ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІЕРСИТЕТ

Інститут комп’ютерних систем

Кафедра інформаційних систем

Лабораторна робота №8

З дисципліни «Операційні системи»

Тема: «Програмування керуванням процесами в ОС Unix»

Виконав:

Студент групи АІ-202

Полянський М.О.

Перевірили:

Блажко О. А.

Дрозд М.О.

Одесса 2021

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

програмування С.

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.

```
info.c              [----] 28 L:[  1+ 5   6/ 13] *(1
#include <stdio.h>
#include <unistd.h>

int main(void) {
    fprintf(stderr,"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,"nid=%d\n",getuid());
    fprintf(stderr,"gid=%d\n",getgid());
    return 0;

}
```

```
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ gcc info.c -o info
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ ./info | ./info
gpid=27636
sid=25355
pid=27637
ppid=25355
nid=54357
gid=54363
gpid=27636
sid=25355
pid=27636
ppid=25355
nid=54357
gid=54363
```

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.

```
create.c            [----] 46 L:[  1+ 9  10/ 12] *(231 / 263b) 0034 0x0
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void) {
    pid_t pid = fork();
    if (pid == 0)
<------>printf("I am child! pid=%d, ppid=%d\n",getpid(),getppid());
    else
<------>printf("I am parent! pid=%d, ppid=%d\n",getpid(),pid);
    return 0;
}
```

```
ikola@vpsj3IeQ Кількість прибутих]$ gcc create.c -o create
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ ./create
I am parent! pid=30361, ppid=30362
I am child! pid=30362, ppid=30361
```

```
create.c            [----] 66 L:[  1+10  11/ 16] *(261 / 354b) 0041 0x029
#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("I am parent! pid=%d, child  pid=%d\n",getpid(),pid);
<------>execve("/bin/echo",echo_args,environ);
<------>fprintf(stderr,"Error!\n");
    }
    return 0;
}
```

```
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ gcc create.c -o create
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ ./create
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ I am parent! pid=31690, child
pid=0
I am ECHO!
```

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.

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

static void sig_usr(int signo) {
    if (signo == SIGUSR1 )
------>fprintf(stderr,"Got signal %d\n",SIGUSR1);
}

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

```
12477 12367 S+   ./get_signal
12561 11264 R+   ps -u polyanskij_mikola -o pid,ppid,stat,cmd
```

```
[polyanskij_mikola@vpsj3IeQ laba_8]$ gcc get_signal.c -o get_signal
[polyanskij_mikola@vpsj3IeQ laba_8]$ ./get_signal
```

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.

```
send_signal.c      [----] 27 L:[  1+ 6
```

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

pid_t pid = 12477;

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

```
[polyanskij_mikola@vpsj3IeQ laba_8]$ gcc send_signal.c -o send_signal
[polyanskij_mikola@vpsj3IeQ laba_8]$ ./send_signal
OK!
```

```
[polyanskij_mikola@vpsj3IeQ laba_8]$  gcc get_signal.c -o get_signal
[polyanskij_mikola@vpsj3IeQ laba_8]$ ./get_signal
Got signal 10
Got signal 10
Got signal 10
```

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.

```
sirota.c          [----] 13 L:[  1+19  20/ 21] *(433 / 435b) 0010 0x00A                [*][X]
#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 of Polynskiy pid=%d\n",getpid(),pid);
<------>sleep(10);
<------>_exit(0);
    }
    else {
<------>for (i=0;i<19;i++){
<------>    fprintf(stderr,"I am child of Polynskiy with pid=%d. My parent of Polyanskiy pid = %d\n"
<------><------>,getpid(),getppid());
<------>    sleep(1);
<------>}
    }
    return 0;
```

```
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ gcc sirota.c -o sirota
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ ./sirota
I am parent! pid=9175, child of Polynskiy pid=9176
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 9175
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$ I am child of Polynskiy with pi
d=9176. My parent of Polyanskiy pid = 1
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 1
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 1
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 1
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 1
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 1
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 1
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 1
I am child of Polynskiy with pid=9176. My parent of Polyanskiy pid = 1
[polyanskij_mikola@vpsj3IeQ Кількість прибутих]$
```

Висновок: в ході лабораторної роботи отримали навички в управлінні процесами в ОС Unix на рівні мови