МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
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
Лукашак Д. О.

Перевірив:
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, 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 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 – 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:

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.

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


void main() {
    fprintf(stderr, "\nИдентификатор группы процессов лидера сессии - %d\n", getsid(0));
    fprintf(stderr, "Идентификатор группы процессов, к которой принадлежит процесс - %d\n", getpgrp());
    fprintf(stderr, "Идентификатор процесса - %d\n", getpid());
    fprintf(stderr, "Идентификатор родительского процесса - %d\n", getppid());
    fprintf(stderr, "Идентификатор пользователя процесса - %d\n", getuid());
    fprintf(stderr, "Идентификатор группы пользователя процесса - %d\n\n", getgid());

    sleep(5);
}
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ gcc -o get_info get_info.c
[lukashak_daniil@vpsj3IeQ C Codes]$ ./get_info

Идентификатор группы процессов лидера сессии - 28460
Идентификатор группы процессов, к которой принадлежит процесс - 28818
Идентификатор процесса - 28818
Идентификатор родительского процесса - 28460
Идентификатор пользователя процесса - 54352
Идентификатор группы пользователя процесса - 54358

[lukashak_daniil@vpsj3IeQ C Codes]$ ./get_info | ./get_info

Идентификатор группы процессов лидера сессии - 28460
Идентификатор группы процессов, к которой принадлежит процесс - 28926
Идентификатор процесса - 28927
Идентификатор родительского процесса - 28460
Идентификатор пользователя процесса - 54352
Идентификатор группы пользователя процесса - 54358


Идентификатор группы процессов лидера сессии - 28460
Идентификатор группы процессов, к которой принадлежит процесс - 28926
Идентификатор процесса - 28926
Идентификатор родительского процесса - 28460
Идентификатор пользователя процесса - 54352
Идентификатор группы пользователя процесса - 54358
```

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 Lukashak», а процес-нащадок повинен видати повідомлення типу «Child of Lukashak» через виклик команди echo.

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

extern char** environ;

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

    if (pid > 0)
        printf("\nParent of Lukashak\n");
    else if (!pid) {
        char* args[] = {"echo", "Child of Lukashak\n", NULL};
        execve("/bin/echo", args, environ);

        printf("Execve error");
        return -1;
    }
    else {
        fprintf(stderr, "Fork error");
        return -1;
    }

    return 0;
}
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ gcc -o "/home/lukashak_daniil/C Codes/create_child" "/home/lukashak_daniil/C Codes/create_child.c"
[lukashak_daniil@vpsj3IeQ C Codes]$ ./create_child

Parent of Lukashak
[lukashak_daniil@vpsj3IeQ C Codes]$ Child of Lukashak
```

Завдання 3. Обмін сигналами між процесами

3.1 Створіть С-програму, в якій процес очікує отримання сигналу SIGUSR2 та виводить повідомлення типу «Process of Lukashak got signal» після отримання сигналу. Запустіть створену С-програму.

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.

Receive:

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


void sig_receive(int signal) {
    if (signal == SIGUSR2)
        printf("Process of Lukashak got signal\n");
}


int main() {
    if (signal(SIGUSR2, sig_receive) == SIG_ERR) {
        fprintf(stderr, "Signal error");
        return -1;
    }

    pid_t pid = getpid();
    FILE* file = fopen("pid_task3", "wb");
    fwrite(&pid, sizeof(pid_t), 1, file);
    fclose(file);

    printf("\n");
    while (1) pause();

    return 0;
}
```

Send:

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


int main() {
    pid_t pid;
    FILE* file = fopen("pid_task3", "rb");
    fread(&pid, sizeof(pid_t), 1, file);
    fclose(file);

    if (!kill(pid, SIGUSR2)) {
        printf("\nSignal sent to process with pid = %d\n\n", pid);
        return 0;
    }
    else {
        fprintf(stderr, "Kill error");
        return -1;
    }
}
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ gcc -o "/home/lukashak_daniil/C Codes/signal_receive" "/home/lukashak_daniil/C Codes/signal_receive.c"
[lukashak_daniil@vpsj3IeQ C Codes]$ ./signal_receive

Process of Lukashak got signal

[lukashak_daniil@vpsj3IeQ C Codes]$ gcc -o "/home/lukashak_daniil/C Codes/signal_send" "/home/lukashak_daniil/C Codes/signal_send.c"
[lukashak_daniil@vpsj3IeQ C Codes]$ ./signal_send

Signal sent to process with pid = 5019
```

signal_send виконувався у другому терміналі, тому маємо два скріншота.

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 Lukashak» і додатково виводити PPID процесу-батька.

Значення n – 8.

Перевірте роботу програми, вивчіть вміст таблиці процесів і зробіть відповідні висновки.

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


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

    if (pid > 0)
        sleep(8 + 1);
    else if (!pid) {
        printf("\n");
        for (int i = 0; i < 2 * 8 + 1; i++) {
            printf("Child of Lukashak. PPID = %d\n", getppid());
            sleep(1);
        }
        printf("\n");
    }
    else {
        fprintf(stderr, "Fork error");
        return -1;
    }

    return 0;
}
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ gcc -std=c99 -o "/home/lukashak_daniil/C Codes/parent_stop" "/home/lukashak_daniil/C Codes/parent_stop.c"
[lukashak_daniil@vpsj3IeQ C Codes]$ ./parent_stop

Child of Lukashak. PPID = 11642
Child of Lukashak. PPID = 11642
Child of Lukashak. PPID = 11642
Child of Lukashak. PPID = 11642
Child of Lukashak. PPID = 11642
Child of Lukashak. PPID = 11642
Child of Lukashak. PPID = 11642
Child of Lukashak. PPID = 11642
Child of Lukashak. PPID = 11642
[lukashak_daniil@vpsj3IeQ C Codes]$ Child of Lukashak. PPID = 1
Child of Lukashak. PPID = 1
Child of Lukashak. PPID = 1
Child of Lukashak. PPID = 1
Child of Lukashak. PPID = 1
Child of Lukashak. PPID = 1
Child of Lukashak. PPID = 1
Child of Lukashak. PPID = 1
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
lukasha+  4205  0.0  0.1 115548  2132 pts/6    Ss   00:28   0:00 bash
lukasha+ 11642  0.0  0.0   4212   348 pts/0    S+   01:51   0:00 ./parent_stop
lukasha+ 11643  0.0  0.0   4216    84 pts/0    S+   01:51   0:00 ./parent_stop
lukasha+ 11649  0.0  0.0 155476  1876 pts/6    R+   01:51   0:00 ps u
lukasha+ 28858  0.0  0.1 115676  2148 pts/0    Ss   Apr20   0:00 /bin/bash
```

Висновки: бачимо два створених процеса, що сплять.

P.S. Варто зауважити, що програма завершується не через _exit (0), а через return 0, так як різниці немає.

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 Lukashak».

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

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


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

    if (pid > 0)
        sleep(10);
    else if (!pid)
        printf("\nI am Zombie-process of Lukashak\n\n");
    else {
        fprintf(stderr, "Fork error");
        return -1;
    }

    return 0;
}
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ gcc -o "/home/lukashak_daniil/C Codes/zombie_child" "/home/lukashak_daniil/C Codes/zombie_child.c"
[lukashak_daniil@vpsj3IeQ C Codes]$ ./zombie_child &
[1] 12259
[lukashak_daniil@vpsj3IeQ C Codes]$
I am Zombie-process of Lukashak
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
lukasha+  4205  0.0  0.1 115548  2132 pts/6    Ss   00:28   0:00 bash
lukasha+ 12259  0.0  0.0   4212   352 pts/0    S    01:58   0:00 ./zombie_child
lukasha+ 12260  0.0  0.0      0     0 pts/0    Z    01:58   0:00 [zombie_child] <defunct>
lukasha+ 12271  0.0  0.0 155476  1876 pts/6    R+   01:58   0:00 ps u
lukasha+ 28858  0.0  0.1 115676  2148 pts/0    Ss+  Apr20   0:00 /bin/bash
```

Висновки: бачимо два процеса, батько та нащадок, де нащадок перетворився у процес-зомбі.

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 Lukashak is finished».

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

Значення n – 8.

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

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


void signal_handler(int signal) {
    if (signal == SIGCHLD)
        wait(NULL);
}


int main() {
    sigset(SIGCHLD, &signal_handler);

    pid_t pid = fork();

    if (pid > 0)
        sleep(3 * 8);
    else if (!pid)
        printf("\nChild of Lukashak is finished\n\n");
    else {
        printf("Fork error");
        return -1;
    }

    return 0;
}
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ gcc -o "/home/lukashak_daniil/C Codes/zombie_control" "/home/lukashak_daniil/C Codes/zombie_control.c"
[lukashak_daniil@vpsj3IeQ C Codes]$ ./zombie_control &
[1] 14838
[lukashak_daniil@vpsj3IeQ C Codes]$
Child of Lukashak is finished
```

```
[lukashak_daniil@vpsj3IeQ C Codes]$ ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
lukasha+  4205  0.0  0.1 115548  2132 pts/6    Ss   00:28   0:00 bash
lukasha+ 14869  0.0  0.0 155476  1876 pts/6    R+   02:26   0:00 ps u
lukasha+ 28858  0.0  0.1 115676  2160 pts/0    Ss+  Apr20   0:00 /bin/bash
```

Висновки: ми отримали навички в управлінні процесами в ОС Unix на рівні мови програмування С. Одним з найскладніших завдань виявилось третє завдання, так як потрібно добре орієнтуватися не тільки в отриманні чи посиланні сигналу, а в двох темах одночасно.