

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:
Голованчук Н.Ю.

Перевірили:
Блажко О.А
Дрозд М.О.

Одеса 2021

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

Хід роботи:

1. Перелік завдань до лабораторної роботи
2. Результатами виконання пунктів завдань
3. Висновки

Перелік завдань до лабораторної роботи:

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.

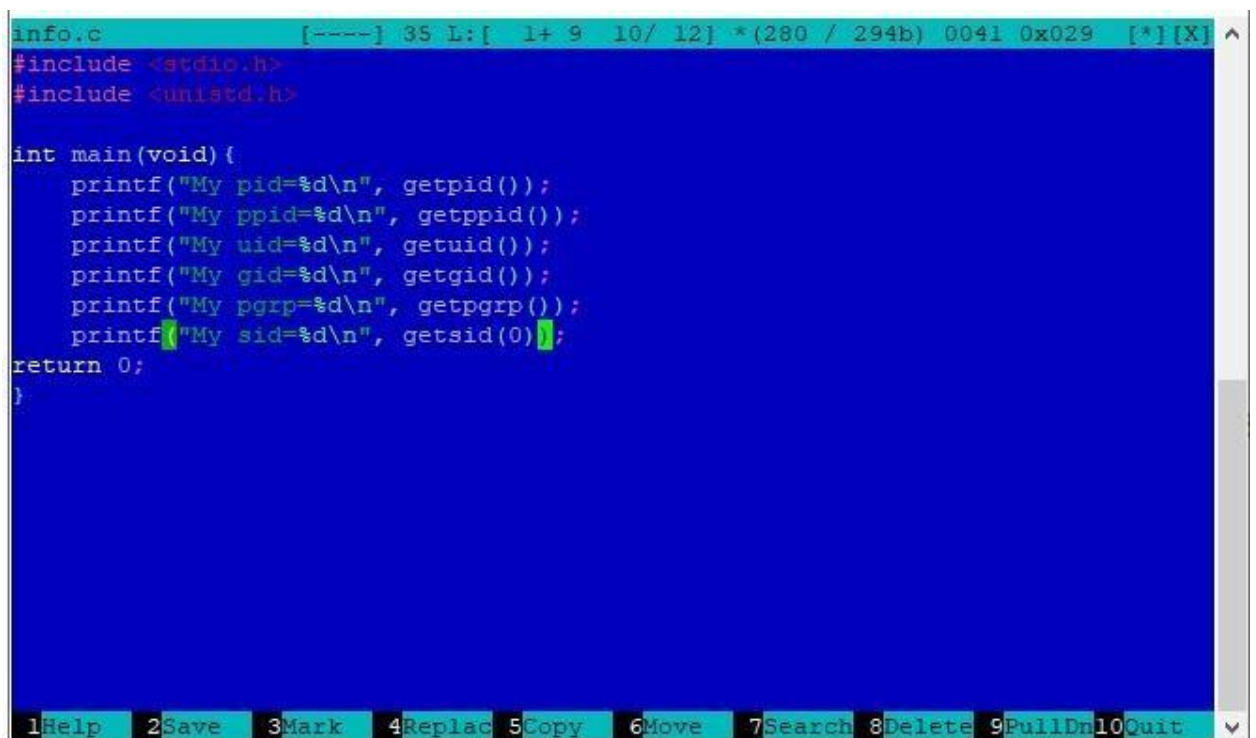
Перевірте роботу програми, вивчіть вміст таблиці процесів і зробіть відповідні висновки.

Результати виконання завдань:

Завдання 1 Перегляд інформації про процес

Створимо C-програму, яка виводить на екран ідентифікатори:

Програма:



```
info.c [-----] 35 L:[ 1+ 9 10/ 12] *(280 / 294b) 0041 0x029 [*][X] ^
#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(0));
return 0;
}
```

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

Результат виконання програми:

```

-bash-4.2$ gcc info.c -o info
      In function '':
      error: too few arguments to function '
      printf("My sid=%d\n", getsid());

In file included from :
      note: declared here
      extern __pid_t getsid (__pid_t __pid) __THROW;

-bash-4.2$ gcc info.c -o info
-bash-4.2$ ./info
My pid=14256
My ppid=14043
My uid=54365
My gid=54371
My pgrp=14256
My sid=14043
-bash-4.2$

```

Завдання 2 Перегляд таблиці процесів

Створимо 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.

Програма:

```
mc [demtsun_andrij@vpsj3leQ.s-host.com.ua]:~
create.c [-----] 4 L: [ 1+ 0 1/ 19] *(4 / 490b) 0108 0x06C [*][X] ^
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

extern char** environ;

int main(void) {
    char* echo_args[] = {"echo", "I am Demtsun", 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;
}
```

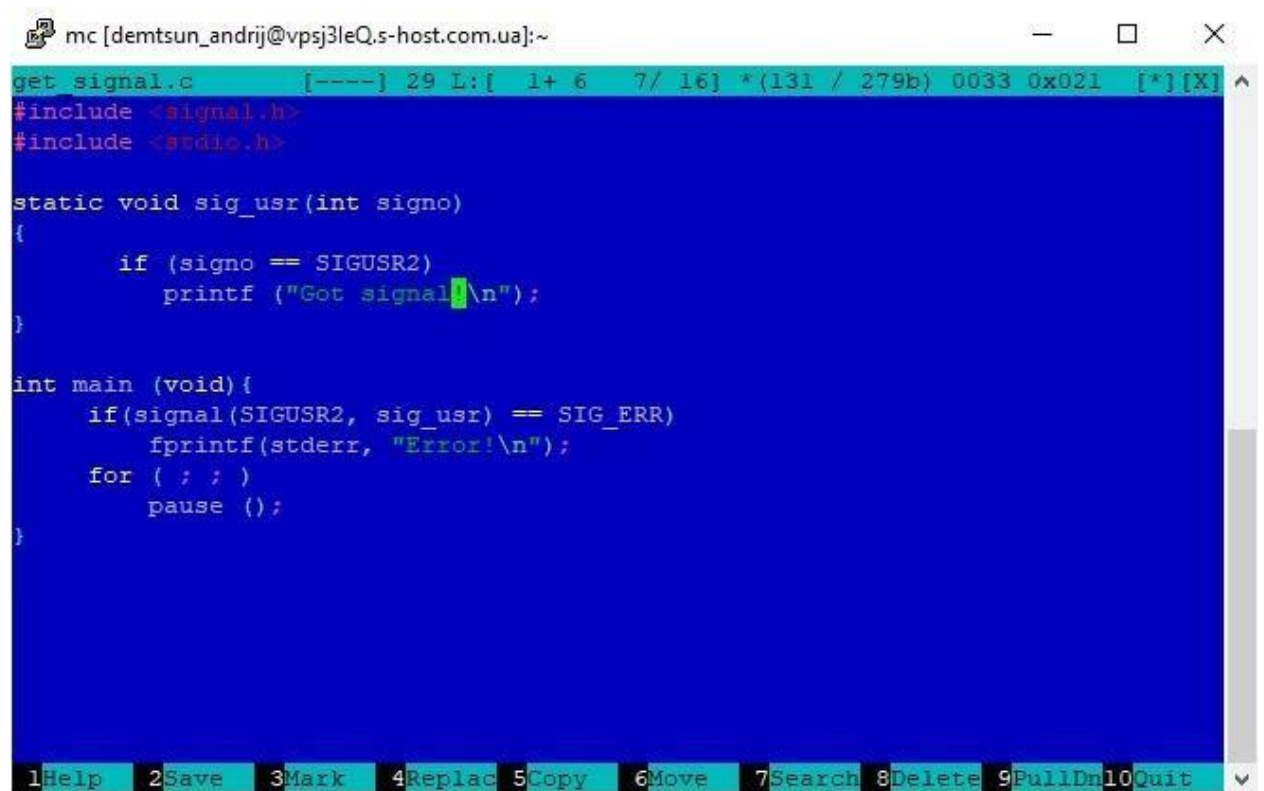
Результат виконання програми:

```
-bash-4.2$ gcc create.c -o create
-bash-4.2$ ./create
I am parent! my pid =29631
I am child! my pid = 29632
I am Demtsun
-bash-4.2$
```

Завдання 3 Обмін сигналами між процесами

3.1 Створимо 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. Запустимо програму.

Програма 3.1:



```
mc [demtsun_andrij@vpsj3leQ.s-host.com.ua]:~
get_signal.c [----] 29 L:[ 1+ 6 7/ 16] *(131 / 279b) 0033 0x021 [*][X] ^
#include <signal.h>
#include <stdio.h>

static void sig_usr(int signo)
{
    if (signo == SIGUSR2)
        printf ("Got signal\n");
}

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

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

Дізнаємось pid:

3.2 Створимо С-програму, яка надсилає сигнал SIGUSR2 процесу, запущеному в попередньому пункту завдання.

Програма 3.2:

```
mc [demtsun_andrij@vpsj3leQ.s-host.com.ua]:~
send signal.c [----] 1 L: [ 1+13 14/ 14] *(263 / 263b) <EOF> [*] [X] ^
#include <stdio.h>
#include <signal.h>
#include <errno.h>

pid_t pid = 26216;

int main(void)
{
    if (!kill(pid, SIGUSR2))
        printf("Send signal %d!\n", pid);
    else
        fprintf(stderr, "Error by processing %d!\n", pid);
return 0;
}

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

Результат виконання програм 3.1 і 3.2:

PuTTY (inactive)

```
gcc get_signal.c -o get_signal
gcc get_signal.c -o get_signal
gcc get_signal.c -o get_signal
gcc get_signal.c -o get_signal
gcc get_signal.c -o get_signal
^C
-bash-4.2$ gcc get_signal.c -o get_signal
-bash-4.2$ ./get_signal
Got signal SIGUSR2!
gcc get_signal.c -o get_signal
^C
-bash-4.2$ gcc get_signal.c -o get_signal
-bash-4.2$ ./get_signal
Got signal SIGUSR2!
gcc get_signal.c -o get_signal
^C
-bash-4.2$ gcc get_signal.c -o get_signal
-bash-4.2$
-bash-4.2$ ./get_signal
^C
-bash-4.2$ ^C
-bash-4.2$ gcc get_signal.c -o get_signal
-bash-4.2$ ./get_signal
Got signal!
```

PuTTY (inactive)

```
error: expected declaration specifiers before ' ' token
}
error: expected ' ' at end of input
-bash-4.2$ gcc send_signal.c -o send_signal
-bash-4.2$ ./send_signal
Send signal: Process 24045 has been killed!
-bash-4.2$ gcc send_signal.c -o send_signal
In function ' ':
warning: unknown escape sequence: '\321' [enabled by default]
t)
printf("Send signal!\t");
-bash-4.2$ gcc send_signal.c -o send_signal
-bash-4.2$ ./send_signal
Send signal!
-bash-4.2$ gcc send_signal.c -o send_signal
-bash-4.2$ ./send_signal
Send signal 25568!
-bash-4.2$ gcc send_signal.c -o send_signal
-bash-4.2$ ./send_signal
Send signal 26216!
-bash-4.2$
```

Завдання 4 Створення процесу-сироти

mc [demtsun_andrij@vpsj3leQ.s-host.com.ua]:~

```
sirota.c [-----] 55 L:[ 1+ 9 10/ 21] *(200 / 452b) 0010 0x00A
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>

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

Результат виконання програми:

```
-bash-4.2$ gcc sirota.c -o sirota
-bash-4.2$ ./sirota
I am child pid = 27639. My parent ppid = 27638
I am child pid = 27639. My parent ppid = 27638
I am child pid = 27639. My parent ppid = 27638
I am child pid = 27639. My parent ppid = 1
-bash-4.2$ I am child pid = 27639. My parent ppid = 1
I am child pid = 27639. My parent ppid = 1
```

Висновок: Під час виконання лабораторної роботи було отримано практичні навички в управлінні процесами в ОС Unix на рівні мови програмування C.