ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

SEMESTRÁLNÍ PRÁCE Z

KIV/UPS

Úvod do počítačových sítí

Síťová počítačová hra Logo quiz

Martin Hron - A11B0379P
hronmar@students.zcu.cz

5. ledna 2014

Obsah

1 Zadání

Vytvořte program realizující síťovou hru Logo quiz. Vytvořte server, který bude obsluhovat více hráčů i her současně, minimální počet hráčů je 3. Vytvořte klienta, který bude komunikovat se serverem a bude schopen reagovat na výzvy od serveru. Komunikace mezi klientem a serverem bude probíhat pomocí protokolu TCP.

1.1 Zásady pro vypracování

- Úlohu naprogramujte v programovacím jazyku C anebo Java. Pokud se jedná o úlohu server/klient, pak klient bude v Javě a server v C.
- Výstupy serveru budou v alfanumerické podobě, klient může komunikovat i v grafice (není podmínkou).
- Server řešte pod operačním systémem Linux, klient může běžet pod OS Windows XP. Emulátory typu Cygwin nebudou podporovány.
- Server musí být schopen obsluhovat požadavky více klientů souběžně.
- V případě použití nespojovaných služeb (UDP) vyřešte na úrovni aplikačního protokolu problematiku ztráty příp. duplicity dat (např. číslování, metoda okénka, apod.).
- Každý program bude doplněn o zpracování statistických údajů (přenesený počet bytů, přenesený počet zpráv, počet navázaných spojení, počet přenosů zrušených pro chybu, doba běhu apod.).

2 Specifikace

U klienta není patrné, zda-li se pro připojení do hry má zadávat IP adresa a port jako parametry při spuštění, nebo až následně na výzvu v GUI. V obou možnostech je samozřejmě ověřována validita. Proto jsem si určil možnost obou řešení. Ze zadání také není patrné, zda-li se má klient připojovat pouze přes IP adresu nebo přes hostname a proto jsem si určil, že se budu připojovat pouze přes IP adresu. Dále není specifikováno zadávání jména hráče a proto jsem si specifikoval, že se jméno bude zadávat na výzvu od serveru.

U serveru není patrné, zda-li se má jako parametr zadávat IP adresa i port, nebo pouze port. Specifikoval jsem si pouze zadání a ověření portu.

3 Popis řešení

V této části si popíšeme řešení těchto bodů:

1. Připojení a vstup do hry

2. Uchovávání informací na serveru
3. Vyhodnocování odpovědí a konečného pořadí hráčů
4. Způsob zpracování zpráv
5. Formát a typy zpráv
6. Ukončení klienta/serveru,
7. Interakce serveru s uživatelem

3.1 Připojení k serveru a vstup do hry

Připojení k serveru může probíhat dvojím způsobem. První způsobem je zadání IP adresy serveru a portu, na kterém server naslouchá, jako parametry příkazové řádky při spouštění klienta. Tyto parametry se nejprve podrobí validitě a v případě úspěšného ověření se pokusí spojit se serverem. V případě neúspěchu, jak už při ověření validity, nebo při spojení se vypíše příslušná hláška a program se ukončí.

Druhým způsobem je zadávání IP adresy a portu až v GUI. Po zadání a kliknutí na tlačítko **Hrat** se nejprve ověří validita IP a portu a v případě neúspěchu se zobrazí dialogové okno s příslušnou hláškou. Pokud ověření validity proběhne správně, pokusí se klient spojit. Případný neúspěch při spojení opět oznámí v dialogovém oknu.

Po úspěšném spojení jak prvním, tak druhým způsobem se zobrazí hlavní okno s dialogem **Cekej na ostatni hrace**, server si zažádá o jméno hráče a u klienta se zobrazí input dialog s výzvou zadání jména. Správnost jména se ověřuje jak na klientovi, tak na serveru a po zadání jména s jinými než s **alfanumerickými znaky** je nutné zadat jméno správně. Po validně zadaném jménu se na serveru ověřuje, zda-li hráč s daným jménem existuje a pokud ne, zařadí hráče do první hry, kde je volno a hráč čeká na ostatní připojení dalších hráčů a spuštění hry. V případě, že hráč se zadaným jménem již existuje oznámí tuto skutečnost klientovi a zažádá si o nové jméno.

3.2 Uchovávání informací na serveru

Uchovávání informací o hráčích, hrách a otázkách je realizováno v datových typech **strukturách**. Každá struktura nese potřebné informace k realizaci jednotlivých her, ale také k chodu serveru. Veškeré informace sdružuje **hlavní spojový seznam her**. Každá hra v tomto seznamu má své **id**, kvůli identifikaci, dále pak **číslo aktuálně hrané otázky** a **pole otázek**. Každá hra také obsahuje **spojový seznam hráčů**, kteří v dané hře hrají a také **referenci na další hru**. Ostatní parametry této struktury není nutné zmiňovat.

Každý hráč má své **jméno** kvůli identifikaci a také **socket** kvůli komunikaci. Dále pak struktura obsahuje informaci o **skóre** hráče a také informaci o **aktivitě** hráče, tedy zda-li je hráč aktivní nebo již vypadl ze hry. Ostatní parametry této struktury není nutné zmiňovat. **Spojový seznam hráčů** také není nutné příliš popisovat, nese pouze dvě reference a to na hráče a na další položku seznamu hráčů.

Struktura otázky nese 4 informace. První z nich je **název loga**, kvůli informaci o sdělení klientovi, které logo má zobrazit. Dále pak nese 3 **možnosti odpovědí** z nichž jedna je vždy správná a zbylé dvě jsou náhodně vybrané. Pole otázek je vždy generováno **náhodně**, a to tak, aby se otázky neopakovali.

Server nese ještě jednu strukturu s názvem **Odpověď**. Tato struktura kromě referencí na hru, hráče a otázku, nese především **správnou** a **zvolenou** odpověď a také **čas**, za který hráč odpověděl na danou otázku. Statistiky serveru nejsou reprezentovány žádnou strukturou, jsou realizovány jako globální proměnné modulu `statistiky.c`.

3.3 Vyhodnocování odpovědí a konečného pořadí hráčů

Vyhodnocování odpovědí probíhá na základě **správnosti** odpovědi a také **času** odpovědi. Nejprve se vždy určí hráči, kteří odpověděli. Z těchto hráčů se následně vyberou ti hráči, kteří odpověděli správně. Pokud hráči odpověděli **špatně**, mají automaticky +0 bodů. Správně odpovídající hráči se seřadí **vzestupně** podle času odpovědi, tedy od nejmenšího času odpovědi k největšímu. Nejvíce bodů pak dostane hráč, který odpověděl nejrychleji, o bod méně pak hráč odpovídající jako druhý atd. Maximální počet bodů za správně odpovězenou otázku je úměrný počtu hráčů.

Konečné pořadí hráčů je určováno primárně podle **počtu bodů**. Nejprve se tedy hráči seřadí **sestupně** podle počtu bodů, tedy od hráče s největším počtem bodů k tomu s nejmenším počtem bodů. Dalším kritériem je pak **aktivita hráče**. Sestupně seřazení hráči podle bodů se seřadí ještě jednou a to tak, aby aktivní hráči byli v popředí a neaktivní v pozadí. Ve výsledku tedy na prvních pozicích budou ti hráči, který jsou aktivní a mají nejvíce bodů. Na konci žebříčku pak budou neaktivní hráči, bez ohledu na počet bodů. Pokud mají všichni hráči stejný počet bodů a jsou aktivní, nikdo nevyhrál a hra končí remízovým stavem. Pokud 1. a 2. hráč mají stejný počet bodů a 3. hráč méně bodů, 3. hráč prohrál a první 2 jsou v remízovém stavu. V případě, že zůstane ve hře pouze jeden aktivní hráč, automaticky vyhrává.

3.4 Způsob zpracování zpráv

Aby se zamezilo nesmyslným zprávám, končí každá zpráva **dvěma středníky** (*zpráva;;*). Dalším faktorem ošetření nežádoucí zprávy je příjmání řídícího znaku, který určuje předem definovanou akci. V případě, že tedy přijde řídící znak zakončený dvěma středníky (např. *A;;*) a řídící znak není definován, je považován za nesmyslnou zprávu. Pokud je řídící znak definován, provede se buď další přijímání, nebo rovnou nějaká akce.

Ve většině případů se po příjmu řídícího řetězce přijímá velikost další zprávy[1]. Tato velikost je opět zakončena dvěma středníky (např. *12;;*). Velikost bufferu pro příjem velikosti zprávy je určena na **maximální očekávanou velikost**. Pokud je tedy očekávána velikost příjmu mezi 10 až 99, pak je velikost bufferu 4 znaky (nesmí se zapomínat na příjem 2 středníků, jinak by byla zpráva nesmyslná). Za předpokladu, že délka zprávy je

[1] velkost další zprávy dále nazývám „velikost příjmu“

menší než předpokládaná velikost, je nutné doplnit zprávu velikosti příjmu o **nuly**, aby délka odpovídala předpokládané velikosti bufferu (např. očekávám číslo od 10 do 99, ale skutečná velikost je 7, pak pošlu *07;;*).

Velikost příjmu se používá jako velikost bufferu pro příjem zprávy. Nutno podotknout, že v této velikosti jsou započítávány i konečné středníky a proto ji není nutné navyšovat o 2 znaky. Po vytvoření bufferu pro příjem zprávy se daná zpráva přijme a ověří se opět dva konečné středníky. V případě úspěchu se přijatá zpráva odstraní o zakončující středníky a následně se rozdělí podle znaku „|“ („pipe“) a zpracuje její obsah. V případě neúspěchu je zpráva označena za nesmyslnou. Tento postup je stejný jak na klientovi, tak na serveru. Například poslání otázky vypadá v jednotlivých krocích přibližně takto:

1. `0;;`
2. `37;;`
3. `Abarth.png|Abarth|Milka|Rolls-Royce;;`

3.5 Formát a typy zpráv

V této části si popíšeme formáty a typy zpráv jak posílané na ze serveru na klienta, tak z klienta na server. Jak již bylo řečeno v předchozí podkapitole, nejprve se vždy přijímá řídící znak a na základě toho se rozhodne jestli se vykoná rovnou nějaká akce nebo se bude přijímat další informace. Nejprve si tedy v tabulce vždy definujeme řídící znaky, typ akce jaký představují a pokud se hned po tom odesílá velikost příjmu, definujeme v tabulce maximální možnou velikost zprávy **včetně zakončovacích středníků**. Na základě tabulky s řídícími řetězci si pak definujeme formát posílané zprávy.

3.5.1 Zprávy posílané ze serveru na klienta

#	*Řídící zpráva*	*Typ akce*	*Max. velikost příjmu*
1	J;;	žádost o jméno	–
2	N;;	žádost o nové jméno (hráč existuje)	–
3	Z;;	zkoumání života spojení klienta	–
4	O;;	poslání otázky	99
5	P;;	poslání jmen hráčů	999
6	M;;	poslání můžeš hrát	–
7	C;;	poslání čekej	99
8	B;;	body hráčů	999
9	K;;	konec hry, výsledné pořadí s body	999
10	V;;	poslání výhry (ostatní hráči skončili)	–
11	X;;	kill	–
12	U;;	server končí	–

Tabulka 1: Řídící řetězce posílané ze serveru na klienta

V následující tabulce si uvedeme vždy číslo zprávy vztahující se k tabulce 1 a formát posílané zprávy. Následně si v tabulce 3 ukážeme reálné příklady zpráv. V obou tabulkách se u zpráv 8 a 9 je možno posílat místo bodů hráče -1. Tato hodnota signalizuje, že hráč skončil, případně ztratil spojení. Zprávy 8 a 9 se od sebe liší především tím, že u zprávy 9 se posílají seřazení hráči podle kritérii vyhodnocení.

Číslo zprávy	*Formát odesílané zprávy*
4	<název loga>.png|<1. odpověď>|<2. odpověď>|<3. odpověď>;;
5	<jméno hráče>|<jméno hráče>|<jméno hráče>;;
7	<zobrazovaná zpráva>;;
8	<jméno>|<body>|<jméno>|<body>|<jméno>|<body>;;
9	<jméno>|<body>|<jméno>|<body>|<jméno>|<body>;;

Tabulka 2: Formáty posílaných zpráv ze serveru na klienta

3.5.2 Zprávy posílané z klienta na server

V následující tabulce si uvedeme vždy číslo zprávy vztahující se k tabulce 4 a formát posílané zprávy. Následně si v tabulce 6 ukážeme reálné příklady zpráv.

Číslo zprávy	*Příklad odesílané zprávy*
4	Abarth.png|Abarth|Milka|Rolls-Royce;;
5	Jiri|Manas|Blabol;;
7	Cekej na ostatni hrace;;
8	Jiri|23|Manas|10|Blabol|-1;;
9	Manas|24|Jiri|23|Blabol|-1;;

Tabulka 3: Příklady posílaných zpráv ze serveru na klienta

#	*Řídící zpráva*	*Typ akce*	*Max. velikost příjmu*
1	J;;	poslání jména	99
2	Z;;	odpověď na zkoumání života	–
3	O;;	odpověď na otázku	999
4	K;;	poslání konce spojení	–

Tabulka 4: Řídící řetězce posílané z klienta na server

Číslo zprávy	*Formát odesílané zprávy*
1	<jméno hráče>;;
3	<jméno hráče>|<název loga>.png|<číslo odpovědi>;;

Tabulka 5: Formáty posílaných zpráv z klienta na server

Číslo zprávy	*Příklad odesílané zprávy*
1	Manas;;
3	Manas|Abarth.png|1;;

Tabulka 6: Příklady posílaných zpráv z klienta na server

3.6 Ukončení klienta/serveru

K ukončení klienta může vést několik věcí. Mimo ukončení na základě ztráty spojení, je možné klienta ukončit zavřením hlavního okna, kdy klient pošle serveru zprávu o ukončení, ukončí se a server následně zneaktivní daného hráče. Toto zavření lze provést i zavřením dialogového okna čekej, kde ukončení probíhá obdobným způsobem. Klient se také ukončí na základě 4 zpráv od serveru (zpráva o ukončení serveru, kill, konec hry a zpráva o výhře na základě vzdání se ostatních hráčů). Na tyto 4 zpráva se reaguje pouze ukončením spojení.

Server je možné ukončit kromě násilného ukončení, také příkazy `exit` a `exit-w`. Tyto příkazy si popíšeme v následující sekci.

3.7 Interakce serveru s uživatelem

Na serveru je vytvořeno vlákno, které reaguje na několik definovaných příkazů od uživatele. Tyto příkazy jsou:

1. **seznam** – vypíše seznam všech her a všech hráčů příslušné hry
2. **kill** – možnost poslat kill hráči
3. **stat** – vypíše statistiky serveru
4. **cas** – vypíše dobu běhu serveru
5. **online** – vypíše všechny online hráče
6. **exit** – dojde k ukončení serveru (může dojit k neuvolnění paměti)
7. **exit-w** – dojde k ukončení serveru (čeká 65 s na ukončení spojení neznámých hráčů)
8. **help** – vypíše nápovědu

Především příkazy `kill`, `exit` a `exit-w` je nutné trochu přiblížit. Po zadání příkazu `kill` přijde výzva na zadání jména a po zadání jména se pošle danému hráči zpráva o killnutí. Pokud hráč neexistuje vypíše se pouze příslušná hláška.

Příkaz `exit` a `exit-w` se od sebe příliš neliší. Oba dva příkazy pošlou všem známým hráčům zprávu o ukončení serveru a smažou všechny hry a hráče, kteří byli ve hrách a ukončí server. Hlavním rozdílem je ten, že u `exit-w` se čeká na ukončení neznámých hráčů[2] a dochází ke správnému uvolnění paměti. U příkazu `exit` se na neznámé hráče nečeká a může dojít k neuvolnění paměti právě u těchto hráčů. Pokud při provedení `exit` žádní neznámí hráči k serveru připojeni nejsou, provede se správné uvolnění paměti a nemusí se čekat 65 s jako v případě `exit-w`.

4 Popis implementace

4.1 Server

4.1.1 Soubory hrac.c a hrac.h

Tyto moduly reprezentují především samotného hráče a práci s ním. Definují nový datový typ `HRAC`, který obsahuje nejen hlavní informace pro hraní jako je *jmeno*, *skore* či *aktivita_hrace*, ale také informace potřebné k chodu serveru. Mezi ty patří například parametr *socket*, aby bylo možné danému hráči poslat zprávu nebo parametry *vlakno* a *vlakno_zivot*,

[2]rozdíl mezi známým a neznámým hráčem je ten, že známý hráč již byl zařazen do hry a je možno k němu přistoupit prostřednictvím hlavního spojového seznamu her, u neznámého hráče se čeká na přijetí jména hráče, tento hřáč není veden v žádném seznamu a je nutné počkat než hráči vyprší timeout, aby se automaticky smazal

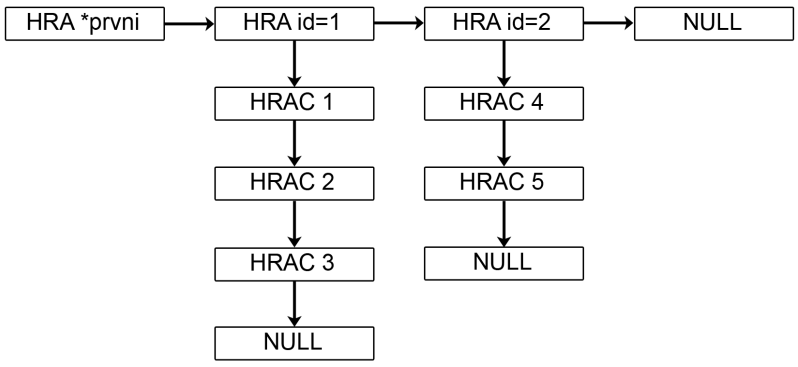
aby bylo možné zjistit jestli daný hráč již odpověděl na otázku a skončilo tak jeho přijímací vlákno, případně zda-li skončilo vlákno ověřující život spojení. Tyto soubory také obsahují struktury reprezentující spojový seznam hráčů (datový typ `SEZNAM_HRACU`). K oboum datovým typů jsou v modulu **hrac.c** implementovány funkce pro práci s nimi. Jedná se především o funkce pro vytvoření a smazání a to ,jak hráče tak seznamu hráčů, ale také funkce pro výpis jednotlivých struktur.

4.1.2 Soubory otazky.c a otazky.h

V těchto modulech najdeme především struktury reprezentující otázku a také funkce potřebné pro práci s otázkami. Datový typ `OTAZKA` nese 4 základní informace a to hlavně *logo* a pak 3 odpovědi, mezi nimiž je vždy jedna správná. V modulu **otazky.c** se především nachází globální pole se všemi názvy souborů s logy (`polozky`) a modul pak obsahuje funkci pro vytvoření pole otázek (`OTAZKA **pole_otazek(int pocet_otazek)`), která volá potřebné funkce, aby vygenerovala náhodné otázky, které se nebudou opakovat. Tento modul pak dále obsahuje funkce pro inicializaci a načtení položek do globálního pole z adresáře. Dále jsou zde funkce pro vytvoření (mallocování), smazání a výpis otázek.

4.1.3 Soubory hra.c a hra.h

V modulu **hra.h** je definován datový typ `HRA`, který je nejdůležitější v celém programu. Tento datový typ obsahuje především parametr *id*, kvůli identifikaci hry, pak 2 parametry *pocet_hracu* a *pocet_otazek*, které není potřeba popisovat a také parametry *otazky* (pole otázek) a *aktualni_otazka*, které reprezentují informace o otázkách dané hry. Dále pak každá hra obsahuje seznam hráčů (parametr *prvni*), resp. referenci na první položku seznamu, svůj semafor, kvůli ošetření časového souběhu vláken dané hry a v jako poslední obsahuje každá hra referenci na další hru (parametr *dalsi*). Díky poslednímu parametru je pak v modulu **hra.c** udržován globální spojový seznam her, ze kterého jsou dohledatelné všechny hrané hry, a všichni hráči, kteří byli zařazeni do dané hry. Na první položku v seznamu se v modulu odkazuje globální pointer `HRA *prvni`. Spojový seznam her včetně „rozvinutého“ seznamu hráčů je vidět na obrázku 1.



Obrázek 1: Spojový seznam her a hráčů

Tento seznam je považován za kritickou sekci a proto je v modulu **hra.c** globální semafor, který je využíván ve všech funkcích kde by mohl nastat problém časového souběhu vláken. Tento modul obsahuje mimo funkcí pro práci se datovým typem `HRA`, také funkce potřebné pro vytvoření daného spojového seznamu her, funkce pro zařazení vytvořeného hráče do volné hry a také funkce pro smazání spojového seznamu. V neposlední řadě obsahuje modul funkce pro obsluhu vlákna, které od neznámého připojeného hráče, vyžádá jméno, na základě jména ověří, zda-li hráč již neexistuje a následně hráče pomocí již výše zmíněných metod zařadí do hry. Dále jsou zde již jen různé funkce pro výpis jak seznamu, tak samostatné hry a také funkce pro start hry která odstartuje vlákno s obsluhou hry.

4.1.4 Soubory obsluha_hry.c a obsluha_hry.h

V souboru **obsluha_hry.h** jsou pouze hlavičky funkcí, a proto se zde budeme věnovat pouze souboru **obsluha_hry.c**. Tento soubor obsahuje skupinu funkcí reprezentujících obsluhu hrané hry a také posílání zpráv. V modulu jsou především dvě hlavní funkce pro posílání zpráv a to `void poslat_vsem(SEZNAM_HRACU *seznam, char *zprava, int delka)` a `void poslat_hraci(HRAC *hrac, char *zprava, int delka)`, které volají ostatní funkce, které chtějí sdělit příslušnou zprávu klientovi. To jsou především funkce pro poslání hráčů a otázky, pro poslání bodů za předchozí otázku, pro poslání konečného počtu bodů nebo konce pokud ostatní hráči vzdali a funkce pro poslání čekání na odehrání ostatních hráčů a pro poslání můžeš hrát.

Nejdůležitější funkcí je zde funkce `void *obsluha(void *arg)`. Tato funkce je hlavní funkcí vlákna pro obsluhu hry a řídí samotnou hru. Využívá především funkci `void prijmej_odpovedi(HRA *hra)`, která si za pomoci startování jiných vláken „obstará“ odpovědi na otázky a následně pomocí funkce pro vyhodnocení oznámí hráči, jaké skóre dostal za předchozí otázku. Funkce `prijmej_odpovedi` využívá především funkcí z modulu **prijmani.h** a **prijmani.c**

4.1.5 Soubory prijmani.c a prijmani.h

Soubor **prijmani.h** obsahuje datový typ `ODPOVED`, který má především parametry *cas*, *spravna_odpoved* a *zvolena_odpoved*. Tyto parametry jsou důležité pro správné vyhodnocování odpovědí na otázky. Dále pak nese pomocné parametry (reference) *hra*, *hrac* a *otazka*. Soubor **prijmani.c** pak obsahuje funkce, které s touto strukturou pracují. Mimo funkcí pro vytvoření a smazání odpovědi, modul obsahuje funkce pro přijímání zpráv od klienta a následné zpracování těchto zpráv. Jedná se především o funkci `char prijmy_ridici(ODPOVED *odpoved)` pro příjem řídící zpráva od klienta a funkci `int prijmy_odpoved(ODPOVED *odpoved, struct timeval tv1)`, pro příjem délky zprávy a samotné zprávy. Modul obsahuje hlavně funkci `void *prijmej(void *arg)`, která řídí celé přijímání odpovědi na otázku. Soubor má ještě 2 důležité funkce pro zkoumání života spojení s hráčem, které se využívají především v době, kdy server dostal odpověď a čeká, než odehrají ostatní hráči. Jako poslední jsou zde metody pro příjem jména, které se využívají v modulu **hra.c** při zařazování hráče do hry.

4.1.6 Soubory statistiky.c a statistiky.h

Tyto soubory se starají o udržování statistik serveru a také o logy. Pro zápis do logů serveru jsou zde 2 funkce a to `void zapis_logy(char *zprava)` a `void zapis_logy_volba(int volba, HRA *hra, HRAC *hrac)`. První z nich zapisuje do logů zprávu, u které není nutná informace o ze nějaké hry nebo hráče. Druhá funkce má přesně definované výpisy podle parametru `volba`. Tyto volby jsou popsány v komentářích zdrojových souborů. Logy se zapisují do souboru *vystup.log*. Modul **statistiky.c** se stará především o tyto statistiky:

- *počet přijatých bytů*
- *počet odeslaných bytů*
- *počet poslaných zpráv* (jako zpráva se počítá i poslání řídícího řetězce i délky zprávy)
- *počet obdržených zpráv* (jako zpráva se počítá i poslání řídícího řetězce i délky zprávy)
- *počet nesmyslných zpráv*
- *počet připojení*
- *počet hráčů zařazených do hry*
- *počet vytvořených her*
- *doba běhu serveru*

Kvůli ošetření kritických sekcí je zde globální semafor, který slouží jak pro statistiky tak pro logy. Jako poslední je zde ještě funkce pro zápis statistik serveru do souboru *statistiky.stat*.

4.1.7 Soubory obsluha_serveru.c a obsluha_serveru.h

Tyto moduly slouží především pro interakci mezi serverem a uživatelem. Modul **obsluha_serveru.h** sice neobsahuje definici datového typu, ale obsahuje 2 globální proměnné typu `extern`. První z nich je `port` a druhá je `konec_serveru`. Obě dvě jsou zde z důvodu ukončení serveru. Proměnná `konec_serveru` zastaví vykonávání nekonečné smyčky přijímání nových spojení, kde je blokující funkce `accept`. Aby se nemuselo čekat, než přijde další spojení a funkce `accept` dovolila opuštění nekonečné smyčky přijímání, vytváří se tzv. falešné spojení, kdy server vytvoří připojení sám na sebe a hned se na toto připojení ukončí. K tomuto důvodu se využívá externí proměnná `port`. Nutno podotknout, že falešné připojení se nezapočítává do statistiky.

Modul **obsluha_serveru.c** obsahuje především funkci `void *obsluha_serveru(void *arg)`, který zajišťuje interakci mezi serverem a uživatelem. Modul dále obsahuje funkce vykonávající činnosti popsané v kapitole 3.7.

4.1.8 Soubor serveru.c

Hlavní soubor celého programu. Obsahuje pouze 2 funkce, z nichž jedna je funkce `main` a druhá je funkce `int overPort(char *str)`. Druhá funkce, jak napovídá název je, k ověření správně zadaného portu jako parametru příkazové řádky. Funkce `main` především vytváří server socket, přijímá spojení a volá příslušné funkce pro inicializace a destrukci globálních semaforů a podobných věcí potřebných pro běh serveru. Nutno podotknout, že se zde také nastavuje přes funkci `setsockopt`, timeout pro send a recv na 63s, které se využívají zároveň jako timeouty pro odehrání otázky.

4.2 Klient

4.2.1 Třída Hrac

Tato třída reprezentuje hráče a obsahuje pouze atributy `skore`, `jmeno`, `skoreLB` a `jmenoLB`. Všechny atributy obsahují příslušné gettry a settry. Třída má ještě dvě důležité metody, kde první je `public void setHlavniHrac()`, která nastavuje zbarvení labelu na oranžovo pro identifikaci hlavního hráče na klientu a druhá metoda je `public void setNeaktivni()`, která nastavuje zbarvení labelu na červeno, pro identifikaci vypadlých hráčů.

4.2.2 Třída Otazka

Tato třída reprezentuje především panel otázky a obsahuje potřebné atributy pro vykreslení otázky. Kromě metody `public JPanel getLogoPanel()`, která vrací naformátovaný panel s otázkou, obsahuje třída ještě jednu důležitou metodu a to je `public void setOtazka(String []odpovedi)`, která na základě přijaté otázky nastaví logo a odpovědi na aktuální otázku.

4.2.3 Třída ZvolenaOdpoved

Tato třída plní jediný účel a to akci po kliknutí na tlačítko odpovědi v panelu otázky. V překryté metodě `public void actionPerformed(ActionEvent e)` se vezme číslo odpovědi, název aktuální otázky a jméno hlavního hráče a pošle se zpráva serveru.

4.2.4 Třída DialogCekej

Třída **DialogCekej** funguje jako indikace hráči, že se čeká na ostatní hráče. Třída dědí od třídy **JDialog** a měla tak plnit funkci modálního dialogu. Třída obsahuje především 3 hlavní metody. První z nich je `public void setStav(String titulek, String zprava)`, která nastavuje titulek okna a zobrazovanou zprávu. Zbylé dvě metody jsou `public void zmiz()` a `public void ukaz()`. Metody v principu pouze nastavují buď aktivní nebo neaktivní hlavní okno a skrývají nebo zobrazují dialogové okno čekání.

4.2.5 Třída Stopwatch

Tato třída reprezentuje časovač a zároveň indikátor, kolik sekund zbývá hráči na odehrání otázky. Kromě metody `public JPanel vytvorPanelStopky()`, která vrací naformátovaný panel s indikací času, obsahuje třída ještě metody `public void init()` a `public void zastavCas()`. Jak napovídají názvy metod, první inicializuje časovač (objekt třídy **Timer**) a druhá tento časovač zastavuje.

4.2.6 Třída HlavniOkno

Třída **HlavniOkno** zobrazuje panely ze tříd **Stopwatch** a **Otazka** a také panel se všemi hráči ve hře. Třída obsahuje atributy s referencemi na hlavního hráče, pole hráčů, otázku, stopky a dialog čekej. K těmto atributům obsahuje taky příslušné settry a gettry.

4.2.7 Třída Spojeni

Tato třída reprezentuje činnosti vztahující se ke spojení. Tedy navazuje a ukončuje spojení se serverem a posílá a přijímá zprávy. Spojení se navazuje již v konstruktoru třídy a v případě vyhození vyjímky se tato vyjímka posílá o úroveň výše. Třída má především 3 hlavní metody. První z nich je metoda `public void posli(String zprava)`, která odesílá zprávy na server. Druhá z těchto metod je metoda `public String prijem()`, která přijímá řídící řetězce a na základě metody `private String rozhodni(char []znak)` se vrací buď samotný řídící znak nebo null v případě příjmu nedefinovaného řídícího řetězce. Poslední z těchto metod je metoda `public String prijemDelky(int delka)`, která přijímá zprávu přesně definované délky a v případě přijetí špatné zprávy vrací null. Třída obsahuje ještě metodu `public void zavriSpojeni()`, která ukončuje spojení se serverem. V této třídě jsou ještě dvě statické metody pro ověření portu a IP adresy.

4.2.8 Třída Prijmani

Tato třída reprezentuje hlavní přijímací vlákno. V překryté metodě `public void run()` se ve smyčce přijímá vždy řídící znak na základě volání metody `prijmy()` z třídy **Spojení** a podle přijatého řídícího řetězce se rozhodne jakou z metody bude volat dále. Většina volaných metod vykoná rovnou nějakou akci, zbylé metody většinou přijímají nejprve délku následující přijímané zprávy, následně přijmou zprávu a až pak vykonají akci. Jediná metoda `private void posliJmeno(boolean existujici)` vyvolá input dialog a posílá na server jméno hráče. Funkčnost jednotlivých metod provádějících tyto činnosti není nutné popisovat, u většiny metod lze poznat činnost již z názvu metody.

4.2.9 Třídy UvodniOkno a HrajAction

Třída **UvodniOkno** představuje okno pro zadání IP adresy a portu. Obsahuje pouze 2 labely a dvě textová pole a tlačítko *Hraj*, které provádí akci definovanou ve třídě **HrajAction**. Tato akce spočívá v ověření IP adresy a portu metodami ze třídy **Spojení** a v případě

úspěchu se spojuje se serverem. V případě neúspěchu oznamuje případné prohřešky dialogovým oknem. V případě úspěšného spojení se serverem se vytváří objekty tříd **Prijmani** a **HlavniOkno** a úvodní okno končí. V případě neúspěšného spojení se opět zobrazí dialogové okno s příslušnou hláškou.

4.2.10 Třída Hlavni

Hlavní třída celého programu. Obsahuje jedinou metodu a to metodu `main`. Tato metoda nejprve ověřuje počet parametrů. Pokud jsou parametry 2 (1. IP adresa a 2. port), vykoná funkce obdobnou činnost jako třída **HrajAction**, pouze s tím rozdílem, že případné chyby vypisuje na konzoli. Pokud je zadáno více jak 2 parametry, vypisuje se na konzoli příslušná hláška a v ostatních případech (tedy žádný parametr) se vytváří instance třídy **UvodniOkno**.

5 Uživatelská dokumentace

5.1 Překlad

V adresáři se nachází 3 adresáře **c_src**, **java_src** a **logo** a také 2 soubory (**Makefile** a **build.xml**). V adresáři **c_src** se nachází zdrojové kódy serveru a v adresáři **java_src** se nachází zdrojové kódy klienta. V posledním adresáři (**logo**) jsou loga nutná pro správný běh serveru, klient si nese obrázky v jaru. Překlad může probíhat dvojím způsobem[3].

5.1.1 1. způsob – přeložení serveru i klienta najednou

Pro tento účel slouží již zmíněný soubor **Makefile** stačí v příkazovém řádku zavolat `make` a provede se automatický překlad jak serveru, tak klienta[4]. Zároveň se vytvoří adresáře **java_bin** a **c_bin**, ve kterých jsou vytvořené binární soubory. Dále se vytvoří spustitelný soubor *server* a také jar soubor *client.jar*.

5.1.2 2. způsob – přeložení serveru a klienta zvlášť

Přeložení serveru: Pro přeložení serveru stačí ve složce **c_src** zavolat `make` a prostřednictvím **Makefile**[5]

Přeložení serveru: Pro přeložení klienta stačí v zavolat `ant` ve složce se souborem **build.xml**, který provede automatický překlad[6]. Po překladu je vidět, že se vytvoří adresář **java_bin** a také soubor *client.jar*.

[3] pro bezproblémový překlad je nutné mít nainstalovanou javu OpenJDK verze 7

[4] při tomto překladu se volá nástroj `ant` a je nutnou podmínkou mít tento nástroj nainstalován

[5] Pozn. tento **Makefile** je jiný než **Makefile** v kořenové složce, ale **Makefile** v kořenové složce je na tomto souboru závislý

[6] i v tomto případě je nutnou podmínkou mít nainstalován program `ant`

5.2 Spuštění

5.2.1 Spuštění serveru

Spuštění serveru se provádí příkazem `./server <port>`, kde `<port>` je port, kterém bude daný server naslouchat.

5.2.2 Spuštění klienta

Spuštění klienta je možno dvěma způsoby. První z nich je pomocí příkazu `java -jar client.jar <ip> <port>`, kde `<ip>` je IP adresa a na kterou se má daný klient připojit a `<port>` je port, na kterém daný server naslouchá (obrázek 4 v sekci 5.3.2). Druhý způsob spuštšní klienta je pouze příkazem `java -jar client.jar` a IP adresa a port se zadávají až v úvodním oknu (obrázek 4 v sekci 5.3.2).

5.3 Ovládání

5.3.1 Ovládání serveru

Server spustíme a čekáme na připojení hráčů. Připojení hráčů je vidět na obrázku 2. Server komunikuje s uživatelem pomocí pár jednoduchých příkazů. Tyto příkazy lze vypsat příkazem `help`. Server reaguje na tyto příkazy:

1. **seznam** – vypíše seznam všech her a všech hráčů příslušné hry
2. **kill** – možnost poslat kill hráči
3. **stat** – vypíše statistiky serveru
4. **cas** – vypíše dobu běhu serveru
5. **online** – vypíše všechny online hráče
6. **exit** – dojde k ukončení serveru (může dojít k neuvolnění paměti)
7. **exit-w** – dojde k ukončení serveru (čeká 65 s na ukončení spojení neznámých hráčů)
8. **help** – vypíše nápovědu

Tyto příkazy jsou podrobněji popsané v kapitole 3.6. Jak vypadá například příkaz exit je vidět na obrázku 3. V poslední řadě je třeba zmínit dva soubory, které server vytváří. Jsou jimi **vystup.log** a **statistika.stat**. V prvním jmenovaném je zapsána činnost serveru a ve druhém statistiky serveru.

```
manas@debian:~/pokus$ ./server 10000
Vytvarim soubor s logy - "vystup.log"
Startuju server
Bind OK
Listen OK
Inicializuji loga
Loga nactena uspesne
Vytvarim hrace se socketem 5.
Vytvarim hrace se socketem 6.
Vytvarim hrace se socketem 7.
Vytvarim novou hru - id = 1
Pridavam do hry 1 hrace A se socketem 5
Pridavam do hry 1 hrace B se socketem 6
Pridavam do hry 1 hrace Blabol se socketem 7
Startruju hru 1
```

Obrázek 2: Ukázka připojení hráčů na server

```
exit
Ukoncuji server
Zaviram server socket
Hrac A: ukoncuji
Hrac B: ukoncuji
Hrac C: ukoncuji
Hrac A: ztrata spojeni pri prijmu ridiciho retezce
Hrac B: ztrata spojeni pri prijmu ridiciho retezce
Hrac C: ztrata spojeni pri prijmu ridiciho retezce
Konec hry 1
Mazu hru 1.
Mazu hrace A
Mazu hrace B
Mazu hrace C
Mazu pole otazek
Mazu pole polozek
Pamet u her uvolnena.
Pamet v obsluze hry uvolnena.
Pamet v modulu hrac uvolnena.
Pamet v modulu otazka uvolnena.
Pamet v prijmani uvolnena
Vytvarim soubor se statistikou - "statistika.stat"
Zaviram soubor se statistikou - "statistika.stat"
Koncim
Zaviram soubor s logy - "vystup.log"
```

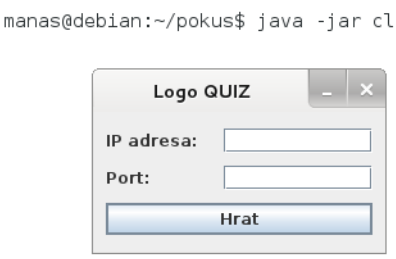
Obrázek 3: Ukázka příkazu exit

5.3.2 Ovládání klienta

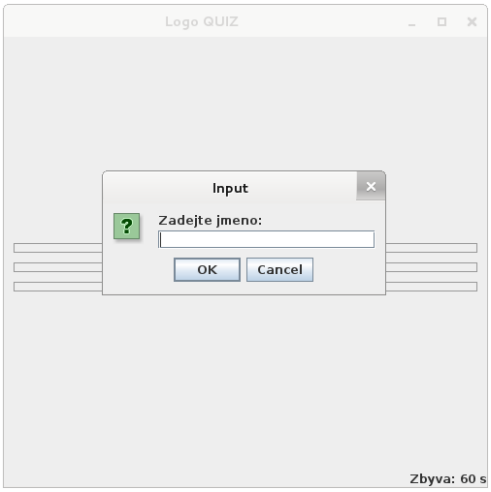
Spuštění klienta 1. způsobem můžeme vidět na obrázku 4 a spuštění druhým způsobem můžeme vidět na obrázku 5. Zejména u druhého způsobu je nutné do pole **IP adresa** zadat IP adresu serveru a do pole **Port** zadat port serveru. Po úspěšném spojení si server ihned zažádá o jméno to můžeme vidět na obrázku 6.

```
manas@debian:~/pokus$ java -jar client.jar 127.0.0.1 10000
Pripojuju se na : 127.0.0.1 se jmenem : localhost
```

Obrázek 4: Ukázka spuštění 1. způsobem

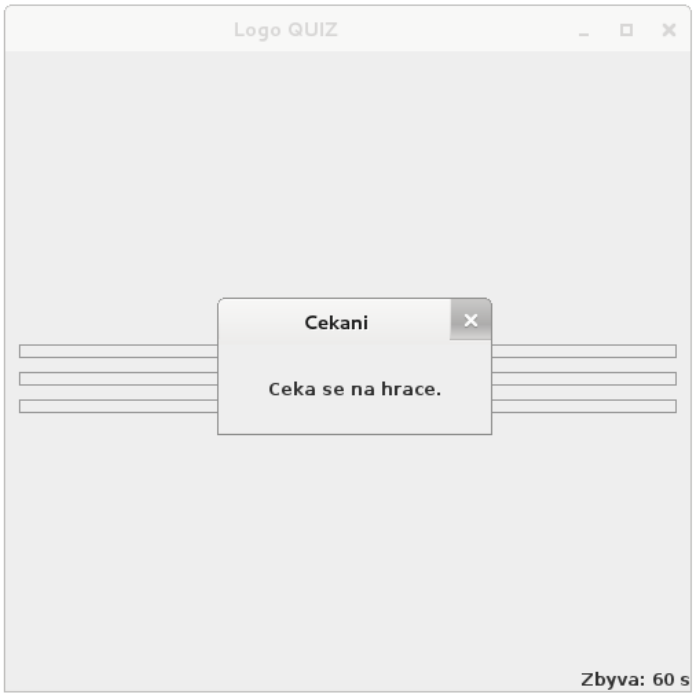


Obrázek 5: Ukázka spuštění 2. způsobem

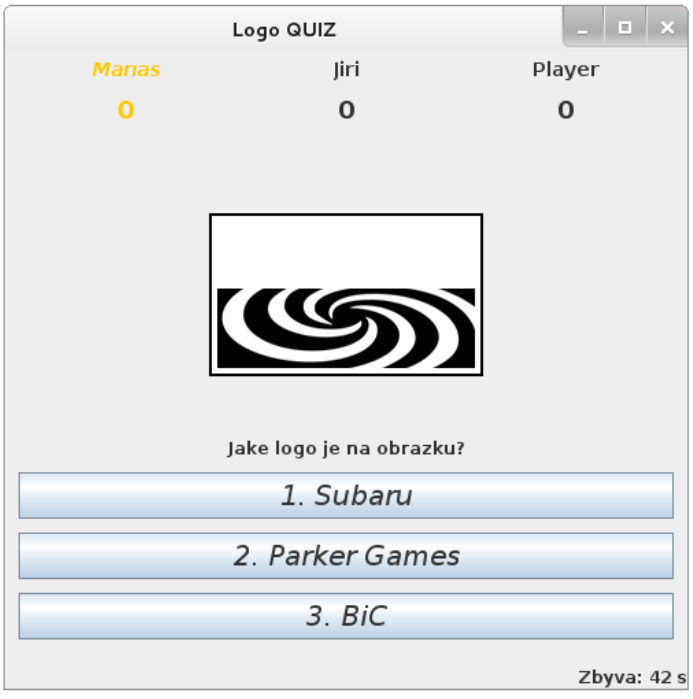


Obrázek 6: Ukázka zadávání jména

Po zadání správného jména se zobrazí dialogové okno **Čekej** jako je vidět na obrázku 7. To znamená, že je třeba počkat než se připojí zbylí hráči do hry. Pokud se hráči nechce čekat, stačí okno zavřít. Po připojení ostatních hráčů se v hlavním okně zobrazí otázka a spustí se odpočet do, kterého je nutné odpovědět, jinak hráč ve hře končí. V horní části hlavního okna si můžeme všimnout oranžově zbarveného hráče. Toto zbarvení signalizuje hráče, hrajícího na daném klientovi. Nyní již můžeme odpovídat na otázku kliknutím na příslušné **tlačítko odpovědi**. Tento stav je vidět na obrázku 8.



Obrázek 7: Ukázka čekání na připojení dalších hráčů



Obrázek 8: Ukázka vstupu do hry

Po úspěšném odehrání se zobrazí opět dialogové okno s hláškou, která říká, že se čeká na odehrání ostatních hráčů. To je vidět na obrázku 9. Takto se odehraje 15 otázek. Pokud během hraní některý hráč vypadne, jeho jméno a body zčervenají, to je vidět na obrázku 10.

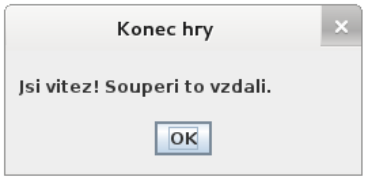
Obrázek 9: Ukázka čekání na odehrání ostatních hráčů

Obrázek 10: Ukázka ztráty hráče

V případě, že se nestihne odehrát do vypršení timeoutu, je to oznámeno jak v dolní části hlavního okna, tak dialogovým oknem. Tento stav je vidět na obrázku 11. Pokud hráč odehraje všechny otázky, dostane zprávu o vyhodnocení hry a zjistí jak dopadl. Ukázku této zprávy je vidět na obrázku 12. V případě, že zbylí hráči skončí hru dříve, hráč, který zůstává ve hře automaticky vyhrál. Tento stav je vidět na obrázku 13.

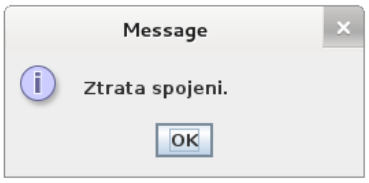
Obrázek 11: Ukázka vypršení timeoutu

Obrázek 12: Ukázka vyhodnocení hry

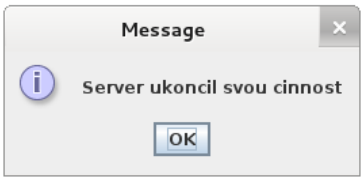


Obrázek 13: Ukázka vzdání se ostatních hráčů

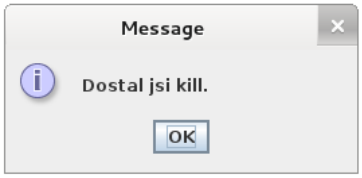
Ve hře mohou nastat ještě 3 stavy. Jeden z nich je **ztráta spojení**. Pokud dojde ke ztrátě spojení, je to oznámeno hláškou jako na obrázku 14. Dalším stavem je **ukončení činnosti serveru**. Tento stav se projeví pokud někdo vypnul server oznámení vypadá jako na obrázku 15. Posledním stavem, který může nastat je, že hráč dostane **kill** od serveru. Tento stav je vidět na obrázku 16.



Obrázek 14: Ukázka ztráty spojení



Obrázek 15: Ukázka ukončení činnosti serveru



Obrázek 16: Ukázka killnutí hráče

6 Závěr

Oba programy jak klient, tak server by měli být plně funkční a měli by splňovat zadání. Server je schopen obsloužit více her najednou a teoreticky je jedno pro kolik hráčů bude každá hra i kolik bude obsahovat otázek. Prakticky by ale hra pro více jak 5 hráčů mohla začít nudit z důvodu čekání na ostatní hráče a i více jak 30 otázek by mohlo navozovat efekt „nekonečné hry“.

Při tvorbě semestrální práce se vyskytlo mnoho potíží a to především neošetřením časového souběhu nebo špatným uvolňováním paměti. Nakonec se mi za pomoci nástroje `valgrind` povedlo uvolňování paměti opravit a ošetřit i kritické sekce. Během semestrální práce jsem se naučil pracovat s vlákny v jazyce C a také jsem se naučil pracovat se spojovaným protokolem TCP. Díky semestrální práci jsem se zase o trochu více naučil v jazyce C a také jsem si naprogramoval svou první „větší“ aplikaci typu klient–server. Práce byla velice zajímavá a především obrovsky přínosná a to i přes svou velkou náročnost.