УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе № 8

по предмету «Архитектура компьютерной техники и операционных систем»

Вариант 7

Выполнил:

Панкратьев Е.С.

гр. 251003

Проверил:

Аврамец Д. В.

Минск 2023

# задание

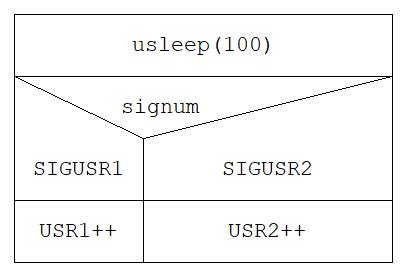
Создать дерево процессов.

Дерево процессов из “1->(2,3,4,5,6) 6->(7,8) “.

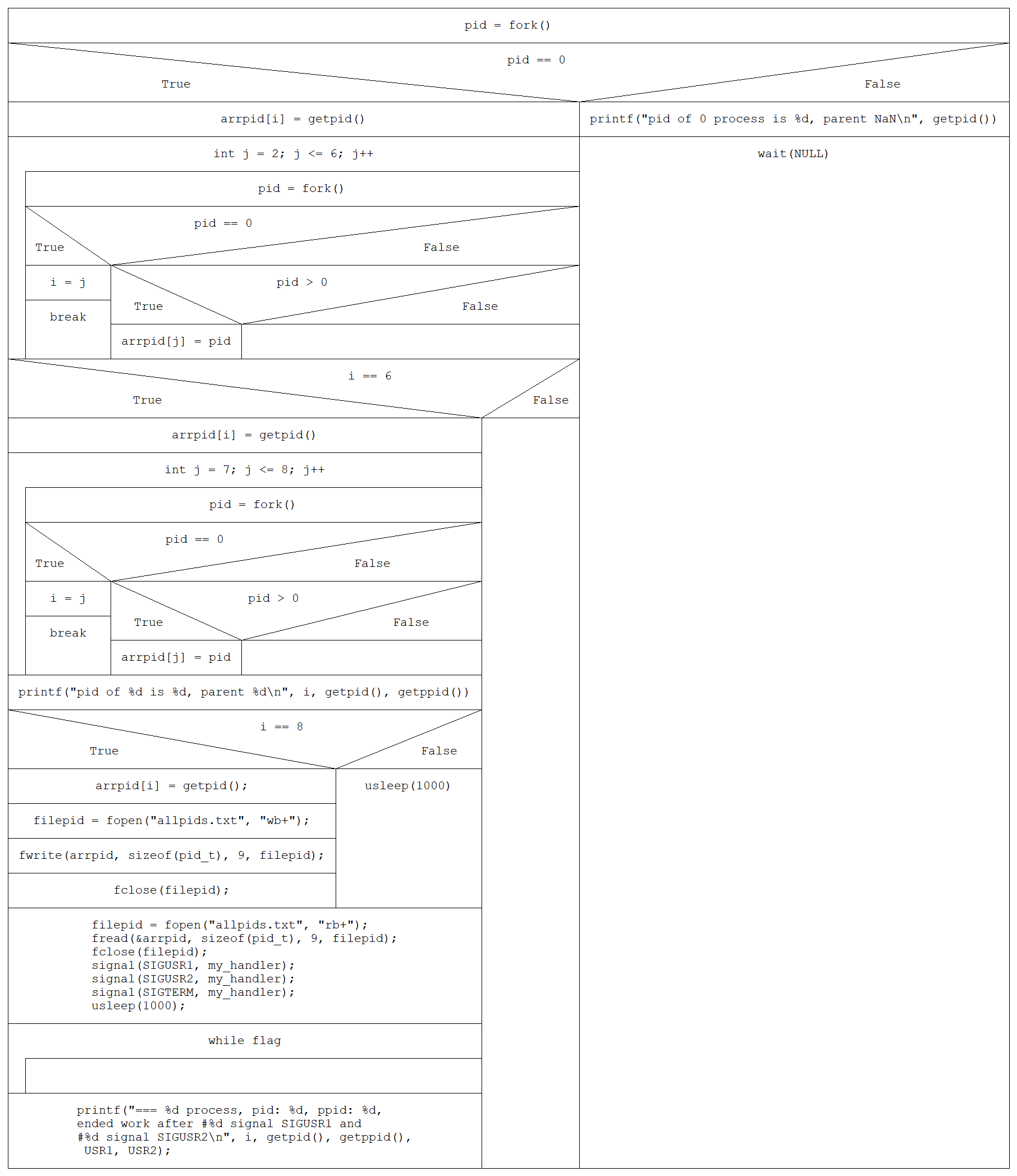
Последовательность обмена сигналами из “*1->(2,3,4,5,6) SIGUSR2 6->(7,8) SIGUSR1 8->1 SIGUSR2”*.

# Схема алгоритма

***Схема алгоритма функции my\_handler***



***Схема алгоритма функции main***

******

# Код программы для процессов

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <time.h>

#include <signal.h>

#include <sys/wait.h>

#include <stdbool.h>

#define PROC\_COUNT 9

#define SIGCOUNT 101

static int i = 1;

static int USR1 = 0, USR2 = 0;

static bool flag = false;

pid\_t arrpid[PROC\_COUNT] = {};

void my\_handler(int signum) {

usleep(100);

switch (signum) {

case SIGUSR1:

printf("process: %d, pid: %d, ppid: %d, signal: SIGUSR1, time: %ld\n", i, getpid(), getppid(), clock());

USR1++;

break;

case SIGUSR2:

printf("process: %d, pid: %d, ppid: %d, signal: SIGUSR2, time: %ld\n", i, getpid(), getppid(), clock());

USR2++;

flag = true;

break;

default:

flag = true;

break;

}

}

int main(int argc, char \*argv[]) {

pid\_t pid;

FILE \*filepid;

// 0 -> 1

pid = fork();

if (pid == 0) {

arrpid[i] = getpid();

for (int j = 2; j <= 6; j++) {

pid = fork();

if (pid == 0) {

i = j;

break;

}

else if (pid > 0)

arrpid[j] = pid;

}

if (i == 6) {

arrpid[i] = getpid();

for (int j = 7; j <= 8; j++) {

pid = fork();

if (pid == 0) {

i = j;

break;

}

else if (pid > 0)

arrpid[j] = pid;

}

}

printf("pid of %d is %d, parent %d\n", i, getpid(), getppid());

if (i == 8) {

arrpid[i] = getpid();

filepid = fopen("allpids.txt", "wb+");

fwrite(arrpid, sizeof(pid\_t), 9, filepid);

fclose(filepid);

}

else

usleep(1000);

filepid = fopen("allpids.txt", "rb+");

fread(&arrpid, sizeof(pid\_t), 9, filepid);

fclose(filepid);

signal(SIGUSR1, my\_handler);

signal(SIGUSR2, my\_handler);

signal(SIGTERM, my\_handler);

usleep(1000);

pid\_t group;

int h;

switch (i) {

case 1: // 1 -> (2,3,4,5,6)

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

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

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

setpgid(arrpid[5], arrpid[2]);

setpgid(arrpid[6], arrpid[2]);

group = getpgid(arrpid[2]);

for (h = 1; h <= SIGCOUNT; h++) {

killpg(group, SIGUSR2);

printf(">> %d process, pid: %d, ppid: %d, send signal SIGUSR2 to group %d, time %ld\n", i, getpid(), getppid(), group, clock());

usleep(1000);

}

killpg(group, SIGTERM);

break;

case 6: // 6 -> (7,8)

setpgid(arrpid[7], arrpid[7]);

setpgid(arrpid[8], arrpid[7]);

group = getpgid(arrpid[7]);

for (h = 1; h <= SIGCOUNT; h++) {

killpg(group, SIGUSR1);

printf(">> %d process, pid: %d, ppid: %d, send signal SIGUSR1 to group %d, time %ld\n", i, getpid(), getppid(), group, clock());

usleep(1000);

}

killpg(group, SIGTERM);

break;

case 8: // 8 -> 1

for (h = 1; h <= SIGCOUNT; h++) {

kill(arrpid[1], SIGUSR2);

printf(">> %d process, pid: %d, ppid: %d, send signal SIGUSR2, time %ld\n", i, getpid(), getppid(), clock());

usleep(1000);

}

kill(arrpid[1], SIGTERM);

break;

}

while (1)

if (flag)

break;

for (int i = 0; i < PROC\_COUNT; i++)

wait(0);

printf("=== %d process, pid: %d, ppid: %d, ended work after #%d signal SIGUSR1 and #%d signal SIGUSR2\n", i, getpid(), getppid(), USR1, USR2);

}

else {

printf("pid of 0 process is %d, parent NaN\n", getpid());

wait(NULL);

}

return 0;

}

# Результаты

