Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

1. samostatná úloha z předmětu

Počítačové sítě

RPC

Jiří Kučera, A08N0092P

kalwi@students.zcu.cz

23. 6. 2009

Zadání první samostatné úlohy - RPC

Sestavte program pro vzdálené volání podprogramů, který bude jako parametry přenášet netriviální datové struktury. Vlastní úlohu pro výpočet si zvolte dle vlastního uvážení (např. pro zadané vrcholy trojúhelníka vypočtěte jeho obvod, obsah a polohu těžiště). Úlohu doporučuji řešit ve třech krocích:

1. Realizujte jednoduchou úlohu synchronního vzdáleného volání. Připravte specifikační soubor s definicí rozhraní (*.x), hlavní program klienta a těla jednotlivých podprogramů. Spojky serveru i klienta vygenerujte pomocí programu rpcgen. Pro ladění se vyplatí nejprve odladit volání podprogramů lokálně, teprve pak provézt úpravu pro vzdálené volání.
2. Realizujte modifikaci pro asynchronní vzdálené volání. Asynchronnost spočívá v tom, že po dobu výpočtu funkce ve vzdáleném uzlu klient nečeká na výsledek, ale může provádět jinou činnost. Asynchronní volání lze realizovat tak, že v prvé fázi se přenesou parametry výpočtu a v druhé fázi se ze serveru získají výsledky. První fázi lze např. realizovat tak, že nastavíme timeout pro odpověď ze serveru na nulu a vyvoláme RPC. Voláním přeneseme pouze parametry, na odpověď serveru nečekáme. Jiná možnost spočívá ve vytvoření speciální samostatné funkce pro přenos parametrů (server potvrdí příjem). Druhou fázi realizujeme tak, že přenos výsledků provedeme dalším voláním RPC (s vyhodnocením dosažitelnosti výsledku), nebo tak, že s v okamžiku předání výsledků server s klientem vymění roli - server voláním RPC předá výsledky klientovi, resp. jeho RPC serveru.
3. V třetí části zadání upravte server tak, aby byl schopen zpracovávat déle trvající požadavky paralelně, tj. vlastní těla podprogramů zpracovávat jako samostatná vlákna nebo samostatné procesy.

Úlohu řeší student samostatně. Konkrétní zadání si zvolí sám tak, aby splnil jednu z podmínek - přenášené parametry musí být netriviálního typu (např.)Výše uvedený postup je třeba chápat jako postupné kroky řešení. Výsledkem budou tři samostatné programy a referát v elektronické podobě. Struktura adresářů by měla být následující: v kořenovém adresáři soubor *.pdf s referátem, případně READ.ME. Ve třech podadresářích kořenového adresáře jednotlivé modifikace zadání. Každá modifikace bude obsahovat podadresář se zdrojovými kódy a podadresář s binárními kódy. Mezi zdrojovými kódy bude i Makefile pro překlad a sestavení.

Mnou vybraná úloha vypočte délku zadaných slov, počet souhlásek, samohlásek a rozdílných písmen.

Programátorská dokumentace

Jako referenční aplikaci při psaní programu jsem použil vzorový příklad ze stránky:

http://www.kiv.zcu.cz/~ledvina/vyuka/PSI/exam-rpc.htm

Zdrojový kód každé z modifikací úlohy sestává ze čtyř souborů:

strpar_client.c

Zdrojový kód klientské aplikace. Čte uživatelský vstup, volá RPC funkci na serveru a prezentuje obdržené výsledky.

strpar_server.c

Zdrojový kód serverové aplikace. Naslouchá na volání RPC funkce, vypočte výsledek a vrátí jej.

strpar.x

Zdrojový kód RPC rozhraní pro komunikaci mezi klientskou a serverovou aplikací.

Makefile

Skript pro přeložení aplikace příkazem `make` a smazání produktů kompilace příkazem `make clean`. Při kompilaci rovněž automaticky vygeneruje RPC spojku nástrojem `rpcgen`.

Uživatelská příručka

Kompilace

Aplikaci je nutno před spuštěním zkompilovat, to se provede příkazem `make`. Příkazem `make clean` je pak možno všechny soubory vytvořené kompilací opět smazat.

Spuštění a používání aplikace

Program sestává ze serverové a klientské aplikace. Pro přehlednější orientaci ve vstupech a výstupech je vhodnější spustit obě aplikace v novém terminálovém okně. Z demonstračních důvodů se klientská aplikace připojuje pouze na `localhost`, pro připojení k jinému serveru je třeba změnit příslušnou proměnnou v klientské aplikaci a program překompilovat.

Nejprve je třeba spustit server:

```
./strpar_server
```

Poté je možno spustit klient:

```
./strpar_client
```

Jako vstup se zadává jedno nebo více slov oddělených mezerami. Vstup je také možno zadat přímo při spouštění klientské aplikace jako argumenty programu:

```
./strpar_client Hello world
```

Po zadání vstupu aplikace vypíše zvlášť pro každé slovo délku zadaného slova, počet rozdílných písmen, počet souhlásek a počet samohlásek.

Závěr

Aplikaci jsem odladil na školním serveru ul403ds10-kiv.fav.zcu.cz, na kterém je nainstalován operační systém Debian 4.1.1-19 (jádro Linux 2.6.18.2, gcc 4.1.2 20061028).

Přesné určení doby trvání vývoje aplikace je zpětně obtížný. Odhaduji, že práce trvala včetně lazení a psaní dokumentace tři až čtyři dny (po 12 – 16 hod.).