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»

Виконав:

Студент групи АІ-202

Боровець Владислав Григорович

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.

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

int main (void){
fprintf(stderr,"I am 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,"gid=%d\n",getgid());
sleep(5);
return 0;

}
```

```
I am process! gpid=27863
sid=25815
pid=27863
ppid=25815
uid=54343
gid=54349
```

Додатково запустіть програму в конвеєрі, наприклад:

./info | ./info

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

int main (void){
fprintf(stderr,"I am 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,"gid=%d\n",getgid());
sleep(5);
return 0;

}
```

```
[borovets_vladislav@vpsj3IeQ ~]$ ./info ; ./info
I am process! gpid=27927
sid=25815
pid=27927
ppid=25815
uid=54343
gid=54349
I am process! gpid=27928
sid=25815
pid=27928
ppid=25815
uid=54343
gid=54349
```

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

```
[borovets_vladislav@vpsj3IeQ ~]$ ./create
Parent of Borovets! pid=31550, child pid=31551
Child of Borovets
```

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>
#include <unistd.h>


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

```
[borovets_vladislav@vpsj3IeQ ~]$ gcc get_signal.c -o get_signal
[borovets_vladislav@vpsj3IeQ ~]$ ./get_signal
pid=3362
```

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.

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

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

pid_t pid = 3362;

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

}
```

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.

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

int main (viod) {
int i;
pid_t pid = fork();
if(pid != 0){
fprintf(stderr,"I am parent! pid=%d, child pid=%d\n",getpid(),pid);
sleep(3);
_exit(0);
}
else{
for(i=0;i<5;i++){
fprintf(stderr,"I am child with pid=%d. My parent pid = %d\n",getpid(),getppid());
sleep(5);
 }
}
```

[Read 21 lines]

```
[borovets_vladislav@vpsj3IeQ ~]$ ./sirota
I am parent! pid=6045, child pid=6046
I am child with pid=6046. My parent pid = 6045
[borovets_vladislav@vpsj3IeQ ~]$ I am child with pid=6046. My parent pid = 1
I am child with pid=6046. My parent pid = 1
I am child with pid=6046. My parent pid = 1
I am child with pid=6046. My parent pid = 1
```

Висновок: отримали навичкі управіння процесами в ОС Unix на рівні мови програмування С.