

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»

**Виконала:** Студентка групи AI-204

Озарчук А.С.

**Перевірили:** Блажко О. А

Дрозд М. О.

## 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 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.

#### **Завдання 5. Створення процесу-зомбі**

Створіть С-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 С-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  $n$  -  $n$  – номер команди студента + номер студента в команді.

Запустіть програму у фоновому режимі, а в окремому терміналі вивчіть вміст таблиці процесів і зробіть відповідні висновки.

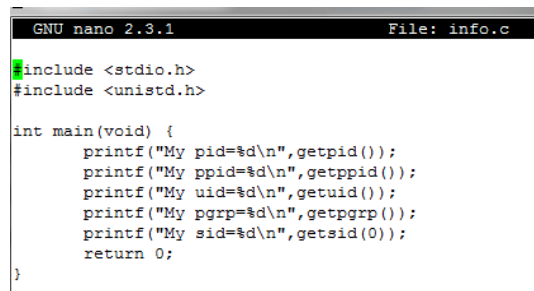
# Виконання завдання

## Завдання 1. Перегляд інформації про процес

Я створила 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.

```
My sid=24693[ozarchuk_anna@vpsj3IeQ ~]$ gcc info.c -o info
[ozarchuk_anna@vpsj3IeQ ~]$ ./info
My pid=26494
My ppid=24693
My uid=54398
My pgrp=26494
My sid=24693
```



```
GNU nano 2.3.1 File: info.c
#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 pgrp=%d\n",getpgrp());
    printf("My sid=%d\n",getsid(0));
    return 0;
}
```

## Завдання 2. Стандартне створення процесу

Я створила 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.

```
[ozarchuk_anna@vpsj3IeQ ~]$ gcc create_any.c -o create_any
[ozarchuk_anna@vpsj3IeQ ~]$ ./create_any
Parent pid =30737
Child pid=30738
Child of Ozarchuk!
[ozarchuk_anna@vpsj3IeQ ~]$ █
```

```

GNU nano 2.3.1                               File: create_any.c

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

extern char** environ;

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

```

### Завдання 3. Обмін сигналами між процесами

3.1 Створила 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.

Запустила створену С-програму.

3.2 Створила С-програму, яка надсилає сигнал SIGUSR2 процесу, запущеному в попередньому пункту завдання. Запустила створену С-програму та проаналізуйте повідомлення, які виводить перша програма. Завершила процес, запущеному в попередньому пункту завдання.

```

[ozarchuk_anna@vpsj3IeQ ~]$ ps -u ozarchuk_anna -o pid,stat,cmd
PID STAT CMD
4592 S+  nano set_signal_any.c
4855 S+  ./get_signal_any
4862 R+  ps -u ozarchuk_anna -o pid,stat,cmd
23293 S  sshd: ozarchuk_anna@pts/17
23294 Ss  -bash
24692 S  sshd: ozarchuk_anna@pts/35
24693 Ss  -bash
32023 S  sshd: ozarchuk_anna@pts/87
32025 Ss  -bash
[ozarchuk_anna@vpsj3IeQ ~]$ gcc set_signal_any.c -o set_signal_any
[ozarchuk_anna@vpsj3IeQ ~]$ ./set_signal_any
Send signal!
[ozarchuk_anna@vpsj3IeQ ~]$ █

```

```

[ozarchuk_anna@vpsj3IeQ ~]$ gcc get_signal_any.c -o get_signal_any

```

```

GNU nano 2.3.1 File: get_signal_anya.c

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

static void sig_usr(int signo)
{
    if (signo == SIGUSR2)
        printf("Process of Ozarchuk got signal!\n");
}

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

    return 0;
}

```

```

GNU nano 2.3.1 File: set_signal_anya.c

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

int main(void)
{
    pid_t pid = 4855;
    if (!kill(pid, SIGUSR2))
        printf("Send signal!\n");
    else
        fprintf(stderr, "Error!\n");

    return 0;
}

```

#### Завдання 4. Створення процесу-сироти

Я створила 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. Перевірила роботу програми, вивчила вміст таблиці процесів і зробила відповідні висновки.

```

GNU nano 2.3.1 File: sirota_anya.c

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

int main(void)
{
    int i;
    pid_t pid = fork();
    if (pid == 0)
    {
        printf("Ozarchuk child pid=%d\n", getpid());
        for (i=0; i<19; i++){
            printf("Child of Ozarchuk pid=%d\n, My parent=%d\n", getpid(), getppid());
            sleep(9);
        }
    }
    else {
        printf("Parent of Ozarchuk=%d\n", getpid());
        sleep(1);
        _exit(0);
    }
    return 0;
}

```

```
[ozarchuk_anna@vpsj3IeQ ~]$ gcc sirota_anya.c -o sirota_anya
[ozarchuk_anna@vpsj3IeQ ~]$ ./sirota_anya
Parent of Ozarchuk=10465
Ozarchuk child pid=10466
Child of Ozarchuk pid=10466
, My parent=10465
[ozarchuk_anna@vpsj3IeQ ~]$ Child of Ozarchuk pid=10466
, My parent=1
Child of Ozarchuk pid=10466
, My parent=1
Child of Ozarchuk pid=10466
, My parent=1
```

## Висновок

Під час виконання Лабораторної роботи №8 мною були отримані навички в управлінні процесами в ОС Unix на рівні мови програмування C.