ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА «ІНФОРМАЦІЙНІ СИСТЕМИ»

Лабораторна робота №8

З дисципліни

«Операційні системи»

Тема

«Програмування керуванням процесами в ОС Unix»

Виконав:

Студент групи АІ-203
Вояковський Д. П.
Перевірили:
Дрозд М.О. Блажко О.А.

Одеса 2021

Завдання для виконання

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. Процеснащадок 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.

1

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

int main(void) {
....
    printf("pid=%d\n", getpid());
    printf("ppid=%d\n", getppid());
    printf("uid=%d\n", getuid());
    printf("gid=%d\n", getgid());
    printf("pgrp=%d\n", getpgrp());
    printf("sid=%d\n", getsid(0));
....
    return 0;
}
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ gcc info.c -o info
[voyakovskij_dmitro@vpsj3IeQ ~]$ ./info
pid=26053
ppid=24907
uid=54363
gid=54369
pgrp=26053
sid=24907
```

2

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

int main(void)
{
    pid_t pid = fork();
    if (pid == 0) printf("Child of Voyakovskiy pid=%d, ppid=%d\n",getpid(),getppid());
    else printf("Parent of Voyakovskiy pid=%d, ppid=%d, child pid=%d\n",getpid(),getppid(),pid);
....
    return 0;
}
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ gcc create.c -o create
[voyakovskij_dmitro@vpsj3IeQ ~]$ ./create
Parent of Voyakovskiy pid=27871, ppid=27508, child pid=27872
Child of Voyakovskiy pid=27872, ppid=27871
```

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

extern char ** environ;

int main(void)
{
    char* echo[] = {"echo","It's echo",NULL};
    pid_t pid = fork();
    if (pid == 0) printf("Child of Voyakovskiy\n");
    else
    {
<------>printf("Parent of Voyakovskiy\n");
<------>execve("/bin/echo",echo,environ);
<------>fprintf(stderr, "Error");
<------>return 1;
    }
....
    return 0;
}
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ gcc create.c -o create
[voyakovskij_dmitro@vpsj3IeQ ~]$ ./create
Parent of Voyakovskiy
Child of Voyakovskiy
It's echo
```

3

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

static void sig_usr(int signo)
{
    if (signo == SIGUSR2) printf("Process of Voyakovskiy got signal\n");
}
int main(void)
{
------>if (signal(SIGUSR2, sig_usr) == SIG_ERR)
------>    fprintf(stderr,"Неможливо перехопити сигнал SIGUSR2\n");
------>for( ; ; ) pause();
}
```

```c
#include <errno.h>
#include <signal.h>
#include <stdio.h>
pid_t pid=3237;
int main(void)
{
    if (!kill(pid,SIGUSR2))
------>printf("Signal sent\n");
    else fprintf(stderr,"Error\n");
}
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ gcc get_signal.c -o get_signal
[voyakovskij_dmitro@vpsj3IeQ ~]$ ./get_signal
Process of Voyakovskiy got signal
Process of Voyakovskiy got signal

[voyakovskij_dmitro@vpsj3IeQ ~]$ ps -u voyakovskij_dmitro -o pid,stat,cmd
  PID STAT CMD
 1810 S    sshd: voyakovskij_dmitro@pts/2
 1811 Ss   -bash
 3237 S+   ./get_signal
 3283 R+   ps -u voyakovskij_dmitro -o pid,stat,cmd
27473 S    sshd: voyakovskij_dmitro@pts/0
27474 Ss   -bash
27506 S+   /usr/bin/mc -P /tmp/mc-voyakovskij_dmitro/mc.pwd.27474
27508 Ss   bash -rcfile .bashrc
[voyakovskij_dmitro@vpsj3IeQ ~]$ gcc send_signal.c -o send_signal
[voyakovskij_dmitro@vpsj3IeQ ~]$ ./send_signal
Signal sent
[voyakovskij_dmitro@vpsj3IeQ ~]$ ./send_signal
Signal sent
```

4

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

int main(void)
{
    int i;
    pid_t pid = fork();
    if (pid != 0)
    {
------>printf("I am parent pid=%d\n", getpid());
------>sleep(4);
------>_exit(0);
    }
    else
    {
------>for (i=0;i<7;i++)
------>{
------>    printf("Child of Voyakovskiy ppid=%d\n", getppid());
------>    sleep(1);
------>}
    }
return 0;
}
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ gcc orphan.c -o orphan
[voyakovskij_dmitro@vpsj3IeQ ~]$ ./orphan
I am parent pid=7327
Child of Voyakovskiy ppid=7327
Child of Voyakovskiy ppid=7327
Child of Voyakovskiy ppid=7327
Child of Voyakovskiy ppid=7327
Child of Voyakovskiy ppid=1
[voyakovskij_dmitro@vpsj3IeQ ~]$ Child of Voyakovskiy ppid=1
Child of Voyakovskiy ppid=1
```

5

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

void sighandler (void )
{
    printf("I am Zombie-process of Voyakovskiy\n");
    wait(0);
}

int main(void)
{
    int i;
    sigset (SIGCHLD, &sighandler);
    pid_t pid = fork();
    if (pid == 0) _exit(0);
    else
    {
------>fprintf(stdout,"Start\n");
------>sleep(2);
------>fprintf(stdout,"Finish\n");
    }
    return 0;
}
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$ gcc zombie.c -o zombie
[voyakovskij_dmitro@vpsj3IeQ ~]$ ./zombie
Start
I am Zombie-process of Voyakovskiy
Finish

[voyakovskij_dmitro@vpsj3IeQ ~]$ ps -u voyakovskij_dmitro -o pid,ppid,stat,cmd
  PID  PPID STAT CMD
 1810  1772 S    sshd: voyakovskij_dmitro@pts/2
 1811  1810 Ss+  -bash
 9602  9578 S    sshd: voyakovskij_dmitro@pts/3
 9603  9602 Ss   -bash
12268 27508 S    ./zombie
12269 12268 Z    [zombie] <defunct>
```

6

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

void sighandler ( int sig )
{
    printf("Signal handler for signal = %d", sig);
    wait(0);
}

int main(void)
{
    sigset (SIGCHLD, &sighandler);
    pid_t pid = fork();
    if (pid == 0)
    {
<------>printf("Child of Voyakovskiy is finished");
<------>_exit(0);
    }
    else
    {
<------>fprintf(stderr,"\nStart");
<------>sleep(9);
<------>fprintf(stderr,"\nFinish\n");
    }
    return 0;
}
```

```
[voyakovskij_dmitro@vpsj3IeQ ~]$
Start
Finish
Signal handler for signal = 17
[voyakovskij_dmitro@vpsj3IeQ ~]$ ps -u voyakovskij_dmitro -o pid,ppid,stat,cmd
  PID  PPID STAT CMD
 1810  1772 S    sshd: voyakovskij_dmitro@pts/2
 1811  1810 Ss+  -bash
 9602  9578 S    sshd: voyakovskij_dmitro@pts/3
 9603  9602 Ss   -bash
13359  9603 R+   ps -u voyakovskij_dmitro -o pid,ppid,stat,cmd
27473 27433 S    sshd: voyakovskij_dmitro@pts/0
27474 27473 Ss   -bash
27506 27474 S+   /usr/bin/mc -P /tmp/mc-voyakovskij_dmitro/mc.pwd.27474
27508 27506 Ss+  bash -rcfile .bashrc
```

Висновки: під час виконання лабораторної роботи усі завдання були помірної складності.