler = SigHandler;

if(sigaction(SIGCHLD,&act,NULL)==-1)

{

perror("Error using sigaction");

return -1;

}

if(sigemptyset(&set)==-1)

{

perror("Error using sigemptyset");

return -1;

}

if(sigaddset(&set,SIGCHLD)==-1){

perror("Failed to add SIGCHLD to set"); return -1; } int ret = fork(); if(ret==-1)

{

perror("Failed to create a child");

return -1; } if(ret==0) {

printf("Child: Hi\n");

}else{ mywait();

printf("Parent: After child's termination\n"); }

}

**Output:**

