**SIGNALS**

**LAB #** **11**

**Fall 2023**

**CSE-302L**

**Systems Programming Lab**

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

“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**

Sunday, January 28, 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**CSE 302L: SYSTEMS PROGRAMMING LAB**

**LAB ASSESSMENT RUBRICS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria & Point Assigned** | **Outstanding 2** | **Acceptable 1.5** | **Considerable 1** | **Below Expectations 0.5** | **Score** |
| **Attendance and**  **Attentiveness in Lab**  PLO08 | Attended in proper Time and attentive in Lab | Attended in proper Time but not attentive in Lab | Attended late but attentive in Lab | Attended late not attentive in Lab |  |
| **Capability of writing Program/Algorithm/Drawing Flow Chart**  PLO1, PLO2, PLO3, PLO5 | Right attempt/ no errors and well formatted | Right attempt/ no errors but not well formatted | Right attempt/ minor errors and not well formatted | Wrong attempt |  |
| **Result or Output/ Completion of target in Lab**  PLO9 | 100% target has been completed and well formatted. | 75% target has been completed and well formatted. | 50% target has been completed but not well formatted. | None of the outputs are correct. |  |
| **Overall, Knowledge**  PLO10, | Demonstrates excellent knowledge of lab | Demonstrates good knowledge of lab | Has partial idea about the Lab and procedure followed | Has poor idea about the Lab and procedure followed |  |
| **Attention to Lab Report**  PLO4, | Submission of Lab Report in Proper Time i.e., in next day of lab, with proper documentation. | Submission of Lab Report in proper time but not with proper documentation. | Late Submission with proper documentation. | Late Submission very poor documentation |  |

**Instructor:**

|  |  |
| --- | --- |
| Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |

**Implement wait ( ) function**

Objectives:

The objectives of this lab are to gain a practical understanding of key system programming concepts, including

* The pause function
* The sigsuspend function
* The sigwait function

Tasks:

**Task 1**: By changing the default behavior of SIGCHLD (without using pause or sigsuspend or sigwait)

**Code in C:**

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int child\_terminated=0;

// Signal handler for SIGCHLD

void sigchld\_handler(int signo) {

(void)signo;

}

int main() {

struct sigaction sa;

sa.sa\_handler = sigchld\_handler;

sigemptyset(&sa.sa\_mask);

sa.sa\_flags = 0;

if (sigaction(SIGCHLD, &sa, NULL) == -1) {

perror("Error setting up signal handler");

exit(EXIT\_FAILURE);

}

// Fork a child process

pid\_t child\_pid = fork();

if (child\_pid == -1) {

perror("Error forking process");

exit(EXIT\_FAILURE);

} else if (child\_pid == 0) {

// Child process

printf("Child process is running\n");

sleep(2); // Simulate some work in the child process

printf("Child process is done\n");

exit(EXIT\_SUCCESS);

} else {

// Parent process

printf("Parent process waiting for child to finish...\n");

while (!child\_terminated) {

sleep(10000);

}

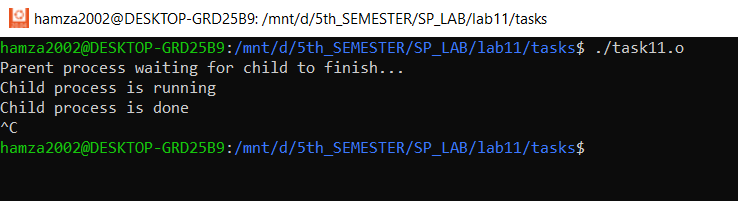
printf("Parent process continuing after child terminated\n");

}

return 0;

}

**Output:**

****

**Task 2**: Using pause () function

**Code in C:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <signal.h>

sigset\_t set;

void myHandler(int);

void myWait(void);

int main(void)

{

struct sigaction newAction;

newAction.sa\_handler = myHandler;

newAction.sa\_flags = 0;

sigfillset(&set);

sigdelset(&set, SIGINT); // Only for debugging

sigprocmask(SIG\_BLOCK, &set, NULL);

sigaction(SIGCHLD, &newAction, NULL);

pid\_t pid = fork();

if(pid < 0){

return 1;

}

else if( pid == 0){

return 13;

}

else{

myWait();

return 0;

}

}

void myHandler(int signo)

{

printf("Child is terminated by receiving signal number: %d.\n", signo);

}

void myWait(void)

{

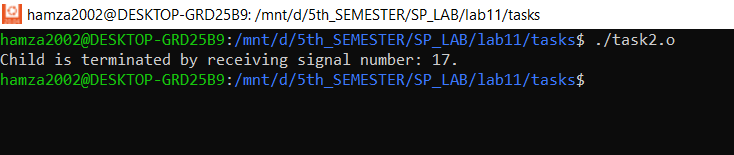
sigdelset(&set, SIGCHLD);

sigprocmask(SIG\_SETMASK, &set, NULL);

pause();

}

**Output:**

****

**Task 3**: Using signal suspend option

**Code in C:**

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

// Signal handler for SIGCHLD

void sigchld\_handler(int signo) {

(void)signo; // Suppress unused parameter warning

// No need to do anything in the signal handler

}

int main() {

// Set up the signal handler for SIGCHLD

struct sigaction sa;

sa.sa\_handler = sigchld\_handler;

sigemptyset(&sa.sa\_mask);

sa.sa\_flags = 0;

if (sigaction(SIGCHLD, &sa, NULL) == -1) {

perror("Error setting up signal handler");

exit(EXIT\_FAILURE);

}

// Fork a child process

pid\_t child\_pid = fork();

if (child\_pid == -1) {

perror("Error forking process");

exit(EXIT\_FAILURE);

} else if (child\_pid == 0) {

// Child process

printf("Child process is running\n");

sleep(2); // Simulate some work in the child process

printf("Child process is done\n");

exit(EXIT\_SUCCESS);

} else {

// Parent process

printf("Parent process waiting for child to finish...\n");

// Set up a mask that blocks SIGCHLD

sigset\_t mask;

sigemptyset(&mask);

sigaddset(&mask, SIGCHLD);

// Suspend execution until a signal is received (SIGCHLD in this case)

while (sigsuspend(&mask) == -1 && errno == EINTR);

// Parent process continues after child termination

printf("Parent process continuing after child terminated\n");

// Wait for child to completely exit (cleanup)

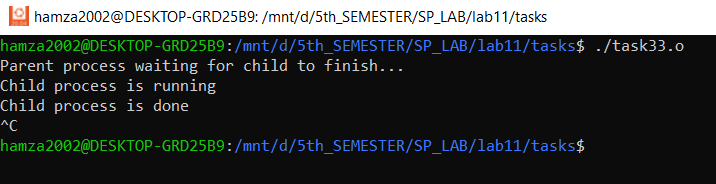
waitpid(child\_pid, NULL, 0);

}

return 0;

}

**Output:**

****

**Task 4**: Using sigwait

**Code in C:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <signal.h>

sigset\_t set;

void myWait(void);

int main(void)

{

sigemptyset(&set);

sigaddset(&set, SIGCHLD);

sigprocmask(SIG\_BLOCK, &set, NULL);

pid\_t pid = fork();

if(pid < 0){

return 1;

}

else if (pid > 0){

myWait();

}

return 0;

}

void myWait(void)

{

int x;

sigwait(&set, &x); // Wiat for blocked signal.

printf("Child is terminated by receiving signal number: %d.\n", x);

}

**Output:**

**A computer screen with blue text

Description automatically generated**

Reference:

To view my codes, please refer to my GitHub account:  [https://github.com/aimalexe/DCSE/tree/main/semester\_5\_(fall-23)/systems\_programming\_lab/lab\_reports](%20https://github.com/aimalexe/DCSE/tree/main/semester_5_(fall-23)/systems_programming_lab/lab_reports) .

Conclusion:

In summary, this laboratory experience has provided a comprehensive exploration of various fundamental system programming concepts including signals. These newfound insights are valuable assets that will enhance my problem-solving abilities and future project contributions.

The End.