Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

Osztott rendszerek specifikációja és implementációja

IP-08bORSIG

Dokumentáció az *1.* beadandóhoz

Szalóki Sándor
H8L59S

2017. október 26.

1. Kitűzött feladat

A harcoknak évek óta vége, törékeny béke uralkodott a bolygón. Míg korábban az emberek hősei és megmentői voltunk, akik segítettek a rászorulókon, most az ügynökeinket (tévesen) háborús bűnökkel vádolják, így inkognitóban kell élniük az életüket. A konfliktus azonban egyre fokozódik, így a bújkálásaitoknak véget kell vetni, ehhez azonban megbízható formáját kell választanunk a kommunikációnak.

A kiküldött üzenetek tartalmára vonatkozóan külön titkosítás a protokollban nincsen, mivel annak megfelelő értelmezése egy előre definiált fogalomrendszer mentén jelenti a pontos információkat (azaz mások számára önmagában értelmetlen vagy érdektelen), ehhez azonban a kapott adat megbízhatósága létfontosságú.

Ennek érdekében két féle jelből áll össze a pontos információ: A főparancsnokság (HQ) által, és csak tőlük kibocsátott ellenőrző kód (hash), valamint a kiküldött üzenet.

Minden ügynökünk tisztában van az ellenörző kódot előállító művelettel, így a kapott üzenetre ezt alkalmazva meg tud bizonyosodni arról, hogy azt menet közben nem módosították. A számüzetés miatt azonban nem áll rendelkezésükre a korábbi csúcstechnológia, így csupán a lassabb hardverek biztosította környezeten tudnak dolgozni. Annak érdekében, hogy minél hamarabb tudják a kapott információt hitelesíteni, megfelelő szoftveres támogatásra lesz szükségük, melyet maguknak kell implementálni. Örömmel konstatálták, hogy a számítógépekben a processzor több maggal rendelkezik, ezáltal több végrehajtási szálat is támogat egyidőben, így párhuzamosítva az említett módszert lényegesen gyorsabban juthatnak el céljaikhoz.

Adott egy input fájl, melyben bizalmas adatokat tartalmazó szöveget találhatunk. Arról azonban nem tudunk közvetlen meggyőződni, hogy az információk helyesek és nem kerültek módosításra, így a kriptográfiában megismertek módjára hashelnünk kell az adatokat, hogy ellenőrizhessük, hogy az a hivatalos ellenőrzőkóddal megegyezik e, hogy nyugodtan támaszkodjunk az ott olvasottra.

Az inputfájl felépítése az alábbi: Az első sorban egy N, nemnegatív egész szám olvasható, ezt követően összesen N sornyi szöveges információt találhatunk. Arra vonatkozóan, hogy egy sorban pontosan hány szó található, nincs közvetlen információnk, de mindegyikben legalább 1, és legfeljebb 100.

Egy szó hashértéke a benne szereplő betűk hashértékének összegeként áll elő. Egy sor/szöveg hashértéke a benne szerplő szavak hashértékének szóközzel vett konkatenációjaként definiálható.

Feltehetjük, hogy a szövegben az angol ábécé betűit használó szavak (tehát magyar/orosz/koreai stb. készletbe tartozó karakterek nem), valamint általános írásjelek (pont, vessző, kérdő- és felkiáltójel, aposztróf, kettőspont stb.) találhatóak.

Annak érdekében, hogy hatékonyan működjön a programunk, a hasht ne szavanként, hanem soronként állítsuk elő, így egyszerre több sorhoz tartozó értéket párhuzamosan tudunk kiszámolni. A program olvassa be az adatokat, majd N folymatot indítva számítsa ki az egyes sorokhoz tartozó hash-értékeket, majd az így kapott adatokat írja ki az

"output.txt" fájlba.

2. Felhasználói dokumentáció

A program feltételezi, hogy a futtatható fájl mellett jelen van egy input.txt, melynek a formátuma helyes. A program futása végén az eredményt egy output.txt fájlba írja.

2.1. Rendszer-követelmények, telepítés

A programunk több platformon is futtatható, dinamikus függősége nincsen, bármelyik, manapság használt PC-n működik. Külön telepíteni nem szükséges, elég a futtatható állományt elhelyezni a számítógépen.

2.2. A program használata

A program a generált .exe fájl futtatásával használható akár intézőből is, ekkor a fájl melletti input.txt fájl tartalmát dolgozza fel. A program indítható command prompt-ból is, akkor adható paraméterként fájl, amit fel kívűnunk dolgozni.

A futtatható állomány mellett kell elhelyezni az *input.txt* nevű fájlt, mely a bemeneti adatokat tartalmazza, a fenti specifikációnak megfelelően. Figyeljünk az ebben található adatok helyességére és megfelelő tagolására, mivel az alkalmazás külön ellenőrzést nem végez erre vonatkozóan. A futás során az alkalmazás mellett található *output.txt* fájl tartalmazza a kapott eredményt, ahol az i-ik sor a bemeneti fájl i-ik sorának szavainak hash értékei állnak.

3. Fejlesztői dokumentáció

A program nagyban épül a C++ 11-es STL függvényeire. A gyorsaság érdekében a program soronként párhuzamosítja a számolást, erre std::future-t használunk. Az egyszerűség érdekében funkcionális programozásból ismert map, illetve fold műveleteket az std::transform, illetve az std::accumulate függvényekkel alkalmazzuk.

3.1. Megoldási mód

A feladatot több részre osztjuk. Első lépésben beolvassuk az adott fájlból a sorokat. A beolvasott sorokat párhuzamosítva soronként feldolgozzuk. A szálon a számolást megvárjuk, majd összeszedjük a hash-elt eredményeket. Miután minden sor kész, kiírjuk az eredményt az output.txt fájlba.

3.2. Implementáció

Első lépésben a sorokat beolvassuk egy `std::vector<std::string>` vectorba. Következő lépésben használva az `std::transform` függvényt létrehozzuk az `std::vector<std::future<std::str` vectort. Ebben a vectorban a sorokra alkalmazzuk a hash függvényt. A hash függvény működését 2 lépésben érjük el. A sor minden szavára alkalmazzuk a hash függvényt, majd ezeket egymás mellé írjuk. Szavak hashelésére `std::accumulate` függvényt használjuk. Következő lépésben megvárjuk a szálakon a hash végét, majd kiírjuk az eredményt az output.txt fásjlba.

3.3. Fordítás menete

A programunk forráskódját a `main.cpp` fájl tartalmazza. A fordításhoz elengedhetetlen egy `C++11` szabványt támogató fordítóprogram a rendszeren. Ehhez használhatjuk az *MSVC*, *g++* és *clang* bármelyikét. A fordítás menete (4.9.2-es verziójú g++ használata esetén) a következő: `'g++ main.cpp -std=c++11'`. A speciális, `-std=c++11` kapcsoló azért szükséges, mert alapértelmezés szerint ez a verziójú fordítóprogram még a régi, C++98-as szabványt követi, melyben a felhasznált nyelvi elemek még nem voltak jelen.

3.4. Tesztelés

A program tesztelésésre egy egyszerű test scriptet írtam, amely kihasználja, hogy a program paraméterül várhat egy fájlt. Egy 550 ezer karakteres fájlt módosítottam különböző tesztesetek előállítására. A sorok számával megadható a szálak száma, így 1-től 12-ig elkészítettem a megfelelő sorú fájlokat. (Ekkor az 1 sor szimulálja a szekvenciális esetet).

Szálak száma	Másodperc
1	1.91041
2	1.0498
3	0.887191
4	0.702312
5	0.897408
6	0.672741
7	0.687076
8	0.610146
9	0.633025
10	0.618567
11	0.669199
12	0.630693