Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Факультет Безопасности (ФБ)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

ПОТОКИ. СИНХРОНИЗАЦИЯ ПОТОКОВ И ПРОЦЕССОВ.

Отчёт по практической работе № 5 по дисциплине  
«Системное программирование»

Студент группы 726

\_\_\_\_\_\_\_\_\_\_ М.И. Васильева

\_\_\_\_\_\_\_\_\_\_

Принял

к.т.н., доцент кафедры КИБЭВС

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ Я.Е. Мещеряков

оценка

\_\_\_\_\_\_\_\_\_\_

дата

1 Введение

Цель работы: Познакомиться с основными функциями POSIX API для работы с потоками, особенностями потоков в операционных системах Unix.

Ход работы:

1. Ознакомление с теоретическим материалом по работе с потоками

2. Разработка и написание программ на языках Python3 и С++

3. Отладка программ

4. Оформление отчёта

5. Сдача лабораторной работы

Задание:

1. Изучить теоретические сведения, самостоятельно изучить способы

работы с процессами в POSIX.

1. Разработать и написать программы на языке Python3 и С++, которые используют несколько потоков для решения задачи.
2. Создать Dockerfile для выполнения написанных программ в операционной среде Debian.
3. На moodle загрузить архив, содержащий: Отчет, все файлы проекта.
4. Загрузить все необходимые файлы в репозиторий на github.
5. Написать отчет и защитить у преподавателя.

2 Ход работы

В ходе работы были написаны 2 программы на языке Python3 и 1 программа на языке С++ в соответствии с вариантом 8, представленным ниже.

Задание для написания скрипта:

Написать две программы. Первая – вычисляет контрольную сумму файла. Вторая – вычисляет контрольную сумму всех файлов в директории, при этом обработка каждого отдельного файла осуществляется с помощью первой программы в отдельном процессе.

Результат работы программы, реализующей алгоритм высчитывания контрольной суммы файла представлен на рисунке 1.

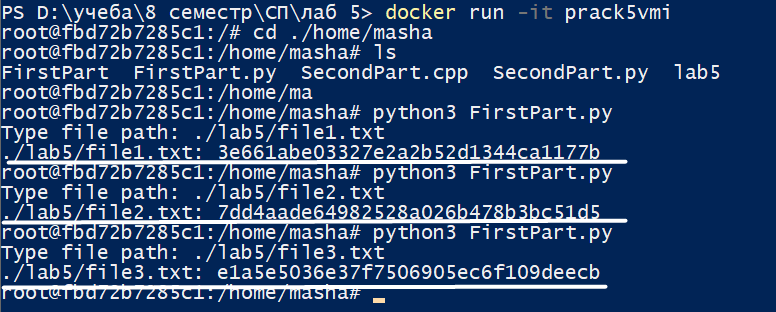


Рисунок 1 – Результат работы первой программы

Результат работы второй программы на языке программирования Python3 программы, реализующей вычисление контрольной суммы всех файлов в директории. Обработка каждого отдельного файла вычисляется с помощью первой программы в отдельном потоке (рисунок 2).

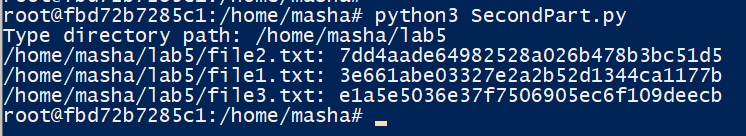


Рисунок 2 - Результат работы второй программы на Python3

Работа 2 программы, написанной на C++ представлена на рисунке 3.

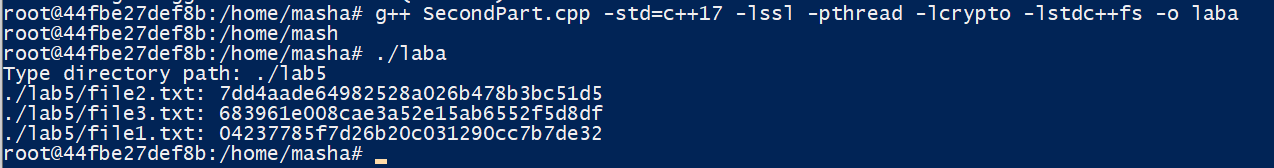


Рисунок 3 – Результат работы 2 программы на C++

3 Заключение

В ходе работы были изучены с основные функции POSIX API для работы с потоками на языке Python и С++, особенностями потоков в операционных системах Unix.

Все необходимые файлы для воспроизведения работы данной практической работы были загружены в репозиторий: https://github.com/mariaigorevna/Threads.

Приложение А

**import** hashlib

**def** hash\_file\_print**(**file\_path**):**

**with** open**(**file\_path**,** 'rb'**)** **as** f**:**

hash\_str **=**3D hashlib**.**md5**()**

**while** **True:**

data **=**3D f**.**read**(**8192**)**

**if** **not** data**:**

**break**

hash\_str**.**update**(**data**)**

**print(**file\_path **+** ": " **+** hash\_str**.**hexdigest**());**

**if** \_\_name\_\_ **=**3D**=**3D "\_\_main\_\_"**:**

file\_path **=**3D input**(**"Type file path: "**)**

hash\_file\_print**(**file\_path**)**

Листинг А.1 – Исходный код первой Python программы

**import** os

**import** threading

**from** FirstPart **import** hash\_file\_print

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

dir\_path **=** input**(**"Type directory path: "**)**

files\_arr **=** os**.**listdir**(**dir\_path**)**

threads\_list **=** **[]**

j **=** 0

**for** file\_name **in** files\_arr**:**

threads\_list**.**insert**(**j**,** threading**.**Thread**(**target**=**hash\_file\_print**,** args**=(**dir\_path**+**'/'**+**file\_name**,)))**

threads\_list**[**j**].**start**()**

j **+=** 1

**for** i **in** range**(**j**):**

threads\_list**[**i**].**join**()**

Листинг А.1 – Исходный код второй Python программы

#include <iostream>

#include <vector>

#include <filesystem>

#include <thread>

#include <unistd.h>

#include <openssl/md5.h>

#include <fstream>

void print\_MD5**(**std**::**string hashMe**,** std**::**string filePath**)**

**{**

int i**;**

MD5\_CTX md5Handler**;**

unsigned char md5digest**[**MD5\_DIGEST\_LENGTH**];**

Листинг А.1 – Исходный код второй C++ программы (Начало)

MD5**((**const unsigned char**\*)**hashMe**.**c\_str**(),** hashMe**.**size**(),** md5digest**);**

printf**(**"%s: "**,** filePath**.**c\_str**());**

**for** **(**i **=** 0**;** i **<** MD5\_DIGEST\_LENGTH**;** i**++)**

**{**

printf**(**"%02x"**,** md5digest**[**i**]);**

**}**

printf**(**"\n"**);**

**}**

int main**()**

**{**

std**::**string path**;**

std**::**string data**,** tempBuf**;**

std**::**vector**<**std**::**string**>** vecPaths**;**

std**::**vector**<**std**::**thread**>** vecThreads**;**

std**::**cout **<<** "Type directory path: "**;**

std**::**cin **>>** path**;**

**for** **(**const auto **&**entry **:** std**::**filesystem**::**directory\_iterator**(**path**))**

**{**

vecPaths**.**push\_back**(**entry**.**path**());**

**}**

**if** **(**vecPaths**.**size**()** **==** 0**)**

**{**

std**::**cerr **<<** "Error: Empty directory\n"**;**

**return** 1**;**

**}**

**for** **(**auto filePath **:** vecPaths**)**

**{**

data **=** ""**;**

std**::**fstream tmpFile**(**filePath**);**

**if** **(**tmpFile**.**is\_open**())**

**{**

**while** **(**getline **(**tmpFile**,** tempBuf**))**

data **+=** tempBuf **+** '\n'**;**

data**.**pop\_back**();**

**try** **{**

vecThreads**.**emplace\_back**(**print\_MD5**,** data**,** filePath**);**

**}** **catch** **(**const std**::**exception**&** e**)** **{**

std**::**cerr **<<** "Error: " **<<** e**.**what**()** **<<** std**::**endl**;**

**}**

**}**

**}**

**for** **(**std**::**thread **&**threadIter **:** vecThreads**)**

**{**

**if** **(**threadIter**.**joinable**())**

threadIter**.**join**();**

**}**

**return** 0**;**

**}**

Листинг А.1 – Исходный код второй C++ программы (Конец)