МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И

МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования **«Московский технический университет связи и информатики»**

**(МТУСИ)**

Кафедра «Математическая кибернетика и информационные технологии»

**ОТЧЕТ**

по лабораторной работе №2

“**Процессы и потоки в ОС GNU/LINUX**”

дисциплина «Операционные системы»

Выполнил:

студент группы БВТ2101

Старостин И.Д.

Проверила:

ст. пр. кафедры МКиИТ

Королькова Т.В.

Москва 2023

**Цель работы**: изучение системных средств порождения и взаимодействия

процессов в ОС GNU/LINUX.

**Задание:**

1. Изучить теоретическую часть лабораторной работы.
2. Написать программу, создающую два дочерних процесса с использованием двух вызовов fork(). Родительский и два дочерних процесса должны выводить на экран свой pid и pid родительского процесса и текущее время в формате: часы:минуты:секунды. Используя вызов system(), выполнить

команду ps -x в родительском процессе. Найти свои процессы в списке запущенных процессов.

Первый дочерний процесс, помимо указанного вывода, также выводит информацию по варианту (табл.1), используя вызов system().

Второй дочерний процесс после вывода основной информации должен быть замещен задачей, создающей два дочерних потока (п.3 задания).

1. Написать программу, создающую два дочерних потока. Родительский процесс и два дочерних потока должны выводить на экран свой id и pid родительского процесса и текущее время в формате: часы:минуты:секунды.

**Выполнение:**

Задание будет выполнятся на ОС Arch Linux и языке C.

Программа состоит из двух файлов. Для начала необходимо скомпилировать код threads.c так, чтобы исполняемый файл имел название: threads

$ gcc threads.c -o threads

Программа также находится в github-репозитории:

[https://github.com/Barsux/OS\_L2](https://github.com/Barsux/OS_L1)

Ниже представлен код программы main.c:

#include <stdio.h>

#include <unistd.h>

#include <time.h>

#include <stdlib.h>

void printstat(int num){

time\_t ts;

time(&ts);

struct tm \* time = localtime(&ts);

int pid = getpid();

int ppid = getppid();

if (num == 0) printf("PID Родительского Процесса: %d, Время: %02d:%02d:%02d\n", pid, time->tm\_hour, time->tm\_min, time->tm\_sec);

else printf("Дочерний процесс: N%d, PID Процесса: %d, PID родителя: %d, Время: %02d:%02d:%02d\n", num, pid, ppid, time->tm\_hour, time->tm\_min, time->tm\_sec);

}

int main() {

pid\_t child1, child2;

child1 = fork(); // Создание первого дочернего процесса

if (child1 < 0) {

perror("Ошибка при создании первого дочернего процесса.\n");

return 1;

}

else if (child1 == 0) {

// Код для первого дочернего процесса

printstat(1);

}

else {

child2 = fork(); // Создание второго дочернего процесса

if (child2 < 0) {

perror("Ошибка при создании второго дочернего процесса.\n");

return 1;

} else if (child2 == 0) {

printstat(2);

// Замещаем процесс другой исполняемой программой.

execl("threads", (char \*)NULL);

} else {

// Код для родительского процесса

int status;

printstat(0);

printf("Выполнение команды 'ps -x':\n");

system("ps -x");

printf("\nВыполнение команды 'route':\n");

system("route");

waitpid(child1, &status, 0); // Ожидание завершения первого дочернего процесса

waitpid(child2, &status, 0); // Ожидание завершения второго дочернего процесса

}

}

return 0;

}

Далее представлен код программы threads.c:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <time.h>

// Вывод статистики по потокам

void pthrintstat(){

time\_t ts;

time(&ts);

struct tm \* time = localtime(&ts);

int pid = getpid();

int ppid = getppid();

printf("3. Дочерний процесс: N%d, PID Процесса: %d, PID родителя: %d, Время: %02d:%02d:%02d\n", pthread\_self(), pid, ppid, time->tm\_hour, time->tm\_min, time->tm\_sec);

}

int main(){

// Создаю идентификаторы потоков

pthread\_t thread1, thread2;

// Запускаю потоки с вызовом функции

pthread\_create(&thread1, NULL, pthrintstat, NULL);

pthread\_create(&thread2, NULL, pthrintstat, NULL);

// Ожидание завершения потоков

pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL);

return 0;

}

Результат работы программы:

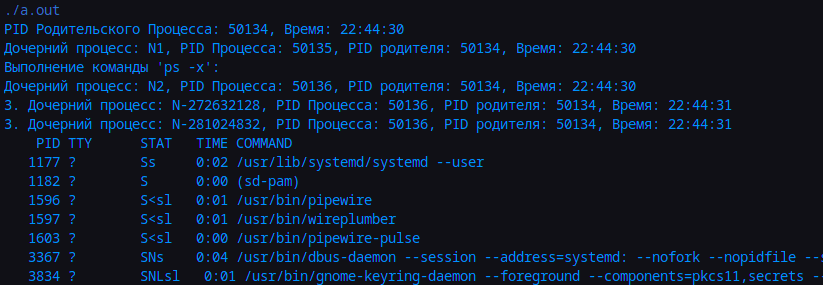


Рисунок 1 - Результат работы программы.

**Вывод:** Я научился работать с процессами и потоками в языке СИ на ОС Linux.