

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 керування процесами в ОС Unix»

Виконав:

Студент групи AI-202

Сідельніков М. В.

Перевірив:

Блажко О. А.

Одеса-2021

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

Завдання:

Завдання 1 Перегляд інформації про процес.

1. 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.
2. Завершіть створення програми включенням функції `sleep(5)` для забезпечення засинання процесу на 5 секунд.
3. При створенні повідомлень використовуйте функцію `fprintf` з виведенням на потік помилок.
4. Після компіляції запусіть програму.
5. Додатково запусіть програму в конвеєрі.

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.

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

Завдання 4 Створення процесу-сироти.

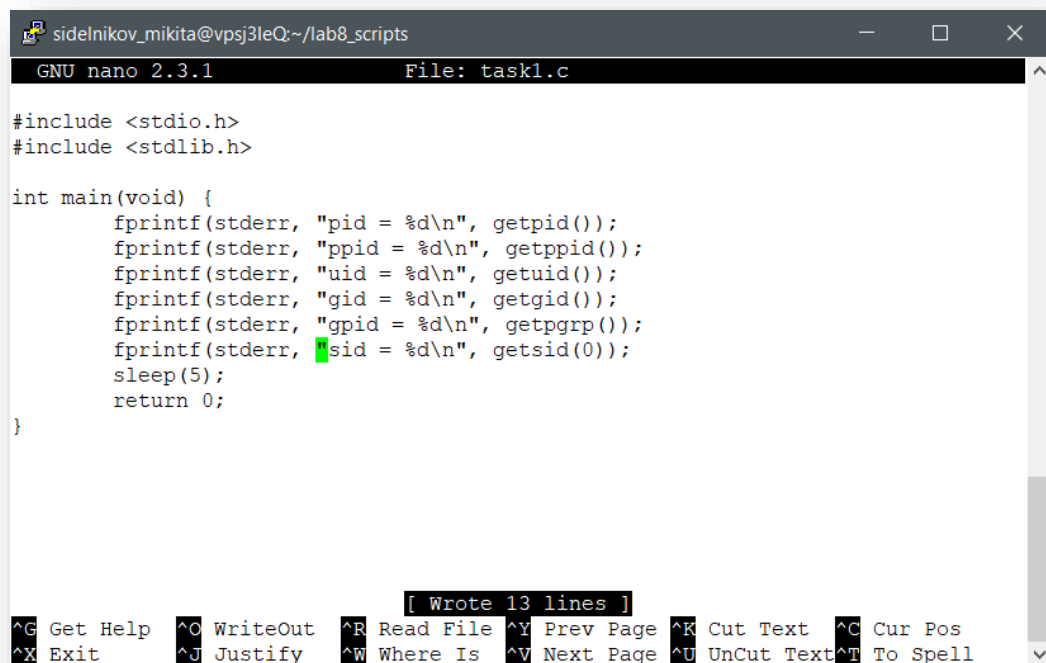
Створіть C-програму, в якій процес-батько несподівано завершується раніше процесу-нащадку. Процес-батько повинен очікувати завершення $n+1$ секунд. Процес-нащадок повинен в циклі $(2*n+1)$ раз із затримкою в 1 секунду виводити повідомлення, наприклад, «Parent of Ivanov», за шаблоном як в попередньому 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.

1. 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.
2. Завершіть створення програми включенням функції `sleep(5)` для забезпечення засинання процесу на 5 секунд.
 3. При створенні повідомлень використовуйте функцію `fprintf` з виведенням на потік помилок.
 4. Після компіляції запусіть програму.
 5. Додатково запусіть програму в конвеєрі.



```
sidelnikov_mikita@vpsj3leQ:~/lab8_scripts
GNU nano 2.3.1 File: task1.c

#include <stdio.h>
#include <stdlib.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, "gpuid = %d\n", getpgrp());
    fprintf(stderr, "sid = %d\n", getsid(0));
    sleep(5);
    return 0;
}

[ Wrote 13 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

```
sidelnikov_mikita@vpsj3IeQ:~/lab8_scripts
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ gcc task1.c -o info
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ ./info | ./info
pid = 28648
ppid = 23625
uid = 54351
gid = 54357
gpid = 28647
sid = 23625
pid = 28647
ppid = 23625
uid = 54351
gid = 54357
gpid = 28647
sid = 23625
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$
```

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.

```
sidelnikov_mikita@vpsj3IeQ:~/lab8_scripts
GNU nano 2.3.1 File: task2.c

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

extern char** environ;

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

[ Read 17 lines ]
^G Get Help      ^C WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spell
```

```
sidelnikov_mikita@vpsj3IeQ:~/lab8_scripts
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ gcc task2.c
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ ./a.out
I am parent of Sidelnykov! pid = 32652, child pid = 32653
I am child of Sidelnykov!
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$
```

Завдання 3 Обмін сигналами між процесами.

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.

```
sidelnikov_mikita@vpsj3IeQ:~/lab8_scripts
GNU nano 2.3.1 File: task3.c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

static void sig_usr(int signo) {
    if (signo == SIGUSR2)
        printf("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;
}

[ Read 17 lines ]
^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spell
```

```
sidelnikov_mikita@vpsj3leQ:~/lab8_scripts
[sidelnikov_mikita@vpsj3leQ lab8_scripts]$ nano task3.c
[sidelnikov_mikita@vpsj3leQ lab8_scripts]$ gcc task3.c -o get
[sidelnikov_mikita@vpsj3leQ lab8_scripts]$ ./get
pid = 7271
```

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.

```
GNU nano 2.3.1 File: task3-send.c Modified
#include <signal.h>
#include <stdio.h>

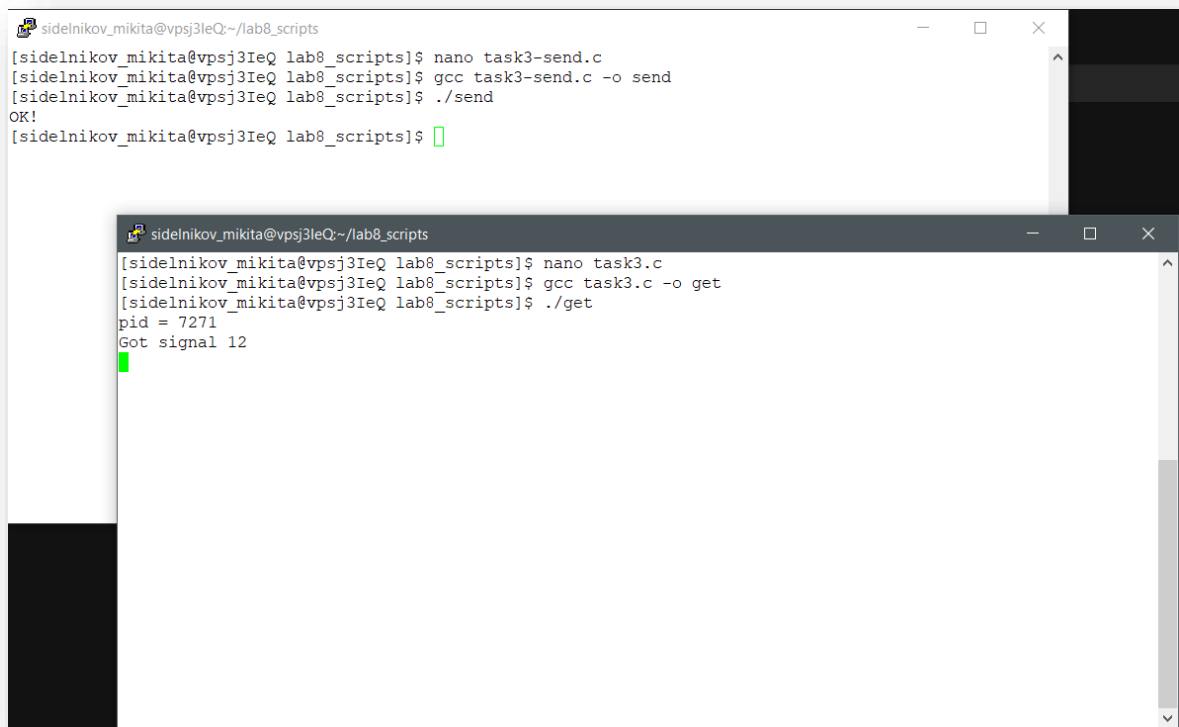
pid_t pid = 7271;

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

[Read 12 lines]

^G Get Help	^O WriteOut	^R Read File	^Y Prev Page	^K Cut Text	^C Cur Pos
^X Exit	^J Justify	^W Where Is	^V Next Page	^U UnCut Text	^T To Spell

Перша програма отримала сигнал з другої та вивела у консоль номер сигналу.



The image shows two overlapping terminal windows from a Linux environment. The top window shows the compilation and execution of a program named 'task3-send.c'. The user runs 'nano task3-send.c', then 'gcc task3-send.c -o send', and finally './send', which outputs 'OK!'. The bottom window shows the compilation and execution of 'task3.c'. The user runs 'nano task3.c', then 'gcc task3.c -o get', and finally './get'. The output shows 'pid = 7271' and 'Got signal 12'.

```
sidelnikov_mikita@vpsj3IeQ:~/lab8_scripts
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ nano task3-send.c
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ gcc task3-send.c -o send
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ ./send
OK!
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$

sidelnikov_mikita@vpsj3IeQ:~/lab8_scripts
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ nano task3.c
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ gcc task3.c -o get
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ ./get
pid = 7271
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.


```
sidelnikov_mikita@vpsj3IeQ:~/lab8_scripts
GNU nano 2.3.1 File: task4.c

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

int main(void) {
    int i;
    pid_t pid = fork();
    if (pid != 0) {
        printf("I am parent! pid = %d, child pid = %d\n", getpid(), pid);
        sleep(8);
        _exit(0);
    } else {
        for (i = 0; i < 15; i++) {
            printf("I am child with pid = %d. My parent pid = %d\n", getpid(), getppid());
            sleep(1);
        }
    }
    return 0;
}
```

```
sidelnikov_mikita@vpsj3IeQ:~/lab8_scripts
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ gcc task4.c
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ ./a.out
I am parent! pid = 5685, child pid = 5686
I am child with pid = 5686. My parent pid = 5685
I am child with pid = 5686. My parent pid = 5685
I am child with pid = 5686. My parent pid = 5685
I am child with pid = 5686. My parent pid = 5685
I am child with pid = 5686. My parent pid = 5685
I am child with pid = 5686. My parent pid = 5685
I am child with pid = 5686. My parent pid = 5685
I am child with pid = 5686. My parent pid = 5685
[sidelnikov_mikita@vpsj3IeQ lab8_scripts]$ I am child with pid = 5686. My parent pid = 1
I am child with pid = 5686. My parent pid = 1
I am child with pid = 5686. My parent pid = 1
I am child with pid = 5686. My parent pid = 1
I am child with pid = 5686. My parent pid = 1
I am child with pid = 5686. My parent pid = 1
I am child with pid = 5686. My parent pid = 1
```

Коли батьківський процес несподівано перервався та не завершив роботу child процесу, батьківським став root процес (PID = 1).

Висновки: під час виконання лабораторної роботи було отримано навичок в управлінні процесами в ОС Unix на рівні мови програмування C.