Notes sur le protocole XDR

Bruno Jacob, LIUM, M1

19 Décembre 2003

L'objet du protocole XDR (*eXternal Data Representation*), défini par SUN, est de définir une représentation standard des objets. Cette représentation est indépendante de l'architecture des machines. Par conséquent, elle convient aux échanges entre machines utilisant des représentations différentes pour les mêmes types de données.

1 Intérêt d'une représentation standard

Lorsque des applications sur des machines différentes doivent s'échanger des informations typées, le passage d'une représentation "réseau" à la représentation interne des types d'objets pose un problème. La réception d'un même type de données sur deux machines différentes peut ne pas donner un résultat unique.

2 Principe général d'utilisation

Un processus voulant transmettre une suite de valeurs procède à une sérialisation (ou un encodage) de ces différentes valeurs.

Chaque type d'objet est associé à sa représentation XDR par le processus "émetteur" qui appelle une fonction spécifique pour chaque type d'objet.

Un flot XDR peut être vu comme à un flot TCP/IP.

A l'autre bout de la chaîne, le processus "récepteur" peut extraire du flot XDR les représentations XDR des objets transmis et peut appliquer des fonctions de conversion pour obtenir la représentation locale.

3 Le type XDR

C'est le type d'un flot. Quand on veut manipuler les fonctions et les types relatifs à XDR, on doit respecter l'ordre suivant pour faire les inclusions des fichiers :

```
#include <rpc/types.h>
#include <rpc/xdr.h>
```

4 Nature d'un flot XDR

Un flot XDR est dédié :
- soit à l'encodage (type `XDR_ENCODE`)
- soit au décodage (type `XDR_DECODE`)
- soit à la libération des zone mémoire des objets alloués dynamiquement (type `XDR_FREE`)

5 Création d'un flot

5.1 Les flots en mémoire

La fonction

```
void  xdrmem_create( XDR  * xdrptr,
                     const caddr_t adr,
                     const lg,
                     const enum xdr_op type_op);
```

permet d'initialiser l'objet `*xdrptr`. La zone d'adresse `adr` et de longueur `lg` est déclarée comme un flot XDR. Le type `type_op` détermine la nature des opérations à faire sur ce flot (`XDR_ENCODE`, `XDR_DECODE`, `XDR_FREE`).
Le protocole RPC utilise les flots en mémoire pour les échanges de messages sous les protocoles UDP/IP. Les messages sont construits en mémoire avant leur transmission sur le réseau par un `sendto` sur une socket.

5.2 Les flots XDR et les fichiers

Les fichiers gérés avec un buffer par la bibliothèque standard, et en particulier `stdin` et `stdout`, peuvent être associés à un flot XDR.

```
void xdrstdio_create( XDR *xdrptr,
                      FILE *fd,
                      const enum xdr_op type_op);
```

6 Destruction d'un flot

La fonction suivante détruit un flot XDR associé :

```
void xdr_destroy(XDR *xdrptr);
```

7 Opérations de sérialisation et de désérialisation

7.1 Principe général

Il consiste à extraire ou à ajouter dans un flot XDR des informations (ou objets) en réalisant un encodage ou un décodage. A chaque type d'objet <type> correspond un décodage/encodage spécifique réalisé par une fonction xdr_<type> qui est soit prédéfinie, soit définie par l'utilisateur.
Toutes les fonctions prédéfinies renvoient la valeur 1 (TRUE) ou 0 (FALSE) selon si l'opération s'est bien déroulée ou pas et les fonctions définies par l'utilisateur doivent renvoyer le même type de résultat.
Dans la bibliothèque XDR, le type bool_t est prédéfini ainsi que les valeurs TRUE et FALSE.

7.2 La fonction xdr_void

Cette fonction permet d'appeler des fonctions sans paramètres.

7.3 Le traitement des types scalaires de base

L'encodage/décodage de chaque type de base <type> est réalisé par la fonction :

```
void xdrstdio_<type>( XDR * xdrptr,
                      <type> * objptr);
```

encodage : interprète la valeur pointée par objptr comme étant du type <type> et l'ajoute au flot XDR.

décodage : extrait du flot XDR un objet de type <type> et affecte la résultat à l'adresse objptr.

La liste des fonctions de décodage/encodage des types scalaires prédéfinis dans XDR est donnée dans le tableau 1 page 4.

7.4 Les énumérations

xdr_enum : fonction de correspondance entre le type "énumérations" du C (jusqu'à maintenant des entiers) et leur représentation externes

Type C	Fonction XDR associée
char	xdr_char
int	xdr_int
unsigned int	xdr_u_int
short int	xdr_short
unsigned short	xdr_u_short
long int	xdr_long
unsigned long	xdr_u_long
float	xdr_float
double	xdr_double

TAB. 1 – Liste de types de base XDR

xdr_bool : cas particulier pour les objets booléens (FALSE⇔0 et TRUE⇔1).

7.5 Les données opaques

Quand un processus client reçoît et renvoie un objet dont il ne connaît pas la structure, on désigne cet objet comme une donnée opaque. En général, on s'en sert pour identifier des objets du serveur ; le serveur fournit des identificateurs d'objets au client qui les utilisera tels quels pour référencer des objets.

```
bool_t xdr_opaque( XDR *xdrptr,
                   caddr_t ptr,
                   const uint_t lg);
```

7.6 Les chaînes de caractères

```
bool_t xdr_string( XDR *xdrptr,
                   char **ptr,
                   const uint_t lgmax);
```

L'encodage : encode la chaîne de caractères pointée par `ptr`. Si la longueur de la chaîne ne dépasse pas `lgmax` alors cette fonction renvoie FALSE (0).

Le décodage : la chaîne de caractères décodée est affectée à une zone mémoire d'adresse `ptr`. La taille de cette zone est de `lgmax` octets.

- Si `ptr` = NULL : la zone est allouée dynamiquement
- Si `ptr` ≠ NULL : la zone est supposée déjà allouée et sa taille suffisamment grande pour contenir le résultat

7.7 Les tableaux d'octets

Par rapport aux chaînes de caractères :
- ils ne se terminent pas par '/0';
- la taille des éléments est une unité de chargement.

```
bool_t xdr_bytes( XDR *xdrptr,
                  char **ptr,
                  uint_t *lgptr,
                  const uint_t lgmax);
```

`lgptr` est un pointeur sur un entier donnant
- à l'encodage : la longueur de la suite à encoder
- au décodage : la taille réservée pour `ptr`

7.8 Les tableaux d'objets de type quelconque

Dans la fonction suivante

```
bool_t xdr_array( XDR *xdrptr,
                  caddr_t *tabptr,
                  uint_t  *tailleptr,
                  const uint_t taillemax,
                  const uint_t tailleelt,
                  const xdrproc_t xdr_type );
```

l'utilisateur précise quelle est la fonction qu'il faut appliquer dans `xdr_type` pour encoder/décoder tous les éléments du tableau `tabptr`.

7.9 Les structures définies par les utilisateurs

Si un utilisateur définit lui-même une structure du type :

```
struct struct_1{
  type_1 champ_1 ;
  type_2 champ_2 ;
    :       :
  type_n champ_n ;}
```

il doit construire la fonction qui encode et décode cette structure :

```
xdr_struct_1( XDR * xdrptr ,
              struct struct_1 * ptr )
{
  return( xdr_type_1(xdr_&ptr->champ_1) &&
          xdr_type_2(xdr_&ptr->champ_2) &&
               :              :
          xdr_type_2(xdr_&ptr->champ_2) );
}
```

7.10 Les unions à discriminant

La bibliothèque XDR permet la prise en compte des structures d'union à condition qu'elle comporte un champ discriminant : comme chaque constituant de l'union donne lieu a une interprétation possible de cette structure, elle doit comporter un champ particulier qui indique l'interprétation à prendre en compte. L'encodage et le décodage d'une telle structure union sont réalisés par la fonction :

```
xdr_union( XDR  *xdrptr,
           enum_t  *discrimptr,
           char  *ptr,
           const struct xdr_discrim *selectptr,
           const  xdrproc_t  (*selectdefaut));
```

Paramètres :

1. xdrptr : pointeur sur le flot XDR
2. discrimptr : pointeur sur le champ discriminant
3. ptr : pointeur sur l'objet
4. selectptr : tableau de selection
5. selectdefaut : fonction par defaut

avec la structure

```
struct xdr_discrim
{
  enum_t value ;
  bool_t (*proc)() ;
};
```

Cette structure permet de faire la correspondance entre une valeur du champ discriminant et la fonction XDR à activer pour décoder/encoder le membre de l'union correspondant. Le nombre d'éléments pouvant être quelconque, la fin de ce tableau est indiqué par une valeur spéciale : `_dontcare_ , NULL`.
Si la valeur du discriminant pointé par `discrimptr` n'est pas dans cette structure alors on appliquera une fonction par défaut pointée par `selectdefaut`.

7.11 la prise en compte des pointeurs

7.11.1 Structure non récursive

La fonction :

```
bool_t xdr_reference( XDR *xdrptr,
                      caddr_t *ptrptr,
                      uint_t  taille,
                      const xdrproc_t xdrfonc);
```

permet d'encoder ou de décoder un pointeur `et` l'objet pointé.
Paramètres :

1. xdrptr : pointeur sur flot XDR
2. ptrptr : pointe sur le pointeur à encoder ou sur zone qui contiendra le pointeur décodé
3. taille : taille de l'objet pointé
4. xdrfonc : fonction XDR du type de l'objet pointé

Cette fonction renvoie une erreur si `(*ptrptr=NULL)` en encodage. Si `(*ptrptr=NULL)` en décodage alors la fonction alloue la zone necessaire.

7.11.2 Structures récursives

Cela revient à prendre en compte l'encodage du pointeur NULL, ce que ne fait pas `xdr_reference`. C'est le rôle de la fonction :

```
bool_t xdr_pointer( XDR *xdrs,
                    char **objpp,
                    uint_t objsize,
                    const xdrproc_t xdrobj);
```

qui possède les mêmes paramètres que `xdr_reference`. Cette fonction permet d'encoder les structures récursives comme les arbres ou les listes chaînées.

Exemple :

```
typedef struct maillon
{
  int val;
  struct maillon *next;
} liste;

bool_t xdr_liste(XDR*pt_xdr, liste*pt_liste)
{
  return (
  xdr_int(pt_xdr,&pt_liste->val) &&
  xdr_pointer (pt_xdr,&pt_liste->next,sizeof(liste),xdr_liste)
  );
}
```

7.12 La gestion mémoire

Certaines fonctions allouent de la mémoire dynamiquement au moyen de `malloc()` lors du décodage (xdr_array, xdr_bytes, xdr_pointer, xdr_reference, xdr_string), il peut être utile de la libérer.

```
void xdr_free ( xdr_proc_t proc,
                char* objet);
```

Paramètres :

1. proc : fonction XDR associee a l'objet a desallouer
2. objet : pointeur sur l'objet a desallouer

`Exemple :`

```
char *string = NULL;
XDR xdr;

xdrstdio_create(&xdr, file, XDR_DECODE);
xdr_string(&xdr, &string,20);
xdr_free(xdr_string,&string);
```