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 1-го курсу
Гр. АІ-205
Щербаков Г.С.
Перевірили:
Блажко О.А.

Одеса 2021

Мета роботи: отримання навичок в управлінні процесами в ОС

Unix на рівні мови програмування С.

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, що викликав цю функцію
− ідентифікатор групи процесу, що викликав цю функцію.

Завдання 2 Стандартне породження процесу
Створіть С-програму, яка породжує процес-нащадок. У програмі процес-батько повинен видати повідомлення типу «Parent of Ivanov», а процес-нащадок повинен видати
повідомлення типу «Child of Ivanov», 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 SIGUSR1 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.

Виконання завдань:

Завдання 1.

```
  GNU nano 2.3.1                          File: info.c

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

int main(void){
  pid_t getpgrp(void);
  pid_t getpid(void);
  pid_t getsid(pid_t pid);
  pid_t getppid(void);
  uid_t getuid(void);
  gid_t getgid(void);
  printf("My sid =%d\n", getsid(0));
  printf("My pgrp =%d\n", getpgrp());
  printf("My pid =%d\n", getpid());
  printf("My ppid =%d\n", getppid());
  printf("My uid =%d\n", getuid());
  printf("My gid =%d\n", getgid());

}
```

```
[sherbakov_georgij@vpsj3IeQ ~]$ ./info
My sid =26656
My pgrp =27832
My pid =27832
My ppid =26656
My uid =54411
My gid =54417
```

Завдання 2.

```
[sherbakov_georgij@vpsj3IeQ ~]$ ./fork
Parent of Sherbakov
Child of Sherbakov
I am ECHO!
```

```
  GNU nano 2.3.1                          File: fork.c

#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("Child of Sherbakov\n");
   else{
      printf("Parent of Sherbakov\n");
      execve("/bin/echo", echo_args, environ);
   }
 return 0;
}
```

Завдання 3.1.

```
  GNU nano 2.3.1                       File: get_signal.c

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

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

int main(void) {
  if(signal(SIGUSR2, sig_usr) == SIG_ERR)
      fprintf(stderr, "Error!");
  for( ; ; )
    pause();
  return 1;
}
```

```
[sherbakov_georgij@vpsj3IeQ ~]$ touch get_signal.c
[sherbakov_georgij@vpsj3IeQ ~]$ nano get_signal.c
[sherbakov_georgij@vpsj3IeQ ~]$ gcc get_signal.c -o get_signal
[sherbakov_georgij@vpsj3IeQ ~]$ ./get_signal
[sherbakov_georgij@vpsj3IeQ ~]$ ps -u sherbakov_georgij -o pid,stat,cmd
  PID STAT CMD
26655 S    sshd: sherbakov_georgij@pts/0
26656 Ss   -bash
29234 S+   ./get_signal
29398 S    sshd: sherbakov_georgij@pts/3
29399 Ss   -bash
29500 R+   ps -u sherbakov_georgij -o pid,stat,cmd
[sherbakov_georgij@vpsj3IeQ ~]$
```

Завдання 3.2.

```
[sherbakov_georgij@vpsj3IeQ ~]$ ./send_signal
Sent signal.
[sherbakov_georgij@vpsj3IeQ ~]$
[sherbakov_georgij@vpsj3IeQ ~]$ ps -u sherbakov_georgij -o pid,stat,cmd
  PID STAT CMD
26655 S    sshd: sherbakov_georgij@pts/0
26656 Ss+  -bash
29234 T    ./get_signal
29398 S    sshd: sherbakov_georgij@pts/3
29399 Ss   -bash
30840 T    ./get_signal
31202 R+   ps -u sherbakov_georgij -o pid,stat,cmd
[sherbakov_georgij@vpsj3IeQ ~]$
```

```
 GNU nano 2.3.1                    File: send_signal.c

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

pid_t pid = 29234;
int main(void) {
 if(!kill(pid,SIGUSR1))
    printf("Sent signal.\n");
 else
    fprintf(stderr, "Error!");
return 1;
}
```

Завдання 4.

```
 GNU nano 2.3.1                      File: task4.c

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

int main(void)
{
 int i;
 pid_t pid = fork();
 if (pid != 0) {
  printf("Parent of Sherbakov with ppid= %d", getppid());
  sleep(6);
  _exit(0);
}
else {
 for(i=0; i<11; i++){
   printf("Child of Sherbakov\n");
}
}
return 0;
}
```

```
[[sherbakov_georgij@vpsj3IeQ ~]$ ./task4
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov
Child of Sherbakov

[sherbakov_georgij@vpsj3IeQ ~]$
```

Висновок: у ході виконання цієї лабораторної роботи я надбав навички з управління процесами в ОС Unix засобами командної оболонки. При виконанні завдань ніяких труднощів не виникло.