Master 1 Informatique - Réseaux et Communications - TD 05
Remote Procedure Call (RPC)

L'appel procédural à distance (RPC) permet à des processus clients d'exécuter des procédures localisées dans d'autres processus, appelés processus serveurs.

Les systèmes RPC sont des outils de programmation, qui facilitent la programmation des applications basées sur le modèle client/serveur. Une application RPC simple est composée d'un seul client qui communique avec un seul serveur. Le processus client fait appel au serveur en passant des arguments. Un appel procédural synchrone permet au client d'attendre les résultats retournés par le serveur.

Le but de ce TD est de spécifier/développer quelques applications client/serveur simples basées sur le modèle de communication RPC.

1 Nombres premiers

On souhaite développer une application client/serveur dans laquelle le client demande à un serveur de vérifier si un nombre est premier. Écrivez le contrat de l'application (*prime.x*), le code du client et celui du serveur. **Rappel :** un nombre entier plus grand que 1 est un nombre premier si ses seuls diviseurs sont 1 et lui même.

2 Min Max d'un vecteur

On souhaite développer une application client/serveur dans laquelle le client demande à un serveur de lui calculer le minimum et le maximum d'un vecteur de nombres entiers. Proposez une spécification dans un fichier *minmax.x* du protocole de communication entre le client et le serveur.

3 Inscriptions à un concours

Le but de cet exercice est de développer une application client/serveur pour la gestion des inscriptions à distance.

1. Donnez le contrat d'une telle application comportant les procédures suivantes :
   - **ajouter_un_client** : cette procédure rajoute un client à la liste des clients inscrits. Un client est défini par 3 paramètres de type chaîne de caractères : nom, prénom et adresse, et un paramètre de type réel age.
   - **liste_des_clients** : cette procédure retourne la liste des clients inscrits.
   - **supprimer_client** (optionnelle) : cette procédure supprime un client (annule une inscription) de la liste des inscrits. Elle a un paramètre : le nom du client à supprimer.
   - **nb_clients** (optionnelle) : cette procédure retourne le nombre de clients inscrits.
2. Donnez un schéma algorithmique pour le client. L'accent est mis sur la description de la connexion au serveur et des appels de procédures distantes. Puis, décrivez les opérations implémentées par le serveur.

4 Aide

La compilation d'un contrat avec la commande *rpcgen* génère trois ou quatre fichiers. Ces fichiers réalisent la déclaration de constantes et types utilisés par le client et le serveur, l'encodage et le décodage des arguments et l'implémentation des stubs.

Un aperçu du code généré pour l'application de la section 1 vous est fourni. Il constitue une aide à la réalisation des applications du TD.

prime.h

```
#include <rpc/rpc.h>
#define PRIMEPROG 0x2000009a
#define PRIMEVERS 1
#if defined(__STDC__) || defined(__cplusplus)
#define IS_PRIME 1
extern  bool_t * is_prime_1(int *, CLIENT *);
extern  bool_t * is_prime_1_svc(int *, struct svc_req *);
extern int primeprog_1_freeresult (SVCXPRT *, xdrproc_t, caddr_t);

#else
...
#endif
```

prime_prime_clnt.c

```
#include <memory.h> /* for memset */
#include "prime.h"
...
bool_t * is_prime_1(int *argp, CLIENT *clnt){
   static bool_t clnt_res;
   memset((char *)&clnt_res, 0, sizeof(clnt_res));
   if (clnt_call (clnt, IS_PRIME,(xdrproc_t) xdr_int, (caddr_t) argp,(xdrproc_t) xdr_bool,
                  (caddr_t) &clnt_res, TIMEOUT) != RPC_SUCCESS) {
     return (NULL);
   }
   return (&clnt_res);
}
```

prime_prime_svc.c

```
#include "prime.h"
#include <sys/socket.h>
...
int main (int argc, char **argv){
  register SVCXPRT *transp;
  pmap_unset (PRIMEPROG, PRIMEVERS);
  transp = svcudp_create(RPC_ANYSOCK);
  if (transp == NULL) {
    fprintf (stderr, "%s", "cannot create udp service."); exit(1);
  }
  if (!svc_register(transp, PRIMEPROG, PRIMEVERS, primeprog_1, IPPROTO_UDP)) {
    fprintf (stderr, "%s", "unable to register (PRIMEPROG, PRIMEVERS, udp)."); exit(1);
  }
  transp = svctcp_create(RPC_ANYSOCK, 0, 0);
  if (transp == NULL) {
    fprintf (stderr, "%s", "cannot create tcp service."); exit(1);
  }
  if (!svc_register(transp, PRIMEPROG, PRIMEVERS, primeprog_1, IPPROTO_TCP)) {
    fprintf (stderr, "%s", "unable to register (PRIMEPROG, PRIMEVERS, tcp)."); exit(1);
  }
  svc_run ();
  fprintf (stderr, "%s", "svc_run returned");
  exit (1);
}
```