Projekt do předmětu ISA:

Filtrující DNS resolver

Login: xdanca01

Jméno: Petr

Příjmení: Dančák

Rok: 2020/21

Obsah

[1. Úvod 3](#_Toc53164605)

[2. Návrh a implementace 3](#_Toc53164606)

[2.1. Zpracování argumentů 3](#_Toc53164607)

[2.2. Server - start 3](#_Toc53164608)

[2.3. Server – navázání komunikace 3](#_Toc53164609)

[2.4. Server – komunikace s DNS serverem 3](#_Toc53164610)

[3. Návod na použití 4](#_Toc53164611)

[3.1. Příklady spuštění 4](#_Toc53164612)

# Úvod

Cílem projektu je implementace DNS resolveru, který bude poslouchat na zadaném portu, přeposílat dotazy na zadaný server a filtrovat zprávy, které budou pod/doménou nějaké z domén ve specifikovaném souboru.

# Návrh a implementace

Projekt se skládá z vícero jazyků, které můžeme rozdělit do 2 skupin. Hlavní část, tedy DNS resolver je naprogramován v jazyce c++ a tvoří ho jediný soubor main.cpp. Druhou část tvoří jednotlivé testy napsané ve skriptovacím jazyce bash.

## Zpracování argumentů

Zpracování argumentů probíhá ve funkci main a k implementaci jsou použity knihovny ***stdlib.h*** a ***string.h*** Předané argumenty se postupně parsují konstrukcí if, else, která využívá funkcí ***strcmp()*** a ***strtol()***. Tyto funkce se používají pro porovnání očekávaných argumentů s předanými a pokud se některý argument neshoduje, tak dojde k vrácení chybového kódu, který je v případě parsing 3, a ohlášení chyby na standardní chybový výstup.

## Server - start

Hlavní část serveru se také vyskytuje ve funkci main, ve které server začíná otevřením socketu funkcí ***socket()*** s parametry ***AF\_*INET** (makro pro IPv4), ***SOCK\_DGRAM*** (makro pro UDP), ***17*** (označení protokolu UDP) a jeho následovným nastavením na odposlech i odesílání (funkcí ***setsockopt()***) a nastavením portu, který je defaultně při nezadání argumentu port nastaven na hodnotu 53 a musíme ho zároveň převést z host byte order do network byte orderu funkcí ***htons()***. Dále socketu nastavíme, ať poslouchá na všech rozhraních hodnotou ***INADDR\_ANY***, kterou musíme převést z lokálního byte orderu do síťového byte order funkcí ***htonl()***. Přiřazení portu, rozhraní na kterém má poslouchat a jestli se jedná o IPv4, nebo IPv6 se přiřadí funkcí ***bind()***. Po nastavení socketu si deklaruji určitý počet vláken, který je daný makrem ***THREADS***, který jsem si defaultně nastavil na 100. Jakmile server vstoupí do nekonečného loopu, tak čeká na příchozí komunikaci.

## Server – navázání komunikace

Jakmile server obdrží nějaká data, tak si uloží adresu s portem odesílatele a zkontroluje typ požadavku. Server podporuje pouze typ požadavku ***A*** a na všechny ostatní odpovídá přímo odesílateli erorem typu ***NOTIMP***, který je popsaný v RFC a vrátí se na začátek loopu, kde opět čeká na navázání komunikace. Jestliže server dostane požadavek typu ***A***, tak zkontroluje obsah dat, jestli doména, kterou máme přeložit se nevyskytuje v souboru s filtrovanými doménami, a nebo jestli není poddoménou některé z domén v souboru. Pokud je dotaz filtrovaný, tak se opět odpovídá zpět odesílateli a to s erorem typu ***REFUSED*** a pokud dojde k internímu erroru, tak se odpovídá erorem typu ***SERVFAIL***. Jinak se vytvoří struktura pomocí funkce ***malloc()*** a ta se vyplní potřebnými parametry pro zpracování v jiné funkci. Jakmile se struktura naplní, tak se vytvoří nové vlákno, které začne funkcí ***send\_next()*** a dostane vytvořenou strukturu.

## Server – komunikace s DNS serverem

Pro komunikaci s DNS serverem se vždy vytvoří nový socket, který dostane náhodný port, aby jednotlivá vlákna mohla nezávisle komunikovat. Socketu se přiřadí komunikace oběma směry a zároveň hodnota timeoutu, po kterou čekáme na odpověď od DNS serveru. Data pošleme DNS serveru na výchozí port 53 a čekáme na odpověď po dobu timeoutu. Jestliže po tuto dobu nám nepřijde odpověď ze strany serveru, tak odpovíme zpět odesílateli s erorem typu ***SERVFAIL***, uvolníme alokovanou pamět a ukončíme práci vlákna. Jinak vrátíme odesílateli odpověď od DNS serveru.

# Návod na použití

Pro správné přeložení programu je nutné mít Makefile a složku DNS ve stejném adresáři. Program se přeloží commandem ***make*** a smaže se commandem ***make clean***. Program očekává dva povinné argumenty ***‘-s’***, za kterým následuje ip adresa serveru, na který se mají přeposílat dotazy a argument ***‘-f’***, za kterým se specifikuje cesta k souboru s doménami, které se mají filtrovat. Třetí argument je volitelný a zadá se pomocí přepínače ‘-p’ a za ním se specifikuje port, na kterém má program komunikovat. Záleží na pořadí argumentů, které je “-s -p -f”.

## Příklady spuštění

make

main -s <server> [-p <port>] -f <soubor\_s\_filtry>

make test

make clean