

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра вычислительной техники**

**ОТЧЕТ**

**по лабораторной работе №3**

по дисциплине «Организация процессов и программирование в среде Linux»

Тема: Создание и идентификация процессов

Студент гр.8305

Преподаватель

\_\_\_\_\_ Зайков Д.Г  
\_\_\_\_\_ Разумовский Г.В

Санкт-Петербург

2021

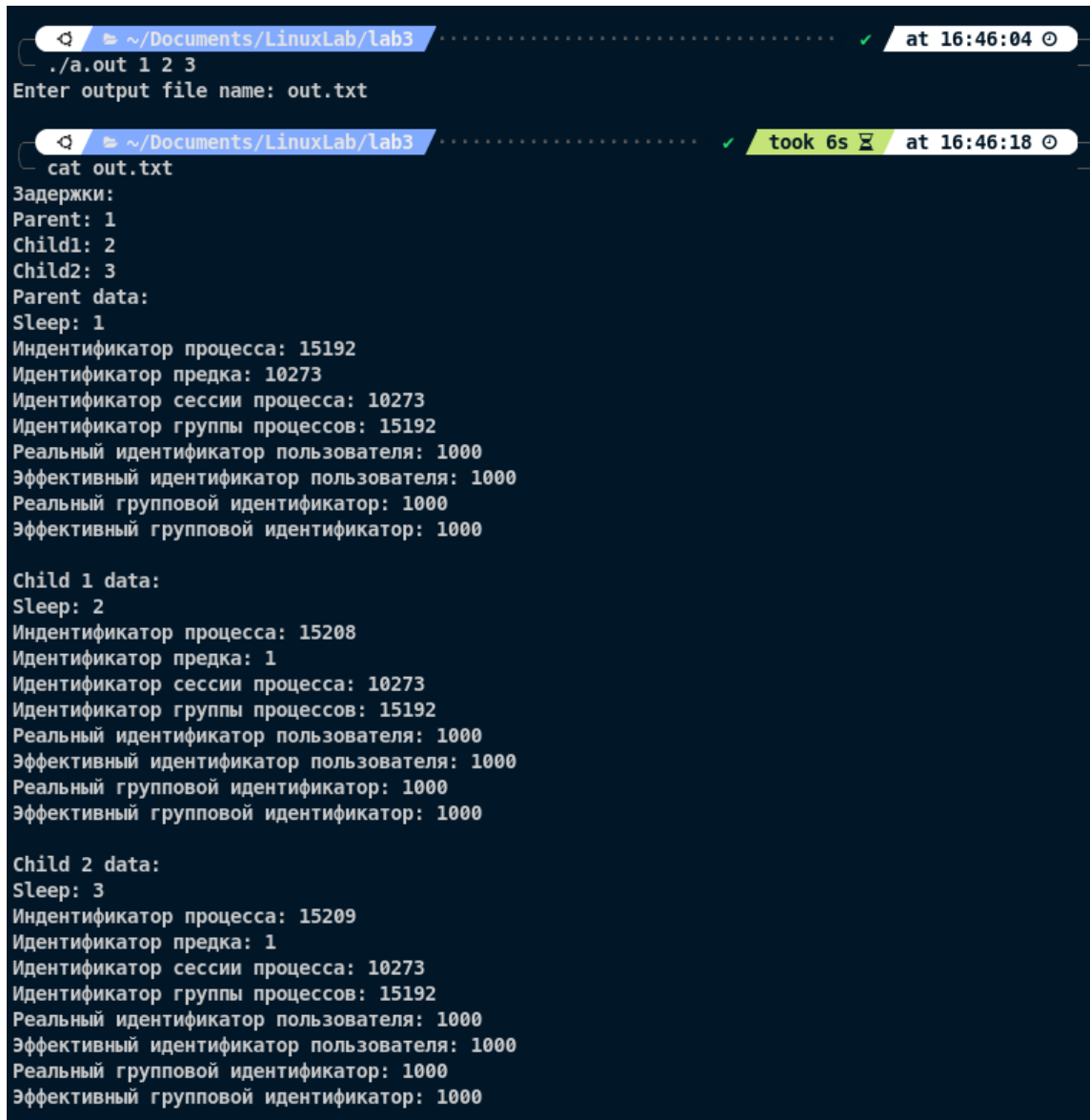
**Цель работы.** Изучение и использование системных функций, обеспечивающих порождение и идентификацию процессов.

**Задание.**

1. Разработать программу, которая порождает 2 потомка. Первый потомок порождается с помощью `fork`, второй – с помощью `vfork` с последующей заменой на другую программу. Все 3 процесса должны вывести в один файл свои атрибуты с предварительным указанием имени процесса (например: Предок, Потомок1, Потомок2). Имя выходного файла задается при запуске программы. Порядок вывода атрибутов в файл должен определяться задержками процессов, которые задаются в качестве параметров программы и выводятся в начало файла.
2. Откомпилировать программу и запустить ее 3 раза с различными сочетаниями задержек.

**Ход работы.** Проведём тестирование программы с разными входными параметрами.

1. Запустим с задержками 1 2 3 для предка, потомка1 и потомка2 соответственно:



```
~/Documents/LinuxLab/lab3 at 16:46:04
./a.out 1 2 3
Enter output file name: out.txt

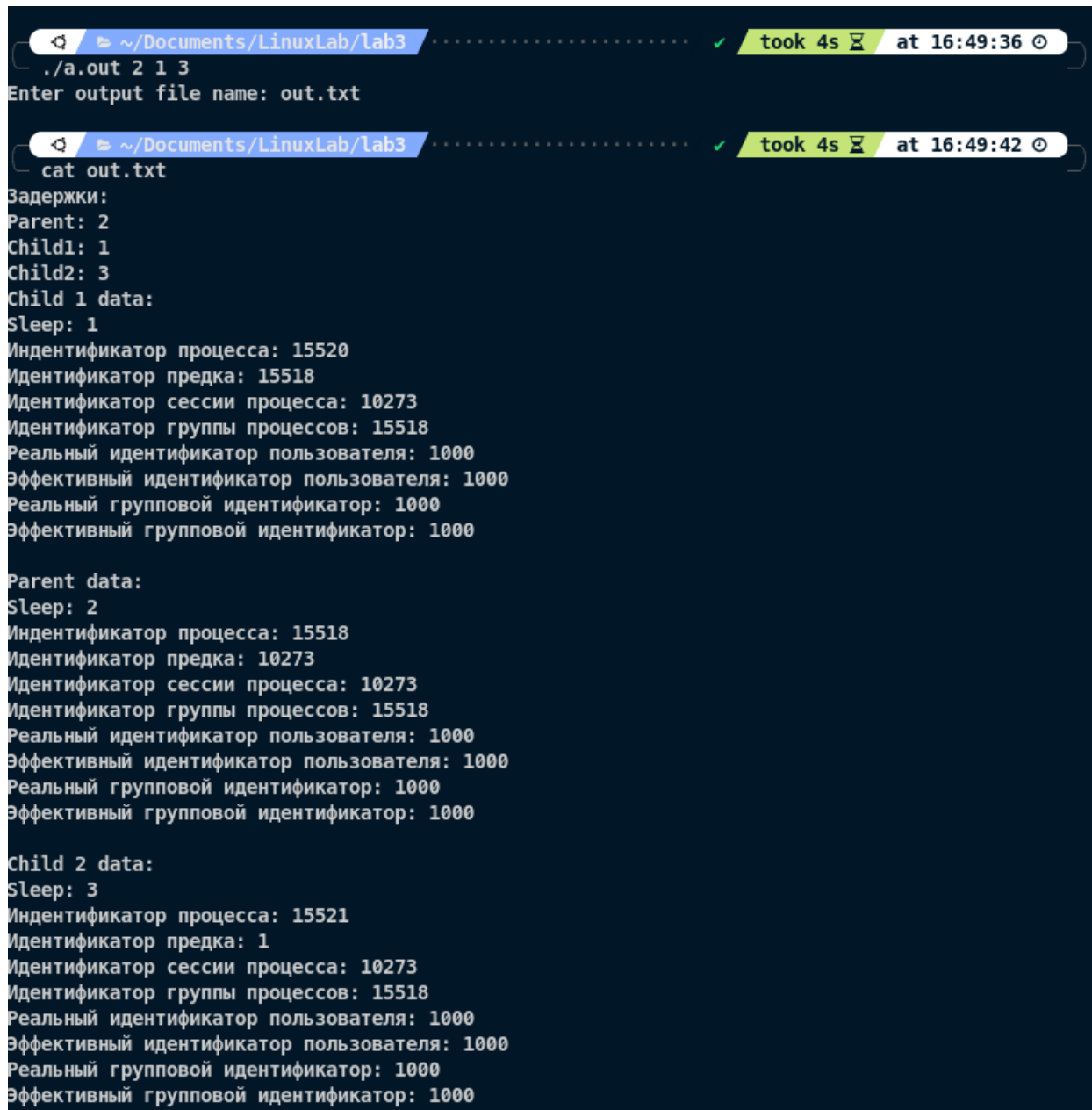
~/Documents/LinuxLab/lab3 took 6s at 16:46:18
cat out.txt
Задержки:
Parent: 1
Child1: 2
Child2: 3
Parent data:
Sleep: 1
Идентификатор процесса: 15192
Идентификатор предка: 10273
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15192
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000

Child 1 data:
Sleep: 2
Идентификатор процесса: 15208
Идентификатор предка: 1
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15192
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000

Child 2 data:
Sleep: 3
Идентификатор процесса: 15209
Идентификатор предка: 1
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15192
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000
```

Рисунок 1. Задержки 1 2 3

2. Запустим с задержками 2 1 3 для предка, потомка1 и потомка2 соответственно:



```
~/Documents/LinuxLab/lab3 ..... ✓ took 4s at 16:49:36
./a.out 2 1 3
Enter output file name: out.txt

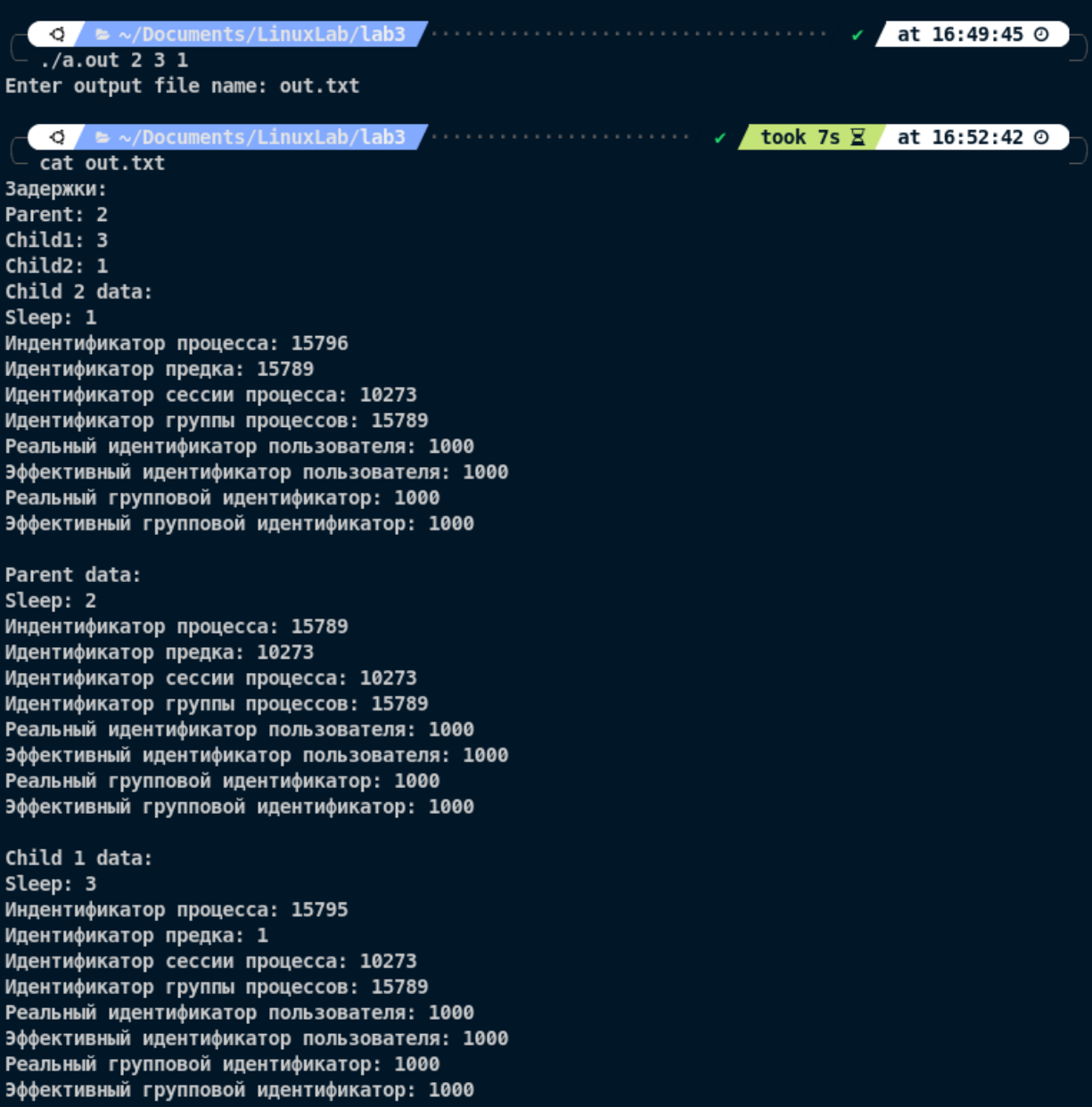
~/Documents/LinuxLab/lab3 ..... ✓ took 4s at 16:49:42
cat out.txt
Задержки:
Parent: 2
Child1: 1
Child2: 3
Child 1 data:
Sleep: 1
Идентификатор процесса: 15520
Идентификатор предка: 15518
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15518
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000

Parent data:
Sleep: 2
Идентификатор процесса: 15518
Идентификатор предка: 10273
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15518
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000

Child 2 data:
Sleep: 3
Идентификатор процесса: 15521
Идентификатор предка: 1
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15518
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000
```

Рисунок 2. Задержки 2 1 3

3. Запустим с задержками 2 3 1 для предка, потомка1 и потомка2 соответственно:



```
~/Documents/LinuxLab/lab3 at 16:49:45
./a.out 2 3 1
Enter output file name: out.txt

~/Documents/LinuxLab/lab3 took 7s at 16:52:42
cat out.txt
Задержки:
Parent: 2
Child1: 3
Child2: 1
Child 2 data:
Sleep: 1
Идентификатор процесса: 15796
Идентификатор предка: 15789
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15789
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000

Parent data:
Sleep: 2
Идентификатор процесса: 15789
Идентификатор предка: 10273
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15789
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000

Child 1 data:
Sleep: 3
Идентификатор процесса: 15795
Идентификатор предка: 1
Идентификатор сессии процесса: 10273
Идентификатор группы процессов: 15789
Реальный идентификатор пользователя: 1000
Эффективный идентификатор пользователя: 1000
Реальный групповой идентификатор: 1000
Эффективный групповой идентификатор: 1000
```

Рисунок 3. Задержки 2 3 1

**Вывод.** В ходе выполнения лабораторной работы были изучение и использованы системных функций, обеспечивающих порождение и идентификацию процессов.

### **Приложение. Исходный код программы.**

#### **lab3.cpp**

```
#include <iostream>
#include <fstream>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <iomanip>

int main(int argc, char **argv) {
    std::string fileName;
    int parentDelay = atoi(argv[1]),
        child1Delay = atoi(argv[2]),
        child2Delay = atoi(argv[3]);
    std::cout << "Enter output file name: ";
    std::cin >> fileName;
    std::ofstream outputFile;
    outputFile.open(fileName);
    outputFile << "Задержки: " << "\nParent: " << parentDelay <<
        "\nChild1: " << child1Delay <<
        "\nChild2: " << child2Delay;
    outputFile.close();
    int child1 = fork();
    if (child1 == 0) {
        sleep(child1Delay);
        outputFile.open(fileName, std::ios_base::app);
        outputFile
            << "\nChild 1 data:"
            << "\nSleep: " << child1Delay
            << "\nИдентификатор процесса: " << getpid()
            << "\nИдентификатор предка: " << getppid()
            << "\nИдентификатор сессии процесса: " << getsid(getpid())
```

```

        << "\nИдентификатор группы процессов: " << getpgid(getpid())
        << "\nРеальный идентификатор пользователя: " << getuid()
        << "\nЭффективный идентификатор пользователя: " << geteuid()
        << "\nРеальный групповой идентификатор: " << getgid()
        << "\nЭффективный групповой идентификатор: " << getegid() << std::endl;
    outputFile.close();
    exit(EXIT_SUCCESS);
} else {
    int child2 = vfork();
    if (child2 == 0) {
        execl("child", "child", fileName.c_str(), argv[3], NULL);
    } else {
        sleep(parentDelay);
        outputFile.open(fileName, std::ios_base::app);
        outputFile
            << "\nParent data:"
            << "\nSleep: " << parentDelay
            << "\nИдентификатор процесса: " << getpid()
            << "\nИдентификатор предка: " << getppid()
            << "\nИдентификатор сессии процесса: " << getsid(getpid())
            << "\nИдентификатор группы процессов: " << getpgid(getpid())
            << "\nРеальный идентификатор пользователя: " << getuid()
            << "\nЭффективный идентификатор пользователя: " << geteuid()
            << "\nРеальный групповой идентификатор: " << getgid()
            << "\nЭффективный групповой идентификатор: " << getegid() << std::endl;
        outputFile.close();
    }
}

return 0;
}

```

## child.cpp

```
#include <iostream>
#include <fstream>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <iomanip>

int main(int argc, char *argv[]) {
    std::ofstream outputFile;
    outputFile.open(argv[1], std::ios_base::app);
    sleep(atoi(argv[2]));
    outputFile
        << "\nChild 2 data:"
        << "\nSleep: " << atoi(argv[2])
        << "\nИдентификатор процесса: " << getpid()
        << "\nИдентификатор предка: " << getppid()
        << "\nИдентификатор сессии процесса: " << getsid(getpid())
        << "\nИдентификатор группы процессов: " << getpgid(getpid())
        << "\nРеальный идентификатор пользователя: " << getuid()
        << "\nЭффективный идентификатор пользователя: " << geteuid()
        << "\nРеальный групповой идентификатор: " << getgid()
        << "\nЭффективный групповой идентификатор: " << getegid() << std::endl;
    outputFile.close();

    exit(EXIT_SUCCESS);
}
```