Tema 1 - SPRC
RPC Remote Shell

Responsabil tema:
Florin Pop (`florin.pop@cs.pub.ro`)

Termen de predare:
Duminica, 3 Noiembrie, 2013, ora 23.55

---

1 Specificatii functionale

Realizati o aplicatie de tip Remote Shell bazata pe RPC, care permite executarea de comenzi (in limita drepturilor de acces) pe un server aflat la distanta. Arhitectura aplicatiei presupune doua procese care comunica intre ele prin protocolul RPC (modelul client-server):

- un client (`rclient`) care pune la dispozitie un terminal de tip prompt;
- un server (`rserver`) care va executa comenzile cerute de client.

Clientul va afisa un prompt de comanda care arata directorul curent de lucru al serverului (printr-un apel RPC se aduce valoarea variabilei `pwd`). Clientul va interpreta comenzile de intrare si le va procesa local, prin parsarea sirului de caractere ce descrie o comanda, care poate sa fie simpla sau compusa. O comanda se considera compusa daca contine cel putin o data operatorul de secventiere (`;`). In functie de tipul comenzii, clientul va o face cerere de executie server-ului:

- `Exec_simple_command(...)`, pentru o comanda simpla (dar care poate contine redirectari);
- `Exec_command(...)`, pentru o comanda compusa doar prin operatorul de secventiere (`;`).

Clientul va afisa rezultatele intoarse de apelurile RPC (inclusiv notificari despre incheierea normala sau anormala a solicitarilor clientului). Pentru fiecare comanda executata care nu afiseaza nimic la consola (de exemplu la redirectarea in fisier) se va afisa un mesaj prin care se specifica daca comanda s-a executat cu succes sau nu.

Comenzile trimise spre executie nu sunt distructive. Clientul trebuie sa suporte o comanda speciala (`local exit`) care sa permita terminarea acestuia.

Un schelet al aplicatiei client (care descrie sumar comportamenul clientului, functiile folosite se vor implementa de voi) poate fi urmatorul:

```
while(1){
    char buffer[BUFSIZE];    // folosit pentru comanda de intrare
    result* res;             // structura definita in fisierul .x
    rpc_cmd_data pdata;      // structura definita in fisierul .x

    res=Get_remote_pwd();                              // apel RPC
    prompt(res);
    getUserCommand(buffer);                           // apel local

    if ( !parseCommand(buffer, &pdata) ) {
        continue;
    } else {
        switch( commandType(buffer) ) {
           case EXIT:
               return 0;
           case COMMAND:
               res = Execute_command(pdata);       // apel RPC
               break;
           case SIMPLE_COMMAND:
               res = Execute_simple_command(pdata);// apel RPC
               break;
        }
        printResult(res);
    }
}
```

Se poate observa ca raspunsul primit de la server este o structura, pe care trebuie sa o definiti voi. Puteti folosi o lista liniara, simplu inlantuita care sa o folositi pentru raspunsul comenzilor compuse.

2 Specificatii non-functionale

- Server-ul poate primi cereri de la mai multi clienti in paralel. Deoarece comenzile se pot suprapune, voi decideti ce variabile le declarati `static`.
- Erorile de comunicatie (si erorile in general) trebuie rezolvate "gracefully". Se recomanda o implementare defensiva prin care erorile sunt tratate.
- Conexiunea client-server trebuie inchisa la terminarea executiei clientului prin "eliberarea lina" a conexiunii (clientul nu se termina brusc prin comezi de genul `Ctrl+C`).

3 Ipoteze initiale

- IP-ul serverului este cunoscut de la inceput de toti clientii si nu se schimba.
- Nu ne intereseaza aspectele de autentificare (certificate, conexiuni securizate etc.) ale clientilor. Cu alte cuvinte, se presupune ca nu exista clienti malitiosi in retea.
- Comenzile sunt introduse de la tastatura, deci contin numai caractere alfa-numerice. Lungimea unei comenzi, fie ea simpla sau compusa, nu va depasi 1024 caractere.

4 Constrangeri de implementare

- Implementarea se va realiza folosind limbajul C si utilitarul `rpcgen`;
- Fisierul care defineste functionalitatea serverului, functiile si structurile se va numi `rshell.x`;
- Comportamentul clientului se va scrie in fisierul `rclient.c`;
- Comportamentul serverului se va scrie in fisierul `rserver.c`;
- Pentru lansarea in executie a serverului se va folosi urmatoarea comanda:

  ```
  ./rserver &
  ```

- Pentru lansarea in executie a clientului se va folosi urmatoarea comanda (`server_ip` este IP-ul serverului):

  ```
  ./rclient server_ip
  ```

5 Specificatii de deployment

Structura temei trebuie sa includa cel putin urmatoarele:

- sursele de baza in radacina arhivei. Arhiva temei de casa va contine cel putin urmatoarele fisiere: `makefile`, `rclient.c`, `README`, `rserver.c` si `rshell.x`;
- Deployment-ul, compilarea si rularea aplicatiei se vor face folosind `make`. Fisierul `makefile` aflat in radacina arhivei trebuie sa contina cel putin urmatoarele targeturi: `clean`, `build`, `run`. Tema trebuie organizata in asa fel incat deploymentul sa fie posibil pe orice masina (cu alte cuvinte, sa fie de tipul "extract-compile-and-run", fara alte operatii de executat manual).

6 Observatii

- Inainte de trimiterea temei de casa trebuie sa cititi cu atentie regulamentul cursului de SPRC disponibil pe site-ul cursului si sectiunile 4 si 5.
- Tema nu este complexa si va sfatuim sa nu o complicati inutil prin diverse presupuneri.
- Intrebarile cu privire la rezolvarea temei de casa se vor pune pe forumul dedicat acesteia, forum disponibil pe site-ul cursului. NU publicati bucati de cod sursa pe forum!