Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Dokumentace k semestrální práci z UPS

Sítová počítačová hra KVÍZ

Plzeň, 2014
David Košek, A11B0409P, kosek@students.zcu.cz

Obsah

1 Zadání

Naprogramujte jednoduchou síťovou počítačovou hru (KVÍZ) pro tři hráče s možností hraní po síti se serverem, který může být spuštěn na libovolném počítači, na který „vidí"všichni klienti, kteří chtejí hrát – při hraní po internetu by tedy měl mít server veřejnou IP adresu, klienti mohou být za routerem s privátní IP, komunikace probíhá přes TCP protokol. Najednou může být spuštěno i více her. Hra je určena pro tři hráče v jedné hře.

1.1 Zásady 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 Analýza

V této části se budeme zabývat tím, jak daný problém rěšit. Dále, co je nutné si specifikovat tj. co není řečeno v zadání. Ze zadání je patrné, že je třeba řešit následující problémy:

1. Připojení klienta k serveru,
2. Zpracování očekávaných zpráv ze serveru/klienta,
3. Zpracování neočekávaných zpráv ze serveru/klienta ,
4. Ukončení klienta/serveru,
5. Uchovávání informací na serveru,
6. Způsob odesílaní zpráv a typy zpráv.

2.1 Specifikace

U klienta jsem si specifikoval to, že před připojením do hry žádám klienta o přezdívku, kterou má potom ve hře. Dále ho žádám o číslo hry, do které chce, tím chci zaručit to, aby tři klienti (kamarádi) mohli být v jedné hře a hrát proti sobě. Další parametr je port a IP adresa serveru. Všechny tři poslední parametry jsou ošetřeny, aby nedošlo ke zbytečným problémům.

U serveru není co specifikovat. Pouze to, že v argumentu si můžu zadat port na kterém server vysílá. V případě nezadání argumentu, je port nastaven defaultně na 10 000. Port musí být v rozsahu 1025 - 65 535.

2.2 Řešení problémů

2.2.1 Připojení klienta

Klient se připojí na server po zadání portu a ip adresy serveru. V případě zadání špatných hodnot se klient nepřipojí a nastane vyjímka, která vypíše

chybu a klient se ukončí. V případě úspěchu se hráč připojí na server a odešle svojí přezdívku a číslo hry. Po připojení se testuje, zda není hra plná, pokud není, hráč je připojen a server odešle klientovi, aby nehrál. Ten požadavek zpracuje a zakáže stisknutí tlačítek a následně odešle na server požadavek, zda nejsou už tři ve hře, aby hra mohla začít. Když jsou tři, nastaví se jména hráčů v pořadí, jak se připojili a prvnímu hráči se povolí tlačítka, aby mohl začít hrát. V přídapě plné hry server odešle, že je hra obsazena a klientovi zobrazí dialog, který po zavření ukončí hru.

2.2.2 Zpracování zpráv na serveru/klientu

Zprávy si posílám v textové podobě, které později parsuji. Jednotlivé zprávy jsou oddělené středníkem, které spojuji v jeden `string` a následně odesílám. Po přijetí zprávy u klienta testuji daný retězec na první znak, v případě shody řetězec parsuji a vracím `stringové` pole, kde dále testuji zda řetezec odpovídá očekávané zprávě, v případě shody vyhodnotím danou zprávu, v případě neúspěchu zprávu zahodím a nevyhodnocuji jí. U serveru to probíhá velmi podobně, po přijetí zprávy testuji daný řetězec, v případě úspěchu vyhodnocuji daný požadavek, v případě neúspěchu řetězec ignoruji.

2.2.3 Ukončení serveru/klienta

Klient se ukončí po odehrání všech kol tj. nezbyde žádná otázka, proběhne vyhodnocení výsledků a klient se ukončí, na server se pošle zpráva o ukončení hry. Server zprávu přijme, vyhodnotí a vymaže příslušné hráče v dané hře. V případě ukončení klienta zavřením okna, se na server odešle zpráva o ukončení, server zprávu zpracuje a odešle ostatním hráčům ve hře zprávu o tom, že hráč opustil hru, jeho body se nastaví na -99999. Když zbydou dva hráči a jeden se ukončí, poslednímu třetímu hráči přijde zpráva, která vyvolá dialog o tom, že poslední hřác vyhrál. Když se ukončí server, na který jsou připojený klienti, tak jim nastane vyjímka, která spustí dialog s chybovou hláškou. Server kdykoliv můžu opět spustit, bez jakékoli časové prodlevy.

2.3 Uchování informací

Na serveru je nutné si uchovávat informace o hráčích a o hrách, které jsou právě připojený na server. K tomu mi slouží globální dvourozměrné pole. Do toho si ukládám číslo hry, klientský socket, následuje 0 nebo 1, záleží zda je hráč na tahu nebo ne, 0 - není, 1 - je. Předposlední parametr v řádku pole je pořadí, kolikátý je hráč v dané hře. Poslední parametr je index, zda je hra spuštěna, aby nechodázelo k připojení do hry, v případě, že ji jeden hráč již opustil. Přezdívku si ukládám do jiného pole, protože předchozí pole je typu `integer`. U klienta si není třeba uchovávat žádné informace o zbývajících hráčích.

2.4 Odesílání zpráv a typy zpráv

Zpráva se mezi klientem a serverem odesílám jako text, kde jsou jednotlivé části oddělené středníkem. Používám více formátů pro odesílání zpráv, avšak nejčastější formát zpráv je tvaru "znak;o_co_jde;potrebne_info". Počet položek zprávy je proměnný podle typu zprávy, které jsou popsány v tabulce č.1. Kdy, který typ zprávy použiju je popsáno v tabulce č.2. Příklady, jak by takovéhle typy zpráv mohly vypadat je v tabulce č.3.

č.	*Typ zprávy*	*Počet položek*	*Min.*	*Max. délka*
1	Připojení klienta k serveru	2	3	13
2	Odpověď na připojení klienta (úspěch)	2	8	8
3	Odpověď na připojení klienta (neúspěch)	3	41	41
4	Požadavek na spuštění hry	1	1	1
5	Odpověď jsme 3	5	10	40
6	odpověď nejsme 3	2	6	6
7	Odeslání prvnímu klientu, aby začal	2	6	6
8	Odeslání ostatním hráčům, hraje první	3	14	14
9	Požadavek o otázku	2	2	2
10	Odeslání otázky klientovi	6	60	176
11	Odeslání zvolené odpovědi	4	48	99
12	Vyhodnocení otázky na serveru (dobře,špatně)	5	21	22
13	Odeslání, ať hraje další	1	1	1
14	Nastavení dalšího	2	6	6
15	Zneaktivování ostaních	2	12	12
16	Ukončení klienta	1	1	1
17	Informace o ukončení klienta ostatním	3	13	13

Tabulka 2.1: Typ zprávy a odpovídající počet položek při odesílání.

Typ	*Kdy využiju*	*Odpověd na zprávu*
1	Po spuštění klienta a vyplnění údajů	Jsi připojen do hry, nebo je plná
2	Po úspěšném připojení do hry	Jsi ve hře, ale nehraj, čekej
3	Po neúspěšném připojení do hry	Hra je plná, nebo si narušitel
4	Po každé, když se někdo připojil	Jsme tři, nejsme tři
5	Když jsme tři	Nastavíme jména hráčů a začneme
6	Zkouším spustit hru, ale nejsme 3	Čekáme na ostatní hráče
7	V případě, že jsme 3,řekneme, aby první začal	Hra začala
8	Hraje jeden hráč, ostatním to sdělíme	Čeká se na hráče až odehraje
9	Po zvolení klientovo skupiny	Server odešle požadovanou otázku
10	Po přijetí otázky od serveru	Zobrazí se hráči požadovaná otázka
11	Po odeslání zvolené odpovědi	Server vyhodnocuje otázku
12	Po přijetí zda jsem odpověděl dobře, špatně	Přičtení, odečtení bodů
13, 14	Po přičtení nebo odečtení bodů	Nastavní dalšího hráče, jako aktivního
15	Zaktivuji jednoho, zbylé zneaktivním	Vždy hraje pouze 1
16, 17	Po zmáčknutí křížku, alf+F4..	Rozešle hráčů zprávu o ukončení

Tabulka 2.2: Typ zprávy a její případ využití.

Typ	*Příklad*
1	”cislo_hry;prezdivka”
2	”Z;nehraj”
3	”Z;obsazeno;Ty ulicniku.. uz o tobe vim...”
4	”T”
5	”Z;jmena;prvniho;druhyho;tretiho”
6	”Z;malo”
7	”Z;hraj”
8	”Z;hrajedalsi”
9	”z1”
10	”Z;otazka;hadanka;a;b;c”
11	”P;otazka;zvolenaMoznost”
12	”Z;odpoved;spravne;z1;3”
13	”D”
14	”Z;hraj”
15	”Z;hrajedalsi”
16	”X”
17	”Z;odpojilse;cislo”

Tabulka 2.3: Ukázka formátu jednotlivých typů zpráv.

3 Popis implementace

V této části si popíšeme jednotlivé třídy a jejich metody/funkce. Klientskou implementaci jsem se snažil logicky rozdělit do jednotlivých tříd podle funkcionality. Server je v jednom souboru.

3.1 Klient

Implementace klienta je rozdělena do 4 tříd.

1. Apk.java
2. Gui.java
3. Komunikace.java
4. Spojeni.java

Třída Apk
Hlavní třída, ve které se nachází metoda *main*. Na začátku běhu programu je nutné si vyžádat určité parametry od uživatele, proto jsou zde metody pro ověření vstupů. Metoda *nactiPrezdivku* je volaná jako první, zde načítáme přezdívku a ověřujeme, zda je maximální délka v pořádku, když není, je uživatel znovu požádán o zadání přezdívky. Další je třeba načíst číslo hry, port a IP adresu. K tomu slouží metody *nactiPort*, *nactiHru* a *nactiIP*. První dvě zmiňované metody fungují velmi podobně, jen s tím rozdílem, že vyskakovací podmínka je v jiném rozmezí. V poslední metodě ověřuji IP adresu pomocí regulárního výrazu. Po načtení parametrů vytvářím instanci spojení se serverem, gui a vlákno pro vyhodnocování zpráv přijatých ze serveru. V metodě *main* jsou postupně volány všechny potřebné metody pro správný chod klienta.

Třída Gui
Tato třída zaručuje správné zobrazení grafického prostředí uživateli. Je složeno z jednotlivých JButton, na kterých je zobrazen příslušný obrázek, každý JButton, který může být aktivní obsahuje ActionListener, který reaguje na událost a předává infomace o tom, na jaké tlačítko bylo kliknuto pomocí

zpráv. Pro zobrazení skóre každého hráče slouží JTextfield, který můžu mít nadpis a v našem případě je tento nadpis přezdívka hráče. Pro vypsání informačních zpráv slouží JTextArea, do které stále přidávám text, nic nemažu. Nacházejí se zde i metody pro zobrazení dialogů, v případě, když hra skončí, spadne server či je hra obsazená. Tento soubor obsahuje také třídu Event implementovanou ActionListener, sloužící hlavně pro vyhodnocení zmáčnutého tlačítka. Dále je zde i metoda pro vypsání možností k otázce.

Třída Komunikace
V této třídě běží tělo vlákna vytvořené pro vyhodnocování zpráv. Vlákno běží v nekonečné smyčce, kde na začátku načítá stringové pole, zpracované již metodou ve třídě Spojeni a zde s ním dále pracujeme. Podle druhé položky v tomto poli rozhodujeme, co se má stát, jsou volány příslušné metody. Jsou zde metody pro připsání bodů za správnou odpověd, analogicky odečtení za nesprávnou odpověď. Dále metoda pro nastavení tlačítek aktivnímu hráči a zneaktivování tlačítek, když hráč není na tahu. Ještě je zde metoda pro nastavení bodů hráči, který opustil hru. Bez této třídy by klient sice příjmal zprávy, ale nedokázal by je vyhodnocovat a reagovat na ně.

Třída Spojeni
Tato třída nás připojí k serveru a odesílá číslo hry a přezdívku hráče. Dále jsou tu dvě metody. Jedna pro příjmání zpráv a druhá pro odesílání zpráv. Pokud příjmeme neočekávanou zprávu, rovnou zde jí zahodíme v opačném případě ji parsujeme podle středníku a ukládáme do stringového pole. K odesílání na server využíváme *OutputStreamWriter* a pro příjmání *InputStreamReader*.

3.2 Server

Zde se nachází všechny atributy a funkce pro správný chod serveru. Nazačátku si definuji strukturu pro otázky. Každá otázka obsahuje otázku, možnosti a,b,c, správnou odpověď a odpovídající skupinu. Následuje funkce pro zjištění aktivního hráče ve hře. Funkce, která nastaví dalšího hráče ve hře na akitvní je *nastav_aktivni*. Pro přidání otázky do pole slouží funkce *vytvor_otazku*. Zde je i funkce pro zapisování do souboru, která je nutná volat po každé, když se někdo připojí, odpojí, přijme zprávu a odešle zprávu, vše je poté uloženo v souboru *serverInfo.txt*. Vytvořené vlákna čekají ve funkci *vyrizeni_klienta* na přijetí o tom, že se někdo připojil pomocí funkce *accept*, po

připojení se vlákno ujme klienta a obsluhuje ho, při úspěšném připojení do hry vlákna jde do funkce *prijem*, kde se odehravá komunikace s klienty. Pro naslouchání na serveru je nastartováno vlákno, které čte příkazy vložené do terminálu a v případě správného výrazu se vyhodnotí pomocí funkce *poslouchej*, která je tělem vlákna a *online*, která vypíše hráče, co jsou připojeny k serveru. Poslední funkcí je main, jde o hlavní funkci, jako první vytvářím server socket pomocí *socket*, dále nastavuji server socket a port hostitelského počítače pomocí *bind*, kam se přidává ještě parametr adresa. Následuje funkce *listen*, díky které začínáme poslouchat pro klienty. Zprávy příjmám pomocí funkce *recv* a odesílám je pomocí *send*. Pro lepší konfiguraci serveru jsem využil *setsockopt*.

4 Uživatelská dokumentace

4.1 Překlad

4.1.1 Server v C pod Linuxem

Pro překlad v linuxu je připraven soubor makefile. Překlad probíhá pomocí překladače GCC a pro bezproblémový překlad je nutné mít zdrojový (.c) soubor v jedné složce se souborem makefile, jako v mém případě v c_src. Překlad je pak řízen z terminálu zadáním příkazu:

```
make
```

Pokud vše proběhne bez problémů, vytvoří se spustitelný soubor server v adresáři c_bin.

4.1.2 Klient v Javě pod Linux

Pro překlad v linuxu je připraven soubor makefile. Překlad probíhá pomocí příkazu javac a pro bezproblémový překlad je nutné mít zdrojový (.java) soubory v jedné složce se souborem makefile java_src. Překlad je pak řízen z terminálu zadáním příkazu:

```
make
```

Pokud vše proběhne bez problémů, vytvoří se soubory class pro všechny zdrojové soubory a vytvoří se spustitelný soubor s koncovnkou jar, vše se poté nachází ve složce java_bin.

4.2 Obshluha programu

4.2.1 Spuštění

Spuštění serveru v Linuxu probíhá z terminálové řádky příkazem:

./server <input>

<input> je port, který může být zadán, ale nemusí, pak je nastaven defaultně.

Spuštění klienta probíhá stejně jak v Linuxu, tak ve Windowsech.

java -jar Kviz.jar

Po spuštění, už stačí zadat pouze potřebné parametry a hra může začít.

4.2.2 Ovládání klienta/serveru

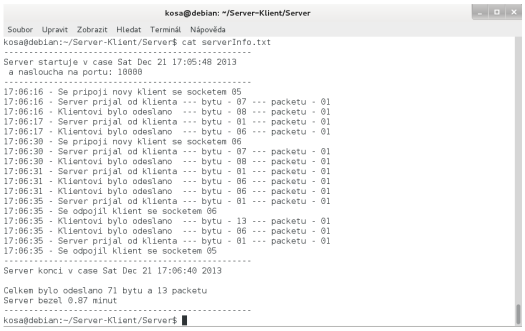
Server spustíme a čekáme, na připojení hráčů, obrázek č.1. Na obrázku č.2 vidíme příkaz, na který server reaguje, všechny příkazy lze vypsat příkazem *help*. Server neustále zapisuje do souboru, co se děje na serveru. Ukázku tohoto souboru můžeme vidět na obrázku č.3.

```
Port je nastaven defaultně 10000
Bind - OK
Listen - OK
Hura nove spojeni
Prijato spojeni z adresy 127.0.0.1
1 klient a ma socket: 5, id hry je 1 a je 1.hráčem a leží na řádku 0 a přezdívka
 je David
Hura nove spojeni
Prijato spojeni z adresy 127.0.0.1
2 klient a ma socket: 6, id hry je 1 a je 2.hráčem a leží na řádku 1 a přezdívka
 je Shorty
Hura nove spojeni
Prijato spojeni z adresy 127.0.0.1
3 klient a ma socket: 7, id hry je 1 a je 3.hráčem a leží na řádku 2 a přezdívka
 je Kiv
```

Obrázek 4.1: Ukázka serveru, po připojení tří hráčů.

```
whoisonline
Vypis vsech pripojenych hracu
------------------------------
Hrac David se socketem 5 je ve hre 1 a hra je aktivni ano
Hrac Shorty se socketem 6 je ve hre 1 a hra je aktivni ano
Hrac Kiv se socketem 7 je ve hre 1 a hra je aktivni ano
------------------------------
```

Obrázek 4.2: Ukázka serveru, po zadání příkazu *whoisonline*.



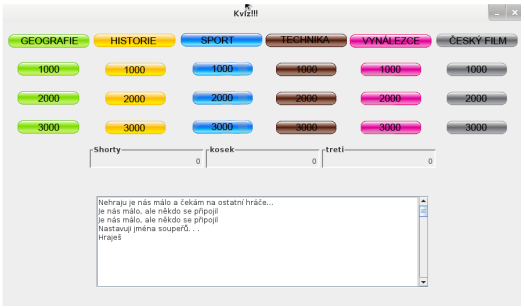
Obrázek 4.3: Ukázka výpisu textového souboru.

Klient po spuštění je požádán o zadání parametrů, přezdívka, číslo hry, port a ip adresa serveru. Příklad, jak by to mohlo vypadat je na obrázku č.4.

```
kosa@debian:~/Server-Klient/Klient$ java Apk
Zadej přezdívku: Shorty
Zadej číslo hry: jedna
Nezadal jsi číslo hry
Zadej číslo hry: 1
Zadej port: auto
Nezadal jsi port
Zadej port: 99999
Port je přes rozsah (1024-65535)
Zadej port: 10000
Zadej ip: u.p.s.k.a
Chybná IP adresa
Zadej ip: 3.2.1.A
Chybná IP adresa
Zadej ip: 127.0.0.1
```

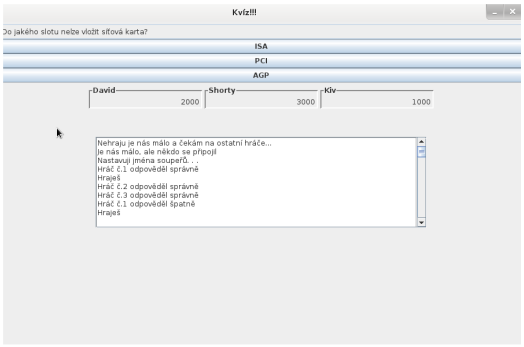
Obrázek 4.4: Ukázka přihlášení na server.

Poté klient čeká, až budou tři ve hře, a pak je hra spuštěna. Nastaví se jména ve hře a prvnímu hráči se zpřístupní tlačítka, obrázek č.5.



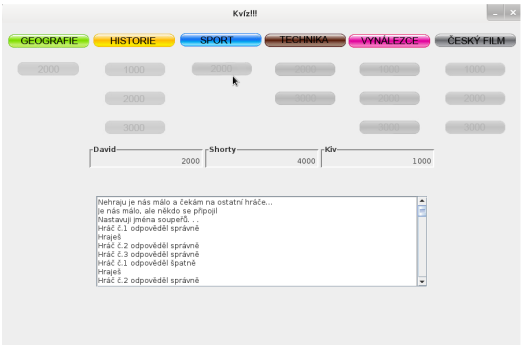
Obrázek 4.5: Nastavení jmen a zpřístupnění tlačítek prvnímu hráči.

Po zvolení otázky se klientovi zobrazí požadovaná otázka i s možnostmi, obrázek č.6.



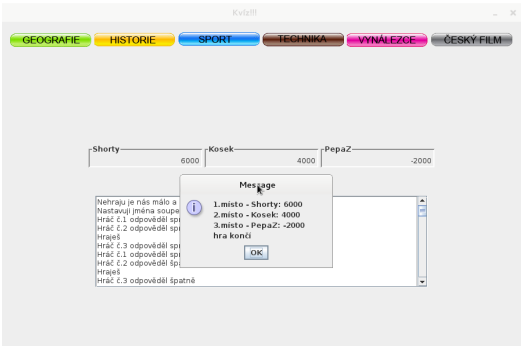
Obrázek 4.6: Ukázka otázky s možnostmi.

Po zvolení odpovědi server vyhodnotí otázku a předá zprávu dalšímu hráči o tom, že je na tahu. Přitom dojde k přičtení/odečtení bodů, obrázek č.7.

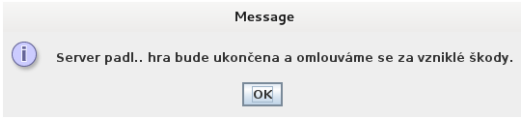


Obrázek 4.7: Ukázka otázky s možnostmi.

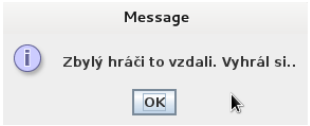
Hra končí po odehrání všech otázek. Nastane vyhodnocení podle příslušných bodů, obrázek č.8. V případě, že se server nečekaně ukončí, zobrazí se chybová hláška na obrázku č.9. V jiném případě, když se klient odpojí a zbydou dva, tak pokračují ve hře, když se odpojí i druhý hráč, poslední hráč ve hře automaticky vyhrál, obrázek č.10.



Obrázek 4.8: Vyhodnocení hry.



Obrázek 4.9: Dialog, který se zobrazí všem uživatelům v případě ukončení serveru.



Obrázek 4.10: Dialog pro hráče, který zůstal sám ve hře.

5 Závěr

I přestože jsem nepoužil nejefektivnější metody, podařilo se mi dosáhnout uspokojivých výsledků. Program by bylo možné ještě dále vylepšit. Semestrální práce byla velice přínosná. Myslím si, že jsem se částečně naučil programovat v jazyce C a poznal jsem jeho světlé (rychlost) i stinné (hledání chyb) stránky. Díky dokumentaci jsem se seznámil i s typografickým systémem TEX, resp. LATEX. S prací jsem spokojen.