FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce

předmět: KIV/UPS

15.01.2022
Michaela Benešová
A19B0008P

zadání: Síťová počítačová hra kvíz

1. Zadání semestrální práce

Téma: vybírá si student sám -> já jsem vybrala téma *Kvíz*

Zásady vypracování semestrální práce:

Úlohu naprogramujte v programovacím jazyku C/C++ anebo Java. Pokud se jedná o úlohu server/klient, pak klient bude v Javě a server v C/C++.

Komunikace bude realizována textovým nešifrovaným protokolem nad TCP protokolem.

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.

Realizujte konkurentní (paralelní) servery. Server musí být schopen obsluhovat požadavky více klientů souběžně.

Součástí programu bude trasování komunikace, dovolující zachytit proces komunikace na úrovni aplikačního protokolu a zápis trasování do souboru.

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.).

Zdrojové kódy organizujte tak, aby od sebe byly odděleny části volání komunikačních funkcí, které jste vytvořili na základě zadání, od částí určených k demonstraci funkčnosti vašeho řešení (grafické rozhraní).

Kvíz a jeho funkčnost:

- název hry Kvízeček
- max. počet lidí ve hře 3
- hráč odpovídá na jednoduché otázky pomocí klikání myší
- může se připojit na běžící server pomocí vyplnění ip adresy a portu
- hráč si zvolí heslo a přezdívku
- v případě odpojení v běhu hry se může znovu připojit (načtou se mu veškeré informace, především jeho získané body)

2. Programátorská část

Úvod

Hra kvíz se skládá ze serverové části a klientské. Server je naprogramován v jazyce *C*. Klient a GUI v jazyce *python*. Celkem se aplikace skládá ze čtyř tříd, které spolu vzájemně komunikují.

msg_codes.py

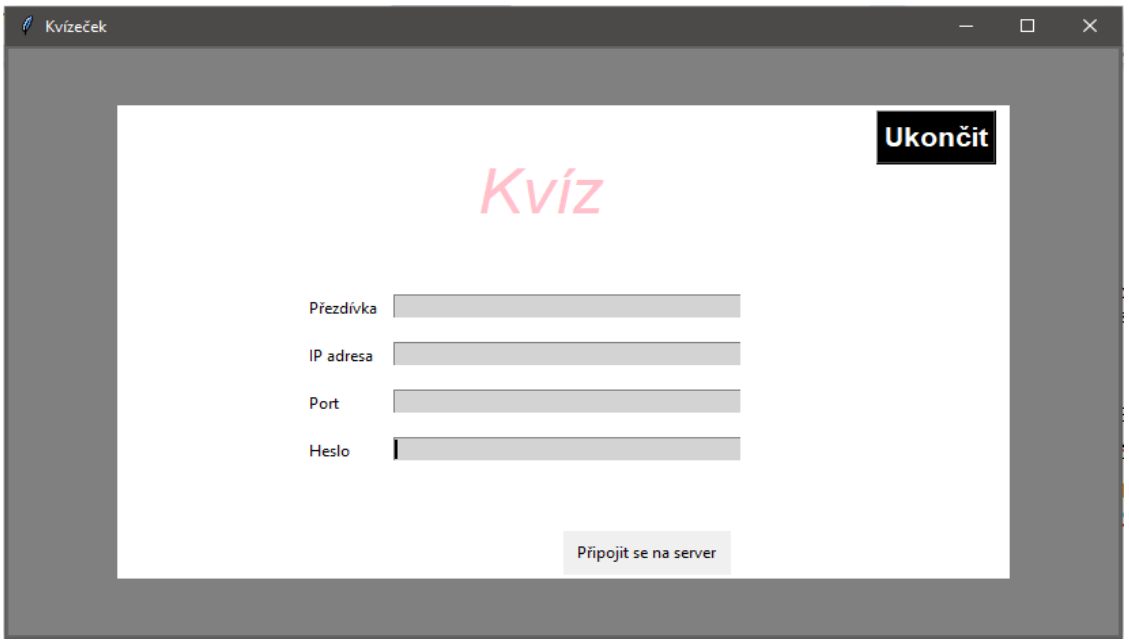
Tento skript obsahuje jednotlivé kódy zpráv a jejich číselné označení. Kódy slouží ke společné komunikaci serveru a klienta, aby dokázali zpracovat své vzájemné požadavky.

Používané kódy:

REQ_ID = 1
SEND_QUIZ_ANSWER = 6
CONNECT_TO_GAME = 2
RECONNECT_TO_GAME =3
CREATE_NEW_ROOM = 4
START_GAME =5
SEND_QUIZ_ANSWER = 6
PING = 7
LEAVE_GAME = 8
NEXT_QUESTION = 9
BACK_TO_MENU = 10
SHOW_TABLE = 11
ERR = -1

Quiz.py

Tento skript slouží ke tvorbě GUI. Využívá knihovnu tkinter. Obsahuje metody na vytvoření jednotlivých oken, potřebných textů, labelů a tlačítek. Tlačítka vykonávají příslušné funkce -> volají buď třídní metody a nebo metody z klienta.



Obr.1: Ukázka GUI

client.py
Tato třída představuje klienta, aneb prostředníka v komunikaci mezi serverem a GUI.

Pokud je hráč úspěšně připojen, běží metoda recieve_from_server() pro každého hráče ve vlastním vlákně, tak aby se stále naslouchalo zprávám ze serveru. Zároveň pokud dojde na serveru k úspěšnému přidělí ip, rozběhne se další vlákno pro pingování serveru, aby se vědělo, zda server běží. Pokud by nedal odpověď do 1s, objeví se hláška, že server nefunguje a hráč bude nucen se vrátit na úvodní obrazovku.

Ke komunikaci jsou využívány kódy zpráv. Většina akcí se skládá ze dvou metod -> jedna žádá server, druhá zpracovává odpověď.

server.c

Server má nadefinované základní konstanty v konstruktoru a spustí vlákno, které zpracovává požadavky z klienta a odpovídá na něj zpět. Obsahuje struktury pro hráče, hru a otázku. Dochází zde k veškerým akcím -> přiděluje se hráči id, kontroluje se odpověď na otázku, odesílají se potřebné informace na klienta, který je zpracuje a přizpůsobí GUI, například zasílá ID místnosti adminovi hry (zakladateli), aby věděl, kam se mají spoluhráči připojit.

Jak to celé funguje?

V konstruktoru skripu *Quiz.py* se nadefinují jednotlivé proměnné, které se budou dále využívat. Poté se zavolá metoda pro zobrazení hlavní stránky hry.

Po vyplnění údajů a kliknutí na *Připojit se na server* se zavolá metoda klienta a ten se pokusí spojit se serverem. Podle odpovědi se buď objeví zpráva, že server nefunguje a vyžádá si nové načtení a nebo dojde k úspěšnému připojení.

Vytvoří se vlastní vlákno, ve kterém bude po celou dobu běhu programu klient naslouchat zprávám ze serveru. Zároveň se vytvoří ještě jedno vlákno v klientské třídě, které bude kontrolovat, zda je server aktivní.

Zobrazí se poté i další obrazovka, kde si hráč vybere, zda se chce připojit do již existují místnosti, či si vytvořit novou.

Pokud hráč založil novou hru, bude moci rozhodovat o startu hry. Jakmile hru nezaložil, objeví se mu věta *“Počkej na start hry”*. Hra

započne, objevují se tři otázky v náhodném pořadí. Bod získává ten, kdo odpoví první správně na otázku.

Po skončení hry, má hráč možnost se vrátit zpět na úvodní obrazovku a znova se připojit na server. Dochází ke smazání údajů o celé hře na serveru, hráč případně získá i nové ID.

V případě odchodu ze hry, má možnost se v průběhu hry do ní znova připojit, tím, že na úvodní obrazovce vyplní svoje jméno a heslo. Pokud by se do hry nevrátil, znamená to, že hra již skončila a může založit svou vlastní hru a nebo se připravit do již existující.

Komunikace a formáty zpráv (protokol) jsou uvedené v příloze na konci tohoto dokumentu.

3. Uživatelská část

Spuštění serveru

Soubor server.c je nejdříve nutné přeložit příkazem (v příkazové řádce ve složce, kde je umístěn):

gcc server.c -lpthread -o server -w

Následně se server spustí:

./server

PORT je pevně nastaven na 10002.
IP adresa se mění (podle toho, kde je server spuštěn) a lze zjistit pomocí příkazové řádky, zadáním příkazu ipconfig/all .

Spuštění klienta

K úspěšnému spuštění je třeba mít nainstalovaný python3 a knihovnu tkinter.

WINDOWS:
Získání python3: Na oficiálních stránkách pythonu provést instalaci nejnovější verze 3.x.x. Provést instalaci. Ověření -> v cmd: *pip -V* zobrazí se verze a umístění.

Získání knihovny tkinter: v příkazové řádce napsat *pip install tk*.
--- po nainstalování potřebného jazyka a knihovny
Spuštění: Ve složce, kde se nachází hra spustit příkazovou řádku a napsat:

python3 Quiz.py

DEBIAN based LINUX:
Získání python3 příkazy v terminálu: *sudo apt update, sudo apt install software-properties-common, sudo add-apt-repository ppa:deadsnakes/ppa, sudo apt update,sudo apt install python3.8 ->* kontrola:*python --version*

Získání knihovny tkinter: v terminálu napsat **sudo apt-get install python-tk**
--- po nainstalování potřebného jazyka a knihovny
Spuštění: Ve složce, kde se nachází hra spustit terminál a napsat:

python3 Quiz.py

Poté se hráči zobrazí úvodní obrazovka GUI (viz Obr. 1, výše), vyplní jméno, heslo, ip adresu a port, kde server běží. Po správném vyplnění si může užívat hru Kvízeček.

4. Závěr a možné vylepšení

Hra Kvízeček je zábavná jednoduchá kvízová hra nabízející zábavu s přáteli a zároveň získání nových vědomostí. Funguje plynule a přehledně. GUI je vytvořeno přívětivě pro uživatele.

Mezi možná budoucí vylepšení se řadí tvoření otázek, oddělení otázek do databáze, možnost přidání vlastní sady otázek a nastavení libovolného počtu hráčů v místnosti.

Protokol

Formát zprávy na server:

odesílá se textový řetězec
msg = id_hráče, kód zprávy číselně či textově,parametr

odeslání na server: self.soc.sendall(msg.encode())

Formát zprávy ze serveru:

odesílá se textový řetězec
msg = kód zprávy číselně či textově, parametr - v případě více parametrů oddělení středníkem

odeslání na klienta: write(socket, msg, strlen(msg)) - délka zprávy

ukázka kódu - veškeré parametry převedeny na řetězce, žádost o zobrazení místnosti v případě admina:

```
char game_id_text[8];
            sprintf(game_id_text, "%d", game_id);
char *msg_for_admin = malloc(strlen("4,") + strlen(game_id_text) +
strlen(text_number) + 1);
            memcpy(msg_for_admin, "4,", strlen("4,"));
            memcpy(msg_for_admin + strlen("4,"), game_id_text,
strlen(game_id_text));
            memcpy(msg_for_admin + strlen("4,") + strlen(game_id_text),
";", strlen(";"));
            memcpy(msg_for_admin + strlen("4,") + strlen(game_id_text)
+ strlen(";"), text_number, strlen(text_number)+1);
write(games[game_id].array_p[0].socket, msg_for_admin,
sizeof(msg_for_admin));
            free(msg_for_admin);
```

Na server

Ze serveru

připojení:
self.soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
self.soc.connect((self.HOST, self.PORT))

Žádost o ID:
-1,REQ_ID,jméno+heslo

zpracování ID:
nový hráč: 1,id

reconnect (má stejné jméno a heslo jako někdo předtím a hra ve které byl je aktivní nebo čeká na start):

hra v lobby

1, id
2,počet_hráčů

hra ve hře:

1, id

načte se momentální otázka ↓
5, otázka;
opověď-opdověď-
odpověď, odpověď

Ve vlastním vlákně od získání id *běží ve while(proměnná=True), aby se získala informace o tom, kdyby server přestal fungovat***:**
id_hráče,PING,x

Vytvoření nové místnosti:
id_hráče,CREATE_NEW_ROOM,x

Vrácení id místnosti zakladateli:
4,id_místnosti

Připojení do stávající místnosti:
id_hráče,CONNECT_TO_GAME, id_místnosti

Odpověď hráčům, kromě zakladatele:
2,počet_hráčů
Odpověď zakladateli:
4,id_místnosti;počet_hráčů

Spuštění hry:
id_hráče, START_GAME,x

První otázka *všem hráčům***:**
5,otázka;odpověď-odpověď-odpověď-odpověď

Odeslání odpovědi:
id_hráče,SEND_QUIZ_ANSWER,číslo odpovědi (1 až 4)

špatná odpověď -> nic se nestane
správná odpověď ↓ (všem hráčům)
6,body;správná odpověď

---

spuštění dalšího kola
vytvoření vlákna, které se uspí -> slouží jako odpočet do další otázky
zavolání metody next_round(game_id)
pokud nejsou odehrána veškerá kola, spustí se další otázka
jinak se ukáže vyhodnocení

Další otázka *v případě zbývajících kol, všem hráčům*↓
5,otázka;odpověď-odpověď-odpověď-odpověď

Vyhlášení v případě dohrání hry↓
Vítěz:
11,Vyhrál jsi
Poražení hráči:
11,Prohrál jsi

Opuštění hry:
id_hráče,LEAVE_GAME,x
dojde ke smazání údajů o hráči, jen v případě, pokud ukončuje klienta po dohrání hry
jinak se jeho údaje zachovávají, kvůli reconnectu
po opuštění hry v lobby a hře zůstává počet hráčů neměnný, jelikož se hráč může znovu připojit (reconnect)

Odeslání erroru:
-1,text