Tugas Besar

IF3097 Jaringan Komputer

File Transfer Protocol

Deadline: 7 Desember 2012 Pukul 23:59

Pengantar

File Transfer Protocol atau yang lebih dikenal dengan FTP bukanlah server seperti yang dikenal secara umum melainkan protokol yang digunakan untuk mentransfer data dari server ke komputer yang lain melalui koneksi TCP. Protokol ini sering digunakan untuk server yang difungsikan untuk menyimpan data saja.

Tugas kali ini adalah mengimplementasikan protokol FTP sederhana di lingkungan Linux menggunakan Socket API. Bahasa yang digunakan adalah C. Peserta dilarang menggunakan C++ dan semua *library* terkait network programming yang merupakan *wrapper* untuk Socket API, seperti modul network programming Qt, ACE, atau boost::asio.

Secara garis besar ada empat komponen yang perlu diperhatikan pada tugas ini, yaitu client, server, pesan komunikasi, dan protokol transfer *file*.

Spesifikasi

Client dan server masing-masing merupakan program yang diimplementasikan menggunakan Socket API Linux dan berasosiasi dengan satu direktori di *filesystem* program tersebut dijalankan. Direktori tersebut merupakan direktori *root* dari client atau server. Manipulasi data terjadi di direktori tersebut. Pada tugas ini client yang terkoneksi pada server hanya berjumlah satu buah.

Server

Server mengerti pesan sesuai format yang terdefinisi dari client yang terkoneksi ke dirinya. Server harus membalas pesan tersebut sesuai dengan format yang terdefinisi. Server dapat dimatikan dengan perintah **shutdown**. Ketika perintah tersebut dilakukan koneksi dengan client yang terhubung harus diputus terlebih dahulu.

Secara *default* direktori *root* server adalah direktori ketika server dinyalakan.
Direktori *default* dapat diganti dengan parameter -d dan memberikan lokasi direktori *root* ketika menjalankan server. Contoh: ./server -d /home/user/n-train

Client

Client adalah program yang dapat menerima masukan pengguna untuk memasukkan pesan-pesan dengan format terdefinisi yang akan dikirimkan ke server. Proses koneksi ke server dilakukan dengan membuat koneksi TCP ke server. Setelah terhubung ke server yang ditandai dengan pesan yang sesuai, client baru dapat melakukan transfer data.

Client dapat dimatikan dengan perintah **shutdown**. Ketika perintah tersebut dilakukan koneksi ke server harus diputus terlebih dahulu. Client dapat mengganti current working directory dengan perintah **cd** <pathname>. Jika *pathname* valid, current working directory client pindah ke *pathname*. Jika *pathname* tidak valid, current working directory client memberikan pesan kesalahan di terminalnya.

Secara *default* direktori *root* client adalah direktori ketika client dinyalakan.
Direktori *default* dapat diganti dengan parameter -d dan memberikan lokasi direktori *root* ketika menjalankan client. Contoh: ./client -d /home/user/seeya

Pesan

Komponen

Dua buah komponen yang perlu ada di dalam pesan yang dikirimkan

1. **code**: kode yang membedakan masing-masing pesan yang dikirim; 4 digit untuk pesan yang dikirimkan oleh client dan 3 digit untuk pesan balasan dari server; dan
2. **parameter**: bagian pesan yang menampung pesan-pesan yang membutuhkan parameter seperti *pathname*.

Setiap pesan yang dikirimkan dan diterima **harus** ditampilkan di terminal client atau server.

Client

Pesan yang dikirimkan oleh client selalu berisi code dengan 4 huruf kapital

1. CONNECT (CONN) -> CONN <ip-address>

   Pesan ini digunakan untuk melakukan koneksi TCP ke server. Argumen pesan ini adalah alamat IP dari server. Jika client tidak dapat melakukan koneksi ke server dalam waktu 10 detik, client menampilkan pesan "server not found" di terminalnya.
2. RETRIEVE (RETR) -> RETR <pathname>

   Pesan ini menyebabkan server mengirimkan salinan *file* yang dispesifikasikan di *pathname* yang merupakan argumen pesan ini. Jika *pathname* tidak valid, server akan mengirimkan pesan kesalahan. Jika *pathname* valid, server akan mengirimkan pesan transfer ok dan mentransfer *file* ke current working directory client. Jika ada *file* bernama sama di current working directory client, *file* tersebut akan ditimpa dengan data baru.
3. STORE (STOR) -> STOR <pathname>

   Pesan ini menyebabkan client mengirimkan salinan berkas yang dispesifikasikan di *pathname* yang merupakan argumen pesan ini. Jika *pathname* tidak valid, client memberikan pesan kesalahan di terminalnya. Jika *pathname* valid, client akan mentransfer *file* ke current working directory server. Jika ada *file* bernama sama di current working directory server, *file* tersebut akan ditimpa dengan data baru.
4. LOGOUT (QUIT) -> QUIT

   Pesan ini menterminasi sesi dan memutuskan koneksi ke server. Jika mengerjakan bonus dan transfer *file* sedang terjadi sesi akan diterminasi setelah transfer selesai.

5. LIST (LIST) -> LIST [pathname]

   Pesan ini membuat server mengirimkan daftar direktori dan *file* di pathname. Argumen *pathname* opsional. Jika *pathname* tidak diberikan, server mengirimkan daftar direktori dan *file* di current working directory server.
6. CHANGE WORKING DIRECTORY (CWD) -> CWD <pathname>

   Pesan ini membuat server mengganti current working directory server.

Server

Pesan balasan dari server selalu berupa pesan dengan code 3 digit angka

1. 150

   Pesan ini merupakan balasan untuk pesan yang terdefinisi di client selain pesan CONN, QUIT, dan CWD jika *pathname* yang diberikan valid. Client harus menunggu satu balasan lagi sebelum dapat mengirimkan pesan lain. Setelah pesan ini dikirimkan proses transfer data terjadi.
2. 250

   Pesan merupakan pesan yang dikirimkan setelah server selesai mengirimkan atau menerima data.
3. 200 <comment>

   Pesan ini merupakan balasan untuk operasi CONN, QUIT, dan CWD yang berhasil. Argumen *comment* diisi dengan komentar apa yang berhasil dilakukan dalam bahasa Inggris.

   Untuk pesan CONN, *comment* diisi dengan “connected to <ip-address>”.

   Untuk pesan QUIT, *comment* diisi dengan “connection to <ip-address> closed”.

   Untuk pesan CWD, *comment* diisi dengan “directory changed to <pathname>”.
4. 500

   Pesan ini merupakan balasan untuk pesan yang tidak terdefinisi di pesan client di atas.
5. 501

   Pesan ini merupakan balasan untuk kesalahan pada argumen pesan secara khusus pada tugas ini adalah pada *pathname*. Pesan ini dikirimkan jika *pathname* yang dikirimkan oleh client tidak valid di server

Catatan
Current working directory adalah lokasi direktori saat ini pada server atau klien.

Pathname dapat dimulai dari *root* dengan karakter '/' atau dari current working directory server. Jika *pathname* dimulai dengan karakter '/', *pathname* tersebut mengacu ke *root* direktori server atau client. Jika tidak, *pathname* tersebut mengacu ke current working directory server atau client. Sebagai contoh jika current working directory client berada di /document dan client mengirimkan pesan STOR halo.txt, *file* yang dimaksud mempunyai *pathname* /document/halo.txt. Jika client mengirimkan pesan STOR /image/halo.jpg, *file* yang dimaksud berada di direktori image dan bernama halo.jpg.

Jika *pathname* tidak valid di sisi server, server akan mengirimkan balasan bahwa *pathname* tidak valid. Jika *pathname* tidak valid di sisi client, client akan menampilkan pesan *pathname* tidak valid di terminalnya.

Protokol Transfer File

Tugas ini memerlukan operasi I/O untuk melakukan transfer *file*. *File* yang dikirimkan tidak hanya berupa *file* teks, tetapi juga dapat berupa *file binary*. Jadi, jangan gunakan mesin karakter atau mesin kata untuk membaca dan menulis *file*. Gunakanlah fungsi-fungsi *library* C standar untuk membaca dan menulis *file*.

Operasi pengiriman *file* berukuran kecil tidak akan menjadi masalah. Masalah akan terjadi ketika *file* yang dikirimkan berukuran besar. Oleh karena itu, kirimkanlah *file* bagian per bagian. Protokol pengiriman *file* per bagian ini diserahkan kepada peserta. Modifikasi protokol ini dapat mempengaruhi kecepatan pengiriman. Akan menarik jika protokol ini melibatkan multi thread pada pembacaan *file* dan pengiriman data.

Bonus

Bonus yang dapat dikerjakan hanya satu, yaitu client dan server tidak terblok ketika melakukan transfer data. Ketika transfer data terjadi, client dan server masih dapat menerima masukan dari pengguna.

Pada sisi client hal di atas dapat diimplementasikan dengan menggunakan dua koneksi ke server. Koneksi pertama digunakan untuk mengirimkan pesan dan menerima balasannya. Koneksi kedua digunakan untuk mentransfer data. Kedua koneksi tersebut harus dijalankan pada dua thread berbeda. Pada sisi server hal di atas dapat diimplementasikan dengan memiliki dua buah thread berbeda untuk menerima masukan pengguna dan untuk menerima koneksi dari client.

Bonus ini dapat diimplementasikan di client atau server saja. Jika bonus ini diimplementasikan di client dan server berarti server memiliki dua thread untuk menerima masukan pengguna dan untuk menerima koneksi dari client. Thread untuk menerima koneksi dari client akan membuka dua koneksi dengan client dimana masing-masing koneksi berjalan di thread masing-masing pada client.

Semua cara implementasi yang diberikan hanya merupakan tips. Implementasi yang dapat dilakukan tidak terbatas pada tips yang telah diberikan.

Laporan

Komponen yang harus ada di laporan adalah penjelasan mengenai implementasi client, server, protokol transfer *file*, dan *screenshot* jalannya program.

Deliverable

Tugas dikirimkan ke situs http://kuliah.informatika.org. Yang perlu dikumpulkan adalah sebuah direktori terkompres bernama nomor kelompok masing-masing yang berisi:
bin: berisi executable program client dan program server;
src: berisi source code program client dan program server;
doc: berisi readme cara menjalankan program dan laporan.
Deadline tugas ini tanggal 7 Desember 2012 pukul 23:59

Demo dan Kompetisi

Demo dilakukan mulai dari tanggal 10 Desember 2012. Selain pemeriksaan kelengkapan fitur sesuai spesifikasi, akan ada kompetisi kecepatan pengiriman *file* menggunakan *file* yang disediakan oleh asisten. Kelompok yang mentransfer *file* tersebut dengan waktu yang paling singkat akan mendapatkan sesuatu dari asisten.