Structuur van computer programma's II
Opgave project shttpd

Dirk Vermeir
Departement Informatica
Faculteit Wetenschappen
Vrije Universiteit Brussel

Juni 2002

Inhoudsopgave

1 Inleiding: shttpd **2**

2 Achtergrond: client-server **2**

3 Opgave: shttpd **3**

4 Functionele eisen shttpd **3**
4.1 Aanroepen, configuratie 3
4.2 Het HTTP protocol 3
4.3 Formaat log file 4

5 Niet-functionele eisen **5**

6 Vragen **6**

7 Indienen **6**

1 Inleiding: shttpd

(Deze tekst is ook beschikbaar als een postscript bestand **stru2-2001b.ps**. dat bvb. op wilma kan afgedrukt worden).

`Shttpd` is een eenvoudig **http** server programma dat een subset van het HTTP/1.0 protocol implementeert.

2 Achtergrond: client-server

Het programma's maakt gebruik van de normale internet protocollen en standaarden: tcp/ip,dns. Zeer summier:

- Een machine op het net heeft een numeriek *adres*, meestal voorgesteld als een rij van 4 bytes. Bvb. is het internet adres van tinf2 134.184.65.2.

- D.m.v. de *DNS* service kunnen een of meerdere namen geassocieerd worden met een internet adres. Bvb. is de officiële naam van de machine met adres 134.184.65.2 *tinf2.vub.ac.be*. Zulke namen zijn hiërarchisch gestructureerd (de machine "*tinf2*" behoort tot het *domein* "*vub.ac.be*" dat zelf weer een deeldomein is van "*ac.be*", hetwelk een deel is van het "top" domein "*be*". Men kan bvb. d.m.v. het "nslookup" programma nagaan wat het adres is dat hoort bij een naam (of omgekeerd).

- Een machine op het net heeft een aantal *poorten* waarlangs een verbinding kan gemaakt worden met een poort van een andere machine. Sommige poorten worden, bij conventie, gebruikt voor bepaalde diensten. Bvb. wordt de poort nummer 80 veelal gebruikt door web servers. Wanneer bvb. konqueror (of netscape) vanop de machine 134.184.65.2 (tinf2) een verbinding wenst te maken naar de webserver op elvas.vub.ac.be (adres 134.18.65.31) met de web server van tinf2, dan wordt er een verbinding geopend tussen een poort, bvb 8026, op 134.184.65.2 en poort 80 op 134.18.65.31. D.m.v. het tcp protocol kunnen dan data uitgewisseld worden tussen de toepassingen aan beide kanten van de verbinding.

Voor het netwerk aspect kan gebruik gemaakt worden van het **dvnet** pakket dat o.a. een klasse **`Socket`** (voor clients) en **`SocketServer`** (voor servers definieert). Omdat **`Socket`** afgeleid is van **`iostream`** is het sturen en ontvangen van gegevens via een netwerkverbinding heel makkelijk.

De directory **demo** bevat een volledig uitgewerkt voorbeeld van een "echo server", dit is een server die elke lijn gestuurd door een client terugstuurt, na de tekst te hebben omgezet naar hoofdletters.

3 Opgave: shttpd

1. Maak een eenvoudig web server programma "**shttpd**" dat voldoet aan de onderstaande functionele[1] en niet-functionele[2] eisen ("requirements").

2. Beschrijf kort uw ontwerp in een apart document; waarvan u een afdruk bezorgt aan het secretariaat van de vakgroep vóór de uiterste indiendatum.

4 Functionele eisen shttpd

4.1 Aanroepen, configuratie

Het programma kan op de volgende manier opgeroepen worden

```
shttpd naam-van-configuratie-file
```

wat resulteert in een **daemon** proces dat als eenvoudige webserver functioneert. Het programma kan enkel gestopt worden dmv het "kill" bevel.

Het configuratie bestand bevat een aantal lijnen van de vorm **`naam=waarde`** waarbij **`naam`** en **`waarde`** één van de betekenissen uit tabel 4.1 kan hebben

naam	waarde
documentroot	absoluut pad naar een directory waaronder de documenten die kunnen aangeboden worden zich bevinden
portnr	nummer van de poort gebruikt door de server
logfile	volledig pad naar logfile
typesfile	volledig pad naar "types" bestand

Tabel 1: Configuratie elementen

4.2 Het HTTP protocol

- Het HTTP/1.0 protocol wordt volledig beschreven in het bijgevoegde "RFC[3]" **document** (nummer 1945).

[1] Functionele eisen leggen vast **wat** het programma moet doen.

[2] Niet-functionele eisen leggen vast aan welke andere voorwaarden het programma moet voldoen, bvb. onder welk operating system het moet werken, welke programmeertalen of bibliotheken mogen gebruikt worden, etc.

[3] Request For Comments.

- Merk op dat **shttpd** `Full-Request` en `Full-Response` dient te implementeren. Echter, enkel **GET** en **HEAD** requests moeten minimaal geimplementeerd worden.
- De volgende headers moeten niet ondersteund worden: Allow, Authorization, Content-Type van de client, Content-Encoding, WWW-Authenticate.
- Voor het bepalen van het type van de inhoud van een bestand mag uitgegaan worden van de suffix van de naam van dat bestand. De mapping van suffixes naar types moet opgeslagen worden in een "types" bestand waar elke lijn van de vorm

  suffix type

  is.
- Enkel het aanbieden van "gewone" files moet ondersteund worden,

4.3 Formaat log file

Voor elk verzoek wordt precies één lijn aan de logfile toegevoegd. Zo'n lijn heeft een formaat zoals gellustreerd in Figuur 1.

```
igwe.vub.ac.be - - [12/Feb/2000:13:18:49 +0100] "GET /images/openssl_ics.gif HTTP/1.0"200 2063
134.184.65.2 - - [01/Nov/2000:15:42:44 +0100] "GET /ssl/kiesoos.cgi?rolnr=58769&stjcode=5L10021 HTTP/1.0"302 300
```

Figuur 1: Voorbeeld lijnen in apache logbestand

Een lijn bestaat dus uit een aantal velden gescheiden door spaties. De betekenis van de velden is als volgt:

1. De naam of het adres van de machine van waaruit het verzoek werd gestuurd. Sommige lijnen zullen hier een domeinadres bevatten.
2. Een niet ingevulde veld (−).
3. Een niet ingevulde veld (−).
4. Het tijdstip van het verzoek, tussen vierkante haken (`[]`).
5. Het verzoek zelf, omringd door accenten (`"`). Dit verzoek bestaat zelf weer uit een aantal delen, gescheiden door spaties:
   (a) Het "werkwoord": `GET`, `PUT`, `HEAD`.
   (b) Een padnaam, evt. gevolgd door formuliergegevens na een vraagteken.

(c) Een aanduiding van het protocol (typisch `HTTP/1.0` of `HTTP/1.1`).

6. Een code die het resultaat geeft. De mogelijke codes zijn beschreven in RFC1945.

7. Het aantal bytes dat werd teruggestuurd naar de aanvrager.

5 Niet-functionele eisen

1. Het programma moet volledig en uitsluitend in C++ geschreven worden.

2. Het programma moet gebaseerd zijn op een **object-georienteerd** ontwerp. Functies die langer zijn dan 30 lijnen zullen met argwaan worden bekeken.

3. Het ontwerp moet zodanig zijn dat delen zo veel mogelijk herbruikbaar zijn.

4. Het programma moet correct werken op **wilma**.

5. Enkel de C++ (of C) standaard library en de **dvnet** en **dvutil** libraries mogen gebruikt worden. In dvutil bevinden zich diverse klassen (bvb Date, File) die vermoedelijk in dit project nuttig kunnen zijn.

6. **Eigen implementaties van containers zijn niet toegestaan, enkel standaard library containers zoals map, set, vector, mogen gebruikt worden.**

7. De source (C++) bestanden dienen de suffix ".cc", ".C" of ".cpp" te gebruiken. De "header files" met declaraties dienen de suffix ".h" te gebruiken.

8. Alle source bestanden, samen met een "Makefile", dienen zich in een directory "structuur2/project2001b" te bevinden. De volgende reeks van bevelen zal resulteren in correct uitvoerbare programma's "shttpd" in dit directory:

```
make clean
rm -f shttpd *.o
make all
```

9. Het programma moet *robuust* zijn. Dit wil zeggen dat het ook een redelijke actie moet ondernemen indien de input extreem of foutief is.

6 Vragen

- Zoals steeds kan je vragen en opmerkingen sturen naar **o4556@elvas.vub.ac.be**.
- Indien je zeker wil zijn van een goede aanpak, dan kan je het ontwerp (welke klassen die welke functies ondersteunen) naar **stru2@tinf.vub.ac.be** sturen voor commentaar.

7 Indienen

De uiterste indiendatum (de source code en Makefile worden automatisch opgehaald uit het directory **$HOME/structuur2/project2001b**, zie punt 8 in sectie 5) is maandag 2/9/2002, 23:59.