COMP.CS.410 Käyttöjärjestelmät

Harjoitustyö	70 whois

1. Tehtäväkuvaus

Tee TCP-asiakasohjelma, joka lähettää RFC3912:n mukaisen pyynnön palvelimelle whois.ficora.fi ja tulostaa saamansa vastauksen.

2. Työn taustaa

Harjoitustyössä tutkittava GNU/Linux komentorivikomento on nimeltään “whois”. Tämän komennon perusperiaate on, että sen yhdistäminen palvelinnimeen komentorivillä aiheuttaa kyselyn, josta käy muun muassa sen nimen mukaisesti ilmi, kuka verkkosivua ylläpitää.

2.1 Komennon käyttö komentoriviltä

Työssä kysely osoitetaan palvelimelle ficora.fi, eli Liikenne- ja viestintävirasto Traficomin sivulle. Komentorivillä käytettynä kysely tapahtuisi siis seuraavasti:

```
whois ficora.fi
```

Tämä komento tuottaa harjoitustyön tekohetkellä seuraavanlaisen lopputuloksen:

```
domain.............: ficora.fi
status.............: Registered
created............: 29.6.2001 09:25:00
expires............: 31.8.2027 00:00:00
available..........: 30.9.2027 00:00:00
modified...........: 16.10.2022 16:34:12
RegistryLock.......: no
```

Nameservers

nserver.....: ns2.traficom.fi [87.239.125.187] [2a00:13f0:0:1002:125:184:0:3] [OK]

nserver.....: ns-secondary.funet.fi [128.214.248.132] [2001:708:10:55::53] [OK]

nserver.....: nsp.dnsnode.net [OK]

nserver.....: ns2.z.fi [OK]

nserver.....: ns1.z.fi [OK]

nserver.....: ns1.traficom.fi [87.239.125.186] [2a00:13f0:0:1002:125:184:0:2] [OK]

DNSSEC

dnssec.....: signed delegation

DS Key Tag 1.....: 64365

Algorithm 1.....: 10

Digest Type 1.....: 2

Digest 1.....: 524596E1F7B203F3D15596811531FF4F461F0343976B9236B0186DF575D46B23

DS Key Tag 2.....: 12891

Algorithm 2.....: 10

Digest Type 2.....: 2

Digest 2.....: F69462AADFB1CF0096E1539146E85385E170A64848B1284F844029ACD0D9644C

Holder

name.....: Liikenne- ja viestintävirasto Traficom

register number....: 2924753-3

address.....: PL 320

postal.....: 00059

city.....: TRAFICOM

country.....: Finland

phone.....:

holder email.....:

```
Registrar
registrar.....: Liikenne- ja viestintävirasto

>>> Last update of WHOIS database: 21.10.2022 15:00:12 (EET) <<<
Copyright (c) Finnish Transport and Communications Agency Traficom
```

2.2 RFC3912

RFC:t ovat IETF:n julkaisemia standardidokumentteja, jossa määritellään Internetin protokollia sekä erilaisia käytäntöjä. Tehtävän vaatima whois protokolla on määritelty syyskuussa 2004 RFC3912:ssa. Tämän RFC:n luvussa 3 kuvataan protokollan toimintaa:

```
  client                                          server

open TCP    ---- (SYN) ------------------------------>
            <---- (SYN+ACK) -------------------------
send query  ---- "Smith<CR><LF>" -------------------->
get answer  <---- "Info about Smith<CR><LF>" ---------
            <---- "More info about Smith<CR><LF>" ----
close       <---- (FIN) ------------------------------
            ----- (FIN) ------------------------------>
```

Toimintamalli on siis seuraavanlainen: Asiakas (client) avaa TCP yhteyden (SYN) palvelimelle (server). Palvelin vastaa takaisin (SYN+ACK). Asiakas tietää palvelimen kuuntelevan, ja voi lähettää sille haluamansa kyselyn (query). Palvelin vastaa asiakkaalle antamalla tälle tietoa kohteesta, jonka jälkeen se lähettää lopetusviestin (FIN). Asiakas lopettaa kyselyn vastaavalla viestillä (FIN).

3. Ohjelman toiminta

Ohjelma etenee hyvin suoraviivaisesti. Käynnistettäessä se luo uuden socketin, jonka kautta asiakas ja palvelin voivat olla yhteydessä. Socketin protokollaksi valitaan TCP, kuten tehtäväohjeistuksessa pyydettiin. Seuraavaksi se konfiguroi palvelimen tiedot yhteydenmuodostusta varten. Mikäli kaikki tiedot on onnistuneesti syötetty eikä tule ohjelman ulkopuolisia virheitä vastaan, avataan yhteys asiakkaan ja palvelimen välille.

Asiakas lähettää pyynnön, ja ohjelma tallentaa palvelimelta tulevan vastauksen. Vastaus tulostetaan näytölle, ja socket-yhteys katkaistaan.

Harjoitustyössä toteutettu ohjelma siis jäljittelee luvun 2.2 toimintamallia. Käydään tässä kappaleessa ohjelman osat sekä niiden toiminta kuitenkin yksityiskohtaisesti läpi.

3.1 Ohjelmaan tuodut kirjastot

Ohjelman toiminta riippuu pitkälti siihen tuotujen kirjastojen sisältämistä funktioista ja struktuureista. Käytetyt kirjastot ovat määritelty koodin ensimmäisillä riveillä, ja koodiin itsessään on kirjoitettu selventävät kommentit, mitä funktiota tai struktuuria mistäkin kirjastosta on käytetty.

```
1
2   #include<stdio.h>        // Default output module
3   #include<string.h>       // Function: strlen
4   #include<sys/socket.h>   // Functions: socket, connect, recv, send
5   #include<netinet/in.h>   // Struct: sockaddr_in
6   #include<arpa/inet.h>    // Functions: inet_ntoa, inet_addr
7   #include<netdb.h>        // Struct: hostent
8   #include<unistd.h>       // Function: close
9
```

Ohjelma hyödyntää sockaddr_in ja hostent struktuureja kirjastoista <netinet7in.h> ja <netdb.h>. Selvennetään näiden tarkoitusta hieman.

Sockaddrin on struktuuri, johon tallennetaan tiedot web-palvelimesta: palvelun portin numero, IP-osoite sekä protokolla, joka määrittelee minkä tyyppinen struktuurin olio on.

Hostent on ohjelmassa käytetty apustruktuuri, jonka kautta palvelimen nimi saadaan käännettyä IP-osoitteeksi.

3.2. Toiminta

Ohjelman ajo alkaa sen ainoasta funktiosta, main():sta. Rivillä 16 määritellään valmiiksi bufferit, client_message ja server_reply. Tämän jälkeen luodaan asiakkaan socket rivillä 19. Socket luodaan kirjaston <sys/socket.h> funktiolla. Funktio tarvitsee toimiakseen kolme erilaista parametria: AF_INET, joka kertoo socketin käyttävän protokollaa “Internet domain”, SOCK_STREAM, joka kertoo yhteyden olevan stream-tyyppinen, ja IPPROTO_TCP, joka kertoo socketin käyttävän TCP-protokollaa.

Harjoitustyön tekemisessä hyödynnettiin tietoa IBM:n sivuilta:

Kun client socket käyttää SOCK_STREAM tyyppiä, tulee ohjelman hyödyntää seuraavaa ketjua:

1. Yhdistetään palvelin socketiin connect() funktion avulla
2. Lähetetään kysely send() funktiolla
3. Vastaanotetaan palvelimen viesti recv() funktiolla
4. Suljetaan asiakas socket close() funktiolla

```
10  int main(void)
11  {
12
13      printf("COMP.CS.410: TCP CLIENT FOR QUERYING TRAFICOM AT FICORA.FI\n");
14
15      // Initialize buffers for messages
16      char client_message[100] , server_reply[1500];
17
18      // Create a client socket
19      int client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
20
21      if(client_socket < 0){
22          printf("The program was unable to create a socket!\n");
23          return -1;
24      }
25
```

Mikäli socket on saatu onnistuneesti luotua, alkaa ohjelma alustamaan struktuuria whois palvelimelle rivillä 27. Whois tarvitsee toimiakseen haettavaa sivustoa vastaavan päätteen. Harjoitustyön tapauksessa haetaan suomalaista sivustoa, ficora.fi:tä, joten pääte tulee olla .fi. Jotta whois.fi nimi voidaan kääntää IP-osoitteeksi, tulee hyödyntää hostent apustruktuuria sen tekemiseen. Ip tallennetaan sitten erilliseen muuttujaan ip rivillä 29.

Jotta ohjelma voi yhdistää palvelimeen, tarvitsee sitä käyttävä funktio struktuurista sockaddr_in luodun olion. Määritellään tähän olioon jälleen protokolla AF_INET, hostent struktuurista haettu ip-osoite ip (ensiksi tämä viedään inet_addr funktion läpi, jotta siitä saadaan yhteensopiva) sekä whois-palvelun käyttämä portti, eli 43.

```
// Create structs and fill them with required parameters
struct hostent *hostname_whois;
hostname_whois = gethostbyname("whois.fi"); // .fi ending is required when querying
char* ip = inet_ntoa(*((struct in_addr*) hostname_whois->h_addr_list[0]));

struct sockaddr_in server_info;
server_info.sin_family = AF_INET;
server_info.sin_addr.s_addr = inet_addr( ip );
server_info.sin_port = htons( 43 ); // port number, whois service runs on port 43
```

Viimein ohjelman rivillä 39 päästään avaamaan yhteys. Kuten aikaisemmin todettiin, connect() funktio tarvitsee toimiakseen sockaddr olion “server_info”, mutta lisäksi se tarvitsee myös aikaisemmin luodun socketin “client_socket” sekä tiedon, kuinka “pitkä” socket on. (Harjoitustyön aikana tätä ihmettelin, mutta en saanut kunnollista selvyyttä, miksi tämä vaaditaan).

```
printf("\nConnecting to the server..");

// Connecting to whois server for query
if(connect( client_socket, (const struct sockaddr*) &server_info, sizeof(server_info) ) < 0)
{
    printf("\nConnection to the server failed.");
}
else{
    printf("\nConnection to the server succeeded.");
}
```

Tässä kohtaa ohjelmaa yhteys on muodostettu, eli SYN ja SYN+ACK viestit on vaihdettu asiakkaan ja palvelimen välillä. Jälleen, mikäli yhteyden avaaminen ei onnistu, ohjelma lopettaa toimintansa.

Seuraavaksi, rivillä 51 ohjelmassa muodostetaan viesti, eli kysely, joka palvelimelle halutaan lähettää. Ohjelman ollessa pelkästään ficora.fi:n tutkimista varten, on sen osoite kovakoodattu kyselyyn. Jatkokehitystä miettien tällaiset parametrit voisi vaihtaa käyttäjältä haettaviksi inputeiksi.

Rivillä 54 asiakas pääsee lähettämään kyselyn palvelimelle. Funktio send() käyttää jälleen socketia “client_socket” kyselyn “client_message” lähettämiseen. Näiden lisäksi se vaatii tiedon kyselyn pituudesta, sekä tiedon, mitä lippuja funktiossa käytetään. Tässä tapauksessa lippuja ei käytetä, joten merkitään viimeiseksi parametriksi 0.

```
    // Construct the client message
    sprintf(client_message , "%s\r\n" , "ficora.fi");    Call to function
    // Send a query message to the server
    if( send(client_socket, client_message, strlen(client_message), 0) < 0)
    {
        printf("\nSending the client message failed!");
    }
```

Viestin lähetettyään ohjelma jää kuuntelemaan palvelimen vastausta käyttäen recv() funktiota. Tämä toimii hyvin vastaavasti kuin send(), mutta bufferi “client_message” tulee korvata bufferilla “server_reply”, johon vastaanotettu viesti tallennetaan.

```
58
    // Saving the received reply to the corresponding buffer
    recv(client_socket, server_reply, sizeof(server_reply), 0);

    // Print out the message
    printf("%s", server_reply);

    // Close the socket after receiving the reply
    close(client_socket);

    return 0;
}
```

Lopuksi ohjelma tulostaa vastaanotetun viestin, nyt siis saman mitä luvussa 2.1. Jotta socket ei jää käyntiin, tulee se vielä sulkea close() funktiolla.

3. Loppusanat ja pohdinta

Valitsin harjoitustyön aiheen, koska sen tehtäväkuvaus kuulosti mielenkiintoiselta.

Olin käyttänyt aikaisemmin whois komentoa tietoturvaan liittyvissä tehtävissä, sekä kirjoittanut samantyyppisen ohjelman aikanaan pythonilla. C kielellä kirjoittaminen osoittautui kuitenkin haastavammaksi, sillä C kieli ei ole oma vahvuus. Pidin kuitenkin tehtävän tekemisestä, ja koen, että ymmärrän sockettien toimintaa nyt paremmin.