

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»

Варіант 6(1)

Виконав:
Студент групи АІ-203
Сиваш А.І.

Одеса 2021

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.

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 C-програму.

3.2 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.

Завдання 5 Створення процесу-зомбі

Створіть C-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 C-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 – 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

```
[sivash_andrij@vpsj3IeQ ~]$ mkdir lab8
[sivash_andrij@vpsj3IeQ ~]$ cd lab8
[sivash_andrij@vpsj3IeQ lab8]$ touch info.c
[sivash_andrij@vpsj3IeQ lab8]$
```

```
mc [sivash_andrij@vpsj3IeQ.s-host.com.ua]: ~/lab8
Left      File      Command  Options  Right
<- ~/lab8 .[^]> <- ~/lab8 .[^]>
'n      Name      Size      Modify time      'n      Name      Size      Modify time
/..      UP--DIR      Apr 28 16:56      /..      UP--DIR      Apr 28 16:56
info.c    0      Apr 28 16:56      info.c    0      Apr 28 16:56

UP--DIR                                     UP--DIR
3505M/38G (8%)                             3505M/38G (8%)

Hint: Want to see your *~ backup files? Set it in the Configuration dialog.
[sivash_andrij@vpsj3IeQ lab8]$
1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDn10Quit
```

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

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

```
[sivash_andrij@vpsj3IeQ lab8]$ gcc info.c -o info
[sivash_andrij@vpsj3IeQ lab8]$ ./info
My pid=28427
My ppid=24613
My uid=54372
My gid=54378
```

```
mc [sivash_andrij@vpsj3leQ.s-host.com.ua]:~/lab8
create.c [----] 1 L: [ 1+13 14/ 14] *(289 / 289b) <EOF> [*][X] ^
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void) {

    pid_t pid = fork();
    if (pid==0)
        printf("I am child! my pid = %d\n", getpid());
    else
        printf("I am parent! my pid = %d\n", getpid());
    .....
    return 0;
}

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit v
```

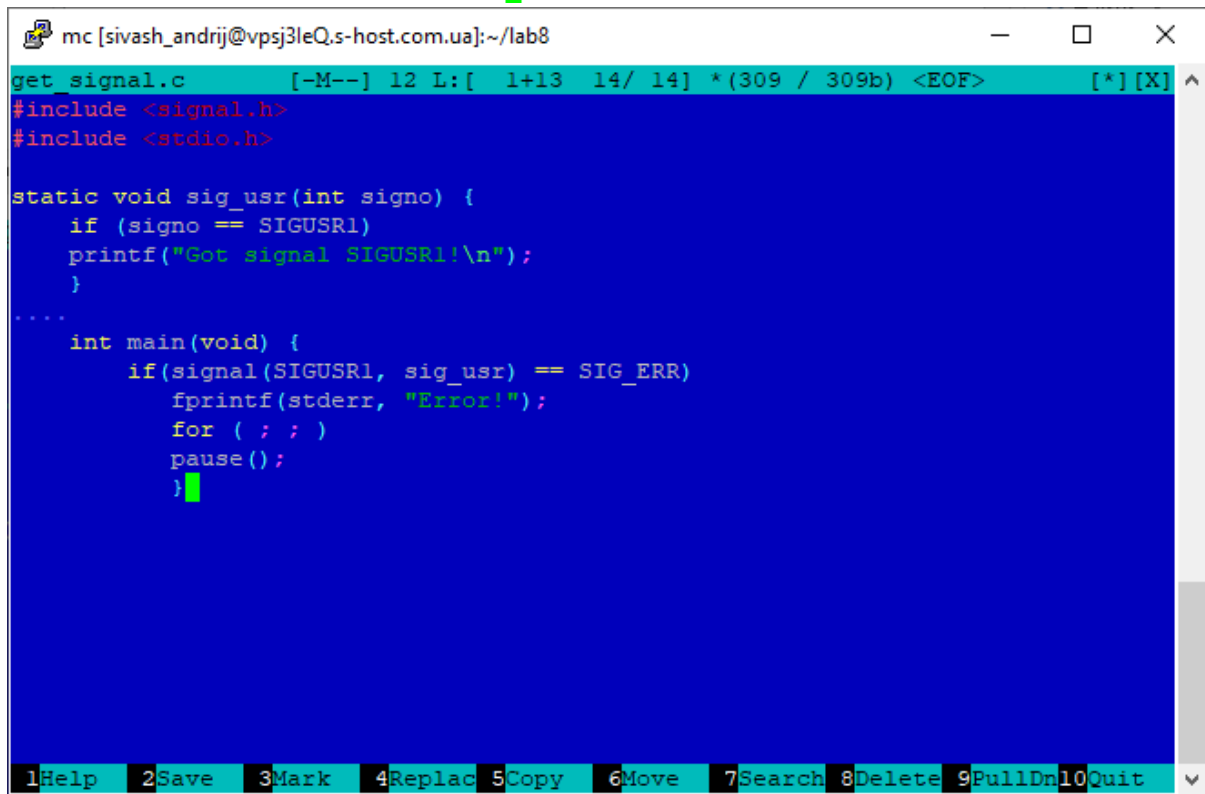
```
[sivash_andrij@vpsj3leQ lab8]$ gcc create.c -o create
[sivash_andrij@vpsj3leQ lab8]$ ./create
I am parent! my pid = 29939
I am child! my pid = 29940
```

```
mc [sivash_andrij@vpsj3leQ.s-host.com.ua]:~/lab8
create.c [-M--] 22 L: [ 1+ 4 5/ 18] *(85 / 480b) 0010 0x00A [*][X] ^
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

extern char** environ;
int main(void) {
    char* echo_args[] = {"echo", "I am ECHO", NULL};
    pid_t pid = fork();
    if (pid==0)
        printf("I am child! my pid = %d\n", getpid());
    else {
        printf("I am parent! my pid = %d\n", getpid());
        execve("/bin/echo", echo_args, environ);
        fprintf(stderr, "Error!");
        return 1;
    }
    return 0;
}

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit v
```

```
[sivash_andrij@vpsj3IeQ lab8]$ gcc create.c -o create
[sivash_andrij@vpsj3IeQ lab8]$ ./create
I am parent! my pid = 30910
I am child! my pid = 30911
I am ECHO
```



```
mc [sivash_andrij@vpsj3IeQ.s-host.com.ua]:~/lab8
get_signal.c  [-M--] 12 L:[ 1+13 14/ 14] *(309 / 309b) <EOF>  [*][X] ^
#include <signal.h>
#include <stdio.h>

static void sig_usr(int signo) {
    if (signo == SIGUSR1)
        printf("Got signal SIGUSR1!\n");
}

....

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

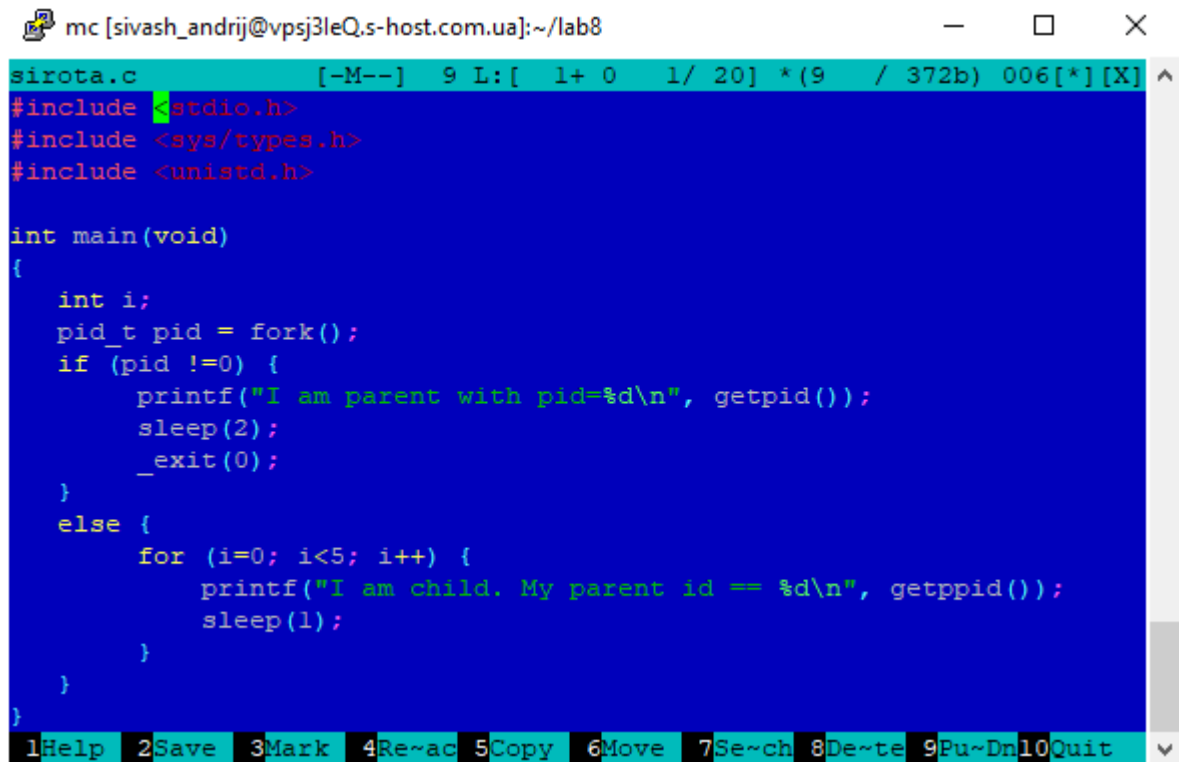
```
[sivash_andrij@vpsj3IeQ lab8]$ gcc get_signal.c -o get_signal
[sivash_andrij@vpsj3IeQ lab8]$ ./get_signal
```

```
[sivash_andrij@vpsj3IeQ lab8]$ ps -u sivash_andrij
```

PID	TTY	TIME	CMD
24612	?	00:00:00	sshd
24613	pts/5	00:00:00	bash
29668	pts/5	00:00:00	mc
29670	pts/7	00:00:00	bash
29692	pts/5	00:00:00	sh
29693	pts/5	00:00:00	vi
29739	pts/5	00:00:00	mc
29741	pts/9	00:00:00	bash
29763	pts/5	00:00:00	sh
29764	pts/5	00:00:00	vi
31780	pts/5	00:00:00	get_signal
32535	pts/5	00:00:00	ps

```
[sivash_andrij@vpsj3IeQ lab8]$ gcc send_signal.c -o send_signal
[sivash_andrij@vpsj3IeQ lab8]$ ./send_signal
Sent signal
[sivash_andrij@vpsj3IeQ lab8]$ ./send_signal
Sent signal
[sivash_andrij@vpsj3IeQ lab8]$
```

```
[sivash_andrij@vpsj3IeQ lab8]$ ./get_signal
Got signal SIGUSR1!
Got signal SIGUSR1!
```



```
mc [sivash_andrij@vpsj3IeQ.s-host.com.ua]:~/lab8
sirota.c [-M--] 9 L:[ 1+ 0 1/ 20] *(9 / 372b) 006[*][X] ^
#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\n", getpid());
        sleep(2);
        _exit(0);
    }
    else {
        for (i=0; i<5; i++) {
            printf("I am child. My parent id == %d\n", getppid());
            sleep(1);
        }
    }
}
```

```
[sivash_andrij@vpsj3IeQ lab8]$ gcc sirota.c -o sirota
[sivash_andrij@vpsj3IeQ lab8]$ ./sirota
I am parent with pid=2798
I am child. My parent id == 2798
I am child. My parent id == 2798
I am child. My parent id == 1
[sivash_andrij@vpsj3IeQ lab8]$ I am child. My parent id == 1
I am child. My parent id == 1
```

Висновок:

Найважчим у даній роботі виявилась робота із процесами.