ШИНА МЕЖМАШИННОГО ВЗАИМОДЕЙСТВИЯ БЛЭКБОКС-ПРОЦЕССОВ ПОСРЕДСТВОМ СЕТЕВОЙ ФАЙЛОВОЙ СИСТЕМЫ

Е. Э. Темиргалеев

В статье приведено решение задачи организации не интенсивного обмена сообщениями между процессами Блэкбокс [1] на машинах локальной сети посредством сетевой файловой системы.

Общие результаты. Разработанная модель реализации шины сообщений предполагает использование общего сетевого каталога (*каталога шины*) для хранения передаваемых сообщений и списка подключённых к шине участников обмена. Каждое сообщение сохраняется в отдельный файл — отправитель пишет его, получатель читает и удаляет. Широковещание осуществляется записью отправителем файла сообщения для каждого получателя.

Свойства модели: *асинхронность* (процессы-участники пишут, читают и удаляют файлы сообщений независимо друг от друга) и *отсутствие гарантий* (факт отправки сообщения (запись файла успешна) не означает факт его получения и обработки, что должно учитываться при программировании взаимодействия участников; отсюда же вытекает и нетребовательность выдерживать упорядоченность записанных конкретным отправителем сообщений при их чтении получателем).

С теоретической точки зрения полученные результаты суть изобретение велосипеда и не представляют ценности, поскольку данный вопрос уже давно изучен в общем виде и изложен во многих работах, например, раздел «2.4. Связь посредством сообщений» (Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. — СПб.: Питер, 2003. — 877 с.: ил. — (Серия «Классика computer science»))[1].

Практический результат решения основной задачи тоже не близок к идеалу. Получить простую реализацию, зиждущуюся исключительно на базовой для системы Блэкбокс абстракции *Files*, не удалось. Выбранное же ради этого транспортное средство в виде сетевых ФС с присущей им латентностью является основным фактором, отрицательно влияющим на скорость и стабильность обмена данными; для улучшения этих характеристик (расширения сферы применения) нужно использовать более адекватные задаче средства передачи, например, сетевые соединения по протоколу TCP/IP[2].

Но и «отрицательный результат — тоже результат». В дополнение к тому, что замена средства передачи данных в основном решении суть чисто технический вопрос, принесли свои плоды и решения сопутствующих задач. В частности, найдена довольно простая схема передачи объектов хранения между процессами Блэкбокс.

---

[1] замечание Рюмшина Б. В.

[2] замечания Рюмшина Б. В.

Объект хранения для представления сообщений. Изначально в качестве более простого рассматривался вариант "ANYREC с полями элементарных типов" и прямой записью в файл [2]. Детальное сравнение плюсов и минусов выявило преимущество *Stores.Store*:

- Применение *Stores.Store* упрощает реализацию компонента и его настройку. (Помимо записи/чтения [2] требуется механизм воссоздания в памяти процесса-получателя "пустой" записи точно того же типа, что сохранил процесс-отправитель. Эта задача не может быть решена без применения низкоуровневых средств с одной стороны, с другой — её решение заложено в *Stores*.) Исключение необходимости применять низкоуровневые средства убирает потребность в специальном интерфейсе разъёма их реализации вне основного компонента, его реализации под нужные платформы, и соответствующей настройке компонента.

- Применение *Stores.Store* расширяет применимость компонента: в качестве объекта хранения могут быть переданы данные любого вида; передаваемые данные платформенно-независимы (см. *Kernel.littleEndian* в [2]).

- Разница в "скорости" "между" ANYREC и *Stores.Store* не существенна. Учитывая время сохранения данных непосредственно в файл, разница между отправкой данных записи с полями элементарных типов одним вызовом *wr(Files.Writer).WriteBytes(...)* или несколькими вызовами *wr(Stores.Writer).Write...(...)* не существенна. (Для чтения аналогично.)

- Проблема рутины — реализации методов *Externalize* и *Internalize* для каждого записевого типа, представляющего сообщение и расширяющего *Stores.Store*, с учётом обычной для этого случая экспортированности всех полей записи, снимается применением библиотечного компонента, автоматизирующего выгрузку экспортированных полей. Соображения относительно "скорости" в этом случае аналогичны изложенным в предыдущем пункте. Пример такого компонента и его применения:

```
DEFINITION PrivRW;
  IMPORT Stores, Meta;
  PROCEDURE ReadItem (VAR rd: Stores.Reader; IN item: Meta.Item);
  PROCEDURE ReadRecord (VAR rd: Stores.Reader; rec: ANYPTR);
  PROCEDURE WriteItem (VAR wr: Stores.Writer; IN item: Meta.Item);
  PROCEDURE WriteRecord (VAR wr: Stores.Writer; rec: ANYPTR);
END PrivRW.
...
  IMPORT Stores, PrivRW;
  TYPE Message* = POINTER TO ABSTRACT RECORD (Stores.Store) END;
  TYPE XMsg* = POINTER TO RECORD (Message) x*, y*: INTEGER END;
```

```
...
PROCEDURE (m: Message) Internalize2- (VAR rd: Stores.Reader), NEW, EMPTY;
PROCEDURE (m: Message) Externalize2- (VAR wr: Stores.Writer), NEW, EMPTY;

PROCEDURE (m: Message) Internalize- (VAR rd: Stores.Reader);
BEGIN
  PrivRW.ReadRecord(rd, m); m.Internalize2(rd)
END Internalize;

PROCEDURE (m: Message) Externalize- (VAR wr: Stores.Writer);
BEGIN
  PrivRW.WriteRecord(wr, m); m.Externalize2(wr)
END Externalize;
```

Схема передачи объектов хранения между процессами Блэкбокс. Требуется прочитать *Stores.Store* из файла, сразу после чего закрыть (и удалить) его. "Стандартная" модель работы Блэкбокс с файлами (*Files.File*) не предполагает их явного закрытия, поэтому объекты хранения (например, *TextModels.Model*) при чтении могут оставлять ссылки на файл, ожидая что (при последующем обращении по ссылке к содержимому) он будет открыт.

Схема приемлемого решения: содержимое исходных файлов приписывать к временному и читать объекты хранения из него; по достижении им определённого объёма — открывать новый временный файл, делая старый кандидатом для сбора мусора.

```
  VAR receiveLocalFile: Files.File; rlfWr: Files.Writer; mfRd: Files.Reader;
...
  PROCEDURE AppendToRLF (msgFile: Files.File; OUT localFile: Files.File;
OUT beg, end: INTEGER);
    CONST maxLen = 4*1024*1024;
    VAR len, rem, rwlen: INTEGER; buf: ARRAY 64*1024 OF BYTE;
  BEGIN
    IF receiveLocalFile = NIL THEN
      receiveLocalFile := Files.dir.Temp(); rlfWr := receiveLocalFile.NewWriter(NIL)
    END;
    beg := rlfWr.Pos(); len := msgFile.Length();
    mfRd := msgFile.NewReader(mfRd); mfRd.SetPos(0);
    rem := len;
    WHILE rem > 0 DO
      rwlen := MIN(LEN(buf), rem);
      mfRd.ReadBytes(buf, 0, rwlen); rlfWr.WriteBytes(buf, 0, rwlen);
      DEC(rem, rwlen)
    END;
    end := rlfWr.Pos();
    localFile := receiveLocalFile;
    IF end >= maxLen THEN receiveLocalFile := NIL; rlfWr := NIL END;
    ASSERT(end - beg = len, 60)
  END AppendToRLF;

  VAR msgFile: Files.File; s: Stores.Store; rd: Stores.Reader;
```

```
  beg, end: INTEGER; localFile: Files.File; res: INTEGER;
...
msgFile := Files.dir.Old(c.loc, name, Files.exclusive); ASSERT(msgFile # NIL, 100);
AppendToRLF(msgFile, localFile, beg, end);
rd.ConnectTo(localFile); rd.SetPos(beg);
rd.ReadStore(s); rd.ConnectTo(NIL);
msgFile.Close; Files.dir.Delete(c.loc, name);
```

Схема реализации. Модель (с каталогом шины работают только процессы-участники взаимодействия):
(1) К шине подключены участники с уникальными именами. Имя участника включается в имя файла сообщения, им отправляемого.
(2) Процесс-участник обозначает свое подключение к шине обеспечением наличия в каталоге шины одноимённого подкаталога.
(3) Отключение от шины осуществляется "сокрытием" каталога — добавлением точки к его имени; имена участников не могут начинаться с точки.
(4) Подключение требует удаления скрытого подкаталога участника, чтобы гарантировать возможность переименования при отключении. Если удаление не возможно, то не возможно и подключение.
(5) Запись файла сообщения для отправки осуществляется созданием пустого файла в каталоге получателя, который после открывается *Files.dir.Old(loc, . . .)* для записи. Неудачное открытие означает отмену посылки.
(6) Три класса файлов сообщений: пишущееся сообщение, записанное и ожидающее получения, получаемое. Принадлежность к классу указывается в имени файла.
(7) Уникальность имени файлов сообщений одного отправителя обеспечивается добавкой случайной цепочки.

Недостающие в абстракции *Files* операции:
(O.1) Создать каталог (2).
(O.2) Переименовать каталог (3).
(O.3) Удалить (пустой) каталог (4).
(O.4) Создать (переписать) пустой файл в данном (существующем) каталоге (5).
(O.5) Получить "чистое" имя[3] (анализ имени при получении сообщений).

Предполагается эксплуатация "полноценных" сетевых ФС, для которых реализация *Files* будет работать без ограничений (сбоев) как поверх "дисковой" ФС, и возможа реализация дополнительных операций (O.2-4).

Библиотечный компонент *PrivFB*. Предоставляет реализацию операций для работы с шиной (подключение, отключение, отправка и широковещание, получение), обеспечивая передачу сообщений в виде объектов хра-

[3]Заметка «Абстракция *Files* системы Блэкбокс: анализ "чистого" имени файла в *Files.FileInfo*» в этом сборнике.

нения (*Stores.Store*). Операции выполняются для именованного соединения с шиной (*Connection*); один процесс может иметь несколько соединений.

```
DEFINITION PrivFB;
  IMPORT Files, Stores;

  TYPE
    Hook = POINTER TO ABSTRACT RECORD ... END; (* разъём реализации O.2-5 *)

    Connection = EXTENSIBLE RECORD
      bus-: Files.Locator; name-: Files.Name; loc-: Files.Locator;
      Handler: MsgHandler; res, locres: INTEGER
    END;

    Msg = RECORD
      ingoing: BOOLEAN; connection: Files.Name; msg: Stores.Store
    END;

    MsgHandler = PROCEDURE (VAR c: Connection; VAR msg: ANYREC);

  PROCEDURE Broadcast (VAR c: Connection; msg: Stores.Store);
  PROCEDURE Configure (VAR c: Connection; bus: Files.Locator; IN name: Files.Name);
  PROCEDURE Connect (VAR c: Connection);
  PROCEDURE Disconnect (VAR c: Connection);
  PROCEDURE Receive (VAR c: Connection);
  PROCEDURE Send (VAR c: Connection; IN receiver: Files.Name; msg: Stores.Store);
  PROCEDURE SetHook (h: Hook);

END PrivFB.
```

Имя файла сообщения:

```
ИмяФайлаСообщения = ИмяОтправителя СлучайныйСуффикс Класс.
СлучайныйСуффикс = hex hex hex hex hex hex hex hex.
hex = "0"..."9" | "A" ... "F".
Класс = "r" | "d" | "w".
```

Свойства и состояния соединения c:
- имя ($c.name$) — не может быть пустым (согласно интерфейса $Files$) и содержать в начале точку (3);
- локатор каталога шины ($c.bus$);
- обработчик сообщений ($c.Handler$);
- локатор каталога соединения ($c.loc$) (2);
- "подключено" = ($c.loc \neq NIL$) и "отключено" = ($c.loc = NIL$);
- коды результатов выполнения операции шины ($c.res$) и файловых ($c.locres$).

Операции для отключенного соединения c:

- Настроить ($Configure(c, bus, name)$).
- Подключить ($Connect(c)$); $a = c.name$:
  (C.2) Удалить каталог "$.a$" (4).
  (C.3) Создать каталог "a" (2).

Операции для подключённого соединения c:

- Отправить сообщение ($Send(c, receiver, msg)$); $a = c.name$, $b = receiver$:
  (S.1) Сформировать имя файла сообщения, уникальное для имеющихся в "b" файлов "a*" — генерируем случайное 32-разрядное число "XXXXXXXX", не являющееся суффиксом имён этих файлов.
  (S.2) Создать пустой файл "aXXXXXXXXw" в каталоге "b". Не выполнено: считаем, что b отключился.
  (S.3) Открыть "aXXXXXXXXw" в "b" Не выполнено: считаем, что b отключился. Выполнено: получен файловый дескриптор, позволяющий закончить запись, даже если b в момент записи отключается.
  (S.4) Записать сообщение (сохранить msg) и закрыть файл.
  (S.5) Переименовать "aXXXXXXXXw" в "aXXXXXXXXd". Не выполнено: считаем, что b отключился. Выполнено: сообщение отправлено.
  (S.*) Выставить коды $c.res$, $c.locres$ и вызвать $c.Handler(c, Msg(FALSE, b, msg))$.

- Широковещать ($Broadcast(c, msg)$).

- Получить сообщения ($Receive(c)$); $b = c.name$, a — имя соединения (из имени файла):
  (R.1) Получить список файлов каталога "b".
  (R.2) Для каждого файла сообщения вида "aXXXXXXXXd":
  (R.2.1) Переименовать файл "aXXXXXXXXd" в "aXXXXXXXXr".
  (R.2.2) Записать содержимое файла "aXXXXXXXXr" в локальный файл f.
  (R.2.3) Прочитать из f сообщение (загрузить msg).
  (R.2.4) Закрыть и удалить файл "aXXXXXXXXr".
  (R.2.5) Вызвать $c.Handler(c, Msg(TRUE, a, msg)$.

- Отключить ($Disconnect(c)$); $a = c.name$:
  (D.1) Переименовать каталог "a" в ".a" (3).

Список литературы

[1] BlackBox Component Builder 1.6-rc6 (Win)//
URL: `http://oberon.ch/zip/SetupBlackBox16-rc6.exe`

[2] Ермаков И. Е. Исходный текст процедур *RecToFile* и *RecFromFile* //
URL: `http://forum.oberoncore.ru/viewtopic.php?p=37413#p37413`