Licence 3 Informatique – Parcours II – Conception d'Applications
2020 – 2021

TD 1 & TP 1 – *RPC* – Utilisation de *RPCgen*

Développement de service ***RPC*** avec ***rpcgen***

Mounir LALLALI mounir.lallali@univ-brest.fr

Rappel : *rpcgen*

Pour éviter d'avoir à programmer la partie réseau de l'application, en particulier celle concernant le serveur, pour faciliter la production des filtres ***XDR***, il existe un utilitaire d'aide au développement des programmes utilisant les ***RPC***. Cet outil s'appelle ***rpcgen*** (cf. ***man rpcgen***). Il s'agit d'un pré-compilateur qui engendre plusieurs fichiers à partir d'une description synthétique des services, description donnée dans un fichier dont le type est ***x***. Le langage de description ressemble au langage ***C***. Par exemple, si le fichier de description s'appelle ***fichier.x***, les fichiers produits par l'exécution de la commande : ***rpcgen fichier.x*** seront les suivants :

- un fichier à inclure dans les programmes du serveur et des clients : ***fichier.h*** ;
- un squelette du code du serveur ***fichier_server.c*** ;
- les procédures réalisant les appels distants pour les clients : ***fichier_client.c*** ;
- les filtres ***xdr*** : ***fichier_xdr.c***.

Il reste donc à écrire deux fichiers :

- un fichier contenant le code des procédures fournies par le serveur qui complétera ***fichier_server.c***, lui-même contenant déjà la fonction ***main***,
- le code du client qui se servira de ***fichier_client.c***, ici on doit écrire la fonction ***main***.

Exercice 1. – Premier exemple

Il s'agit d'écrire un serveur **RPC** qui fournit les services suivants :

- ajouter un nombre à une variable partagée. Ce service sera demandé à l'aide d'une fonction ***one-way*** parce qu'on n'attend pas de réponse de la part du serveur (requête asynchrone),
- lire la valeur de cette variable.

La description pour ***rpcgen*** sera faite dans un fichier ***compteur.x***, son squelette est donné ci-dessous :

```
program COMPTEURPROG {
        version COMPTEURVERS {
        void AJOUT(int) = 1 ;
        int LIRE() = 2 ;
        } = 1 ;
} = ... ;
```

Il restera donc à écrire les fichiers :

- ***compteur_server.c***, qui complète ***compteur_svc.c*** et qui contiendra donc l'exécution des services,
- ***compteur_client.c***, qui doit contenir la fonction ***main*** utilisant ***compteur_clnt.c***, ce dernier fournissant les appels aux services.

Dans cet exercice, nous allons tester le bon fonctionnement du serveur en l'interrogeant avec des clients qui n'utilisent que le service ***Compteur***.

Etapes à suivre

1) Lancer ***rpcgen -a*** en utilisant le fichier ***compteur.x*** donné ci-dessus, on prendra ***0x23456789*** comme numéro de service. Vérifier que les différents fichiers ont bien été créés.

2) On donne ci-dessous le fichier ***compteur_server.c*** .

- La variable partagée qui est incrémentée est ***Compteur***.
- Vérifier que le programme fournissant le service ***Compteur*** est bien écrit.

```
#include "compteur.h"
int compteur = 0 ;
void * ajout_1_svc(int *argp, struct svc_req *rqstp)
{
        static char * result;
        compteur = compteur + *argp;
        printf("_____________________________________________________ \n") ;
        printf("ajout_1_svc : ajout du nombre : %d \n",*argp) ;
        return (void *) &result;
}
int * lire_1_svc(void *argp, struct svc_req *rqstp)
{
        static int  result;
        result = compteur;
        printf("ajout_1_svc : nouveau compteur : %d \n",compteur) ;
        return &result;
}
```

- Utiliser le ***makefile*** pour générer le fichier exécutable ***compteur_server***. Lancer cet exécutable dans un premier terminal.

3) On va maintenant créer l'application cliente qui utilise le service ***Compteur***. D'abord, il faut éditer le programme ***compteur_client.c*** suivant :

```
#include "compteur.h"
int compteur = 0 ;
void compteurprog_1(char *host, int  ajout_1_arg)
{
        CLIENT *clnt;
        void  *result_1;
        int  *result_2;
        char *lire_1_arg;
#ifndef  DEBUG
        clnt = clnt_create (host, COMPTEURPROG, COMPTEURVERS, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
#endif   /* DEBUG */
        result_1 = ajout_1(&ajout_1_arg, clnt);
        if (result_1 == (void *) NULL) {
                clnt_perror (clnt, " ajout_1 call failed");
        }
        result_2 = lire_1((void*)&lire_1_arg, clnt);
        if (result_2 == (int *) NULL) {
                clnt_perror (clnt, " lire_1 call failed");
        }
#ifndef  DEBUG
        clnt_destroy (clnt);
#endif    /* DEBUG */
}
Int main (int argc, char *argv[])
{
        char *host;
        int ajout_nombre = 0;
        if (argc < 3) {
                printf ("usage: %s server_host\n", argv[0]);
                exit (1);
        }
        host = argv[1];
        ajout_nombre = atoi(argv[2]);
        compteurprog_1 (host, ajout_nombre);
        exit (0);
}
```

- Utiliser le ***makefile*** pour générer le fichier exécutable ***compteur_client***.

4) Lancer cette application cliente depuis un autre terminal : ***./compteur_client host_name 5 (***pour connaître le nom du ***hostname,*** taper la commande ***sudo gedit /etc/hostname)***

5) Vérifier le bon fonctionnement en lisant les traces données à l'écran par le service (premier terminal).

Exercice 2. – Développent d'un service *RPC*

On veut développer un serveur ***RPC geometrie*** disposant de trois fonctions de traitement de formes géométriques :

- ***surface_rectangle*** : pour calculer la surface d'un rectangle ;
- ***rectangle creer_rectangle*** **: pour créer un rectangle ;**
- ***inclus*** **: pour vérifier l'appartenance d'un point à un rectangle.**

On supposera que pour un rectangle, le point ***p1*** est le coin inférieur gauche et ***p2*** le supérieur droit.

L'interface ***geometrie.x*** est fournie ci-dessous :

```
struct point { int x; int y; };
struct rectangle { struct point p1; struct point p2; };
struct coordonnees { int x1; int x2; int y1; int y2; };
struct param_inclus { struct rectangle rect; struct point p; };
typedef int booleen;
program GEOM_PROG {
        version GEOM_VERSION_1 {
                int SURFACE_RECTANGLE(rectangle) = 1;
                rectangle CREER_RECTANGLE(coordonnees) = 2;
                booleen INCLUS(param_inclus) = 3;
        } = 1;
} = 0x20000001;
```

Notes

- Nom du programme : ***GEOM_PROG*** avec un identifiant ***20000001*** (en ***hexa***)
- Nom de version : ***GEOM_VERSION_1*** de numéro ***1***
- Les 3 fonctions sont numérotées ***1***, ***2*** et ***3***
- Les structures ***coordonnees*** et ***param_inclus*** ont été créées pour les fonctions nécessitant plus de deux paramètres
- Par convention, on écrit les noms de fonctions, programmes et versions en MAJUSCULE

Questions

Q1. Lancer ***rpcgen –a*** en utilisant le fichier ***geometrie***.x.

Q2. Proposer une implantation pour chacune des trois fonctions dans le fichier ***geometrie_server.c***.

Q3. Proposer un client du service dans le fichier ***geometrie_client.c***.

Q4. Tester ce serveur et son client.