

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій
Кафедра радіоелектронних і комп'ютерних систем

Звіт
про виконання лабораторних робіт №5
“Керування процесами і потоками”

Виконав
студент групи ФeI-23
Гупало Мар'ян
Перевірів
ас. Сінькевич О. О.

Львів – 2019

Завдання 1: Напишіть функцію, виклик якої приведе до знищення всіх процесів-зомбі створених поточним процесом.

Код

```
#include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <unistd.h>

void killZombie(){
    pid_t pid;
    int status;
    printf("\n");
    while((pid = wait(&status)) > 0)    printf("Zombie process with PID %d terminated with status %d\n", pid, status);
    printf("\n");
}

void zombieList(){
    pid_t pid;
    if ((pid = fork()) == -1) exit(-1);
    if (pid == 0){
        char* args[] = {"ps", "uf", "-C", "zombie", NULL};
        execvp("ps", args);
        exit(-1);
    }
    else {
        int status;
        waitpid(pid, &status, 0);
        if (!WIFEXITED(status)){
            exit(-1);
        }
    }
}

int main() {
    pid_t pid;
```

```

int i;

for (i = 1; ; ++i){
    if ((pid = fork()) == -1) return -1;
    else if (pid > 0) {
        if (i == 5){
            zombieList();
            killZombie();
            zombieList();
            exit(0);
        }
    }
}

else {
    printf(" ");
    exit(0);
}
}

return 0;
}

```

Результат

```

f13x@f13xmachine:~/Documents/OSi/5$ gcc zombieKiller.c
f13x@f13xmachine:~/Documents/OSi/5$ ./a.out

```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
Zombie	process	with	PID 4855	terminated	with	status 0				
Zombie	process	with	PID 4856	terminated	with	status 0				
Zombie	process	with	PID 4857	terminated	with	status 0				
Zombie	process	with	PID 4858	terminated	with	status 0				
Zombie	process	with	PID 4859	terminated	with	status 0				

Завдання 2 : Розробіть простий командний інтерпретатор для Linux і Windows. Він

повинен видавати підказку (наприклад, «>»), обробляти введений користувачем командний рядок (що містить ім'я виконуваного файлу програми та її аргументи) і запускати задану програму. Асинхронний запуск здійснюють введенням «&» як останнього символу командного рядка. У разі завершення командного рядка будь-яким іншим символом програма запускається синхронно. Інтерпретатор завершує роботу після введення рядка «exit». Виконання програм, запущених інтерпретатором, може бути перерване натисканням клавіш Ctrl+C, однак воно не повинне переривати виконання інтерпретатора. Для запуску програмного коду в Linux рекомендовано використовувати функцію `execvp()`, що приймає два параметри `prog` і `args`, аналогічні до перших двох параметрів функції `execve()`, і використовує змінну оточення `PATH` для пошуку шляху до виконуваних файлів.

Код

```
#include <signal.h>

#include <stdlib.h>

#include <string.h>

#include <sys/wait.h>

#include <unistd.h>

#include <stdio.h>

pid_t pid;

void exeCommand(char* command){

char* argv[20];

int beakground;

char* token;

token = strtok(command," ");

int i=0;

for(;token!=NULL;i++){

argv[i] = token;

token = strtok(NULL," ");

}

if(!strcmp(argv[i-1],"&")){

beakground = 1;

argv[i-1] = NULL;

}

else{

beakground = 0;
```

```

argv[i] = NULL;

}

if((pid=fork())==-1) exit(1);

if(pid==0){
execvp(argv[0],argv);
exit(1);
}

if(!background) wait(NULL);
}

static void catch_function(int signal){
kill(pid,SIGINT);

printf("\n");
}

int main(){
printf("Starting shell...\n");
signal(SIGINT, catch_function);

char command[5];

while(1){
printf(">");

fgets(command,3, stdin);

if(command[0]=='\0') continue;

if(!strcmp(command, "exit")) break;

exeCommand(command);

}

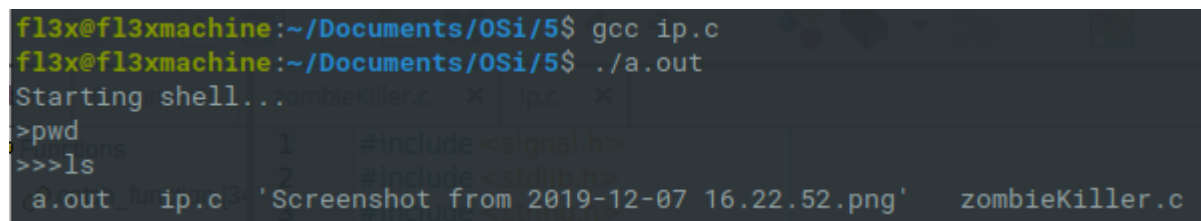
printf("Closing...\n");

return 0;

}

```

Робота



```

f13x@f13xmachine:~/Documents/OSi/5$ gcc ip.c
f13x@f13xmachine:~/Documents/OSi/5$ ./a.out
Starting shell...
>pwd
>>>ls
a.out ip.c

```

Завдання 3 : Розробіть застосування для Linux і Windows, що реалізує паралельне виконання коду двома потоками. Основний потік застосування Т створює потік t Далі кожен із потоків виконує цикл (наприклад, до 30). На кожній ітерації циклу він збільшує значення локального лічильника на одиницю, відображає це значення з нового рядка і призупиняється на деякий час (потік Т – на час w Т , потік t – w t). Після завершення циклу потік Т приєднує t. Як залежать результати виконання цього застосування від значень w Т і w t ? Як зміняться ці результати, якщо потік t не буде приєднано?

Код

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

void *anotherThread(void *value){

int i=0;

for(i=2;i<32;i++){

printf("Second thread, value %d\n",i);

sleep(2);

}

}

int main(){

pthread_t thread;

pthread_create(&thread,NULL,anotherThread,NULL);

int i=0;

for(i=0;i<30;i++){

printf("First thread, value %d\n",i);

sleep(1);

}

pthread_join(thread,NULL);

return 0;
```

Робота

```
fl3x@fl3xmachine:~/Documents/OSi/5$ gcc -pthread threads.c
fl3x@fl3xmachine:~/Documents/OSi/5$ ./a.out
First thread, value 0
Second thread, value 2
First thread, value 1
Second thread, value 3
First thread, value 2
First thread, value 3
Second thread, value 4
First thread, value 4
First thread, value 5
Second thread, value 5
First thread, value 6
First thread, value 7
Second thread, value 6
First thread, value 8
First thread, value 9
Second thread, value 7
First thread, value 10
First thread, value 11
Second thread, value 8
First thread, value 12
First thread, value 13
Second thread, value 9
First thread, value 14
First thread, value 15
Second thread, value 10
First thread, value 16
First thread, value 17
Second thread, value 11
First thread, value 18
First thread, value 19
Second thread, value 12
First thread, value 20
First thread, value 21
```

Висновок: на даній лабораторній роботі я навчився використовувати потоки та застосовувати програмні інтерфейси ОС для керування ними.