分散ソフトウェア開発用ツールキット—ライブラリ—*

6D－6

鈴木寿郎† 中澤修‡ 佐藤泰典‡

(株)沖テクノシステムズラボラトリ† (株)沖電気工業‡

1 はじめに

分散ソフトウェアの開発では，多くの場合，ネットワークにわたるプロセス間通信を実現するために，比較的低水準ないし柔軟性を欠くアプリケーション・インタフェースしか利用できないことが問題である。

著者らはこれに対し，Unix 上の C++ を使ったプロセス間通信のための，抽象化されたインタフェースを提供する DOL (Distributed Objective Language) /C++ライブラリを試作した。

本ライブラリの開発では，柔軟かつ簡単で直感的な外部仕様を目指した。この目的のため，演算子の多重定義など C++ 言語の syntax sugar 用コンストラクトを活用し，また出来る限り資源の管理を自動化することを試みた。本ライブラリは実装のために遠隔手続き呼び出し(RPC)を利用しているが，その詳細はプログラマから隠されている。

本ライブラリを利用するプログラムではただ，通信者型 d_Corr ないし伝達者型 d_Messenger の変数を宣言し，それを送信者ないし受信者とするだけで，プロセス間でメッセージを送受できる。

以下ではまずメッセージ型 d_Message の本文を構成する汎用データ型 d_Data について述べる。次に通信者型と伝達者型，及び通信の方法について述べる。最後に，その応用であるプロセス型 d_Process について簡単に述べる。

2 汎用データ型 d_Data

d_Data は通信メッセージの本文を構成する自己記述的な汎用のデータ型(クラス)である。その実体は型タグとデータ値とからなり，整数や浮動小数点数のスカラやベクタ(1次元配列)を表現できる。d_Data には次の特徴がある。

1. 構築子とキャスト演算子による型変換
2. d_Data 自身のベクタを表現可能
3. 型タグ内の'浅い複写ビット'

数や文字列などをメッセージの本文とする場合，1によりプログラマは d_Data をほとんど意識せずに，透過的にそれを取り扱うことができる。

また 2 により種々の型を比較的容易に模倣し得る。

C++ では動的オブジェクトの解放に細心の注意が必要である。効率のため d_Data に随伴する複写演算では通常，ベクタをポインタだけ複写する浅い複写を行う(d_Data 自身のベクタは例外である)。しかしメッセージの受渡しなど，ベクタの実体の複製を動的に作成する深い複写が便利な場合もある。そのため，深い複写で値を設定する構築子 d_Data(const d_Data&, d_DeepCopyType) と，深い複写で値を代入するメンバ関数 d_Data& becomes(const d_Data&, d_DeepCopyType) も用意した(ここで d_DeepCopyType は d_DeepCopy を唯一の値域とする型である)。そして，実体まで解放すべき深い複写値の解放と，それが不必要な浅い複写値の解放の区別は，プログラマでなく，3 によって d_Data が自動的に行う。これは特に 2 を使って木構造を作る場合に重要である。

なお，機械間のエンディアン等の違いは，RPC が用いる XDR [2] の符号化に吸収されている。[1]

3 通信者型 d_Corr

通信者型 d_Corr (correspondent) は送受信の端点を表す。その実体は識別番号である。ライブラリはこれを内部的に参照カウンタで管理する。

通信者には，値の初期設定時に，任意の文字列で名前を付けることもできる。名前は系内で一意的である。重複した場合は無効値に初期設定される。

*A Toolkit for Development of Distributed Softwares—the Library— † Hisao Suzuki (OKI Technosystems Laboratory), ‡ Osamu Nakazawa and Yasunori Satō (OKI Electric)

[1] 実際，エンディアンの異なる2種類の CPU である i860 と SPARC の間での本ライブラリによる通信に成功している。

系内の通信者の管理と通信等には主従2層の通信サーバ・プロセスが働く。従サーバ POB (post office box) は各ホストの各ユーザごとに，その各プロセスの所有する通信者を管轄し，その通信者に宛てられた受信メッセージを預る私書箱として働く。主サーバ PM (postmaster) は系に1個だけあり，すべての POB を管理し，通信者の名前の一意性を保証する。

本ライブラリを使用した各プロセスは，POB の RPC クライアントになる。各プロセスは，通信者宛のメッセージを，宛先の通信者の POB へ RPC で送信する。宛先の通信者を所有するプロセスが着信待ちに入っている場合，POB は，それにシグナル (無指定時 SIGCONT) を送ってメッセージの着信を通知する。通知されたプロセスは，その POB から着信メッセージを RPC で受け取る。このシグナル通知を活用して，メッセージの着信とその他の入力とを同時に待つことも可能である。

他プロセスの所有する通信者を参照する方法には次の四つがある。3, 4 を伝達者と併せて用いれば通信相手のさまざまな選択方法を実現できる。

1. d_Corr(d_Remote, "c1") のように名前で参照する
2. プロセス fork 後，親又は子の通信者を参照する
3. 受信したメッセージの送信者欄を参照する
4. d_Corr 値を d_Data 値に変換して送受する

4 伝達者型 d_Messenger

1対1通信でなく広報通信を行うには，通信者でなく伝達者型 d_Messenger の値を宛先に使う。

伝達者を名前とともに宣言すると，そのプロセスでその伝達者を担当する名無し通信者が，暗黙のうちに用意される。伝達者宛のメッセージを送信すると，その伝達者の担当通信者を管轄している POB すべてに PM 経由でメッセージが伝達され，そこからその通信者を所有する各プロセスにメッセージが渡される。

下記は，伝達者を利用した簡単な多人数 (随時参加脱退可能) 会話プログラムである。これを実行中の端末から1行入力すると，同じくこれを実行中の他の各端末に，その1行を本文とするメッセージが表示される。

```
#include <stdio.h>
#include "d_Corr.ch"

char s[300], *r;

// 鍵盤入力…実行終了時 (r==s||r==NULL) が成立
void foo() {r = gets(s);}

int main()
{
  d_Message m;
  d_Messenger a("m1");      // 伝達者を宣言
  for (;;) {
    r = s + 1;
    // 鍵盤入力 s とメッセージ m を同時に待つ
    if (a.fetches(m, foo)) {
      printf("%s-%d:%X:<%s>\n",     // m を表示
        m.sender().host(), // 送信者のホスト名
        m.sender().uid(),  // 送信者のユーザ ID
        m.issueNumber(),   // 送信者が付けた発行番号
        (char *)m.text()); // 本文 (文字列に型変換)
    }
    if (r == NULL) break;
    if (r == s && s[0] != '\0') {
      a.carries(s);  // s を広報伝達…ここで送信者には
    }                // a を担当する通信者が使われる
  }
  return 0;
}
```

5 プロセス型 d_Process

d_Corr と d_Messenger は基本的な通信機能を提供するが，プロセス型 d_Process は更に遠隔メンバ関数の実現のための骨組みを提供する。この骨組みは派生クラスでの具体化を前提にしている。

d_Process は d_Corr を用いて実現されている。プロセス fork 時には子プロセスにひとつだけ通信者を譲渡できるが，d_Process は，これを，遠隔メンバ関数を実行するプロセスの生成に利用している。ただし，親からの管理のため，孫プロセスへの再譲渡は (通信者の所有主の判定によって) 禁止している。

6 おわりに

プログラムでの通信データや通信端点の取り扱いの容易さと柔軟性が，分散ソフトウェアの開発に対する本ライブラリの主な利点である。これに比べ，例えば ROME [1] は高水準ではあるが，プログラムの全体的な構成を強く固定しているように思われる。

今後の課題としては，より実行効率を重視した実装の実現などが挙げられる。

参考文献

[1] Scott Burleigh: "ROME: Distributing C++ Object Systems", IEEE Parallel & Distributed Technology, May 1993, pp.21-32.

[2] UNIX Software Operation: "UNIX SYSTEM V RELEASE 4 Programmer's Guide: Networking Interfaces", Prentice-Hall, 1990.