

**RPC : une application basée sur les primitives de haut niveau**

On se propose de développer un utilitaire de lecture de répertoires distants. Un serveur, disposant d'une routine locale "read\_dir()", répond aux requêtes d'un client. Côté client, on a juste besoin de faire un appel distant de la routine du serveur.

- 1) Qu'a-t-on besoin de faire côté serveur ?
- 2) Que fait-on côté client ?

On se propose d'écrire le code du serveur "rls\_svc.c" ainsi que le code du client "rls.c". Le programme client prend 2 valeurs en paramètre : le nom de la machine hôte et une chaîne de caractères, de longueur maximale DIR\_SIZE, correspondant au nom du répertoire à lister.

- 3) Donner le contenu du fichier en-tête "rls.h".
- 4) La routine locale "read\_dir()" du serveur lit le contenu du répertoire donné en paramètre et retourne la liste des fichiers sous forme d'une chaîne de caractères (penser à la fonction sprintf() par exemple). Les données échangées entre le client et le serveur étant de type chaîne de caractères, écrire le filtre XDR "xdr\_dir()" permettant la manipulation des données de type chaîne de caractères.
- 5) Ecrire le code du client, du serveur ainsi que de la fonction read\_dir() permettant de lire le contenu du répertoire passé en paramètre.

**Session :**

- 1) Dans une session où on a négocié les unités fonctionnelles noyau et transmission semi-duplex, que peut faire l'utilisateur qui ne possède pas le jeton des données ?
- 2) Une application de transfert de fichier nécessite des points de reprise au cours des échanges afin de pouvoir exécuter des redémarrages en cas d'erreurs graves. A quelles unités fonctionnelles cette application peut-elle faire appel ?
- 3) Le protocole de session gère un ensemble de jetons qui représente des droits exclusifs acquis sur demande par l'une ou l'autre des entités pour utiliser des services critiques. Sachant qu'il existe 4 jetons (données, terminaison, synchronisation, et gestion d'activité), compléter le tableau suivant en précisant si le jeton est obligatoire (O), si le jeton doit être attribué s'il est disponible (A) ou non (NA).

Service	Données	Terminaison	Synchronisation	Gestion d'activité
Transfert alternat				
Terminaison				
Synchro mineure				
Synchro majeure				
Lancement activité				
Reprise activité				

### Application :

- 1) L'ACSE est l'élément de service de contrôle d'association.
  - a. Quelle correspondance existe-t-il entre une association d'application et une connexion de présentation ?
  - b. Quels sont les services de présentations requis par ACSE ?
  - c. Donner l'enchaînement des primitives ACSE et l'échange des APDUs pour les cas suivants :
    - i. établissement avec succès d'une association ;
    - ii. terminaison d'une association ;
    - iii. rupture d'une association.
  - d. A la réception d'une requête A-ASSOCIATE, une APDU est constituée contenant de nombreux paramètres relatifs aux couches application, présentation et session. Justifier la présence des paramètres suivants :
    - i. les propositions de l'utilisateur pour la session ;
    - ii. le numéro de série de point de synchronisation initial ;
    - iii. l'attribution initiale de jetons ;
    - iv. l'identification de connexion de session.
  
- 2) RTSE est l'élément de service de Transfert Fiable.
  - a. Dans un contexte d'application contenant RTSE, une autre AE peut-elle utiliser directement les services de présentations ou ceux de l'ACSE ?
  - b. Quels sont les paramètres spécifiques (couche session) d'une ouverture d'association RTSE ?
  - c. Donner l'enchaînement des primitives RTSE et l'échange des APDUs pour les cas suivants :
    - i. établissement avec succès d'une ouverture de transfert fiable ;
    - ii. refus d'établissement d'une ouverture de transfert fiable par l'appelé ;
    - iii. transfert fiable ;
    - iv. demande de tour et cession de tour ;
    - v. terminaison d'une association
    - vi. rupture d'une association.

Manuel du programmeur Linux

OPENDIR(3)

NOM opendir - Ouvrir un répertoire.

SYNOPSIS  
#include <sys/types.h>  
#include <dirent.h>

DIR \*opendir (const char \*name);

DESCRIPTION

La fonction opendir() ouvre un flux répertoire correspondant au répertoire name, et renvoie un pointeur sur ce flux. Le flux est positionné sur la première entrée du répertoire.

VALEUR RENVOYÉE  
La fonction opendir() renvoie un pointeur sur le flux répertoire ou NULL si une erreur se produit.

ERREURS

EACCESS  
Accès interdit.

EMFILE

Trop de descripteurs de fichiers pour le processus en cours.

ENFILE

Trop de fichiers ouverts simultanément sur le système.

ENOENT

Le répertoire n'existe pas, ou name est une chaîne vide.

ENOMEM

Pas assez de mémoire.

ENOTDIR

name n'est pas un répertoire

CONFORMITÉ

SVID 3, POSIX, BSD 4.3

VOIR AUSSI

open(2), readdir(3), closedir(3), seekdir(3), telldir(3), scandir(3)

TRADUCTION

Christophe Blaess, 1997.

Manuel du programmeur Linux

Linux

Last change: 23 Octobre 1996

1

CLOSEDIR(3)

**NOM** closedir - Fermer un repertoire.

**SYNOPSIS**  
#include <sys/types.h>

#include <dirent.h>

int closedir (DIR \*dir);

**DESCRIPTION**

La fonction closedir() ferme le flux repertoire associe a dir. Apres cette invocation, le descripteur dir du flux repertoire n'est plus disponible.

**VALEUR RENVOYEE**

La fonction closedir() renvoie 0 si elle reussit, ou -1 si elle echoue, auquel cas errno contient le code d'erreur.

**ERREURS**

EBADF

Le descripteur de flux repertoire dir est invalide.

**CONFORMITE**

SVID 3, POSIX, BSD 4.3

**VOIR AUSSI**

close(2), opendir(3), readdir(3), seekdir(3), telldir(3), scandir(3)

**TRADUCTION**

Christophe Blaess, 1997.

Manuel du programmeur Linux

REaddir(3)

NOT readir - Consulter un répertoire.

```
SYNOPSIS
#include <sys/types.h>
```

```
#include <dirent.h>
```

```
struct dirent *readdir (DIR *dir);
```

#### DESCRIPTION

La fonction `readdir()` renvoie un pointeur sur une structure `dirent` représentant l'entrée suivante du flux répertoire pointé par `dir`. Elle renvoie `NULL` à la fin du répertoire, ou en cas d'erreur.

Les données renvoyées par `readdir()` sont écrasées lors de l'appel suivant à `readdir()` sur le même flux répertoire.

D'après POSIX, la structure `dirent` contient un champ `char d_name[]` de taille non spécifiée, avec au plus `NAME_MAX` caractères avant le caractère nul final. L'utilisation des autres champs de cette structure compromet la portabilité de votre programme.

#### VALEUR RENVOYÉE

La fonction `readdir()` renvoie un pointeur sur une structure `dirent`, ou `NULL` lorsqu'une erreur se produit, ou lorsque la fin du répertoire est atteinte.

#### ERREURS

`EBADF` Le flux répertoire `dir` est invalide.

#### CONFORMITÉ

SVID 3, POSIX, BSD 4.3

#### VOIR AUSSI

`read(2)`, `opendir(3)`, `closedir(3)`, `seekdir(3)`, `telldir(3)`, `scandir(3)`

#### TRADUCTION

Christophe Blaess, 1997.

Linux

Last change: 5 Novembre 1996

1

La structure `dirent` est déclarée comme suit :

```
struct dirent
{
    long d_ino;          /* inode number */
    off_t d_off;        /* offset to this dirent */
    unsigned short d_reclen; /* length of this d_name */
    char d_name [NAME_MAX+1]; /* file name (null-terminated) */
}
```

`d_ino` est un numéro d'i-node.

`d_off` est la distance entre le début du répertoire et cette structure `dirent`.

`d_reclen` est la longueur `d_name`, sans compter le caractère nul final.

`d_name` est le nom de fichier terminé par un caractère nul.