Dokumentace projektu 2: Chatovací klient do IPK 2016/2017
Jméno a příjmení: Marek Šipoš
Login: xsipos03

1 Analýza požadavků

Zadáním úlohy byla implementace jednoduchého chatovacího klienta, který se bude schopen připojit na určitý server (specifikovaný IP adresou) a poté na server zasílat zprávy a zprávy také číst. Klient by měl být schopen provádět zároveň několik úloh:

- Číst standardní vstup a odesílat jej na server
- Číst zprávy ze serveru a vypisovat je na standardní výstup v určitém formátu
- Korektně se ukončit po přijetí signálu `SIGINT`.

2 Implementace

Následující část popisuje samotný způsob implementace aplikace v jazyce C, od parametrů, přes práci se sockety až po ošetřování chybových stavů.

2.1 Načítání parametrů

Vzhledem k tomu, že zadání požadovalo načítání krátkých parametrů, byla použita knihovna `getopt`, která umožňuje jednoduše specifikovat jednotlivé parametry jako řetězec (každý znak představuje jeden parametr) a jejich hodnoty poté načítat.

Dále bylo třeba ošetřit možné chyby – například stav, kdy uživatel zadal nějaký parametr vícekrát, zadal neznámý parametr nebo zadal parametr bez příslušné hodnoty (např. parametr `-u` bez uživatelského jména). Dále byly ošetřeny neplatné vstupy, například uživatelské jméno s netisknutelnými znaky nebo příliš dlouhé uživatelské jméno.

2.2 Připojení

Po zpracování parametrů byl vytvořen soket, který byl nastaven jako neblokující (funkce `fcntl`). K převodu případné jmenné adresy na číselnou byla využita knihovní funkce `gethostbyname`. Výsledná adresa byla spolu s portem 21011 (který byl specifikován zadáním) použita jako základ pro strukturu `sockaddr_in`. Tato adresa se pak použila pro navázání spojení funkcí `connect`, jejíž úspěch v určitém časovém horizontu (10 sekund) byl testován funkcí `select`, která zároveň zajišťovala pasivní čekání (čímž nedošlo ke zbytečnému přetěžování CPU).

2.3 Čtení a zápis (práce s vlákny)

Po navázání úspěšného spojení byla na server odeslána zpráva `username logged in`, kde `username` představovalo uživatelské jméno zadané uživatelem. Poté bylo třeba naráz zajistit čtení zpráv ze serveru, čtení uživatelského vstupu a reakci na signál `SIGINT`. Proto bylo využito vláken. Celkově byla vytvořena vlákna dvě – jedno pro čtení standardního vstupu (funkce `select, send, fgets` a `snprintf`) a odesílání zpráv s předepsaným formátem na server, druhé pro čtení případných zpráv ze serveru, které bylo třeba vypsat na standardní výstup (funkce `select, recv` a `fprintf`). Každé vlákno pro tyto operace využívalo vlastní buffer.

Reakce na signály obstarávalo hlavní vlákno, jehož identifikátor byl uložen do globální proměnné pomocí funkce `pthread_self`. Pro jednoduchou komunikaci mezi vlákny bylo využito uživatelského signálu `SIGUSR1`. V případě, že jedno z vláken ztratilo spojení, odeslalo hlavnímu vláknu uživatelský signál funkcí `pthread_kill`. Hlavní vlákno poté uvolnilo všechny zdroje a ukončilo běh programu s chybovou hláškou o ztrátě spojení. V případě, že přišel signál `SIGINT`, hlavní vlákno obě vedlejší korektně ukončilo, poté uvolnilo zdroje a ukončilo běh programu.

2.4 Testování

Aplikace byla testována za použití referenční aplikace serveru na referenčním operačním systému CentOS 7[1] a přeložena příkazem `gcc` s parametry `-Wall -pedantic -std=c99`.

[1] Image dostupný zde: http://qwe.fit.vutbr.cz/igregr/centos7.ova