SIGNALS

LAB # 11

Fall 2020

CSE302L System Programming Lab

Submitted by: **Shah Raza**

Registration No. : **18PWCSE1658**

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. Madiha Sher

Monday, March 1st, 2021

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).**

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/Results:

```
shahsomething@ubuntu:~/System Programming/labs/Lab 11/Task 1$ ./task1
Child: Hi
Parent: After child's termination
```

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:

```
shahsomething@ubuntu:~/System Programming/labs/Lab 11/Task 2$ ./task2
Child: Hi
Parent: After child's termination
```

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:

```
shahsomething@ubuntu:~/System Programming/labs/Lab 11/Task 3$ ./task3
Child: Hi
Parent: After child's termination
```

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:

```
shahsomething@ubuntu:~/System Programming/labs/Lab 11/Task 4$ ./task4
Child: Hi
Signal number that caused sigwait to return: 17
Parent: After child's termination
shahsomething@ubuntu:~/System Programming/labs/Lab 11/Task 4$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```