IUT 'A' Paul SABATIER

Dpt Informatique

S4

M4102C : Programmation répartie

TP3 : Programmation RPC

Éléments de corrigé

Objectif :

Mise en œuvre du mécanisme simple de RPC du système **Linux**.

Travail demandé :

Développer en langage **C**, une application répartie mettant en œuvre le mécanisme de RPC.

Spécifications :

1- Écrire un programme qui écrit un message dans le fichier poubelle **/dev/null**

2- Transformer le programme pour écrire le message dans le fichier poubelle d'une autre machine en utilisant les RPC

Éléments de corrigé :

1- /* exo1.c : imprime un message dans le fichier poubelle */

```
#include <stdio.h>
int printmessage(char *msg)
{
    FILE *f;
    f = fopen("/dev/null", "w");
    if (f==NULL) {
     { return(0); }
    fprintf(f,  "%s\n", msg);
    fclose(f);
    return(1);
}
```

```
main (int argc,  char **argv)
{
    char * message;
    if (argc!=2)
     {fprintf(stderr, "usage: %s <message>\n", argv[0]);
        exit(1); }
    message = argv[1];
    if(!printmessage(message))
     { fprintf(stderr, "%s: impossible d'imprimer votre
message", argv[0]); exit(1); }
    printf("Message delivre\n");
    exit(0);
}
```

2- /* exo1.x : protocole d'impression de message distant */

```
program MESSAGEPROG {
        version MESSAGEVERS {
                int PRINTMESSAGE(string)=1;
        }=1;
}=0x20000001;
```

---> numero different pour chaque groupe de TP

------> 0x20000001 + numero groupe (en hexa)

compilation avec **rpcgen** :

rpcgen exo1.x /* génération de **exo1.h**, **exo1_clnt.c** et **exo1_svc.c** */

/* exo1_proc.c : mise en œuvre de la procédure distante "printmessage" */

#include <stdio.h>

#include <rpc/rpc.h> /* toujours necessaire */

#include "exo1.h" /* genere par rpcgen */

/* version distante de "printmesssage" */

/* Rmq1 : toute procedure distante accepte en entree un pointeur sur les arguments qu'elle aurait eue en local */

/* Rmq2 : toute procedure distante retourne un pointeur sur les resultats qu'elle aurait retourne en local */

/* Rmq3 : `_1_svc` est rajoute au nom local de la procedure. rpcgen convertit en minuscule le nom de la procedure dans la definition de programme, rajoute le caractere de soulignement (_) et le numero de version (1 dans le cas present). */

```
int *printmessage_1_svc(char **msg, struct svc_req *rqstp )
{   static int result;   /* doit etre static */
    FILE * f;
    f = fopen("/dev/null", "w");
    if (f==NULL)
      { result = 0; return(&result); }
    fprintf(f, "%s\n", * msg);
    fclose(f);
    result = 1;
    return(&result);
}
```

Compilation du service :

```
gcc  -c  exo1_svc.c
gcc  -c   exo1_proc.c
gcc  exo1_svc.o   exo1_proc.o   -o exo1_svc   -lnsl
```

```
/* rexo1.c : version distante de exo1.c */

#include <stdio.h>

#include <rpc/rpc.h>    /* toujours necessaire */

#include "exo1.h"   /* genere par rpcgen */

/* Rmq 1 : une poignee client est creee a l'aide de la routine
de bibliotheque RPC clnt_create(). Cette poignee est transferee
aux routines temporaires qui appellent la poignee distante.*/

/* Rmq 2 : le dernier parametre de clnt_create() peut etre
"visible", ce qui specifie que tout transport visible dans
/etc/netconfig peut etre utilise.*/

/* Rmq 3 : La procedure distante printmessage_1() est appelee
exactement de la meme facon qu'elle est declaree dans
exo1_proc.c, sauf pour la poignee client qui se substitue au
second argument. */

/* Rmq 4 : la procedure distante peut echouer de 2 facons. Il
peut s'agir d'un echec du mecanisme RPC lui-meme ou d'une
erreur dans l'execution de la procedure distante. */

main (int argc,  char **argv)

{

    CLIENT *cl;

    int  *result;

    char *serveur;

    char *message;

    if (argc != 3)

     { fprintf(stderr, "usage: %s <hote> <message>\n",

               argv[0]); exit(1); }

/* sauvegarde des arguments */

    serveur = argv[1];

    message = argv[2];

/* creation de la poignee client utilisee pour appeler
MESSAGEPROG sur le serveur specifie sur la ligne de commande */

    cl = clnt_create(serveur, MESSAGEPROG, MESSAGEVERS, "tcp");
```

```
if (cl==NULL)

/* impossible d'etablir la connexion sur le serveur, impression
du message d'erreur et fin.*/

     { clnt_pcreateerror(server);  exit(1); }

/* Appel de la procedure distante "printmessage" sur le
serveur. */

    result=printmessage_1(&message, cl);

    if (result==NULL)

/* une erreur a eu lieu lors de l'appel du serveur. Impression
du message d'erreur et fin. */

     { clnt_perror(cl, server); exit(1); }

    if (*result==0)

/* le serveur n'a pas pu imprimer le message. Impression du
message d'erreur et fin.*/

     { fprintf(stderr, "%s: impossible d'imprimer votre

                 message", argv[0]); exit(1); }

/* le message a ete imprime sur le /dev/null du serveur */

    printf("Message delivre a distance\n");

    exit(0);

}
```

---

compilation du client :

gcc -c exo1_clnt.c

gcc -c rexo1.c

gcc exo1_clnt.o rexo1.o -o rexo1 -lnsl

<u>Exécution</u> : Le premier test se fera d'abord en local.

1- lancer "**exo1_svc**" dans une fenêtre terminal en **tache de fond**.

2- vérifier que le service est bien enregistré avec la commande :

rpcinfo -p

On doit voir deux lignes du style de :

536870913 1 udp 57344

536870913 1 tcp 36539

3- lancer "**`rexo1 localhost message`**" dans une autre fenêtre terminal.

Si tout va bien me message suivant est affiché :

```
Message delivre a distance
```

4- NE PAS OUBLIER DE TUER LE SERVICE **exo1_svc** APRES UTILISATION

ps -edf | grep $USER .. pour avoir le PID du processus **exo1_svc**

kill -9 …..

<u>Problèmes possibles</u> :

1- nom de machine: RPC: Program not registered

---> le service n'est pas lancé sur cette machine ou il n'a pas le bon numéro

2- nom de machine: RPC: Procedure unavailable

---> la procedure appelée n'a pas le bon numéro de procédure

3- nom de machine: RPC: Unable to receive; An event requires attention

---> le programme de service a peut-être fait un crash...

<u>Exécution</u> : Le second test se fera sur des machines UML.

<u>Note</u> :

Demander à l'enseignant de faire créer par l'administrateur système du département, sur chaque poste de travail, une interface spéciale appelée "**tap0**" à l'@IP **192.168.1.1** qui permet à la machine hôte de communiquer avec une machines virtuelle UML sur son interface **eth0**.

Vérifier avec la commande "**/sbin/ifconfig -a**" que cette interface est présente sur la machine hôte.

Redémarrer la machine **ST1** des TP précédents avec la commande **uml_lancer_serveur**.

Configurer son interface **eth0** pour que cette machine communique avec la machine hôte :

- **ifconfig eth0 192.168.1.10/24**
- **route add default gw 192.168.1.1**

Sur la machine **ST1**, sous le compte **root**, créer le répertoire **P112-TP3**.

Sur la machine hôte, télécharger (cmde **scp**) les programmes exécutables **exo1_svc** et **rexo1** dans le répertoire **P112-TP3** de la machine **ST1**.

Créer un réseau virtuel Ethernet **réseau#1** avec la commande **uml_creer_reseau**.

Redémarrer la machine **ST2** des TP précédents avec la commande **uml_lancer_machine**.

Connecter la machine **ST1** au **réseau#1** par son interface **eth1** à l'@IP **10.0.0.1/24**.
Connecter la machine **ST2** au **réseau#1** par son interface **eth1** à l'@IP **10.0.0.2/24**.
Vérifier que ces machine communiquent.

Sur la machine **ST2**, sous le compte **root**, créer le répertoire **P112-TP3**.

Sur la machine **ST1**, télécharger (cmde **scp**) le programme exécutable **exo1_svc** dans le répertoire **P112-TP3** de la machine **ST2**.

Sur la machine **ST2** lancer le programme exécutable **exo1_svc** en tâche de fond.
Vérifier que le service est bien enregistré avec la commande :

rpcinfo -p

Sur la machine **ST1** lancer le programme exécutable **rexo1** .

Vérifier que l'application répartie fonctionne correctement.

Arrêter les programmes sur chaque machine virtuelle.

Arrêter les 2 machines (cmde **halt**)