УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №8

по предмету «Архитектура компьютерной техники и операционных систем»

Вариант 2

Выполнил:

Бражалович А.И.

Проверил:

[Аврамец](https://iis.bsuir.by/employees/d-avramets) Д.В.

Группа 351004

Минск 2024

**Задание**

***1->(2,3,4)SIGUSR1 2->(5,6)SIGUSR2 6->7 SIGUSR1***

***7->8 SIGUSR1 8->1 SIGUSR2***

**Код программы**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <signal.h>

#include <time.h>

#include <sys/time.h>

#include <sys/types.h>

#include <string.h>

#include <limits.h>

int usr1\_count = 0;

int usr2\_count = 0;

int SIG\_COUNT = 101;

pid\_t pid[9];

int status;

void work() {

while (1) {

usleep(100000);

}

}

void handle\_usr1(int sig) {

usr1\_count++;

struct timeval tv;

gettimeofday(&tv, NULL);

printf("%d pid %d ppid получил USR1 в %ld мкс\n", getpid() - pid[0], getppid()- pid[0], tv.tv\_sec \* 1000000 + tv.tv\_usec);

}

void handle\_usr2(int sig) {

usr2\_count++;

struct timeval tv;

gettimeofday(&tv, NULL);

printf("%d pid %d ppid получил USR2 в %ld мкс\n", getpid()- pid[0], getppid()- pid[0], tv.tv\_sec \* 100000 + tv.tv\_usec);

}

void handle\_sigterm(int sig) {

int flag = 0;

for (int i = getpid() - pid[0] + 1; i < 9; i++) {

if (pid[i] != -1) {

flag = 1;

}

}

if (flag == 0 && getpid() - pid[0] != 8) {

printf("%d Процесс окончен\n", getpid() - pid[0]);

exit(0);

}

}

void send\_signals\_process(pid\_t one\_p, int sig) {

struct timeval tv;

for (int i = 1; i <= SIG\_COUNT; i++) {

kill(one\_p, sig);

gettimeofday(&tv, NULL);

printf("%d pid %d ppid отправил %d в %ld\n", getpid() - pid[0], getppid() - pid[0], sig, tv.tv\_sec \* 100000 + tv.tv\_usec);

usleep(100000);

}

printf("SIGUSR1: %d\nSIGUSR2: %d\n",usr1\_count, usr2\_count);

kill(one\_p, SIGTERM);

}

void send\_signals\_group(pid\_t group, int sig) {

struct timeval tv;

for (int i = 1; i <= SIG\_COUNT; i++)

{

killpg(group, sig);

gettimeofday(&tv, NULL);

printf("%d pid %d ppid отправил %d time %ld\n", getpid() - pid[0], getppid() - pid[0], sig, tv.tv\_sec \* 100000 + tv.tv\_usec);

usleep(100000);

}

sleep(1);

printf("SIGUSR1: %d\nSIGUSR2: %d\n",usr1\_count, usr2\_count);

killpg(group, SIGTERM);

}

//1->(2,3,4)SIGUSR1 2->(5,6)SIGUSR2 6->7 SIGUSR1 7->8 SIGUSR1 8->1 SIGUSR2

void create\_procceses() {

for (int i = 0; i < 9; i++) {

pid[i] = -1;

}

signal(SIGUSR1, handle\_usr1);

signal(SIGUSR2, handle\_usr2);

signal(SIGTERM, handle\_sigterm);

pid[0] = getpid();

pid[1] = fork();

if (pid[1] > 0) {

FILE \*file = fopen("pid1.txt", "w");

fprintf(file, "%d", pid[1]);

fclose(file);

}

if (pid[1] == 0) {

pid[2] = fork();

if (pid[2] > 0) {

pid[3] = fork();

if (pid[3] > 0) {

pid[4] = fork();

if (pid[4] > 0) {

setpgid(pid[2], pid[2]);

setpgid(pid[3], pid[2]);

setpgid(pid[4], pid[2]);

send\_signals\_group(getpgid(pid[2]), SIGUSR1);

waitpid(pid[2], &status, 0);

waitpid(pid[3], &status, 0);

waitpid(pid[4], &status, 0);

printf("%d Процесс окончен\n", getpid() - pid[0]);

exit(0);

}

if (pid[3] == 0) {

work();

}

if (pid[4] == 0) {

work();

}

}

}

if (pid[2] == 0) {

pid[5] = fork();

if (pid[5] > 0) {

pid[6] = fork();

if (pid[6] > 0) {

setpgid(pid[5], pid[5]);

setpgid(pid[6], pid[5]);

send\_signals\_group(getpgid(pid[5]), SIGUSR2);

waitpid(pid[5], &status, 0);

waitpid(pid[6], &status, 0);

printf("%d Процесс окончен\n", getpid() - pid[0]);

exit(0);

}

if (pid[6] == 0) {

pid[7] = fork();

if (pid[7] > 0) {

send\_signals\_process(pid[7], SIGUSR1);

waitpid(pid[7], &status, 0);

printf("%d Процесс окончен\n", getpid() - pid[0]);

exit(0);

}

if (pid[7] == 0) {

pid[8] = fork();

if (pid[8] > 0) {

send\_signals\_process(pid[8], SIGUSR1);

waitpid(pid[8], &status, 0);

printf("%d Процесс окончен\n", getpid() - pid[0]);

exit(0);

}

if (pid[8] == 0) {

FILE \*file = fopen("pid1.txt", "r");

pid\_t pid1;

fscanf(file, "%d", &pid1);

fclose(file);

send\_signals\_process(pid1, SIGUSR2);

printf("%d Процесс окончен\n", getpid() - pid[0]);

exit(0);

}

}

}

}

if (pid[5] == 0) {

work();

}

}

}

waitpid(pid[1], &status, 0);

}

int main ()

{

create\_procceses();

return 0;

}