ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА «ІНФОРМАЦІЙНИХ СИСТЕМ»

Лабораторна робота № 8

з дисципліни «Операційні системи»

Тема: ***«Програмування керуванням процесами в ОС Unix»***

Виконав:

Студент групи АІ-202

Матненко Станіслав Володимирович

Одеса 2020

Мета роботи: отримання навичок в управлінні процесами в ОС Unix на рівні мови програмування С.

Завдання до лабораторної роботи

Завдання 1 Перегляд інформації про процес

Створіть С-програму, яка виводить на екран таку інформацію:

− ідентифікатор групи процесів лідера сесії;

− ідентифікатор групи процесів, до якої належить процес;

− ідентифікатор процесу, що викликав цю функцію;

− ідентифікатор батьківського процесу;

− ідентифікатор користувача процесу, який викликав цю функцію;

− ідентифікатор групи користувача процесу, який викликав цю функцію.

Завершіть створення програми включенням функції sleep(5) для забезпечення засинання процесу на 5 секунд. При створенні повідомлень використовуйте функцію fprintf з виведенням на потік помилок. Після компіляції запустіть програму. Додатково запустіть програму в конвеєрі, наприклад: ./info | ./info

Порівняйте значення групи процесів.

Завдання 2 Стандартне створення процесу

Створіть С-програму, яка створює процес-нащадок, породжуючи процес та замінюючи образ процесу. У програмі процес-батько повинен видати повідомлення типу «Parent of Ivanov», а процес-нащадок повинен видати повідомлення типу «Child of Ivanov» через виклик команди echo, де замість слова Ivanov в повідомленні повинно бути ваше прізвище в транслітерації.

Завдання 3 Обмін сигналами між процесами

3.1 Створіть С-програму, в якій процес очікує отримання сигналу SIGUSR2 та виводить повідомлення типу «Process of Ivanov got signal» після отримання сигналу, де замість слова Ivanov в повідомленні повинно бути ваше прізвище в транслітерації. Запустіть створену С-програму.

3.2 Створіть С-програму, яка надсилає сигнал SIGUSR2 процесу, запущеному в попередньому пункту завдання. Запустіть створену С-програму та проаналізуйте повідомлення, які виводить перша програма.

Завершіть процес, запущеному в попередньому пункту завдання.

Завдання 4 Створення процесу-сироти

Створіть С-програму, в якій процес-батько несподівано завершується раніше

процесу-нащадку. Процес-батько повинен очікувати завершення n+1 секунд. Процес-

нащадок повинен в циклі (2*n+1) раз із затримкою в 1 секунду виводити повідомлення,

наприклад,

«Parent of Ivanov», за шаблоном як в попередньому завданні, і додатково виводити

PPID процесу-батька.

Значення n – номер команди студента + номер студента в команді.

Перевірте роботу програми, вивчіть вміст таблиці процесів і зробіть відповідні

висновки.

Завдання 5 Створення процесу-зомбі

Створіть С-програму, в якій процес-нащадок несподівано завершується раніше

процесу-батька, перетворюється на зомбі, виводячи в результаті повідомлення, наприклад,

«I am Zombie-process of Ivanov», за шаблоном як в попередньому завданні.

Запустіть програму у фоновому режимі, а в окремому терміналі вивчіть вміст

таблиці процесів і зробіть відповідні висновки.

Завдання 6 Попередження створення процесу-зомбі

Створіть С-програму, в якій процес-нащадок завершується раніше процесу-батька,

але ця подія контролюється процесом-батьком.

Процес-нащадок повинен виводити повідомлення, наприклад, «Child of Ivanov is

finished», за шаблоном як в попередньому завданні.

Процес-батько повинен очікувати (3*n) секунд.

Значення n - n – номер команди студента + номер студента в команді.

Запустіть програму у фоновому режимі, а в окремому терміналі вивчіть вміст

таблиці процесів і зробіть відповідні висновки.

Хід роботи

Завдання 1:

Програма:

```
#include <stdio.h>
#include <unistd.h>

int main(void){
    fprintf(stderr,"I am process! gpid=%d\n",getpgrp());
    fprintf(stderr,"sid=%d\n",getsid(0));
    fprintf(stderr,"pid=%d\n",getpid());
    fprintf(stderr,"ppid=%d\n",getppid());
    fprintf(stderr,"uid=%d\n",getuid());
    fprintf(stderr,"gid=%d\n",getgid());
    sleep(5);
    return 0;
}
```

Результат виконання:

```
[matnenko_stanislav@vpsj3IeQ ~]$ gcc info.c -o info
[matnenko_stanislav@vpsj3IeQ ~]$ ./info | ./info
I am process! gpid=28750
sid=23805
pid=28751
ppid=23805
uid=54344
gid=54350
I am process! gpid=28750
sid=23805
pid=28750
ppid=23805
uid=54344
gid=54350
[matnenko_stanislav@vpsj3IeQ ~]$ ./info ; ./info
I am process! gpid=28787
sid=23805
pid=28787
ppid=23805
uid=54344
gid=54350
I am process! gpid=28788
sid=23805
pid=28788
ppid=23805
uid=54344
gid=54350
[matnenko_stanislav@vpsj3IeQ ~]$
```

У випадку виконання конвеєром процеси виконуються в зворотному порядку.

Завдання 2:

Код програми:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

extern char** environ;

int main(void) {
    char* echo_args[] = {"echo","Child of Matnenko",NULL};
    pid_t pid = fork();
    if (pid != 0) {
       printf("Parent Matnenko: pid=%d, child pid=%d\n",getpid(),pid);
        execve("/bin/echo",echo_args,environ);
        fprintf(stderr,"Error!\n");
    }
    return 0;
}
```

Результат роботи:

```
[matnenko_stanislav@vpsj3IeQ ~]$ gcc create.c -o create
[matnenko_stanislav@vpsj3IeQ ~]$ ./create
Parent Matnenko: pid=6423, child pid=6424
Child of Matnenko
[matnenko_stanislav@vpsj3IeQ ~]$
```

Завдання 3:

Коди програм:

1. Отримання сигналу:

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

static void sig_usr(int signo) {
    if (signo == SIGUSR2)
       fprintf(stderr,"Process of Matnenko got signal %d\n",SIGUSR2);
}

int main(void) {
    printf("pid=%d\n",getpid());
    if (signal(SIGUSR2,sig_usr) == SIG_ERR)
        fprintf(stderr,"Error!\n");
    for ( ; ;)
       pause();
    return 0;
}
```

2. Відправка сигналу:

```
#include <signal.h>
#include <stdio.h>

pid_t pid = 7321;

int main(void) {
    if (!kill(pid,SIGUSR2))
       printf("Process of Matnenko send signal!\n");
    else
       fprintf(stderr,"Error!\n");
    return 0;
}
```

Результат виконання:

```
matnenko_stanislav@vpsj3IeQ:~
[matnenko_stanislav@vpsj3IeQ ~]$ gcc get_signal.c -o get_signal
[matnenko_stanislav@vpsj3IeQ ~]$ ./get_signal
pid=7321
Process of Matnenko got signal 12
```

```
matnenko_stanislav@vpsj3IeQ:~
login as: matnenko_stanislav
matnenko_stanislav@91.219.60.189's password:
Last login: Wed Apr 21 10:46:33 2021 from 94.153.9.99
[matnenko_stanislav@vpsj3IeQ ~]$ nano send_signal.c
[matnenko_stanislav@vpsj3IeQ ~]$ gcc send_signal.c -o send_signal
[matnenko_stanislav@vpsj3IeQ ~]$ ./send_signal
Process of Matnenko send signal!
[matnenko_stanislav@vpsj3IeQ ~]$
```

Завдання 4:

Номер групи – 2, номер учасника – 2, $n_1 = 2 + 2 + 1 = 5$, $n_2 = 2 * (2 + 2) + 1 = 9$

Код програми:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
    int i;
    pid_t pid = fork();
    if (pid != 0) {
         printf("Parent with pid=%d, child pid=%d\n",
                  getpid(),pid);
         sleep(5);
         _exit(0);
    }
    else {
           for (i=0;i<9;i++) {
           printf("Parent of of child Matnenko with pid=%d, ppid=%d\n",
                    getpid(),getppid());
           sleep(1);
           }
   }
   return 0;
}
```

Результат виконання:

```
[matnenko_stanislav@vpsj3IeQ ~]$ gcc parent.c -o parent
[matnenko_stanislav@vpsj3IeQ ~]$ ./parent
Parent with pid=11227, child pid=11228
Parent of of child Matnenko with pid=11228, ppid=11227
Parent of of child Matnenko with pid=11228, ppid=11227
Parent of of child Matnenko with pid=11228, ppid=11227
Parent of of child Matnenko with pid=11228, ppid=11227
Parent of of child Matnenko with pid=11228, ppid=11227
[matnenko_stanislav@vpsj3IeQ ~]$ Parent of of child Matnenko with pid=11228, ppi
d=1
Parent of of child Matnenko with pid=11228, ppid=1
Parent of of child Matnenko with pid=11228, ppid=1
Parent of of child Matnenko with pid=11228, ppid=1
[matnenko_stanislav@vpsj3IeQ ~]$
```