Министерство образования и науки Российской Федерации

Югорский государственный университет

Институт цифровой экономики

Отчет о лабораторной работе №1

По дисциплине «Разработка системных приложений»

Выполнил: студент группы 1162б

Стародубов Юрий

Проверил: Шицелов А.В.

Ханты-Мансийск

2019

**Цель:**

Изучить прядок создания, завершения и управления дочерними процессами, операции создания дочернего процесса, загрузки и исполнения новым процессом некоторой программы, а также координация завершения дочернего процесса с главным процессом.

**Ход работы:**

Задание 1. Пояснить, сколько процессов будет создано следующей программой:

void lab1() {

int i;

for (i = 0; i < 3; i++)

{

pid\_t pid = fork();

cout << "I am process " + to\_string(getpid()) + ", My parent is "+to\_string(getppid()) << endl;

}

}

Данной программой будет создано 8 процессов

Задание 2.

Создайте текстовый файл и скомпилировать программу

1. Объясните почему дочерний может считывать файл, открытый родительским процессом?

Дочернему процессу передается дескриптор файла, открытого родителем

1. Измените программу таким образом, чтобы файл открывался на запись и дочерний и родительский процессы записывали в него строку «Строка от дочернего процесса» и «Строка от родительского процесса» соответственно.

void lab22() {

FILE \*fp;

char str[80];

char sss[20];

for (int i = 0; i < 4; i++) {

fp = fopen("/home/piratemaster/plan.txt", "w");

if (fp == NULL)

{

fprintf(stderr, "Could not open file");

scanf(sss);

exit(1);

}

char str1[] = "Parent line", str2[] = "child line";

fputs(str1, fp);

printf("Parent puts: %s\n", str1);

if (fork() == 0)

{

fputs(str2, fp);

printf("Child puts: %s\n", str2);

}

fclose(fp);

}

}

1. Напишите программу, которая бы производила следующее дерево процессов, где корень представляет родительский процесс.

void lab3() {

cout << "I am main process " + to\_string(getpid()) + ", My parent is " + to\_string(getppid()) << endl;

pid\_t PID = fork();

if (PID == 0) {

cout << "I am process " + to\_string(getpid()) + ", My parent is " + to\_string(getppid()) << endl;

pid\_t PID11 = fork();

if (PID11 == 0) {

cout << "I am process " + to\_string(getpid()) + ", My parent is " + to\_string(getppid()) << endl;

}

else {

pid\_t PID12 = fork();

if (PID12 == 0) {

cout << "I am process " + to\_string(getpid()) + ", My parent is " + to\_string(getppid()) << endl;

}

}

}

else {

pid\_t PID2 = fork();

if (PID2 == 0) {

cout << "I am process " + to\_string(getpid()) + ", My parent is " + to\_string(getppid()) << endl;

pid\_t PID21 = fork();

if (PID21 == 0) {

cout << "I am process " + to\_string(getpid()) + ", My parent is " + to\_string(getppid()) << endl;

}

else {

pid\_t PID22 = fork();

if (PID22 == 0) {

cout << "I am process " + to\_string(getpid()) + ", My parent is " + to\_string(getppid()) << endl;

}

}

}

}

}

Вывод: были изучены прядок создания, завершения и управления дочерними процессами, операции создания дочернего процесса, загрузки и исполнения новым процессом некоторой программы, а также координация завершения дочернего процесса с главным процессом.