

Sun RPC/XDR

Remote Procedure Call
eXternal Data Representation

Sun RPC

- trójwymiarowa identyfikacja procedur (nr programu, nr wersji, nr procedury)
- protokół RPC oparty na protokołach warstwy transportowej stosu TCP/IP (TCP lub UDP)
- wiązanie dynamiczne (portmap lub rpcbind)
- kanoniczny format reprezentacji danych (XDR)
- opis interfejsu w języku C-podobnym i przetwarzanie przez narzędzie **rpcgen**

Sun RPC/XDR

2

Tworzenie aplikacji (1)

- przygotowanie pliku z opisem interfejsu w języku RPC (języku narzędzia **rpcgen**), zawierającego:
 - ↳ definicją struktur danych dla przekazywanych parametrów ,
 - ↳ specyfikację procedur zdalnych, obejmującą:
 - * numer programu, numer wersji i numer procedury,
 - * typy argumentów
 - * typ zwracanej wartości.

Sun RPC/XDR

3

Tworzenie aplikacji (2)

- wygenerowanie przez **rpcgen** plików do kompilacji w języku C:
 - ↳ pliku nagłówkowego z definicjami stałych,
 - ↳ plików z namiastką klienta i namiastką serwera,
 - ↳ plików zawierających szkielet procedur zdalnych (fragment programu serwera) i szkielet programu klienta dla potrzeb przetestowania procedur zdalnych,
 - ↳ plik do zarządzania kompilacją (`Makefile`),
 - ↳ plik z funkcjami XDR do konwersji zdefiniowanych typów danych.

Sun RPC/XDR

4

Tworzenie aplikacji (3)

- uzupełnienie lub dokonanie niezbędnych modyfikacji w zawartości wygenerowanych plików
 - ↳ uzupełnienie plików zawierających szkielety procedur zdalnych,
 - ↳ uzupełnienie/modyfikacja programu klienta,
 - ↳ uzupełnienie/modyfikacja pliku Makefile.

Sun RPC/XDR

5

Tworzenie aplikacji (4)

- kompilacja programów,
- testowanie działania procedur zdalnych,
 - ↳ ewentualne przygotowanie aplikacji w wersji zwartej (nie rozproszonej) w celu przetestowania działania samych procedur,
- przygotowanie programu klienta korzystającego z procedur zdalnych w sposób wynikający z potrzeb aplikacji.

Sun RPC/XDR

6

Język RPC — specyfikacja procedur zdalnych

```

program NAZWA_PROGRAMU {
  version NAZWA_WERSJI1 {
    typ nazwa_proc1(typ_arg)=nr_proc;
    typ nazwa_proc2(typ_arg)=nr_proc;
    ⋮
  }=numer_wersji;
  version NAZWA_WERSJI2 {
    typ nazwa_proc1(typ_arg)=nr_proc;
    typ nazwa_proc2(typ_arg)=nr_proc;
    ⋮
  }=numer_wersji;
}=numer_programu;

```

Sun RPC/XDR

7

Dostępne zakresy numerów programów

- 0 – 0x1FFFFFFF — numery standardowe, zdefiniowane (przydzielane) przez firmę Sun,
- 0x20000000 – 0x3FFFFFFF — przydzielone przez użytkownika,
- 0x40000000 – 0x5FFFFFFF — zarezerwowane dla aplikacji, które generują numery programów dynamicznie,
- 0x60000000 – 0xFFFFFFFF — zarezerwowane.

Sun RPC/XDR

8

Przykład opisu prostego interfejsu

```

program REMOTE_KILL {
    version DEFAULT_SIGNALUM {
        int rkill(int) = 1;
    } = 1;
} = 0x20000001;

```

Sun RPC/XDR

9

Przykład opisu interfejsu z definicją struktury danych

```

struct rkill_params{
    int pid;
    int signum;
};

program REMOTE_KILL {
    version SPECIFIED_SIGNALUM {
        int rkill(rkill_params) = 1;
    } = 2;
} = 0x20000001;

```

Sun RPC/XDR

10

Przykład opisu interfejsu dla **rpcgen -N**

```

program REMOTE_KILL {
    version SPECIFIED_SIGNALUM {
        int rkill(int, int) = 1;
    } = 2;
} = 0x20000001;

```

Sun RPC/XDR

11

Standard XDR

- Opis standardu — RFC 1014
- Kanoniczna reprezentacja danych oparta na formacie IEEE
- Deklaratywny język opisu struktur danych (zbliżony do języka C)
- Koncepcja konwersji oparta na
 - ↳ potokach — miejscach przechowywania danych w formacie XDR
 - ↳ filtrach — procedurach konwersji pomiędzy własnym formatem maszyny a formatem kanonicznym (w obu kierunkach)

Sun RPC/XDR

12

Potok XDR

- Potok jest miejscem składowania danych XDR, a dokładniej środkiem dostępu do nich w celu:
 - ↳ zapisu — potok kodujący
 - ↳ odczytu — potok dekodujący
- Rodzaje potoków:
 - ↳ potok na standardowym wejściu-wyjściu — na otwartym pliku
 - ↳ potok w pamięci — w obszarze pamięci opisanym przez adres i rozmiar w bajtach
 - ↳ potok komunikatów (rekordów) — na strumieniu danych, zdefiniowanym wraz z procedurami obsługi tego strumienia (zapis, odczyt)

Sun RPC/XDR

13

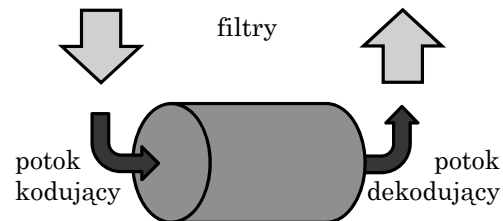
Filtr XDR

- Filtr jest procedurą konwersji, służącą do realizacji dostępu do danych w potoku w celu zapisu (kodująca) lub odczytu (dekodująca) — zależnie od kierunku działania potoku.
- Rodzaje filtrów
 - ↳ filtr prosty — służy do konwersji typów prostych
 - ↳ filtr złożony — służy do konwersji typów złożonych (np. tablicowych, wskaźnikowych)
 - ↳ filtr pochodny — połączenie innych filtrów w ramach jednej procedury filtrującej

Sun RPC/XDR

14

Potoki i filtry XDR



Sun RPC/XDR

15

Filtry proste

- Filtr dla:
 - ↳ typów prostych: `bool_t`, `char`, `short`, `int`, `long`, (również bez znaku) `float`, `double`
 - ↳ typu wyliczeniowego (`enum`)
 - ↳ typu `void`
- Ogólna postać procedury filtrującej (na przykładzie typu `unsigned int`):

```
bool_t
xdr_u_int(XDR* xdrs,
          unsigned int *ptr);
```

Sun RPC/XDR

16

Filtry złożone (1)

- Filtr do konwersji
 - ↳ tablic bajtów o ustalonej lub zmiennej wielkości — `xdr_opaque`, `xdr_bytes`
 - ↳ tablic elementów określonego typu o ustalonej lub zmiennej wielkości — `xdr_vector`, `xdr_array`
 - ↳ łańcuchów znaków — `xdr_string`, `xdr_wrapstring`
 - ↳ typów wskaźnikowych — `xdr_reference`, `xdr_pointer`
 - ↳ unii — `xdr_union` (w praktyce dla statycznie zdefiniowanych typów unijnych stosowany jest filtr pochodny)

Sun RPC/XDR

17

Filtry złożone (2)

- Ogólna postać filtru złożonego:


```
xdr_typ(XDR* xdrs, ptr, ...,
        [xdrproc_t elproc]);
```
- W przypadku typu statycznego (ustalona zajętość pamięci) wskaźnik *ptr*, ma postać `char *ptr`, w przypadku typu o zmiennej wielkości (konieczność dynamicznej alokacji pamięci) `char **ptr`.
- Jeśli typ podstawowy (w przypadku tablic, wskaźników) nie jest z góry określony, konieczne jest wskazanie filtru XDR do konwersji elementu typu podstawowego.

Sun RPC/XDR

18

Filtry złożone do konwersji tablic

```
xdr_array(xdrs, arrp, sizep, maxsize, elsize, elproc)
XDR *xdrs;
char **arrp;
u_int *sizep, maxsize, elsize;
xdrproc_t elproc;
xdr_vector(xdrs, arrp, size, elsize, elproc)
char *arrp;
u_int size, elsize; // pozostałe parametry - j.w.
xdr_bytes(xdrs, sp, sizep, maxsize)
char **sp; // pozostałe parametry - j.w.
xdr_opaque(xdrs, cp, cnt)
char *cp;
u_int cnt;
```

Sun RPC/XDR

19

Filtry złożone do konwersji łańcuchów znaków

```
xdr_string(xdrs, sp, maxsize)
XDR *xdrs;
char **sp;
u_int maxsize;
xdr_wrapstring(xdrs, sp)
≡ xdr_string(xdrs, sp, MAXUN.UNSIGNED );

xdr_wrapstring nie wymaga podania rozmiaru (liczby znaków
łańcucha)
```

Sun RPC/XDR

20

Filtry złożone do konwersji struktur wskaźnikowych

```
xdr_reference(xdrs, pp, size, proc)
XDR *xdrs;
char **pp;
u_int size;
xdrproc_t proc;

xdr_pointer(xdrs, objpp, objsize, xdrobj)
XDR *xdrs;
char **objpp;
u_int objsize;
xdrproc_t xdrobj;
```

`xdr_pointer` w przeciwieństwie do `xdr_reference` interpretuje wskaźnik pusty, co umożliwia obsługę rekurencyjnych struktur danych

Sun RPC/XDR

21

Przykład filtru pochodnego

Definicja struktury

```
struct struktura {
    int x;
    long y;
    char c;
    short s;
};
```

Filtr XDR

```
bool_t
xdr_struktura (XDR *xdrs,
               struktura *objp) {
    if (!xdr_int (xdrs,
                 &objp->x)) return FALSE;
    if (!xdr_long (xdrs,
                  &objp->y)) return FALSE;
    if (!xdr_char (xdrs,
                  &objp->c)) return FALSE;
    if (!xdr_short (xdrs,
                   &objp->s)) return FALSE;
    return TRUE;
}
```

Sun RPC/XDR

22

Zarządzanie pamięcią

- Przekazanie pustego wskaźnika typu `char**` przy konwersji dekodującej spowoduje dynamiczną alokację pamięci przez filtr XDR.
- Zwolnienie obszaru dynamicznie zaalokowanej pamięci przez filtr XDR musi nastąpić w programie aplikacyjnym.
- Ogólna postać funkcji zwalniania pamięci:


```
xdr_free(xdrproc_t proc,
         char* objp);
```

Sun RPC/XDR

23

Tworzenie potoków XDR

- Potok na standardowym wejściu-wyjściu:


```
void xdrstdio_create(XDR *xdrs,
                    FILE *file, enum xdr_op op);
```
- Potok w pamięci:


```
void xdrmem_create(XDR *xdrs, char *addr,
                  u_int size, enum xdr_op op);
```
- Potok komunikatów:


```
void xdrrec_create(XDR *xdrs,
                  u_int sendsize, u_int recvsize,
                  char *handle,
                  int (*readit) (char*, char*, int),
                  int (*writeit) (char*, char*, int));
```

Sun RPC/XDR

24

Potoki komunikatów (1)

- Ustalenie kierunku potoku musi nastąpić po jego utworzeniu (np. `xdrs -> x_op = XDR_ENCODE`)
- Rozmiary buforów (parametry *sendsize* i *recvsize*) mogą mieć wartość 0, co oznacza przyjęcie wartości domyślnych.
- Gdy konieczne jest opróżnienie bufora wyjściowego (wysłanie danych) lub zapełnienie bufora wejściowego, wywołwana jest odpowiednia funkcja (*writeit*, *readit*) z trzema parametrami:
 - ↳ uchwytem *handle*
 - ↳ adresem bufora
 - ↳ liczbą bajtów do zapisu/odczytu

Sun RPC/XDR

25

Potoki komunikatów (2)

- Oznaczenie końca rekordu (przy zapisie)


```
xdrrec_endofrecord(XDR *xdrs,
                    int sendnow);
```
- Pominięcie reszty rekordu (przy odczycie)


```
xdrrec_skiprecord(XDR *xdrs);
```
- Sprawdzenie zakończenia strumienia (przy odczycie)


```
xdrrec_eof(XDR *xdrs);
```

Sun RPC/XDR

26

Definicja struktur danych (1)

XDR (rpcgen)

- stałe


```
const MAX = 512;
```
- typy wyliczeniowe


```
enum COLOR {
    red   = 1,
    green = 2,
    blue  = 4
};
```

język C

```
#define MAX 512

enum COLOR {
    red   = 1,
    green = 2,
    blue  = 4
};

typedef enum COLOR
COLOR;
```

Sun RPC/XDR

27

Definicja struktur danych (2)

XDR (rpcgen)

- struktury

```
struct ST {
    int a;
    int b;
};
```

język C

```
struct ST {
    int a;
    int b;
};

typedef struct ST ST;
```

Sun RPC/XDR

28

Definicja struktur danych (3)

XDR (rpcgen)

- unie


```
union UN switch (int d)
{
  case 1: int a;
  case 2: char b;
  default: short c;
};
```

język C

```
struct UN {
  int d;
  union {
    int a;
    char b;
    short c;
  } UN_u;
};
typedef struct UN UN;
```

Sun RPC/XDR

29

Definicja struktur danych (4)

XDR (rpcgen)

- tablice


```
typedef int tabf[10];
typedef int tabv<10>;
```
- podobnie tablice bajtów


```
typedef opaque btf[10];
typedef opaque btv<10>;
```
- łańcuchy znaków


```
typedef string s10<10>;
typedef string sbo<>;
```

język C

```
typedef int tabf[10];
typedef struct {
  u_int tabv_len;
  int *tabv_val;
} tabv;
• typem bazowym w jest wówczas char
```

Sun RPC/XDR

30