

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних систем
Кафедра «Інформаційних систем»

Лабораторна робота №8

З дисципліни: «Операційні системи»

Тема: «Програмування керування процесами в ОС Unix»

Виконав:

Студент групи AI-205

Кучеренко М.М.

Перевірили:

Блажко О.А.

Дрозд М.О.

Одеса 2021

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

Завдання до виконання

Завдання 1 Перегляд інформації про процес

Створіть C-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.

Завдання 2 Стандартне створення процесу

Створіть C-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 C-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 C-програму.

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

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

Завершіть процес, запущеному в попередньому пункту завдання.

Завдання 4 Створення процесу-сироти

Створіть C-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.

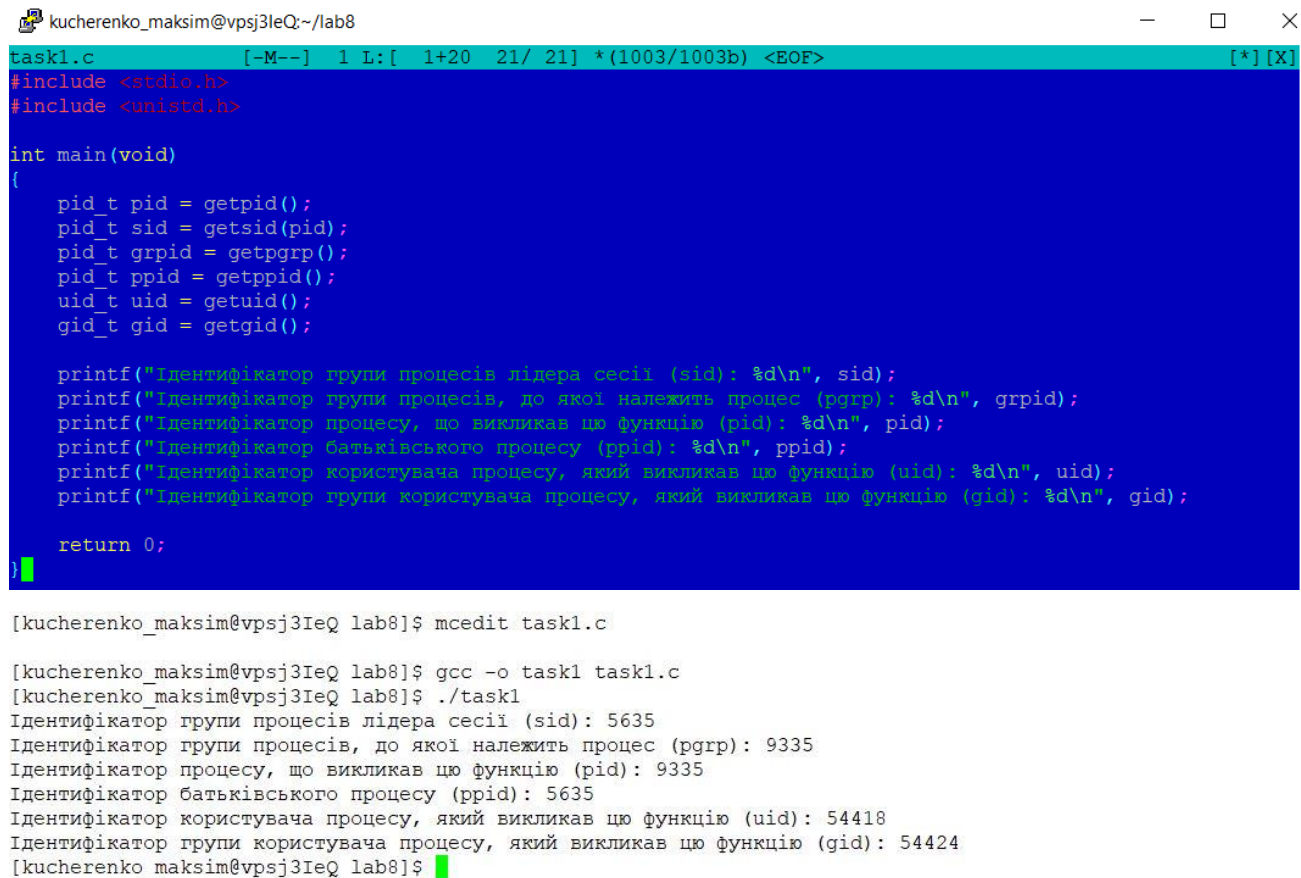
Процес-нащадок повинен виводити повідомлення, наприклад, «Child of Ivanov is finished», за шаблоном як в попередньому завданні.

Процес-батько повинен очікувати (3*n) секунд.

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

Запустіть програму у фоновому режимі, а в окремому терміналі вивчіть вміст таблиці процесів і зробіть відповідні висновки.

Скріншоти виконання завдань:



```
kucherenko_maksim@vpsj3IeQ:~/lab8
task1.c [-M--] 1 L:[ 1+20 21/ 21] *(1003/1003b) <EOF> [*][X]
#include <stdio.h>
#include <unistd.h>

int main(void)
{
    pid_t pid = getpid();
    pid_t sid = getsid(pid);
    pid_t grpid = getpgrp();
    pid_t ppid = getppid();
    uid_t uid = getuid();
    gid_t gid = getgid();

    printf("Ідентифікатор групи процесів лідера cecii (sid): %d\n", sid);
    printf("Ідентифікатор групи процесів, до якої належить процес (pgrp): %d\n", grpid);
    printf("Ідентифікатор процесу, що викликав цю функцію (pid): %d\n", pid);
    printf("Ідентифікатор батьківського процесу (ppid): %d\n", ppid);
    printf("Ідентифікатор користувача процесу, який викликав цю функцію (uid): %d\n", uid);
    printf("Ідентифікатор групи користувача процесу, який викликав цю функцію (gid): %d\n", gid);

    return 0;
}

[kucherenko_maksim@vpsj3IeQ lab8]$ mcedit task1.c

[kucherenko_maksim@vpsj3IeQ lab8]$ gcc -o task1 task1.c
[kucherenko_maksim@vpsj3IeQ lab8]$ ./task1
Ідентифікатор групи процесів лідера cecii (sid): 5635
Ідентифікатор групи процесів, до якої належить процес (pgrp): 9335
Ідентифікатор процесу, що викликав цю функцію (pid): 9335
Ідентифікатор батьківського процесу (ppid): 5635
Ідентифікатор користувача процесу, який викликав цю функцію (uid): 54418
Ідентифікатор групи користувача процесу, який викликав цю функцію (gid): 54424
[kucherenko_maksim@vpsj3IeQ lab8]$
```

kucherenko_maksim@vpsj3leQ:~/lab8

```
task2.c [----] 1 L:[ 1+20 21/ 21] *(387 / 387b) <EOF>
#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 Kucherenko: pid = %d\n", getpid());

    else {
<----->printf("Parent of Kucherenko: pid = %d\n", getpid());
<----->execve("/bin/echo", echo_args, environ);
    }

    return 0;
}
```

kucherenko_maksim@vpsj3leQ:~/lab8

```
[kucherenko_maksim@vpsj3leQ lab8]$ mcedit task2.c

[kucherenko_maksim@vpsj3leQ lab8]$ gcc -o task2 task2.c
[kucherenko_maksim@vpsj3leQ lab8]$ ./task2
Parent of Kucherenko: pid = 10996
Child of Kucherenko: pid = 10997
I am Echo

[kucherenko_maksim@vpsj3leQ lab8]$
```

kucherenko_maksim@vpsj3leQ:~/lab8

```
task3_1.c [----] 1 L:[ 1+17 18/ 18] *(283 / 283b) <EOF>
#include <stdio.h>
#include <signal.h>

static void sig_usr(int signal)
{
    if (signal == SIGUSR2)
<----->printf("Process of Kucherenko got signal\n");
}

int main(void)
{
    if (signal(SIGUSR2, sig_usr) == SIG_ERR)
<----->fprintf(stderr, "Error.");
    for ( ; ; )
<----->pause();

    return 1;
}
```

kucherenko_maksim@vpsj3leQ:~

```
task3_2.c [----] 13 L:[ 1+13 14/ 15] *(197 / 199b) 0010 0x00A
#include <stdio.h>
#include <signal.h>

pid_t pid = 15786;

int main(void)
{
    if (!kill(pid, SIGUSR2))
<----->printf("Sent signal to pid = %d", pid);>

    else
<----->fprintf(stderr, "Error");

    return 1;
}
```

kucherenko_maksim@vpsj3IeQ:~/lab8

```
[kucherenko_maksim@vpsj3IeQ lab8]$ mcedit task3_1.c

[kucherenko_maksim@vpsj3IeQ lab8]$ gcc -o task3_1 task3_1.c
[kucherenko_maksim@vpsj3IeQ lab8]$ ./task3_1
Process of Kucherenko got signal
Process of Kucherenko got signal
Process of Kucherenko got signal
Terminated
[kucherenko_maksim@vpsj3IeQ lab8]$
```

kucherenko_maksim@vpsj3IeQ:~

```
[kucherenko_maksim@vpsj3IeQ ~]$ ps -u kucherenko_maksim -o pid,stat,cmd
  PID STAT CMD
 5634 S   sshd: kucherenko_maksim@pts/3
 5635 Ss  -bash
15817 S   sshd: kucherenko_maksim@pts/2
15818 Ss  -bash
16475 S+   ./task3_1
16503 R+   ps -u kucherenko_maksim -o pid,stat,cmd
[kucherenko_maksim@vpsj3IeQ ~]$ mcedit task3_2.c

[kucherenko_maksim@vpsj3IeQ ~]$ gcc -o task3_2 task3_2.c
[kucherenko_maksim@vpsj3IeQ ~]$ ./task3_2
Sent signal to pid = 16475[kucherenko_maksim@vpsj3IeQ ~]$ ./task3_2
Sent signal to pid = 16475[kucherenko_maksim@vpsj3IeQ ~]$ ./task3_2
Sent signal to pid = 16475[kucherenko_maksim@vpsj3IeQ ~]$ kill 16475
[kucherenko_maksim@vpsj3IeQ ~]$
```

kucherenko_maksim@vpsj3IeQ:~/lab8

```
task4.c [-----] 1 L: [ 1+23 24/ 24] *(413 / 413b) <EOF> [*][X]
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
    int i;
    pid_t pid = fork();
    ....
    if (pid != 0) {
<----->printf("I am parent with pid = %d. My child pid = %d\n", getpid(), pid);
<----->sleep(9);
<----->_exit(0);
    }
    ....
    else {
<----->for (i = 0; i < 16; i++) {
<----->    printf("I am child with pid = %d. My parent id = %d\n", getpid(), getppid());
<----->    sleep(1);
<----->}
    }
    ....
    return 0;
}
```

kucherenko_maksim@vpsj3IeQ:~/lab8

```
task5.c [-----] 43 L: [ 1+16 17/ 21] *(304 / 330b) 0034 0x022
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(void)
{
    pid_t pid = fork();

    if (pid == 0) {
<----->printf("I am child and I am Zombie-process of Kucherenko\n");
<----->_exit(0);
    }

    else {
<----->fprintf(stderr, "Parent start...\n");
<----->sleep(10);
<----->fprintf(stderr, "Parent finish...\n");
    }

    return 0;
}
```

```
kucherenko_maksim@vpsj3IeQ:~/lab8
[kucherenko_maksim@vpsj3IeQ lab8]$ mcedit task5.c

[kucherenko_maksim@vpsj3IeQ lab8]$ gcc -o task5 task5.c
[kucherenko_maksim@vpsj3IeQ lab8]$ ./task5 &
[1] 596
[kucherenko_maksim@vpsj3IeQ lab8]$ Parent start...
I am child and I am Zombie-process of Kucherenko
Parent finish...
█
```

```
kucherenko_maksim@vpsj3IeQ:~
[kucherenko_maksim@vpsj3IeQ ~]$ ps -u kucherenko_maksim -o pid,stat,cmd
PID STAT CMD
596 S ./task5
597 Z [task5] <defunct>
602 R+ ps -u kucherenko_maksim -o pid,stat,cmd
5634 S sshd: kucherenko_maksim@pts/3
5635 S+ -bash
31779 S sshd: kucherenko_maksim@pts/2
31780 Ss -bash
[kucherenko_maksim@vpsj3IeQ ~]$ █
```

```
kucherenko_maksim@vpsj3IeQ:~/lab8
task6.c [----] 1 L: [ 1+30 31/ 31] *(493 / 493b) <EOF>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

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

int main(void)
{
    int i;
    sigset(SIGCHLD, &sighandler);
    ....
    pid_t pid = fork();
    ....
    if (pid == 0) {
<----->printf("Child of Kucherenko is finished\n");
<----->_exit(0);
    }
    ....
    else {
<----->fprintf(stdout, "Parent start...\n");
<----->sleep(24);
<----->fprintf(stdout, "Parent finish...\n");
    }
    ....
    return 0;
}
█
```

```
kucherenko_maksim@vpsj3IeQ:~/lab8
[kucherenko_maksim@vpsj3IeQ lab8]$ gcc -o task6 task6.c
[kucherenko_maksim@vpsj3IeQ lab8]$ ./task6 &
[1] 4130
[kucherenko_maksim@vpsj3IeQ lab8]$ Parent start...
Child of Kucherenko is finished
Signal handler for signal = 17
Parent finish...
[kucherenko_maksim@vpsj3IeQ lab8]$ █
```

```
kucherenko_maksim@vpsj3IeQ:~
[kucherenko_maksim@vpsj3IeQ ~]$ ps -u kucherenko_maksim -o pid,stat,cmd
PID STAT CMD
4142 R+ ps -u kucherenko_maksim -o pid,stat,cmd
5634 S sshd: kucherenko_maksim@pts/3
5635 S+ -bash
31779 S sshd: kucherenko_maksim@pts/2
31780 Ss -bash
[kucherenko_maksim@vpsj3IeQ ~]$ █
```

Висновок: В ході лабораторної роботи були освоєні навички управління процесами в ОС Unix на рівні мови програмування C.