МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
Інститут комп’ютерних систем
Кафедра інформаційних систем

Лабораторна робота №8
З дисципліни: «Операційні системи»
Тема: «Програмування керуванням процесами в ОС Unix»

Виконала:
ст.гр. АІ -204
Сіренко Марія

Перевірили:
Блажко О.А.
Дрозд М.О

Одеса-2021

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

Завдання:

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, що викликав цю функцію

− ідентифікатор групи процесу, що викликав цю функцію.

Завдання 2 Стандартне породження процесу

Створіть С-програму, яка породжує процес-нащадок. У програмі процес-батько повинен видати повідомлення типу «Parent of Ivanov», а процес-нащадок повинен видати повідомлення типу «Child of Ivanov», де замість слова Ivanov в повідомленні повинно бути ваше прізвище в транслітерації.

Завдання 3 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 4 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 5 Контроль 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.
Завдання 6 Обмін сигналами між процесами
6.1 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.
6.2 Створіть С-програму, яка надсилає сигнал SIGUSR1 процесу, запущеному в попередньому пункту завдання.
Запустіть створену С-програму та проаналізуйте повідомлення, які виводить перша програма.
Завершіть процес, запущеному в попередньому пункту завдання.

Виконання завдань:

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, що викликав цю функцію
− ідентифікатор групи процесу, що викликав цю функцію.

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

int main(void)
{
        printf("My sid =%d\n",getsid(getpid()));
        printf("My pgrp=%d\n",getpgrp());
        printf("My pid =%d\n", getpid());
        printf("My ppid =%d\n", getppid());
        printf("My uid =%d\n", getuid());
        printf("My gid =%d\n", getgid());
        return 0;


}
```

```
[sirenko_mariya@vpsj3IeQ lab_8]$ ./info
My sid =25167
My pgrp=25983
My pid =25983
My ppid =25167
My uid =54399
My gid =54405
[sirenko_mariya@vpsj3IeQ lab_8]$
```

Завдання 2 Стандартне породження процесу

Створіть С-програму, яка породжує процес-нащадок. У програмі процес-батько повинен видати повідомлення типу «Parent of Ivanov», а процес-нащадок повинен видати повідомлення типу «Child of Ivanov», де замість слова Ivanov в повідомленні повинно бути ваше прізвище в транслітерації.

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

extern char** environ;

int main(void)
{
        char* echo_args[] = {"echo","Child of Sirenko\n",NULL};
        pid_t pid =fork();

        if(pid==0)
                printf("Child of Sirenko =%d\n",getpid());

        else{
                printf("Parent of Sirenko = %d\n",getpid());
                execve("/bin/echo",echo_args,environ);
        }
        return 0;
}
```

```
[sirenko_mariya@vpsj3IeQ lab_8]$ gcc create.c -o create
[sirenko_mariya@vpsj3IeQ lab_8]$ ./create
Parent of Sirenko = 29608
Child of Sirenko =29609
Child of Sirenko

[sirenko_mariya@vpsj3IeQ lab_8]$
```

Завдання 3 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,

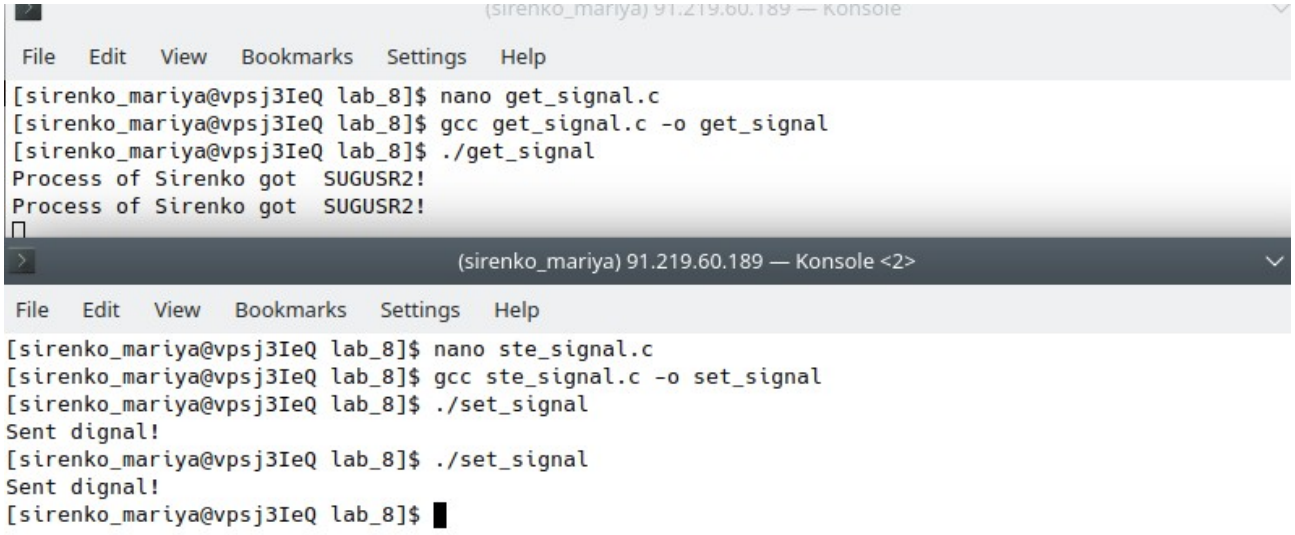
«Parent of Ivanov», за шаблоном як в попередньому завданні, і додатково виводити PPID процесу-батька.

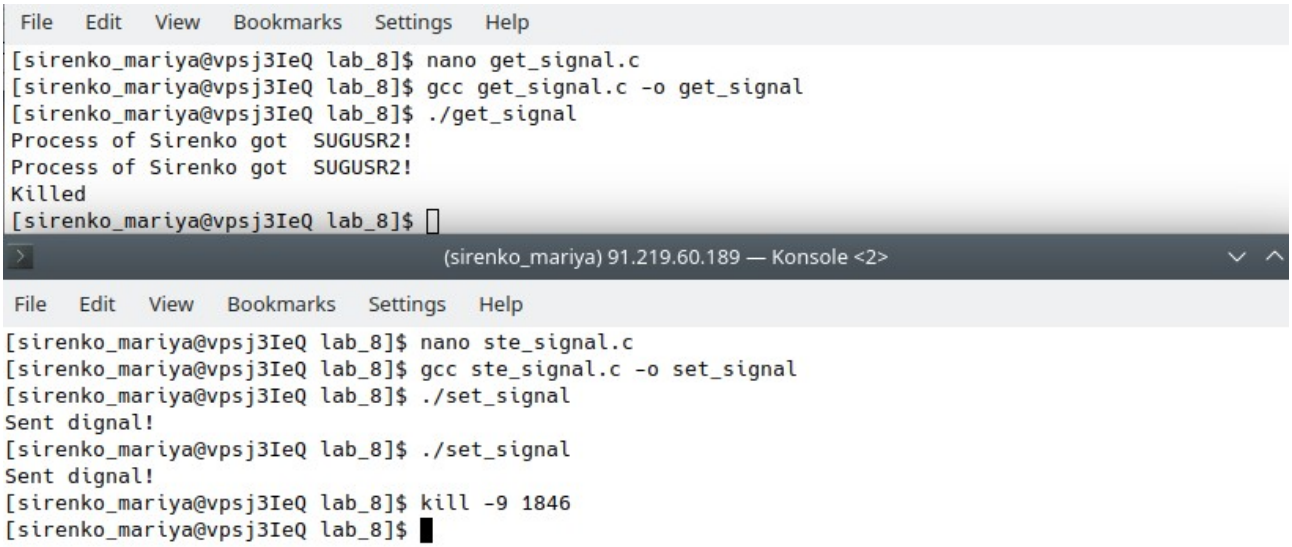
Значення n – номер команди студента + номер студента в команді.

Перевірте роботу програми, вивчіть вміст таблиці процесів і зробіть відповідні висновки.

```c
#include <signal.h>
#include <stdio.h>
pid_t pid =1846;

int main (void){
        if(!kill(pid,SIGUSR2))
        printf("Sent dignal!\n");

        else
        fprintf(stderr,"Error!\n");
return 0;
}
```

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

static void sig_usr(int signo) {
        if(signo ==  SIGUSR2)
        printf("Process of Sirenko got  SUGUSR2!\n");
}

int main (void){
        if(signal(SIGUSR2, sig_usr) == SIG_ERR)
                fprintf(stderr,"Error\n");
        for( ; ; )
                pause();

return 0;
}
```


(sirenko_mariya) 91.219.60.189 — Konsole
File Edit View Bookmarks Settings Help
[sirenko_mariya@vpsj3IeQ lab_8]$ nano get_signal.c
[sirenko_mariya@vpsj3IeQ lab_8]$ gcc get_signal.c -o get_signal
[sirenko_mariya@vpsj3IeQ lab_8]$ ./get_signal
Process of Sirenko got SUGUSR2!
Process of Sirenko got SUGUSR2!
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
[sirenko_mariya@vpsj3IeQ lab_8]$ nano ste_signal.c
[sirenko_mariya@vpsj3IeQ lab_8]$ gcc ste_signal.c -o set_signal
[sirenko_mariya@vpsj3IeQ lab_8]$ ./set_signal
Sent dignal!
[sirenko_mariya@vpsj3IeQ lab_8]$ ./set_signal
Sent dignal!
[sirenko_mariya@vpsj3IeQ lab_8]$


File Edit View Bookmarks Settings Help
[sirenko_mariya@vpsj3IeQ lab_8]$ nano get_signal.c
[sirenko_mariya@vpsj3IeQ lab_8]$ gcc get_signal.c -o get_signal
[sirenko_mariya@vpsj3IeQ lab_8]$ ./get_signal
Process of Sirenko got SUGUSR2!
Process of Sirenko got SUGUSR2!
Killed
[sirenko_mariya@vpsj3IeQ lab_8]$
(sirenko_mariya) 91.219.60.189 — Konsole <2>
File Edit View Bookmarks Settings Help
[sirenko_mariya@vpsj3IeQ lab_8]$ nano ste_signal.c
[sirenko_mariya@vpsj3IeQ lab_8]$ gcc ste_signal.c -o set_signal
[sirenko_mariya@vpsj3IeQ lab_8]$ ./set_signal
Sent dignal!
[sirenko_mariya@vpsj3IeQ lab_8]$ ./set_signal
Sent dignal!
[sirenko_mariya@vpsj3IeQ lab_8]$ kill -9 1846
[sirenko_mariya@vpsj3IeQ lab_8]$

Завдання 4 Створення процесу-зомбі

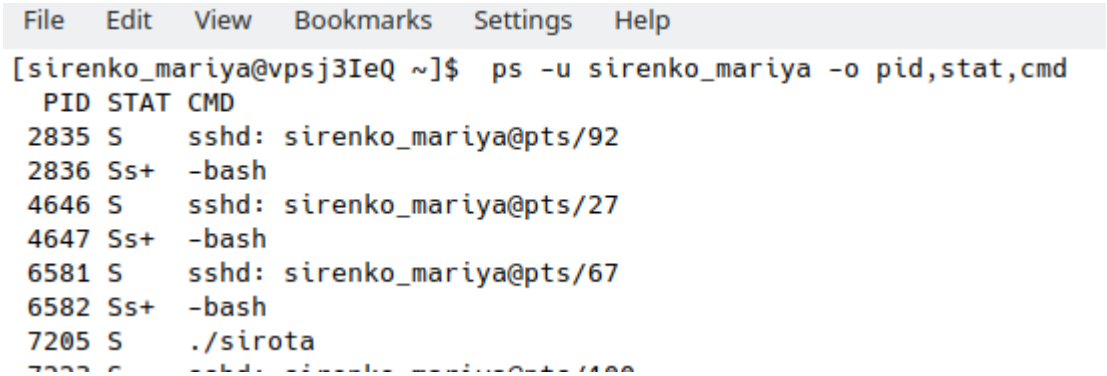
Створіть С-програму, в якій процес-нащадок несподівано завершується раніше процесу-батька, перетворюється на зомбі, виводячи в результаті повідомлення, наприклад, «I am Zombie-process of Ivanov», за шаблоном як в попередньому завданні.

Запустіть програму у фоновому режимі, а в окремому терміналі вивчіть вміст таблиці процесів і зробіть відповідні висновки.

Номер команды 8.2 → n=10

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

int main(void) {
        int i;
        pid_t pid =fork();
        if(pid == 0){
                printf ("Child pid =%d\n",getpid());
                for(i=0;i<1025;i++){
                        printf("I'm Zombie of Sirenko id = %d\n",getppid());
                        sleep(1);
                }
        }
 else {

        printf("Parent process Sirenko id = %d \n",getpid());
        sleep(11);
        _exit(0);
        }
return 0;
```

```
[sirenko_mariya@vpsj3IeQ lab_8]$ nano sirota.c
[sirenko_mariya@vpsj3IeQ lab_8]$ gcc sirota.c -o sirota
[sirenko_mariya@vpsj3IeQ lab_8]$ ./sirota&
[1] 7204
[sirenko_mariya@vpsj3IeQ lab_8]$ Parent process Sirenko id = 7204
Child pid =7205
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 7204
I'm Zombie of Sirenko id = 1
I'm Zombie of Sirenko id = 1
I'm Zombie of Sirenko id = 1
I'm Zombie of Sirenko id = 1
```



Висновок: було отримано навички в управлінні процесами в ОС Unix на рівні мови програмування С.