Министерство высшего образования и науки РФ
ФГБОУ ВО ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра «Информационная безопасность систем и технологий»

# ОТЧЁТ

о лабораторной работе №7

Многопоточные вычисления в программах на языке Си++

Дисциплина: Языки программирования

Группа:18ПИ1

Выполнил: Асаян А.В.

Количество баллов:

Дата сдачи:

Проверил: к.т.н., доцент Лупанов М.Ю.

Пенза, 2019

1 Цель работы

1.1 Освоить компоненты стандартной библиотеки Си++ для создания многопоточных приложений.

2 Задания к практической работе

2.1 В одном университете студенты-гуманитарии использовали пароли, получаемые перестановкой символов в строке 123456789. А студенты из соседнего колледжа написали функцию для подбора таких паролей, если известен хэш от пароля. Код функции приведен в приложении Е. Варианты хэшей паролей студентов-гуманитариев приведены в приложении Ж. Используя функцию из приложения Е напишите программу (однопоточную), которая может последовательно находить пароли для любого количества хэшей от 1 до 8 включительно. Программа не должна интерактивно взаимодействовать с пользователем. Хэши должны либо передаваться через параметры командной строки, либо читаться из файла.

2.2 Модифицируйте функцию из приложения Е таким образом, чтобы она могла быть использована в многопоточной программе. Реализуйте с использованием этой модифицированной функции многопоточную программу, которая может параллельно находить пароли для любого количества хэшей от 1 до 8 включительно. Программа не должна интерактивно взаимодействовать с пользователем. Хэши должны либо передаваться через параметры командной строки, либо читаться из файла.

2.3 Выполните сравнение скорости подбора паролей для программ, разработанных в первом и втором задании. Для этого воспользуйтесь утилитой time. Формат запуска: time ./myprog, где myprog — ваша программа. Сравнение проводить для поиска одного, двух, четырех и восьми паролей на одних и тех же выборках хэшей. Выборку хэшей сделать самостоятельно. Результаты эксперимента свести в таблицу и проанализировать. Использовать таблицу, аналогичную таблице И.1 приложения И.

2.4 Задание повышенной сложности. Модифицируйте функцию из приложения Е таким образом, чтобы она выполняла не полный перебор всего диапазона паролей, а частичный, от одного значения пароля до другого. Реализуйте программу, делящую весь диапазон паролей на 2, 4 или 8 частей и выполняющую подбор одного пароля по частям в параллельных потоках. Программа также должна позволять подбирать пароль в одном потоке, без деления диапазона на части.

2.5 Задание повышенной сложности. Выполните сравнение скорости подбора пароля для первого значения хэш-функции из таблицы Ж.1 приложения Ж для случаев без деления на диапазоны, деления на два, четыре и восемь диапазонов с параллельным подбором в каждом диапазоне с помощью программы time. В качестве границ диапазонов использовать значения 231456789, 341256789, 451236789, 561234789, 671234589, 781234569, 891234567. Результаты эксперимента свести в таблицу и проанализировать. Использовать таблицу, аналогичную таблице И.2 приложения И.

3 Результат выполнения работы

3.1 Была написана программа, которая может последовательно находить пароли для любого количества хэшей от 1 до 8 включительно. Хэши паролей считываются из файла. Код программы представлен ниже:

```
#include <iostream>
#include <algorithm>
#include <crypt.h>
#include <string>
#include <fstream>
using namespace std;
void findPass(string pass, const string& hash)
{
    crypt_data *pCryptData = new crypt_data;
    size_t pos = hash.find_last_of('$');
    string hashHead = hash.substr(0,pos);
    do {
                                                string
newHash(crypt_r(pass.data(),hashHead.data(),pCryptData));
        if (newHash == hash) {
```

```
                        cout<<"Hash:  "<<hash<<endl<<"Pass:
"<<pass<<endl;
                 break;
             }
             }  while  (  next_permutation(  pass.begin(),
pass.end() ) );
         delete pCryptData;
     }
     int main(int argc, char **argv)
     {
     ifstream
Read("/home/asic27/Labaratorka7/two/Debug/hash.txt");
         string pass="123456789";
         string x;
         while((Read>>x).good()) {
             findPass(pass,x);
         }
         return 0;
     }
```

3.2 Функция из приложения Е была модифицирована для работы в многопоточном режиме. В функцию был добавлен мьютекс, для защиты критических участков функции — вывода. Был создан статический объект класса mutex, после чего перед операцией вывода используется класс lock_guard. В функции main были созданы два потока, в которых работает функция. Код программы представлен ниже:

```
#include <iostream>
#include <algorithm>
#include <crypt.h>
#include <string>
#include <fstream>
#include <thread>
#include <chrono>
#include <mutex>
using namespace std;
void findPass(string pass, const string& hash)
{
    crypt_data *pCryptData = new crypt_data;
    size_t pos = hash.find_last_of('$');
    string hashHead = hash.substr(0,pos);
    do {
```

```
                                                        string
newHash(crypt_r(pass.data(),hashHead.data(),pCryptData));
            if (newHash == hash) {
                static mutex mtx;
                {
                    lock_guard<mutex> lock(mtx);
                       cout<<"Hash: "<<hash<<endl<<"Pass:
"<<pass<<endl;
                    break;
                }
            }
            } while ( next_permutation( pass.begin(),
pass.end() ) );
        delete pCryptData;
    }
    int main(int argc, char **argv)
    {
        //th_1;
        //th_2;
                                                       ifstream
Read("/home/asic27/Labaratorka7/two/Debug/hash.txt");
        string pass="123456789";
        string x;
        string y;
        thread th_1;
        thread th_2;
        while((Read>>x).good()) {
            //Read>>x;
            th_1=thread(findPass,pass,x);
           if((Read>>y).good())
            th_2=thread(findPass,pass,y);
            if(th_1.joinable())
                th_1.join();
            if(th_2.joinable())
                th_2.join();
        }
        return 0;
    }
```

3.3 Было произведено сравнение скорости работы программы для следующей выборки хэшей:

1.$1$QbqZMX5p$D6Nzw1jjtb82WY/BR19IY0

2.$1$GUKwA9jP$ImrpHbIDPP12h/YSlxjpF1

3.$1$RNFck8WB$o74hlqGIWEaNydcwsTly..

4.$1$1Vh6fxV6$qqPaDiwV9dByIg7E9PAVD/

5.$1$5bYhR71w$aTGmvHDnr7BiT6QtopyRq.

6.$1$2LV5uRQR$2XOUE7FXmlL9eT47.8jnz.

7.$1$Gnh3X9Pq$a1LsX3VVWJA1YzNRe6dv9.

8.$1$QsbE6tnL$o6ohHuu6svwR7Vx6Y8isa1

*Таблица 1 — Результаты измерений скорости работы программ.*

| Количество хэшей | Однопоточная программа | | Многопоточная программа | |
|---|---|---|---|---|
| | User,c | Real,c | User,c | Real,c |
| 1 | 1,7616 | 1,7620 | 1,7632 | 1,7651 |
| 2 | 2,27148 | 2,27273 | 2,27968 | 1,18574 |
| 4 | 3,9140 | 3,9210 | 3,19252 | 1,59503 |
| 8 | 6,6724 | 6,8400 | 6,38480 | 3,52270 |

Как видно из таблицы, скорость работы однопоточной программы для 1 хэша почти равна скорости многопоточной, однако для нескольких хэшей, скорости работы многопоточной программы в 2 раза больше скорости работы однопоточной. На рисунках 1 и2 приведены результаты работы программ.

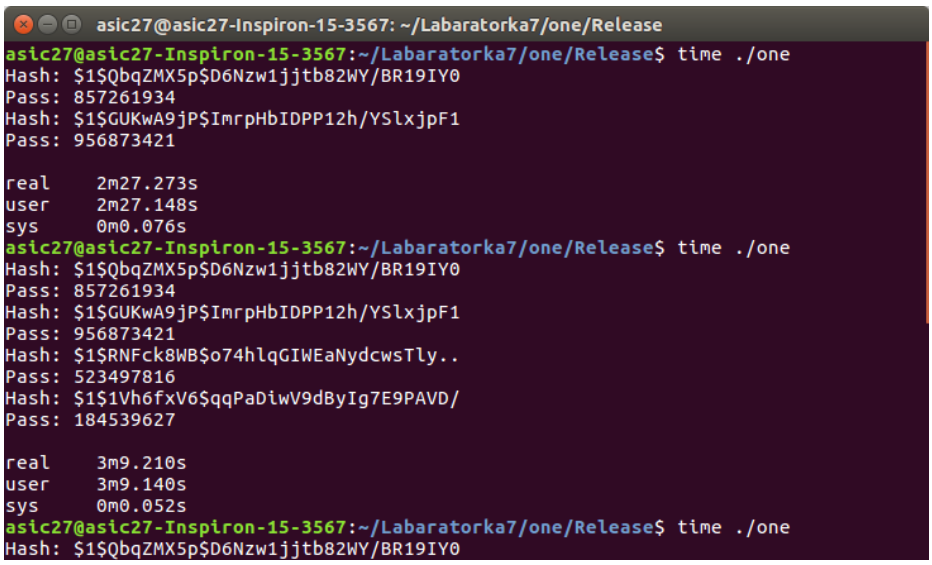


Рисунок 1 -Результаты работы однопоточной программы.

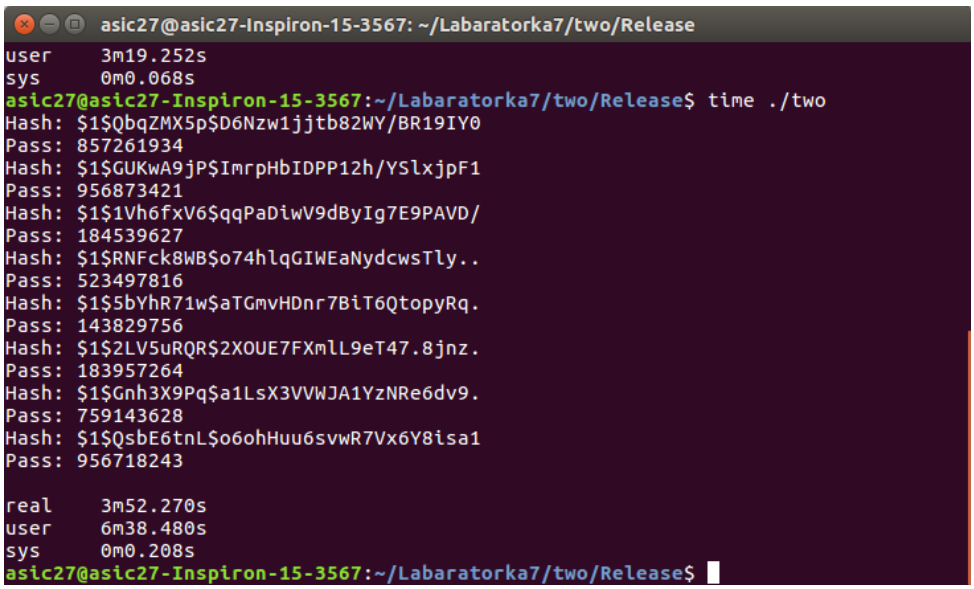


Рисунок 2 — Результаты работы многопоточной программы.

## 4 Вывод

В результате выполнения лабораторной работы были освоены основные компоненты библиотеки C++ для создания многопоточных приложений, были получены практические навыки по использованию класса mutex, lock_guard, thread, методов join().

## ПРИЛОЖЕНИЕ Е

## ФУНКЦИЯ ПОИСКА ПАРОЛЯ ПЕРЕБОРОМ ЗНАЧЕНИЙ

```
    /*
    *  Функция поиска пароля по известному хэшу перестановкой символов пароля
    * Параметры:
    *
```

```
    startPass — начальное значение пароля
    *
    hash - хэш-функция от пароля
    * Функция использует следующие заголовочные файлы:
    *
    algorithm — алгоритмы стандартной библиотеки
    *
    crypt.h — криптографические функции Linux
    */
    void findPass(string pass, const string& hash)
    {
    crypt_data *pCryptData = new crypt_data;
    size_t pos = hash.find_last_of('$');
    string hashHead = hash.substr(0,pos);
    do {
    string
newHash(crypt_r(pass.data(),hashHead.data(),pCryptData));
    if (newHash == hash) {
    cout<<"Hash: "<<hash<<endl<<"Pass: "<<pass<<endl;
    break;
    }
    }    while    (    next_permutation(    pass.begin(),
pass.end() ) );
    delete pCryptData;
    }
```

## ПРИЛОЖЕНИЕ Ж

## ЗНАЧЕНИЯ ХЭШ-ФУНКЦИИ ДЛЯ РАЗЛИЧНЫХ ПАРОЛЕЙ

Длина пароля: 9 символов.

Алфавит пароля: цифры 1, 2, 3, 4, 5, 6, 7, 8, 9.

Дополнительное условие — каждая цифра в пароле встречается ровно один раз.

В таблице Ж.1 приведены примеры значений хэш-функций паролей.

Таблица Ж.1. - Значения хэш-функций паролей

| № | Значение хэш-функции | № | Значение хэш-функции |
|---|---|---|---|
| 1 | $1$h7Skr0Vb$ipc8FG2QEWL88R6MIsJ/10 | 21 | $1$Gnh3X9Pq$a1LsX3VVWJA1YzNRe6dv9. |
| 2 | $1$29fvMdvx$LL.QZs7G3S4Q5ea6.idf11 | 22 | $1$2ZRJ5hDn$4g6I3I3xc7hnTXOZuLdTD1 |
| 3 | $1$37Xhu8sp$jz8AjdBHcqvegV9.PLptt/ | 23 | $1$MdJBM75U$xq8rZyK1ULh/y2tU4gkzk0 |
| 4 | $1$Es9HY7VR$4/WpkGDwKh.wbANw9qZZB0 | 24 | $1$1Vh6fxV6$qqPaDiwV9dByIg7E9PAVD/ |
| 5 | $1$8zbHAF40$VA3.qax9.4qiFZPrvbSA9. | 25 | $1$QsbE6tnL$o6ohHuu6svwR7Vx6Y8isa1 |
| 6 | $1$Bq9uCg8i$5mJ9mGuEzv6TFf4RdCj5u0 | 26 | $1$T3DCQ4fC$PPJ6zkM32y97Uo73rGDw4. |
| 7 | $1$Ok4ymcZA$1jL784kXhgSzhxR2LPw1J/ | 27 | $1$Dcoei9Gg$v2qSQ2NiW/yc2AyfvwRdU1 |
| 8 | $1$Po6qfV6m$e17w/5oU2s3jXE8LvXkMB/ | 28 | $1$GUKwA9jP$ImrpHbIDPP12h/YSlxjpF1 |
| 9 | $1$c2O9StK1$Kqkydq/gxolY/dX90Y6Rw/ | 29 | $1$eHgcBx3e$0e5RvZZQGieOBSeDILGBz/ |
| 10 | $1$vWgvp0lH$ByckSkQb1qIis9pf8uQwP/ | 30 | $1$glIm2JCz$aDdWmtr7k2jfSVH4VjDro. |
| 11 | $1$2LV5uRQR$2XOUE7FXmlL9eT47.8jnz. | 31 | $1$XnGhx94S$inPsacfjVzlJ8vRkqy3Le. |
| 12 | $1$QbqZMX5p$D6Nzw1jjtb82WY/BR19IY0 | 32 | $1$R2fs3oS1$US62dgEyiGmKnh6mTGBW8. |
| 13 | $1$h7Skr0Vb$yOYOrdE.fXX.s4AMEjX9P. | 33 | $1$gaRL2fkT$NpmrF8ZuX60Fj3Adn71X11 |
| 14 | $1$EHE7Lx1a$NSDlTHKAzky.NqJgeyoL60 | 34 | $1$pP1XCkdZ$qDlRuPr4Zc82y5K.uHLy// |
| 15 | $1$nAGbAy3r$EUywXUkf8Uh.DAFYBHO.p. | 35 | $1$laV849Ci$8II//20no5wfXoVGYUPTU/ |
| 16 | $1$tKCHB9Jz$i17M8sfE4tOkCbkOtfB2y. | 36 | $1$pRHzZ7wA$ty8JU0ICd4zq0dURxTkb6/ |
| 17 | $1$rwSst1U3$IKPfxkVVeSksd31EiV3xd/ | 37 | $1$GlCuAo2O$m5dEsCnggil.nJicDkIF/1 |
| 18 | $1$X4iwGPcK$.IuYhAZgQsICFDQw9gA6W/ | 38 | $1$BxMCUB9r$GK61Np7EkmB4miZRORw19/ |
| 19 | $1$5bYhR71w$aTGmvHDnr7BiT6QtopyRq. | 39 | $1$gv1VlMgu$K2IBuyjWuVxhW2J2FiuoH. |
| 20 | $1$26pf8KBS$oLz6obZMna.oe1oxcl/KY1 | 40 | $1$RNFck8WB$o74hlqGIWEaNydcwsTly.. |

# ПРИЛОЖЕНИЕ И
# ФОРМАТЫ ТАБЛИЦ ДЛЯ АНАЛИЗА РЕЗУЛЬТАТОВ

*Таблица И.1. - Результаты сравнения времени подбора для одно- и многопоточных приложений*

| Количество хэшей | Однопоточная программа | | Многопоточная программа | |
|---|---|---|---|---|
| | User,c | Real,c | User,c | Real,c |
| 1 | | | | |
| 2 | | | | |
| 4 | | | | |
| 8 | | | | |

*Таблица И.2. - Результаты сравнения времени подбора одного пароля при распараллеливании задачи*

| Количество диапазонов | Время подбора одного пароля | |
|---|---|---|
| | User,c | Real,c |
| 1 | | |
| 2 | | |
| 4 | | |
| 8 | | |