

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА «ІНФОРМАЦІЙНИХ СИСТЕМ»

Лабораторна робота № 8
з дисципліни «Операційні
системи»

Тема: «Програмування керування процесами в ОС Unix»

Виконав:

Студент групи AI-202
Лобко Данііл Віталійович

Одеса-2021

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

Перелік завдань:

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.

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.

Завдання 3 Обмін сигналами між процесами

3.1 Створіть C-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.

Хід 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.

Створимо програму, що задовольнить умови завдання:

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

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

Запустимо програму. Як ми бачимо, вона працює справно. як ми бачимо, у конвеєрі не однакові значення мас `pid`

```
sid=26313
[lobko_daniil@vpsj3IeQ lobko_lab_8]$ ./infoo | ./infoo
pid=27888
ppid=26313
uid=54348
gid=54354
gpuid=27887
sid=26313
pid=27887
ppid=26313
uid=54348
gid=54354
gpuid=27887
sid=26313
[lobko_daniil@vpsj3IeQ lobko_lab_8]$
```

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.

Створимо програму, що створює процес-нащадок. Викликаємо роботу програми, задовольняючи умови завдання:

```

GNU nano 2.3.1      File: 2.c      Mod
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
extern char** environ;

int main(void) {
    char* echo_args[]={"echo", "I'm echo of child process\n", NULL};
    pid_t pid = fork();
    if(pid!=0){
        printf("Parent of lobko\n\n pid=%d\n child pid=%d\n", getpid(), pid);
        execve("/bin/echo", echo_args, environ);
        fprintf(stderr, "error");
    }
    return 0;
}

```

Перевіряємо:

```

[lobko_daniil@vpsj3IeQ lobko_lab_8]$ ./infoo2
Parent of lobko

pid=648
child pid=649
I'm echo of child process
[lobko_daniil@vpsj3IeQ lobko_lab_8]$

```

Завдання 3 Обмін сигналами між процесами

3.1 Створіть C-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 C-програму.

3.2 Створіть C-програму, яка надсилає сигнал SIGUSR2 процесу, запущеному в попередньому пункту завдання.

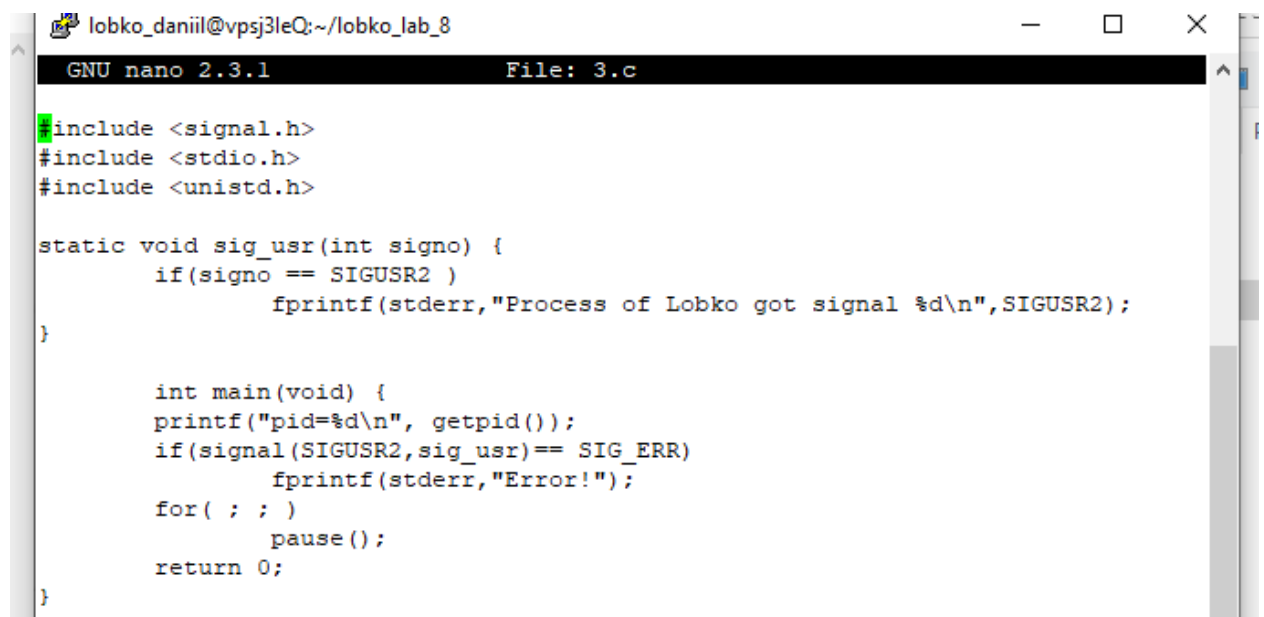
Запустіть створену С-програму та проаналізуйте повідомлення, які виводить перша

програма.

Завершіть процес, запущеному в попередньому пункту завдання.

Виконуємо завдання:

Перша програма:



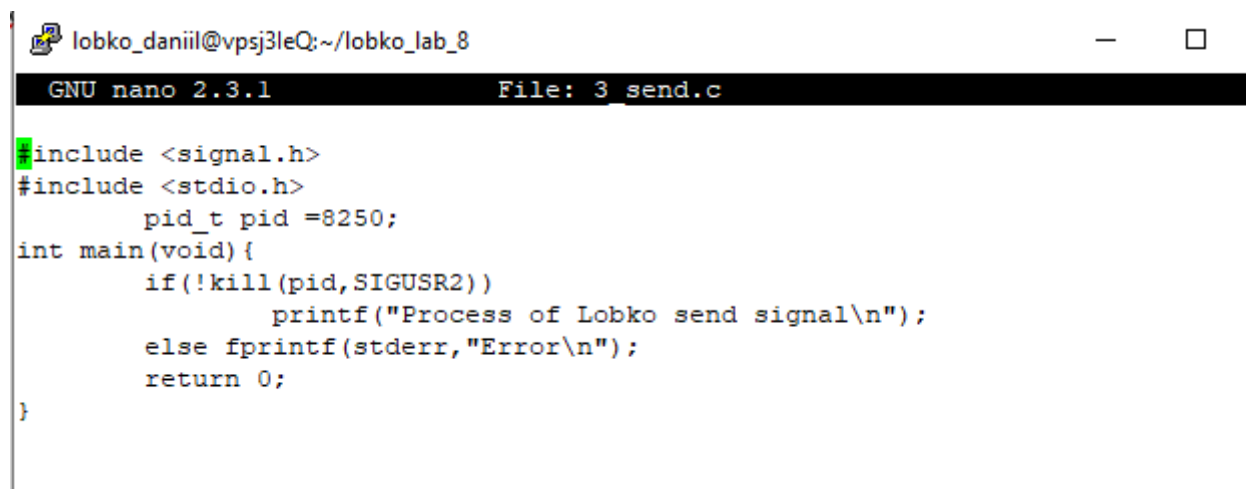
```
lobko_daniil@vpsj3leQ:~/lobko_lab_8
GNU nano 2.3.1 File: 3.c

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

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

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

Друга програма:



```
lobko_daniil@vpsj3leQ:~/lobko_lab_8
GNU nano 2.3.1 File: 3_send.c

#include <signal.h>
#include <stdio.h>
    pid_t pid =8250;
int main(void){
    if(!kill(pid,SIGUSR2))
        printf("Process of Lobko send signal\n");
    else fprintf(stderr,"Error\n");
    return 0;
}
```

```
lobko_daniil@vpsj3ieQ:~/lobko_lab_8
login as: lobko_daniil
lobko_daniil@91.219.60.189's password:
Last login: Wed Apr 21 11:50:48 2021 from 78.26.152.223
[lobko_daniil@vpsj3ieQ ~]$ ls
accountTs.csv lobko_lab_3 lobko_lab_7 Operating-System.-Laboratory-Work-1
lobko_43.csv lobko_lab_5 lobko_lab_8
lobko_4.csv lobko_lab_6 nohup.out
[lobko_daniil@vpsj3ieQ ~]$ cd lobko_lab_8
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ ls
1.c 2.c 3.c 3_send.c a.out info info2 info3 info3send
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ nano 3_send.c
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ gcc 3_send.c -o sent_s
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ ./sent_s
Process of Lobko send signal
[lobko_daniil@vpsj3ieQ lobko_lab_8]$

lobko_daniil@vpsj3ieQ:~/lobko_lab_8
login as: lobko_daniil
lobko_daniil@91.219.60.189's password:
Last login: Wed Apr 21 11:39:14 2021 from 78.26.152.223
[lobko_daniil@vpsj3ieQ ~]$ nano 3.c
[lobko_daniil@vpsj3ieQ ~]$ cd lobko_lab_8
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ nano 3.c
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ ./info3
^C
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ gcc 3.c -o get_s
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ ls
1.c 2.c 3_send.c get_s info info3
1.c 3.c a.out info info2 info3send
[lobko_daniil@vpsj3ieQ lobko_lab_8]$ ./get_s
pid=8250
Process of Lobko got signal 12
```

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.

Код програми:

```
GNU nano 2.3.1 File: 4.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(void)
{
    int i;
    pid_t pid = fork();
    if(pid!=0){
        printf("Im parent with pid =%d. My child pid=%d\n", getpid(),pid);
        sleep(5);
        _exit(0);}

    else {
        for (i=0; i<9; i++) {
            printf("I am child with pid =%d, My parent id =%d\n", getpid(),getppid());
            sleep(1);
        }
    }
    return 0;
}
```

[Read 20 lines]

Результат програми:

```
[lobko_daniil@vpsj3IeQ lobko_lab_8]$ ./4task
Im parent with pid =10025. My child pid=10026
I am child with pid =10026, My parent id =10025
I am child with pid =10026, My parent id =10025
I am child with pid =10026, My parent id =10025
I am child with pid =10026, My parent id =10025
I am child with pid =10026, My parent id =10025
I am child with pid =10026, My parent id =10025
[lobko_daniil@vpsj3IeQ lobko_lab_8]$ I am child with pid =10026, My parent id =1
I am child with pid =10026, My parent id =1
I am child with pid =10026, My parent id =1
I am child with pid =10026, My parent id =1
```

есами в ОС Unix на рівні мови

Висновки: Виконуючи цю лабораторну роботи ми закріпили навички роботи з управління процесами в Linux.

Найскладнішим було завдання 3 завдання через певне недорозуміння завдання та роботу одразу з 2 терміналами.