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 3

Виконала:
Студентка групи АІ-202
Неживих М.О.
Перевірили:
Блажко О.А

Одеса 2021

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

Хід роботи

Завдання 1 Перегляд інформації про процес

Створюємо 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.

Завершуємо створення програми включенням функції sleep(5) для забезпечення засинання процесу на 5 секунд.

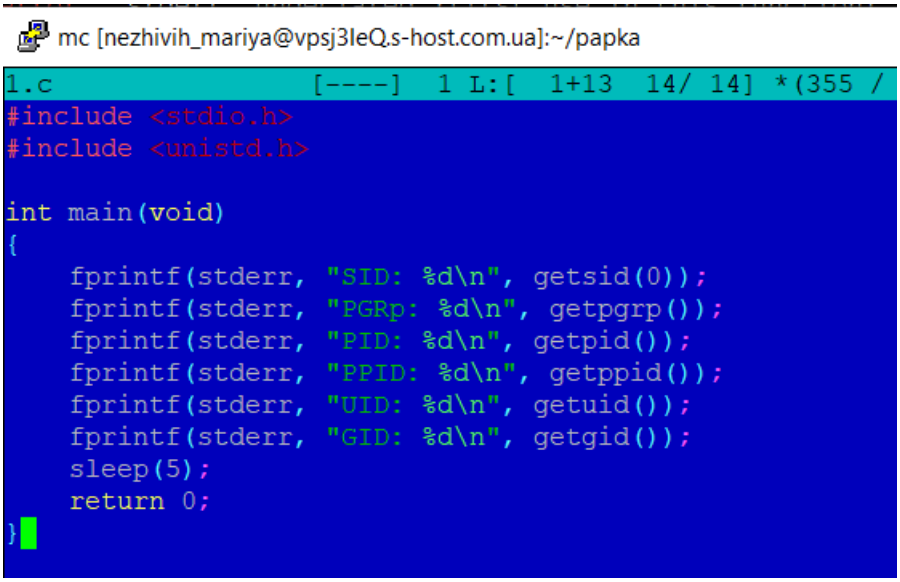
При створенні повідомлень використовуємо функцію fprintf з виведенням на потік помилок.

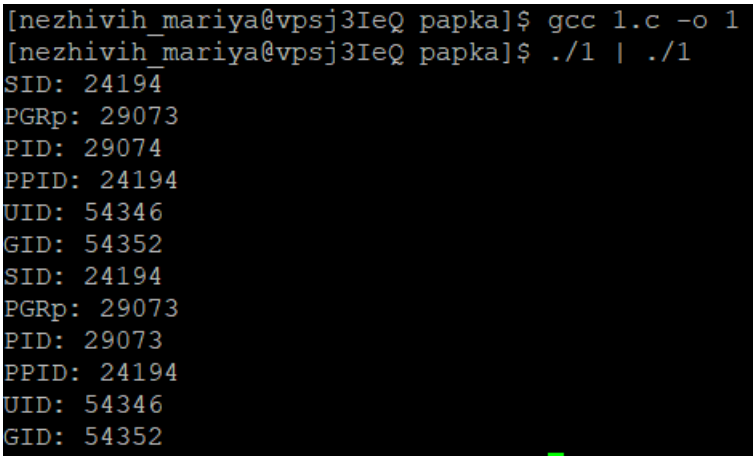
Після компіляції запускаємо програму.

Додатково запускаємо програму в конвеєрі, наприклад:

./info | ./info

Порівнюємо значення групи процесів.





<u>Завдання 2</u> Стандартне створення процесу

Створюємо 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 наше прізвище в транслітерації.

```
2.c                [----]  1 L:[  1+15  16/ 16] *(357 / 357b) <EOF>        [*
#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("parent of nezhivih pid=%d, child pid=%d\n",getpid(),pid);
<------>execve("/bin/echo",echo_args,environ);
<------>fprintf(stderr,"Error!\n");
    }
    return 0;
}
```

```
[nezhivih_mariya@vpsj3IeQ papka]$ gcc 2.c -o 2
[nezhivih_mariya@vpsj3IeQ papka]$ ./2
parent of nezhivih pid=31837, child pid=31838
I am ECHO
```

Завдання 3 Обмін сигналами між процесами

3.1 Створюємо 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 наше прізвище в транслітерації.

Запускаємо створену С-програму.

```
3.1.c              [----]  1 L:[  1+18  19/ 19] *(351 / 351b) <EOF>
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

static void sig_usr(int signo)
{
    if (signo == SIGUSR1)
    fprintf(stderr,"proces of nezhivih got signal %d\n",SIGUSR1);
}

int main(void)
{
    printf("pid=%d\n",getpid());
    if(signal(SIGUSR1, sig_usr) == SIG_ERR)
<------>fprintf(stderr, "Error!");
    for ( ; ; )
<------>pause();
    return 0;
}
```

```
[nezhivih_mariya@vpsj3IeQ papka]$ gcc 3.1.c -o 3.1
[nezhivih_mariya@vpsj3IeQ papka]$ ./3.1
pid=4511
proces of nezhivih got signal 10
proces of nezhivih got signal 10
proces of nezhivih got signal 10
proces of nezhivih got signal 10
proces of nezhivih got signal 10
```

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

Запускаємо створену С-програму та аналізуємо повідомлення, які виводить перша програма.

Завершаємо процес, запущений в попередньому пункту завдання.

```
3.2.c                [----]  1 L:[  1+12  13/
#include <stdio.h>
#include <signal.h>

pid_t pid = 4511;

int main(void)
{
    if (!kill(pid, SIGUSR1))
<------>printf("OK!\n");
    else
<------>fprintf(stderr, "Error!");
    return 0;
}
```

```
[nezhivih_mariya@vpsj3IeQ papka]$ gcc 3.2.c -o 3.2
[nezhivih_mariya@vpsj3IeQ papka]$ ./3.2
OK!
[nezhivih_mariya@vpsj3IeQ papka]$
[nezhivih_mariya@vpsj3IeQ papka]$ ./3.2
OK!
[nezhivih_mariya@vpsj3IeQ papka]$ ./3.2
OK!
[nezhivih_mariya@vpsj3IeQ papka]$ ./3.2
OK!
[nezhivih_mariya@vpsj3IeQ papka]$ ./3.2
OK!
```

<u>Завдання 4</u> Створення процесу-сироти

Створюємо 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.

```
4.c                 [----]  1 L:[  1+23  24/ 24] *(403 / 403b) <EOF>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(void)
{
    int i;
    pid_t pid = fork();
    if ( pid != 0)
    {
<------>fprintf(stderr,"Parent od nezhivih pid=%d ppid=%d \n", getpid(),getppid());
<------>sleep(6);
<------>_exit(0);
    }
    else
    {
<------>for (i=0; i<11; i++)
<------>{
<------>    fprintf(stderr"child of nezhivih pid=%d ppid: %d\n",getpid(), getppid());
<------>    sleep(1);
<------>}
    }
    return 0;
}
```

```
[nezhivih_mariya@vpsj3IeQ papka]$ gcc 4.c -o 4
[nezhivih_mariya@vpsj3IeQ papka]$ ./4
Parent od nezhivih pid=5955 ppid=24194
child of nezhivih pid=5956 ppid: 5955
child of nezhivih pid=5956 ppid: 5955
child of nezhivih pid=5956 ppid: 5955
child of nezhivih pid=5956 ppid: 5955
child of nezhivih pid=5956 ppid: 5955
child of nezhivih pid=5956 ppid: 5955
[nezhivih_mariya@vpsj3IeQ papka]$ child of nezhivih pid=5956 ppid: 1
child of nezhivih pid=5956 ppid: 1
child of nezhivih pid=5956 ppid: 1
child of nezhivih pid=5956 ppid: 1
```

Висновки: У ході лабораторної роботи були отриманні практичні навички в управлінні процесами в ОС Unix на рівні мови програмування С.