Communications sous UNIX - IF4-DIST - ESIEE PARIS

Laurent Perroton - mercredi 12 Avril 2006 - 1:30h - tous documents autorisés

I. RPC : question de cours

Question 1 : Quels sont les caractéristiques des API RPC *haut* et *bas* niveau ? Avantages et inconvénients ? Donner des exemples pour lesquels vous choisiriez plutot l'une que l'autre ?

II. RPC : rls (remote ls) : listage de répertoire distant

On souhaite réaliser un sevice RPC qui prends en entrée un nom de répertoire et qui retourne une liste des noms de fichiers avec leur taille contenus dans ce répertoire. Le contenu du répertoire est donc sur le serveur, d'où le nom du service : *rls*.

Question 1 : La liste des noms d'un répertoire est obtenu avec 3 fonctions : `opendir()` ouvre un répertoire, `readdir()` retourne un par un les noms des fichiers contenus dans ce répertoire, et `closedir()` ferme le répertoire. La fonction `stat()` renvoie une structure contenant des informations sur un fichier, dont sa taille. Lire l'extrait des pages de man en annexe pour comprendre l'utilisation de ces fonctions.

Question 2 : Le service RLS va etre spécifié en RPCL. Expliquer de quels type des données vous allez avoir besoin et donner leur déclarations RPCL.

Question 3 : Donner la spécification RPCL du service RLS (procédure) en utilisant les types de données de la question précédente.

Question 4 : Donner le code de la fonction RPC du serveur qui recoit en entrée le répertoire et construit la liste à retourner. On veillera à ce que la gestion de la mémoire n'entraine pas de fuites (même remarque pour la question suivante).

Question 5 : Donner le code d'un client qui appel le service RPC avec comme nom de répertoire le premier argument passé sur la ligne de commande et affiche le résultat à l'écran.

```
OPENDIR(3) -- 1995-06-11 -- Linux Programmer's Manual

NAME
       opendir - open a directory

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

       DIR *opendir(const char *name);

DESCRIPTION
       The  opendir()  function  opens  a  directory  stream
       corresponding to the directory name, and returns a pointer
       to the directory stream.  The stream is positioned at the
       first entry in the directory.

READDIR(3) -- 1996-04-22 -- Linux Programmer's Manual

NAME
       readdir - read a directory

SYNOPSIS
       #include <sys/types.h>

       #include <dirent.h>

       struct dirent *readdir(DIR *dir);

DESCRIPTION
       The readdir() function returns a pointer to a dirent
       structure representing the next directory entry in the
       directory stream pointed to by dir.  It returns NULL on
       reaching the end-of-file or if an error occurred.

       On Linux, the dirent structure is defined as follows:

         struct dirent {
               ino_t          d_ino;       /* inode number */
               off_t          d_off;       /* offset to the next dirent */
               unsigned short d_reclen;    /* length of this record */
               unsigned char  d_type;      /* type of file */
               char           d_name[256]; /* filename */
         };

CLOSEDIR(3) -- 1995-06-11 -- Linux Programmer's Manual

NAME
       closedir - close a directory

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

       int closedir(DIR *dir);

DESCRIPTION
       The  closedir()  function  closes  the  directory  stream
       associated with dir.  The directory stream descriptor dir is
       not available after this call.

STAT(3) -- 2004-06-23 -- Linux 2.6.7 -- Linux Programmer's Manual

NAME
       stat - get file status

SYNOPSIS
       #include <sys/types.h>
       #include <sys/stat.h>
       #include <unistd.h>

       int stat(const char *path, struct stat *buf);

DESCRIPTION

       stat() stats the file pointed to by path and returns a stat
       structure, which contains the following fields:

        struct stat {
               dev_t     st_dev;     /* ID of device containing file */
               ino_t     st_ino;     /* inode number */
               mode_t    st_mode;    /* protection */
               nlink_t   st_nlink;   /* number of hard links */
               uid_t     st_uid;     /* user ID of owner */
               gid_t     st_gid;     /* group ID of owner */
               dev_t     st_rdev;    /* device ID (if special file) */
               off_t     st_size;    /* total size, in bytes */
               blksize_t st_blksize; /* blocksize for filesystem I/O */
               blkcnt_t  st_blocks;  /* number of blocks allocated */
               time_t    st_atime;   /* time of last access */
               time_t    st_mtime;   /* time of last modification */
               time_t    st_ctime;   /* time of last status change */
        };
```