

МІНІСТЕРСТВО ОСВІТИ Й НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
Інститут комп'ютерних систем
Кафедра інформаційних систем

Лабораторна робота № 8
За дисципліною: "Операційні системи"
Тема: «Програмування керуванням процесами в ОС Unix»

Виконав:
Студент групи АІ-205
Шостак Р.С.
Перевірили:
Блажко О.А.
Дрозд М.О.

Одеса 2021

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

2. Завдання

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, що викликав цю функцію
- ідентифікатор групи процесу, що викликав цю функцію.

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

int main(void) {
pid_t pid = getpid();
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(pid));
return 0;
}

[shostak_roman@vpsj3IeQ ~]$ gcc -o prim2 prim.c
[shostak_roman@vpsj3IeQ ~]$ ./prim2
My pid=16274
My ppid=15833
My uid=54421
My gid=54427
My pgrp=16274
My sid=15833
[shostak_roman@vpsj3IeQ ~]$
```

Завдання 2 Стандартне породження процесу

Створіть С-програму, яка породжує процес-нащадок. У програмі процес-батько повинен видати повідомлення типу «Parent of Ivanov», а процес-нащадок повинен видати повідомлення типу «Child of Ivanov», де замість слова Ivanov в повідомленні повинно бути ваше прізвище в транслітерації.

```

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

extern char** environ; █

int main(void){
    char* echo_args[] = {"echo", "I am Echo\n", NULL};
    pid_t pid = fork ();
    if(pid == 0)
        printf("Child of Shostak: pid=%d\n",getpid());
    else{
        printf("Parent of Shostak: pid=%d\n",getpid());
        execve("/bin/echo",echo_args,environ);
    }
    return 0;
}

```

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

[shostak_roman@vpsj3IeQ ~]\$ gcc -o prim22 prim2.c
[shostak_roman@vpsj3IeQ ~]\$./prim22
Parent of Shostak: pid=18390
Child of Shostak: pid=18391
I am Echo

[shostak_roman@vpsj3IeQ ~]\$ █

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 транслітерації.

Запустіть створену C-програму.

```

getsig.c      [----]  1 L:[ 1+13 14/ 14] *(279 / 279b) <EOF>      [*][X]
#include <stdio.h>
#include <signal.h>

static void sig_usr(int signal){
    if(signal == SIGUSR2)
<----->printf("Process of Shostak got signal\n");
<----->
int main(void){
    if(signal(SIGUSR2, sig_usr)== SIG_ERR)
        fprintf(stderr,"Oshibka");
    for( ; ; )
        pause();
    return 1;
}

```

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

3.2 Створіть C-програму, яка надсилає сигнал SIGUSR1 процесу, запущеному в попередньому пункті завдання. Запустіть створену C-програму та проаналізуйте повідомлення, які виводить перша програма.

Завершіть процес, запущеному в попередньому пункті завдання.

```

sendsig.c      [----]  1 L:[ 1+11 12/ 12] *(188 / 188b) <EOF>      [*][X]
#include <stdio.h>
#include <signal.h>

pid_t pid =20550;

int main(void) {
    if(!kill(pid,SIGUSR2))
<----->printf("sent signal to pid=%d",pid);
else.
    fprintf(stderr , "Oshibka");
return 1 ;
}

```

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

```
[shostak_roman@vpsj3leQ ~]$ mc

[shostak_roman@vpsj3leQ ~]$ gcc -o getsif getsig.c
[shostak_roman@vpsj3leQ ~]$ ps -u shostak_roman -o pid,stat,cmd
 PID STAT CMD
12002 S sshd: shostak_roman@notty
12003 Ss /usr/libexec/openssh/sftp-server
15532 S sshd: shostak_roman@pts/4
15533 Ss -bash
20508 S sshd: shostak_roman@pts/17
20509 Ss -bash
20550 S+ ./getsif
20599 R+ ps -u shostak_roman -o pid,stat,cmd
[shostak_roman@vpsj3leQ ~]$ mc

[shostak_roman@vpsj3leQ ~]$ mc

[shostak_roman@vpsj3leQ ~]$ gcc -o sendsif sendsig.c
[shostak_roman@vpsj3leQ ~]$ ./sendsif
sent signal to pid=20550 shostak_roman@vpsj3leQ -> ./sendsif
sent signal to pid=20550 shostak_roman@vpsj3leQ -> ./sendsif
sent signal to pid=20550 shostak_roman@vpsj3leQ -> $ kill 20550
[shostak_roman@vpsj3leQ ~]$
```

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.

```
[shostak_roman@vpsj3IEQ ~]$ gcc -o sirot sirotka.c
[shostak_roman@vpsj3IEQ ~]$ ./sirot
I am parent with pid=22361. My child pid =22362
I am child with pid=22362. My parent id =22361
I am child with pid=22362. My parent id =22361
I am child with pid=22362. My parent id =22361
I am child with pid=22362. My parent id =22361
I am child with pid=22362. My parent id =22361
```

```
sirotka.c      [----]  9 L:[ 1+20 21/ 21] *(358 / 358b) <EOF>      [*] [X]
#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 with pid=%d. My child pid =%d\n",getpid(),pid);
<----->sleep(10);
<----->_exit(0);
<----->}
<----->else{
<----->for(i=0;i<19;i++) {
<----->printf("I am child with pid=%d. My parent id =%d\n",getpid(),getppid());
<----->sleep(1);
<----->}
<----->}
<----->return 0;
<----->}
```

1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9PullDn 10Quit

|||||