

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»

Виконав:
ст. гр. AI-204
Нестеренко М. О.

Перевірив:
Блажко О. А.

Одеса – 2021

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

2. Завдання

Завдання 1 Перегляд інформації про процес

Створіть C-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.

```
[nesterenko_mikola@vpsj3IeQ lab8]$ ./info
Hello,World!
My pid = 29101
My ppid = 24585
My uid = 54395
My gid = 54395
Mi pgrp = 29101
My sid = 24585
[nesterenko_mikola@vpsj3IeQ lab8]$ ps -u nesterenko_mikola
```

Завдання 2 Стандартне створення процесу

Створіть C-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.

```
[nesterenko_mikola@vpsj3IeQ lab8]$ ./create
Parent pid=10014
child pid=10015
Child of Nesterenko!

[nesterenko_mikola@vpsj3IeQ lab8]$
```


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.

 nesterenko_mikola@vpsj3IeQ:~/lab8

```
GNU nano 2.3.1 File: get_signal.c

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

static void sig_usr(int signo){
    if (signo == SIGUSR1)
        printf("NESTERENKO GOT THE SIGNAL!\n");
}

int main(void){
    if(signal(SIGUSR1,sig_usr) == SIG_ERR)
        fprintf(stderr,"Error!\n");
    for(;;)
        pause();
}
```

```
[nesterenko_mikola@vpsj3IeQ lab8]$ nano get_signal.c
[nesterenko_mikola@vpsj3IeQ lab8]$ gcc get_signal.c -o getsig
[nesterenko_mikola@vpsj3IeQ lab8]$ ./getsig
```

3.2 Створіть С-програму, яка надсилає сигнал SIGUSR2 процесу, запущеному в

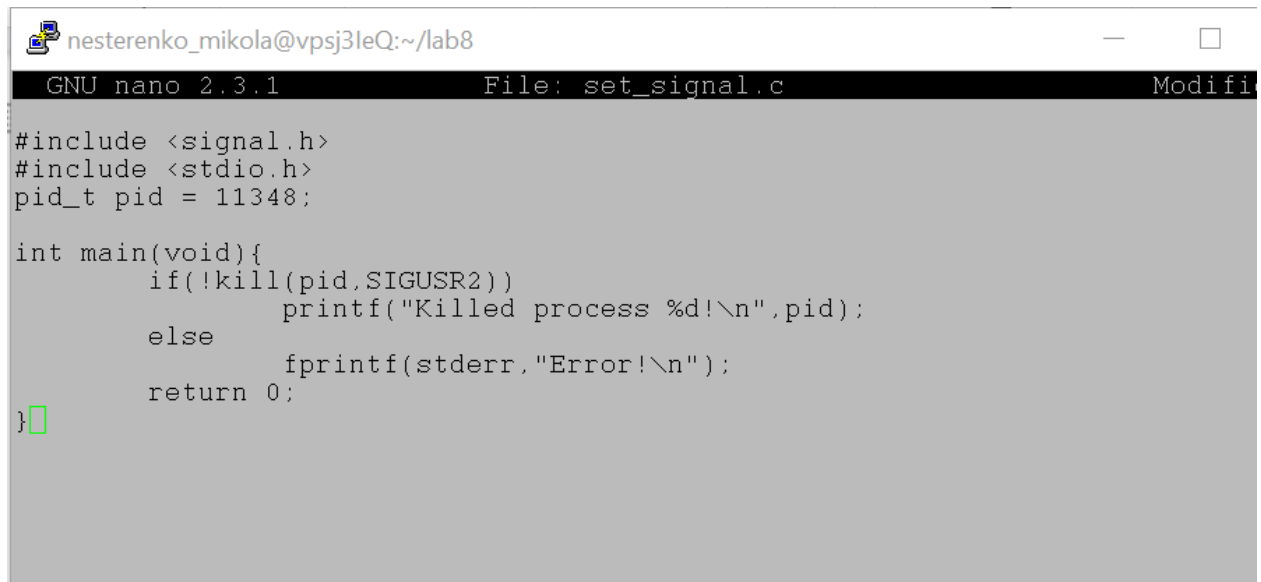
попередньому пункту завдання.

Запустіть створену С-програму та проаналізуйте повідомлення, які виводить перша

програма.

Завершіть процес, запущеному в попередньому пункту завдання.

```
nesterenko_mikola@vpsj3IeQ lab8]$ ps -u nesterenko_mikola -o pid,stat,cmd
PID STAT CMD
920 T ./get_signal
1246 T ./getsig
1348 S+ ./getsig
1531 R+ ps -u nesterenko_mikola -o pid,stat,cmd
4584 S sshd: nesterenko_mikola@pts/34
4585 Ss -bash
4799 S sshd: nesterenko_mikola@pts/45
4800 Ss -bash
```



```
nesterenko_mikola@vpsj3IeQ:~/lab8
GNU nano 2.3.1 File: set_signal.c
#include <signal.h>
#include <stdio.h>
pid_t pid = 11348;

int main(void){
    if(!kill(pid,SIGUSR2))
        printf("Killed process %d!\n",pid);
    else
        fprintf(stderr,"Error!\n");
    return 0;
}
```

```
[nesterenko_mikola@vpsj3IeQ lab8]$ gcc set_signal.c -o killer
[nesterenko_mikola@vpsj3IeQ lab8]$ ./killer
Killed process 11348!
[nesterenko_mikola@vpsj3IeQ lab8]$
```

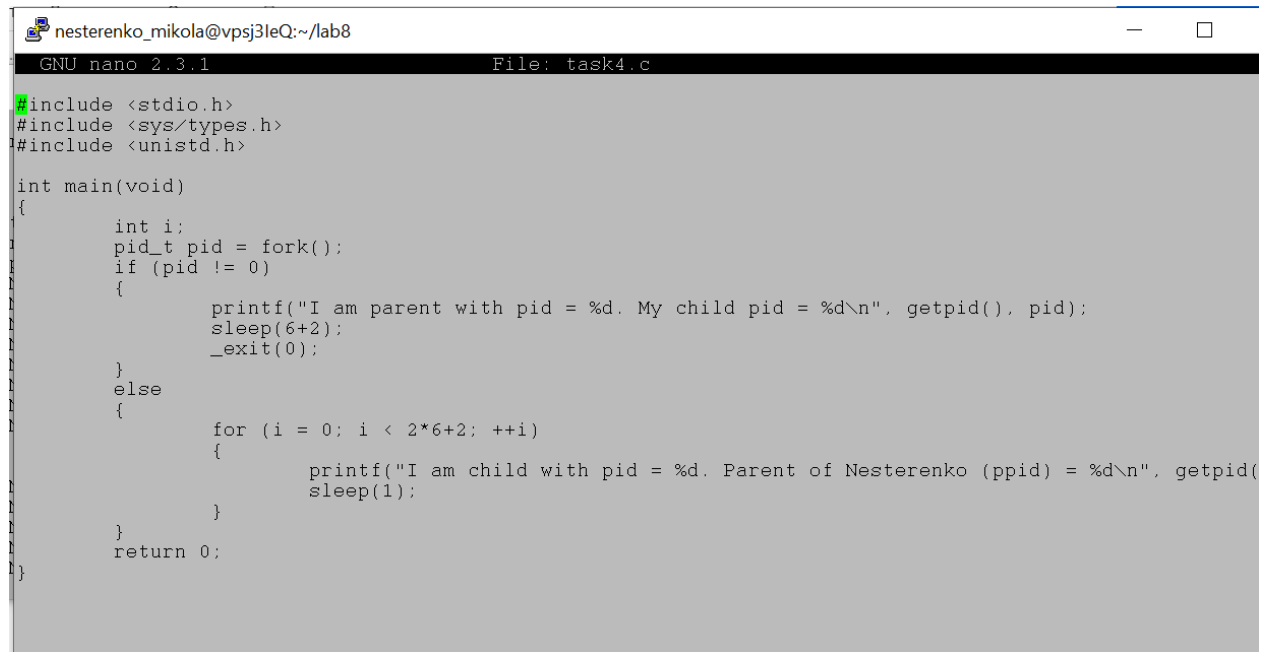
Завдання 4 Створення процесу-сироти

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 $(2 \cdot n + 1)$ раз із затримкою в 1 секунду
ВІВОДИТИ ПОВІДОМЛЕННЯ,

наприклад,



```
nesterenko_mikola@vpsj3IeQ:~/lab8
GNU nano 2.3.1 File: task4.c

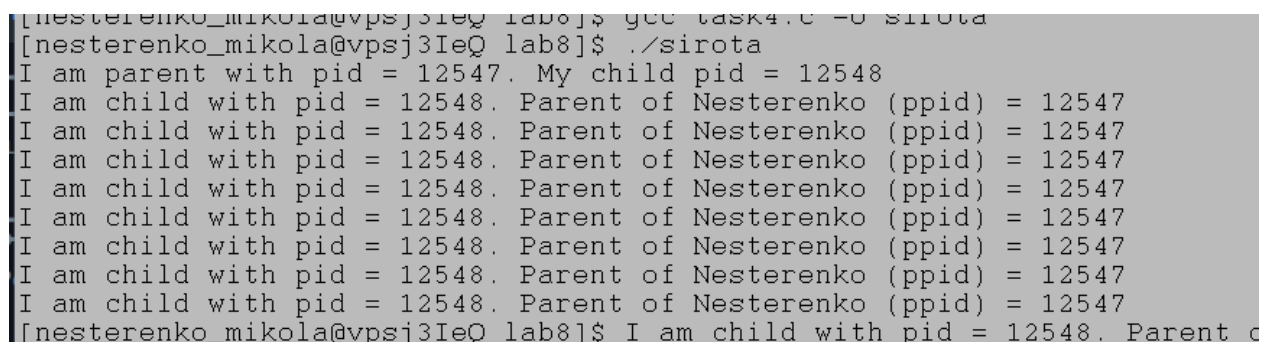
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(void)
{
    int i;
    pid_t pid = fork();
    if (pid != 0)
    {
        printf("I am parent with pid = %d. My child pid = %d\n", getpid(), pid);
        sleep(6+2);
        _exit(0);
    }
    else
    {
        for (i = 0; i < 2*6+2; ++i)
        {
            printf("I am child with pid = %d. Parent of Nesterenko (ppid) = %d\n", getpid(), getppid());
            sleep(1);
        }
    }
    return 0;
}
```

«Parent of Ivanov», за шаблоном як в попередньому завданні, і
ДОДАТКОВО ВІВОДИТИ

PPID процесу-батька.

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



```
nesterenko_mikola@vpsj3IeQ lab8]$ gcc task4.c -o sirota
[nesterenko_mikola@vpsj3IeQ lab8]$ ./sirota
I am parent with pid = 12547. My child pid = 12548
I am child with pid = 12548. Parent of Nesterenko (ppid) = 12547
I am child with pid = 12548. Parent of Nesterenko (ppid) = 12547
I am child with pid = 12548. Parent of Nesterenko (ppid) = 12547
I am child with pid = 12548. Parent of Nesterenko (ppid) = 12547
I am child with pid = 12548. Parent of Nesterenko (ppid) = 12547
I am child with pid = 12548. Parent of Nesterenko (ppid) = 12547
I am child with pid = 12548. Parent of Nesterenko (ppid) = 12547
I am child with pid = 12548. Parent of Nesterenko (ppid) = 12547
[nesterenko_mikola@vpsj3IeQ lab8]$ I am child with pid = 12548. Parent c
```

Висновки: в ході виконання лабораторної роботи були придбані навички в
управлінні процесами в ОС Unix на рівні мови програмування C. На мою
думку найскладнішими завданнями були робота із сигналами.