Structuur van computer programma's II
Opgave project "schat"

Dirk Vermeir
Departement Informatica
Faculteit Wetenschappen
Vrije Universiteit Brussel

Eerste semester academiejaar 2002-2003

Inhoudsopgave

1 Inleiding: "simple chat" **2**

2 Achtergrond: client-server **2**

3 Achtergrond: multi-threading **3**

4 Eisen **3**

5 Functionele eisen schatd **4**
5.1 Aanroepen, configuratie 4
5.2 Gebruikers 4
5.3 Het schat protocol 5
5.4 Formaat log file 6

6 Functionele eisen schat **6**

7 Niet-functionele eisen **7**

8 Hints **8**

9 Vragen **9**

10 Indienen **9**

1 Inleiding: “simple chat”

(Deze tekst is ook beschikbaar als een postscript bestand **stru2-2002.ps**[1]. dat bvb. op wilma kan afgedrukt worden).

`Schat` is een eenvoudig netwerk “chat” systeem om meerdere gebruikers te laten communiceren via een gemeenschappelijk “bord” (dat enkel tekst bevat). Concreet bestaat het systeem uit een “server” (**schatd**) en een “client” (**schat**) programma. Een server proces accepteert connecties van verschillende client processen.

2 Achtergrond: client-server

De programma’s maken 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. mozilla (of 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), 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**[2] pakket dat o.a. een klasse **`Socket`** (voor clients) en **`SocketServer`** (voor servers

[1] `stru2-2002.ps`.

[2] `http://tinf2.vub.ac.be/~dvermeir/software/dv/index.html`.

definieert). Omdat **`Socket`** afgeleid is van **`iostream`** is het sturen en ontvangen van gegevens via een netwerkverbinding heel makkelijk.

De directory **demo**[3] 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 Achtergrond: multi-threading

Omdat bvb. de schat client tegelijkertijd zowel moet "luisteren" naar het toetsenbord (voor bevelen van de gebruiker) en naar eventuele input van de server is het het eenvoudigst om deze activiteiten te implementeren via twee verschillende "threads"[4].

Voor deze toepassing volstaat het een thread te beschouwen als een proces dat zijn address space deelt met een ander proces (dit is trouwens min of meer de implementatie van threads in linux). De uitvoering van een thread wordt verder geregeld zoals voor een proces; een multi-threaded programma kan dus verschillende activiteiten gelijktijdig uitvoeren. Omdat threads dezelfde address space delen is er natuurlijk een probleem om gemeenschappelijke data structuren te beschermen tegen gelijktijdige toegang. Een simpele oplossing gebruikt daarvoor zgn. "monitors", dat zijn stukken code die slechts door één thread tegelijk kunnen uitgevoerd worden.

Voor "threads" en "monitors" zijn klassen beschikbaar in de **dvthread**[5] bibliotheek (Java ondersteunt gelijkaardige klassen). Op die pagina staat ook een programma dat het gebruik van threads en monitors illustreert om concurrente toegang tot een gedeelde buffer te regelen voor verschillende "reader" en "writer" threads.

4 Eisen

1. Maak eenvoudige "chat" server (**schatd**) en client (**schat**) programma's die voldoen aan de onderstaande functionele[6] en niet-functionele[7] eisen ("requirements").

[3] `demo`.

[4] Idem dito voor de server die naar verschillende sockets tegelijk moet kunnen luisteren.

[5] `http://tinf2.vub.ac.be/~dvermeir/software/dv/index.html`.

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

[7] 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.

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

5 Functionele eisen schatd

5.1 Aanroepen, configuratie

Het programma kan op de volgende manier opgeroepen worden

```
schatd naam-van-configuratie-file
```

wat resulteert in een **daemon**[8] proces dat het "schat protocol" ondersteunt.

Het configuratie bestand bevat een aantal lijnen van de vorm **`naam=waarde`** waarbij **`naam`** en **`waarde`** één van de betekenissen uit tabel 1 kan hebben.

naam	waarde
root	paswoord server administratie
portnr	nummer van de poort gebruikt door de server
logfile	volledig pad naar logfile
usersfile	naam bestand met gebruikers info

Tabel 1: Schatd configuratie elementen

Bovendien wordt een lijn die begint met het "#" teken beschouwd als commentaar die door het programma niet moet verwerkt worden.

5.2 Gebruikers

De "usersfile" bevat één lijn per gebruiker, met de volgende layout:

```
naam paswoord commentaar
```

Hier bestaan zowel **`naam`** als **paswoord** uit zowel letters als cijfers (de naam "*root*" kan niet gebruikt worden). De diverse elementen worden gescheiden door spaties of tabs. Het **`commentaar`** veld loopt door tot het einde van de lijn.

Bovendien wordt een lijn die begint met het "#" teken beschouwd als (algemene) commentaar die kan worden overgeslagen.

[8] `http://tinf2.vub.ac.be/~dvermeir/software/dv/dvutil/html/namespaceDv_1_1Util.htm`.

5.3 Het schat protocol

De server accepteert bevelen (via het netwerk) met een syntax zoals getoond in tabel 2.

bevel	=	**join** name passwd
	|	**leave**
	|	**leave** name
	|	**stop**
	|	**write** line
	|	**who**
	;	

Tabel 2: Bevelen aan de server

- Bij een **`join`** bevel zal de server de verbinding associeren met de name, indien deze refereert naar een gekende gebruiker (of root) en het juiste paswoord bevat. Input van een socket waar nog geen gebruiker mee geassocieerd is zal als een fout behandeld worden.
- Bij een **`leave`** bevel zal de server de verbinding sluiten en noteren dat de gebruiker niet meer actief is.
- Een **`leave name`** bevel kan enkel door de *root* gebruiker gegeven worden. Het heeft hetzelfde effect alsof de gebruiker in kwestie het **`leave`** bevel had gebruikt.
- Bij een **`who`** bevel zal de server een lijst tonen van de namen van de gebruikers die op dat moment actief zijn.
- Het **`stop`** bevel kan enkel door de *root* gebruiker gegeven worden. Na dit bevel sluit de server alle connecties en stopt het programma.
- Het **`write`** bevel wordt gevolgd door een spatie en willekeurige tekst tot het einde van de lijn. De server zal de tekst naar alle verbonden gebruikers sturen, voorafgegaan door de naam van de gebruiker die de tekst heeft gestuurd, bvb.

```
fred:  dit is een test
```

5.4 Formaat log file

Voor elk verzoek zal **schatd** precies één lijn aan de logfile toevoegen. Zo'n lijn heeft een formaat zoals gellustreerd in Figuur 1 ("`schatd.conf`" is de naam van het configuratiebestand van de server).

```
2002-07-12 10:04:13 schatd.conf start
2002-07-12 11:00:10 schatd.conf fred joins
2002-07-12 11:59:13 schatd.conf fred writes> a message
...
2002-07-12 11:00:10 schatd.conf fred leaves
2002-07-12 10:04:13 schatd.conf stop
```

Figuur 1: Voorbeeld lijnen in logfile

6 Functionele eisen schat

Het **schat** programma kan op de volgende manier opgeroepen worden

```
schat naam-van-configuratie-file
```

Het configuratie bestand bevat een aantal lijnen van de vorm **naam=waarde** waarbij **naam** en **waarde** één van de betekenissen uit tabel 3 kan hebben.

naam	waarde
hostname	waar schatd loopt
portnr	nummer van de poort gebruikt door schatd
name	gebruiker naam voor "join" bevel
passwd	paswoord voor "join" bevel

Tabel 3: Schat configuratie elementen

Bovendien wordt een lijn die begint met het "#" teken beschouwd als commentaar die door het programma niet moet verwerkt worden.

De aanroep resulteert in een interactief programma dat voortdurend de output van de server toont, behalve wanneer op één van de toetsen "**w**", of "**c**" gedrukt wordt. Dan verschijnt een prompt waarna de gebruiker één lijnt tekst kan ingeven die als volgt wordt verwerkt:

- Indien op "**w**" werd gedrukt, dan wordt de lijn naar de server gestuurd als een "write" bevel.

- Indien op "**c**" werd gedrukt, dan wordt de lijn ongewijzigd naar de server gestuurd. Dit geeft de mogelijkheid om bvb. een "leave" of "who" bevel in te voeren.

Figuur 2 illustreert de output van **schat** (de gebruiker is "lisa").

```
/home/lisa$ schat ~/schat.config
lisa> join
jan> message-1
write: message 2
lisa> message-2
jan> message-3
jan> leave
command: leave
/home/lisa$
```

Figuur 2: Voorbeeld schat sessie

7 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, volgens de richtlijnen in hoofdstuk 11 van het handboek. 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. Bovendien moet het programma zo ontworpen worden dat het makkelijk kan uitgebreid worden. We denken hier bvb. aan één server die verschillende "conversaties" beheert waaraan gebruikers eventueel tegelijkertijd kunnen deelnemen. Het protocol zou ook kunnen uitgebreid worden zodat nieuwe conversaties kunnen opgestart worden, de status van de server en de aanwezige gebruikers kan worden opgevraagd etc.
4. Het programma moet correct werken op **wilma**[9].
5. Enkel de C++ (of C) standaard library en de **dvnet**[10], **dvutil**[11] en **dvthread**[12]

[9] `http://wilma.vub.ac.be/`.
[10] `http://tinf2.vub.ac.be/~dvermeir/software/dv/index.html`.
[11] `http://tinf2.vub.ac.be/~dvermeir/software/dv/index.html`.
[12] `http://tinf2.vub.ac.be/~dvermeir/software/dv/index.html`.

bibliotheken mogen gebruikt worden. In dvutil bevinden zich diverse klassen (bvb. **`Date`**, **`File`**, **`Tty`**) en functie (bvb. **make_daemon**) 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/project2002" te bevinden. De volgende reeks van bevelen zal resulteren in correct uitvoerbare programma's "schat" en "schatd" in dit directory:

```
make clean
rm -f schatd schat *.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.

8 Hints

- Configuratie files kunnen gelezen worden d.m.v de **`Props::Props(istream&)`** constructor uit de **dvutil**[13] bibliotheek.

- Één socket object gebruiken in verschillende threads kan problemen geven, vermoedelijk omdat sommige I/O functies niet reentrant zijn. Indien men bvb. wil dat thread **`a`** probeert te lezen van een socket terwijl thread **`b`** op dezelfde socket wil schrijven dan is het ten zeerste aan te raden dat de threads aparte kopieën van de socket gebruiken. Cfr. de copy constructor voor de **`Socket`** klasse.

- Documentatie voor het ontwerp kan automatisch gegenereerd worden uit de header files (en de commentaar daarin) d.m.v. het **doxygen**[14] programma

[13] `http://tinf2.vub.ac.be/~dvermeir/software/dv/index.html`.
[14] `http://www.doxygen.org/`.

dat beschikbaar is op wilma (voor een afdruk wordt best de latex ouput genomen, maar er wordt ook een html versie gegenereerd). Voorbeeld doxygen configratie files zijn bvb. te vinden bij de source distributie van de **dv**[15] libraries.

Een minimale "doxygen" commentaar is hieronder te zien.

```
/** A simple server class.
 * Lots more detailed comments
 */
class Server {
public:
 /** Constructor.
  * A more detailed description.
  * @param portnr of port on which  server listens
  * @exception runtime_error if construction fails.
  */
  Server(const string& hostname) throw (runtime_error);
  /** A simple function. More comments.
   * @param i a parameter
   * @return some result
   */
  int f(int i);
};
```

9 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.

10 Indienen

De uiterste indiendatum (de source code en Makefile worden automatisch opgehaald uit het directory **$HOME/structuur2/project2002**, zie punt 8 in sectie 7) is zondag 5/1/2003, 23:59. Een nette afdruk van het ontwerpdocument

[15] `http://tinf2.vub.ac.be/~dvermeir/software/dv`.

moet vóór die datum op het secretariaat van de vakgroep afgegeven worden. Later volgt dan een (individuele) mondeling verdediging.