Vysoké učení technické v Brně
Fakulta informačních technologií

Předmět IPK.

Klient-server pro získání informací o uživatelích

Varianta 1

Autor: Bolshakova Ksenia (xbolsh00)

Brno
12.03.2018

Obsah:

1. Úvod

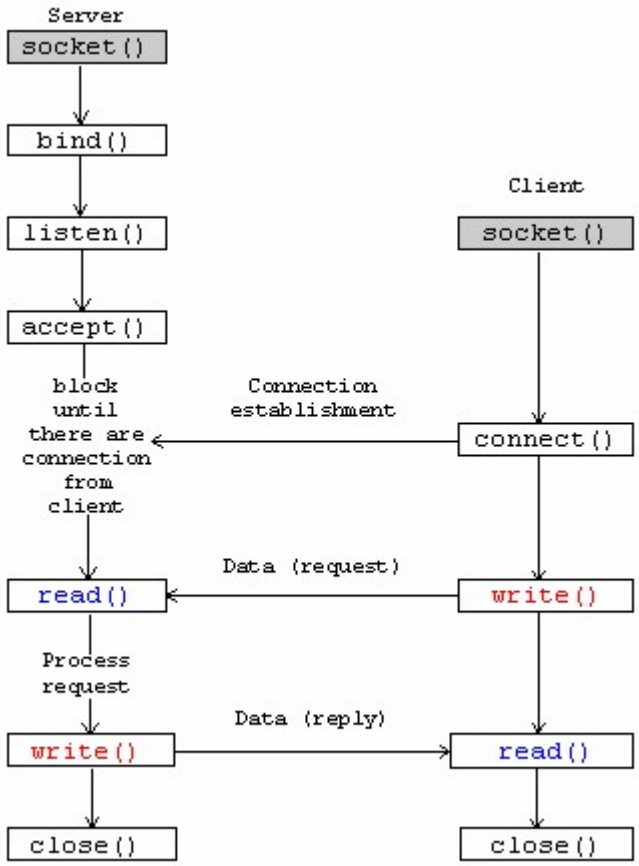
Zadání bylo rozděleno na dva úkoly.

Úkol č. 1 byl seznámit se s kostrami kódů pro programování klientských a serverových síťových aplikací za použití BSD soketů a navrhnout vlastní aplikační protokol pro přenos dat mezi dvěma koncovými body, realizující přenos informací o uživatelích na straně serveru .

Úkol č. 2 byl naprogramovat jak klientskou, tak serverovou aplikaci v C/C++ realizující zprostředkování informace o uživatelích na serveru.

2. Návrh řešní

Po seznámení s informacemi z přednášek a po nastudování příkazů pro systémové volání jsem navrhla vlastní protokol. Kostru protokolu jsem vzala z demo_cvičení předmětu IPK a změnila jsem ji podle potřeb projektu. Mohli jsme použít buď TCP nebo UDP spojení. Vybrala jsem TCP, protože je to spolehlivější varianta.



2.1 Na straně clienta bylo potřeba získat IP adresu serveru pomocí DNS serveru. Dále bylo nutné vytvořit soket a provést komunikaci mezi serverem a klientem.

Pro získání adresy byla použita funkce *gethostbyname()*, v případě chyby bude výpis na stderr.

Pro nalezeni IP adresy serveru používám funkce *htons()*, která konvertuje číslo portu.

Vytvoření soketu bylo pomocí funkce *socket()*, a propojeni mezi sokety bylo vytvořeno pomocí funkce *connect()*, v případě chyby – výpis na stderr.

Pro odesílání a přijetí zprávy využívám funkce *send()* a *recv()*, v případě chyby – výpis na stderr.

2.2 Na straně serveru získáme číslo portu, vytvoříme soket a přidělíme mu adresu pomocí funkce *bind()*.

Pomocí funkce *listen()* označíme soket jako tzv. zásuvku,
tj. zásuvka, která bude použita k přijímání příchozích připojení
žádosti o přijetí.

Připojení se provádí pomoci funkce *accept()*.

Funkce *inet_ntop()* konvertuje IP4 a IP6 adresy z binarniho kodu na text.

Pro odesílání a přijetí zprávy využívám funkce *send()* a *recv()*, v případě chyby – výpis na stderr.

Funkce *close()* uzavírá soket.

3. Implementace

Na začátku jsem musela zpracovat argumenty na vstupu pro clienta a server.

Pro clienta je vstup ve tvaru: ./ipk-client -h *host* -p *port* [-n|-f|-l] *login*

Pro server: ./ipk-server -p port

Zpracovávám pomocí funkce *getopt()*, případě chyby výpis na stderr.

Na straně clienta uchovávám login do bufferu, a pak na konec řetězce přidávám parametr argumentu (-n, -f, -l) a posílám na server.

Na serveru, když dostanu buffer, podívám se na poslední znak řetězce, kde uložen vstupní parametr (n, f, l), a nastavím proměnnou *prom* na specialní číslo, a ten znak z bufferu vymažu. Podle proměnné *prom* funkce *processing_arg()* zpracovává vstupní řetězec. Pomocí knihovny *pwd.h* dostanu informace o uživateli, který má login, uložený v bufferu, a uložím do proměnné *info*, a vrátím ji. Výsledek funkce uložím do pomocné proměnné, a pošlu zpátky do clienta. Aby spravně buffer na straně clienta dostaval loginy v případě argumentu *-l*, a neořezaval to, když těch loginů je hodně, musela jsem udělat dynamický buffer, a taky posilám delku bufferu serveru.

4. Vstupy/Výstupy

-n

./ipk-client -h merlin.fit.vutbr.cz -p 555556 -n xbolsh00
Bolshakova Ksenia,FIT BIT 2r

./ipk-client -h merlin.fit.vutbr.cz -p 555556 -n rysavy
Rysavy Ondrej,UIFS,541141118

-f

./ipk-client -h merlin.fit.vutbr.cz -p 555556 -f xbolsh00
/homes/eva/xb/xbolsh00

./ipk-client -h merlin.fit.vutbr.cz -p 555556 -f rysavy
/homes/kazi/rysavy

-l
./ipk-client -h merlin.fit.vutbr.cz -p 555556 -l xbol
xbolek00
xbolfr00
xbolsh00
xbolti01

V případě chyby nevypíše nic.

Strana serveru
./ipk-server -p 555556

5. Závěr:

Podařilo se mi udělat protokol, a aplikace klient – server, otestovala jsem to na merlinovi, a systemech Unix a Linux, a funguje to. Bohužel těsně před odevzdaním jsem se snažila ještě doladit -l, protože mě nevypisovalo všechny loginy, pokud nezadam zadny arfument, a zapoměla jsem zalogovat projekt, me se rozbilo to vypisovani loginu, ten buffer. Myšlenka byla takova, že jsem založila dynamický buffer, a pak poprve do něho posilám velikost bufferu, kterou musí naallokovat, zatím posilám všechny loginy po jednom.

Na eve to otestovat se mi nepodařilo. Nejprve nešlo mě to přeložit, pak jsem opravila Makefile, a potom se stalo to, žeclient se nechce se připojít se serverem.

6. Zdroje:

přednašky

K UROSE , J. a R OSS , K. W. Computer networking : a top-down approach. 6. vyd. New Jersey: Pearson Education, Inc., 2013. ISBN 0–13–285620–4.

http://man7.org/linux/man-pages/