2022年度
C++プログラミングII

第2回レポート課題

課題

- インターネット上で発生する通信は、①送信元IPアドレス、②送信元ポート番号、③宛先IPアドレス番号、④宛先ポート番号、及び⑤プロトコル番号(※1)の５つの値の組(5-Tupleと呼ぶ)を用いて、一意な通信として識別される。これを通信フローと呼ぶ。

- ここで通信フロー情報が収集済みであり、コンピュータ上のフローテーブルに格納されているものとする。このとき、これらの通信フロー情報を分析するプログラムを次頁の指示に従って作成せよ。

フローテーブルの例

送信元IPアドレス	送信元ポート番号	宛先IPアドレス	宛先ポート番号	プロトコル番号(※1)
172.31.5.64	50500	10.0.10.5	8801	UDP
172.31.5.64	50400	10.0.10.4	8801	UDP
⋮	⋮	⋮	⋮	⋮

(※1)簡単のためプロトコル番号は"TCP"か"UDP"かを識別する文字列が代入される

課題

- 課題1：フローテーブルの先頭に、引数で指定されたフロー情報を格納する関数を実装せよ。
- 課題2：フローテーブルを送信元ポート番号の降順となるようにソートする関数を実装せよ。
- 課題3：HTTPS通信によるフロー数をstd::count_if()を用いてカウントする関数を実装せよ。ここで、HTTPS通信とは、プロトコル番号がTCPでかつ宛先ポート番号が443のものとする。
- 課題4：宛先IPアドレスを引数で受け取り、その宛先IPアドレスと一致する宛先IPアドレスを持つフロー数をstd::count_if()を用いてカウントする関数を実装せよ。
- 課題5：宛先IPアドレスを引数で受け取り、その宛先IPアドレスと一致する宛先IPアドレスを持つフローをstd::find_if()を用いて検索し、標準出力する関数を実装せよ。

netflow.cpp (1)

```
#include <algorithm>
#include <iostream>
#include <string>
#include <vector>

using std::cout, std::string;

class Flow {
    string src_ip;    // 送信元IPアドレス
    string dst_ip;    // 宛先IPアドレス
    string protocol;  // プロトコル番号
    int src_port;     // 送信元ポート番号
    int dst_port;     // 宛先ポート番号

public:
    Flow(string s, string d, string p, int sp, int dp)
        : src_ip{ s }, dst_ip{ d }, protocol{ p }, src_port{ sp }, dst_port{ dp } {}

    string get_src_ip() const { return src_ip; }
    string get_dst_ip() const { return dst_ip; }
    string get_protocol() const { return protocol; }
    int get_src_port() const { return src_port; }
    int get_dst_port() const { return dst_port; }

    void print_flow() const {
        cout << "src ip=" << src_ip << ", dst_ip=" << dst_ip
            << ", protocol=" << protocol << ", src_port=" << src_port
            << ", dst_port=" << dst_port << "¥n";
    }
};
```

netflow.cpp (2)

```
void print(const std::vector<Flow>& ft) {
    for (auto f : ft) {
        f.print_flow();
    }
}

// 課題1：指定したフローを先頭に追加する
auto add_flow(std::vector<Flow>& ft, const Flow& af) {
    関数定義を完成させる
}

// 課題2: フローテーブルを送信元ポート番号の降順にソートする
auto sort_table(std::vector<Flow>& ft) {
    関数定義を完成させる
}
// 課題3：https通信フロー数をカウントする
int count_https(const std::vector<Flow>& ft) {
    関数定義を完成させる

}
```

netflow.cpp (3)

```
// 課題4：指定条件に合致する通信フローをカウントする
int count_ip(const std::vector<Flow>& ft, string dst_ip) {

    関数定義を完成させる

}

// 課題5：指定条件に合致するフローを検索し、出力する
auto dump_flows(const std::vector<Flow>& ft, string dst_ip) {

    関数定義を完成させる

}
```

netflow.cpp (4)

```
int main(int argc, char* argv[]) {
    // フローテーブル
    std::vector<Flow> ft{
        {"172.31.5.64", "10.0.10.2", "tcp", 50200, 443},
        {"172.31.5.64", "10.0.10.3", "tcp", 50300, 443},
        {"172.31.5.64", "10.0.10.4", "udp", 50400, 8801},
        {"172.31.5.64", "10.0.10.5", "udp", 50500, 8801},
        {"172.31.5.64", "10.0.10.7", "udp", 50700, 8801},
        {"172.31.5.64", "10.0.30.8", "tcp", 50801, 443},
        {"172.31.5.64", "10.0.30.8", "tcp", 50802, 443},
        {"172.31.5.64", "10.0.30.8", "tcp", 50803, 443},
    };

    // 課題1：指定したフローを先頭に追加する
    Flow af = { "172.31.5.64", "10.0.10.1", "tcp", 50100, 500 };
    add_flow(ft, af);

    cout << "1: flow table:¥n";
    print(ft);
    cout << "¥n";
```

netflow.cpp (5)

```
    // 課題2: フローテーブルを送信元ポート番号の降順にソートする
    cout << "2:  sorted flow table:¥n";
    sort_table(ft);
    print(ft);
    cout << "¥n";

    // 課題3：https通信フロー数をカウントする
    int cnt_https = count_https(ft);
    cout << "3: flow count of HTTPS: " << cnt_https << "¥n";

    // 課題4：指定条件に合致する通信フローをカウントする
    string target = "10.0.30.8";
    int cnt_filter = count_ip(ft, target);
    cout << "4: flow count of dst_ip=" + target + ": " << cnt_filter <<
"¥n";

    // 課題5：指定条件に合致するフローを検索し、出力する
    cout << "5: flow info of dst_ip=" + target << ":¥n";
    dump_flows(ft, target);

    return 0;
}
```

期待する実行結果

```
1: flow table:
src ip=172.31.5.64, dst_ip=10.0.10.1, protocol=tcp, src_port=50100, dst_port=500
src ip=172.31.5.64, dst_ip=10.0.10.2, protocol=tcp, src_port=50200, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.10.3, protocol=tcp, src_port=50300, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.10.4, protocol=udp, src_port=50400, dst_port=8801
src ip=172.31.5.64, dst_ip=10.0.10.5, protocol=udp, src_port=50500, dst_port=8801
src ip=172.31.5.64, dst_ip=10.0.10.7, protocol=udp, src_port=50700, dst_port=8801
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50801, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50802, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50803, dst_port=443

2:  sorted flow table:
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50803, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50802, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50801, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.10.7, protocol=udp, src_port=50700, dst_port=8801
src ip=172.31.5.64, dst_ip=10.0.10.5, protocol=udp, src_port=50500, dst_port=8801
src ip=172.31.5.64, dst_ip=10.0.10.4, protocol=udp, src_port=50400, dst_port=8801
src ip=172.31.5.64, dst_ip=10.0.10.3, protocol=tcp, src_port=50300, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.10.2, protocol=tcp, src_port=50200, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.10.1, protocol=tcp, src_port=50100, dst_port=500

3: flow count of HTTPS: 5
4: flow count of dst_ip=10.0.30.8: 3
5: flow info of dst_ip=10.0.30.8:
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50803, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50802, dst_port=443
src ip=172.31.5.64, dst_ip=10.0.30.8, protocol=tcp, src_port=50801, dst_port=443
```

課題1の出力

課題2の出力

課題3の出力

課題4の出力

課題5の出力

レポートの構成

- 表紙
  - タイトル（C++プログラミングII　第2回レポート課題）
  - 学籍番号、氏名、提出年月日
- 本文
  1. レポート課題の説明
  2. 課題1の解答（関数の実装コード及び処理内容の説明）
  3. 課題2の解答（関数の実装コード及び処理内容の説明）
  4. 課題3の解答（関数の実装コード及び処理内容の説明）
  5. 課題4の解答（関数の実装コード及び処理内容の説明）
  6. 課題5の解答（関数の実装コード及び処理内容の説明）
  7. 感想など

提出方法

- Course Powerの「**2022年度第2回レポート課題**」内にあるレポート「第2回レポート課題」」から以下を提出する
  - レポートファイル（**pdf**形式）
  - 作成したソースコード

- レポートファイル名はreport2-**学籍番号**-**氏名**.pdfとする
  - **学籍番号**と**氏名**のところは、自分のものに置き換える

- 締め切り：**7月3日(日) 23:59**