

Міністерство освіти і науки України
Одеський національний політехнічний університет
Інститут комп'ютерних систем
Кафедра інформаційних систем

Лабораторна робота №7
З дисципліни «Операційні системи»

Тема: «Програмування керуванням процесами в ОС Unix»

Виконав:
Ст. гр. АІ-204
Дорожкін Михайло

Перевірив(-ла):
Блажко О. А.
Дрозд М.О.

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, який викликав цю функцію;
- ідентифікатор групи користувача процесу, який викликав цю функцію.

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.

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 SIGUSR2 процесу, запущеному в попередньому пункті завдання. Запустіть створену С-програму та проаналізуйте повідомлення, які виводить перша програма.

Завершіть процес, запущеному в попередньому пункті 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.

Виконання роботи:

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.

```
mc [dorozhkin_mihajlo@vpsj3leQ.s-host.com.ua]:~/Lab8
task1.c      [----] 27 L:[ 1+ 9 10/ 13] * (218 / 275b) 0103 0x067  [*] [X]
#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;
}

dorozhkin_mihajlo@vpsj3leQ:~/Lab8
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc task1.c -o task1
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./task1
Pid=24668[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./task1
Pid=24842[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./task1
Pid=24866[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc task1.c -o task1
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./task1
Pid=25072
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc task1.c -o task1
/tmp/cc3tBskM.o: In function `main':
task1.c:(.text+0x36): undefined reference to `getpuid'
collect2: error: ld returned 1 exit status
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc task1.c -o task1
/tmp/ccsQ0lfs.o: In function `main':
task1.c:(.text+0x62): undefined reference to `getgrp'
collect2: error: ld returned 1 exit status
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc task1.c -o task1
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./task1
Pid=26445
Ppid=24365
Uid=54392
Gid=54398
Pgrp=26445
Sid=24365
[dorozhkin_mihajlo@vpsj3leQ Lab8]$
```

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.

The screenshot shows a terminal window with the following content:

```
mc [dorozhkin_mihajlo@vpsj3leQ.s-host.com.ua]:~/Lab8
create.c      [---] 37 L:[ 1+13 14/ 17] * (290 / 316b) 0034 0x022 [*][X]
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

extern char** environ;

int main(void)
{
    char* echo_args[] = {"echo", "Child of Dorozhkin\n", NULL};
    pid_t pid = fork();
    if (pid == 0)
        execve("/bin/echo", echo_args, environ);
    else {
        printf("Parent of Dorozhkin\n");
    }
    return 0;
}

lHelp  2Save  3Mark  4Replace 5Copy  6Move  7Search 8Delete 9PullDn 10Quit
dorozhkin_mihajlo@vpsj3leQ:~/Lab8
Parent of Dorozhkin

[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc create.c -o create
create.c: In function 'main':
create.c:16:9: warning: missing terminating " character [enabled by default]
    printf("Parent of Dorozhkin);
          ^
create.c:16:2: error: missing terminating " character
    printf("Parent of Dorozhkin);
          ^
create.c:17:6: error: expected expression before ')' token
    }
          ^
create.c:17:6: error: expected ';' before ')' token
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc create.c -o create
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./create
Parent of Dorozhkin[dorozhkin_mihajlo@vpsj3leQ Lab8]$ Child of Dorozhkin

gcc create.c -o create
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./create
Parent of Dorozhkin
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ Child of Dorozhkin
```

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.

The screenshot shows two terminal windows side-by-side. The left window displays a command-line session where the user is trying to compile a C program named 'set_signal.c' but receives errors because the file does not exist. The user then creates a directory 'Lab8' and tries again, but still gets errors due to the missing file. Finally, they run the command 'gcc get_signal.c -o get_signal' which results in compilation errors related to undeclared identifiers 'SIGUSER1' and 'SIG_ERR'. The right window shows the user running the command 'gcc create.c -o create', creating a child process, and then running 'gcc get_signal.c -o get_signal'.

```
mc [dorozhkin_mihajlo@vpsj3leQ.s-host.com.ua]:~/Lab8
21882 S      sshd: dorozhkin_mihajlo@pts/18
21883 Ss     -bash
23576 S+   /usr/bin/mc -P /tmp/mc-dorozhkin_mihajlo/mc.pwd.21883
23578 Ss     bash -rcfile .bashrc
24364 S      sshd: dorozhkin_mihajlo@pts/39
24365 Ss     -bash
31765 S+   ./get_signal
32626 R+   ps -u dorozhkin_mihajlo -o pid,stat,cmd
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc set_signal -o set_signal
gcc: error: set_signal: No such file or directory
gcc: fatal error: no input files
compilation terminated.
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ pushd Lab8
bash: pushd: Lab8: No such file or directory
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc set_signal.c -o set_signal
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./set_signal.c
bash: ./: Is a directory
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./set_signal.c
bash: ./set_signal.c: No such file or directory
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./set_signal
bash: ./set_signal: No such file or directory
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./set_signal
Send signal
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ 

dorozhkin_mihajlo@vpsj3leQ:~/Lab8
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./create
Parent of Dorozhkin[dorozhkin_mihajlo@vpsj3leQ Lab8]$ Child of Dorozhkin

gcc create.c -o create
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./create
Parent of Dorozhkin
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ Child of Dorozhkin

[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc get_signal.c -o get_signal
get_signal.c: In function 'sig_usr':
get_signal.c:6:17: error: 'SIGUSER1' undeclared (first use in this function)
    if(signo == SIGUSER1)
        ^
get_signal.c:6:17: note: each undeclared identifier is reported only once for ea
ch function it appears in
get_signal.c: In function 'main':
get_signal.c:12:16: error: 'SIGUSER1' undeclared (first use in this function)
    if (signal(SIGUSER1, sig_usr) == SIG_ERR)
        ^
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc get_signal.c -o get_signal
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./get_signal
Got signal SIGUSER1
```

3.2 Створіть C-програму, яка надсилає сигнал SIGUSR2 процесу, запущеному в попередньому пункті завдання. Запустіть створену C-програму та проаналізуйте повідомлення, які виводить перша програма.

Завершіть процес, запущеному в попередньому пункті 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.

The screenshot shows a terminal window titled 'mc [dorozhkin_mihajlo@vpsj3leQ.s-host.com.ua]:~/Lab8'. The window contains the source code for 'sirota.c' and its execution output.

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

int main(void)
{
    int i;
    pid_t pid = fork();
    if (pid == 0)
    {
        printf("Child pid=%d\n", getpid());
        for (i=0;i<5;i++)
        {
            printf("Im child. My parent id = %d\n", getppid());
            sleep(1);
        }
    }
    else
    {
        printf("Parent pid =%d\n", getpid());
        sleep(6);
        exit(0);
    }
}
```

The terminal output shows two runs of the program:

```
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ gcc sirota.c -o sirota
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./sirota
Parent pid =2204
Child pid=2205
Im child. My parent id = 2204
Im child. My parent id = 2204
Im child. My parent id = 1
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ Im child. My parent id = 1
Im child. My parent id = 1
gcc sirota.c -o sirota
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ ./sirota
Parent pid =2476
Child pid=2477
Im child. My parent id = 2476
[dorozhkin_mihajlo@vpsj3leQ Lab8]$ Im child. My parent id = 1
```