Semestrální práce z předmětu KIV/UPS

Chatovací systém

Tomáš Maršálek, A10B0632P
marsalet@students.zcu.cz

30. prosince 2012

1 Zadání

7. Chatovací systém. Realizujte programy serveru a klienta pro chatování. Chatovaní server a klient bude podporovat přihlášení uživatele pod přezdívkou, komunikaci s ostatními uživateli, pouze s jedním definovaným uživatelem a odhlášení uživatele. Protokol bude obsahovat příkazy LOGIN, LOGOUT, ALL_MSG, PRIV_MSG, USERS, PING a odpovědi OK a ERR.

2 Chatovací protokol

2.1 Přihlášení a odhlášení

Po navázání spojení se serverem je nutné se přihlásit, aby si uživatele server přidal do seznamu.

Formát přihlašovací zprávy je:

```
LOGIN <jméno>
```

Za uživatelské jméno bude použito první slovo které následuje za $LOGIN$. Není možné používat jméno o více slovech. Pokud se přihlášení podařilo, tzn. jméno již není používáno nebo nedošlo k jiným potížím, dostaneme odpověď OK a od této chvíle můžeme používat všechny funkce protokolu. V případě chybové odpovědi ERR můžeme opakovat přihlášení nebo ukončit spojení. K chatovací funkcionalitě protokolu se ale nedostaneme přes správné přihlášení.

Pro odhlášení použijeme zprávu:

```
LOGOUT
```

Pokud ukončíme spojení bez poslání této zprávy, server tento stav rozpozná a uživatel je odstraněn ze seznamu uživatelů i bez korektního odhlášení. Uživatelské jméno tak není zbytečně obsazeno.

2.2 Formát vyměňovaných zpráv

Uživatelé mohou komunikovat prostřednictvím veřejné nebo soukromé zprávy. Veřejná zpráva je rozeslána všem uživatelům mimo odesílatele. Ve svém klientovi tedy nedostane duplikátní zprávu.

```
ALL_MSG <zpráva>
```

Veřejnou zprávu odešleme jednoduše pomocí tohoto dotazu. Zpráva může mít maximální délku 2000 znaků, záleží na nastavení serveru.

Soukromou zprávu odešleme podobným příkazem, jen dodáme jméno příjemce.

```
PRIV_MSG <příjemce> <zpráva>
```

Chceme-li si ověřit spojení se serverem, pošleme zprávu $PING$. Server jednoduše odpoví zprávou OK.

Důležitou součástí protokolu je zjištění přihlášených uživatelů. Mohlo by se zdát, že jde pouze o dodatečnou informaci, ale pro posílání soukromých zprávu se jedná o nezbytnou funkcionalitu. Jednoduše pošleme požadavek na seznam uživatelů.

```
USERS
```

Odpovědí je prostý seznam přihlášených uživatelů bez jakýchkoliv dodatečných informací.

```
Pepa
Magda
Honza
Jirka
Luboš
Václav
Jiřina
```

3 Implementace

3.1 Server

Server je C program využívající knihovnu Berkeley socketů a vláknovou knihovnu pthread.

Při inicializaci je vytvořen socket, který je připojen na výchozí nebo předem zvolený port. Celý program je jeden nekonečný cyklus, který je obsluhován funkcí *select*. Ta v každé iteraci cyklu vybere ten socket, ze kterého přišla zpráva. V případě tohoto serveru existují dva typy socketů. Jedním z

nich je naslouchající socket, který přijímá nová spojení. Navázanému spojení je přiřazen druhý typ socketu - uživatelský, ten odpovídá právě jednomu uživateli. Server si uchovává seznam všech přihlášených uživatelů, jejich uživatelské jméno a uživatelský socket, přes který s ním komunikuje. Pokud *select* vybere naslouchající socket, pouze vytvoří nový uživatelský socket a server čeká, dokud z něj nepřijde požadavek o přihlášení. Po úspěšném přihlášení je přidán do seznamu přihlášených uživatelů. V opačném případě, kdy *select* vybere uživatelský socket, server vyhodnotí požadavek a patřičně se zachová podle výše uvedeného protokolu.

3.2 Datové struktury

Kromě uchovávání seznamu uživatelů je program relativně jednoduchý v ohledu uchovávání dat. Data o uživately jsou uchována ve struktuře *user*, která nese údaje o jeho přihlašovacím jménu a socketu, přes který s ním probíhá komunikace. Díky funkci *getpeername*() zjistíme IP adresu až když ji potřebujeme, není třeba ji uchovávat, ale samozřejmě by to bylo možné. Všichni užavetelé jsou uchováni ve spojovém seznamu. Není očekáváno veliké množství přihlášených uživatelů, proto složitější struktury by ani neměly příliš velký dopad na efektivitu. V praxi se na vytížených chatovacích serverech setkáme s počtem uživatelů v řádech stovek, pro vyšší počty už by bylo vhodné použít stromové nebo hashovací struktury.

4 Logování

4.1 Hlavní log

Pro logování požadavků je vyhrazen speciální soubor *server.log*, který loguje každý požadavek jako čas požadavku, IP adresu žadatele a zprávu požadavku.

4.2 Statistika

Server si uchovává údaje o počtu přenesených bytů, počtu přenesených zpráv, počtu navázaných spojení, počtu úspěšných i neúspěšných přihlášení, době běhu a počet přenosů zrušených kvůli chybě. Tuto informaci zjistíme v interaktivním módu.

5 Použité nástroje a prostředí

Obě aplikace klienta i serveru byly vyvíjeny pod systémem GNU/Linux. Klient byl vyvinut za pomoci IDE Eclipse, pro které byla prvotně vyvinuta GUI knihovna SWT. Server a makefile byly napsány ve Vimu.

6 Uživatelká příručka

Abychom vyzkoušeli tento chatovací protokol, musíme spustit server na určitém portu a následně můžeme připojit libovolné množství klientů.

6.1 Server

6.1.1 Přeložení

Server je napsán v jazyce C pro Unixové prostředí. V adresáři se zdrojovými soubory přeložíme buď pomocí build skriptu

```
$ make
```

nebo ručně

```
$ gcc -pthread -o server *.c
```

6.1.2 Spuštění

Server spustíme bez parametrů, bude použit výchozí port 1234. Port si můžeme zvolit spuštěním serveru s parametrem portu.

```
$ ./server
$ ./server 12345
```

6.1.3 Ovládání

Po spuštění bude server běžet na pozadí interaktivního programu představenému uživateli. Pomocí něj může nenásilně ukončit server, získat informace o vyměněných datech nebo získat seznam právě přihlášených uživatelů.

6.2 Klient

Klient je program s grafickým uživatelským rozhraním v jazyce Java a využívá GUI knihovnu SWT.

6.2.1 Přeložení

6.2.2 Spuštění

Protože se jedná o Java aplikaci, spuštění provedeme z konzole

```
$ java -jar client.jar
```

6.2.3 Ovládání

Při spuštění je uživateli představeno okno s jedinou možností - *Connection*, kde vybere položku *Connectto*. Po zadání parametrů serveru (adresa, port a uživatelské jméno) a potvrzení je při zadání korektních údajích a ještě nepoužitého uživatelského jména úspěšně připojen a rovnou přihlášen, nemusí se starat o záležitosti jako $LOGIN$ a $LOGOUT$.