ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНИЧНИЙ УНІВЕРСИТЕТ
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА «ІНФОРМАЦІЙНИ СИСТЕМИ»

Лабораторна робота №8

з дисципліни

«Операційні системи»

Тема

«Програмування керуванням процесами в ОС Unix»

Виконав:

Студент групи АІ-202

Узун Михайло

Одеса 2021

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

Завдання:

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, який викликав цю функцію;

− ідентифікатор групи користувача процесу, який викликав цю функцію.

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

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

Після компіляції запустіть програму.

Додатково запустіть програму в конвеєрі, наприклад:

./info | ./info

Порівняйте значення групи процесів.

Завдання 2 Стандартне створення процесу

Створіть 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.

Завдання 3 Обмін сигналами між процесами

3.1 Створіть 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.

Запустіть створену С-програму.

3.2 Створіть С-програму, яка надсилає сигнал SIGUSR2 процесу, запущеному в

попередньому пункту завдання.

Запустіть створену С-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.

Хід роботи:

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, який викликав цю функцію;

− ідентифікатор групи користувача процесу, який викликав цю функцію.

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

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

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

int main (void) {
    fprintf(stderr, "Process! gpid=%d\n", getpgrp());
    fprintf(stderr, "sid=%d\n", getsid(0));
    fprintf(stderr, "pid=%d\n", getpid());
    fprintf(stderr, "ppid=%d\n", getppid());
    fprintf(stderr, "uid=%d\n", getuid());
    fprintf(stderr, "guid=%d\n", getgid());
    sleep(5);
    return 0;
}
```

Після компіляції запустіть програму.

```
[uzun_mihajlo@vpsj3IeQ ~]$ gcc info.c -o info
[uzun_mihajlo@vpsj3IeQ ~]$ ./info
Process! gpid=28595
sid=22975
pid=28595
ppid=22975
uid=54356
guid=54362
```

Додатково запустіть програму в конвеєрі, наприклад:

./info | ./info

```
[uzun_mihajlo@vpsj3IeQ ~]$ ./info | ./info
Process! gpid=28661
sid=22975
pid=28662
ppid=22975
uid=54356
guid=54362
Process! gpid=28661
sid=22975
pid=28661
ppid=22975
uid=54356
guid=54362
```

Порівняйте значення групи процесів.

Завдання 2 Стандартне створення процесу

Створіть 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.

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

extern char** environ;

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

```
[uzun_mihajlo@vpsj3IeQ ~]$ gcc create.c -o create
[uzun_mihajlo@vpsj3IeQ ~]$ ./create
Parent of Uzun. pid=32433, child pid=32434
Child of Uzun.

[uzun_mihajlo@vpsj3IeQ ~]$
```

Завдання 3 Обмін сигналами між процесами

3.1 Створіть 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.

```
 GNU nano 2.3.1            File: get_signal.c

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

static void sig_usr(int signo) {
    if (signo == SIGUSR2)
        fprintf(stderr, "Process of Uzun fot signal %d\n", SIGUSR2);
}

int main(void) {
    printf("pid=%d\n", getpid());
    if (signal(SIGUSR2, sig_usr) == SIG_ERR)
        fprintf(stderr, "Error\n");
    for ( ; ; )
        pause();
    return 0;
```

Запустіть створену С-програму.

```
[uzun_mihajlo@vpsj3IeQ ~]$ gcc get_signal.c -o get_signal
[uzun_mihajlo@vpsj3IeQ ~]$ ./get_signal
pid=9022
```

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

```
  GNU nano 2.3.1            File: send_signal.c

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

pid_t pid = 9022;

int main(void) {
    if (!kill(pid, SIGUSR2))
        printf("OK!\n");
    else
        fprintf(stderr, "Error\n");
    return 0;
}
```

Запустіть створену С-програму та проаналізуйте повідомлення, які виводить перша програма.

Завершіть процес, запущеному в попередньому пункту завдання.

```
[uzun_mihajlo@vpsj3IeQ ~]$ gcc get_signal.c -o get_signal
[uzun_mihajlo@vpsj3IeQ ~]$ ./get_signal
pid=9022
Process of Uzun fot signal 12
```

```
[uzun_mihajlo@vpsj3IeQ ~]$ gcc send_signal.c -o send_signal
[uzun_mihajlo@vpsj3IeQ ~]$ ./send_signal
OK!
[uzun_mihajlo@vpsj3IeQ ~]$
```

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.

```
  GNU nano 2.3.1                File: sirota.c

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

int main (void) {
    int i;
    pid_t pid = fork();
    if (pid != 0) {
        fprintf(stderr, "I am parent. pid=%d, child pid=%d\n", getpid(), pid);
        sleep(11);
        _exit(0);
    } else {
        for (i = 0; i < 21; i++) {
                printf("I am child with pid=%d. My parent pid =%d\n", getpid(), getppid());
                sleep(1);
        }
    }
    return 0;
}
```

Значення n – номер команди студента + номер студента в команді.

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

```
[uzun_mihajlo@vpsj3IeQ ~]$ gcc sirota.c -o sirota
[uzun_mihajlo@vpsj3IeQ ~]$ ./sirota
I am parent. pid=10030, child pid=10031
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
I am child with pid=10031. My parent pid =10030
[uzun_mihajlo@vpsj3IeQ ~]$ I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
I am child with pid=10031. My parent pid =1
```

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.