АБСТРАКЦИЯ *Files* СИСТЕМЫ БЛЭКБОКС: АНАЛИЗ "ЧИСТОГО" ИМЕНИ ФАЙЛА В *Files.FileInfo*

Е. Э. Темиргалеев

В заметке приведено описание проблемы, могущей возникнуть при поименной обработке списка файлов "каталога" *list(Files.FileInfo) := Files.dir.FileList(loc(Files.Locator))*, и предложен вариант её решения.

Интерфейс *Files* базируется на постулате типизированности файлов, означающем наличие у каждого файла атрибута *type*, представляемого цепочкой литер *Files.Type*: «Files are typed. This means that each file has a *type* attribute which is a string...» [1, *Files* documentation]

```
DEFINITION Files;
  ...
  TYPE
    Directory = POINTER TO ABSTRACT RECORD
      ...
      (d: Directory) FileList (loc: Locator): FileInfo, NEW, ABSTRACT;
      (d: Directory) GetFileName (name: Name; type: Type;
        OUT filename: Name), NEW, ABSTRACT;
      ...
    END;
    ...
    FileInfo = POINTER TO RECORD
      ... name: Name; ... type: Type; ...
    END;
  ...
  Name = ARRAY 256 OF CHAR;
  Type = ARRAY 16 OF CHAR;
  ...
END Files. [1]
```

Отмечается, что одни ФС непосредственно хранят этот атрибут, в других же он эмулируется включением (*) в имя файла[1]: «On some platforms, the host file system knows about file types (Mac OS), while on others file types are simulated by using file suffixes as extensions (Windows).» [1, *Files* documentation]

Для покрытия обоих подходов единым интерфейсом введена операция *Files.dir.GetFileName*:

«PROCEDURE (d: Directory) **GetFileName** (name: Name; type: Type; OUT filename: Name) NEW, ABSTRACT

Make a file name out of a file and its type.

Windows: filename = name + "." + type

Mac OS: filename = name» [1, *Files* documentation]

[1]Формулировка удалена от точного перевода в контексте рассмотрения *Files* исключительно как абстракции над произвольной ФС: «На некоторых платформах файловая система отдельно помнит типы файлов (Mac OS), на других типы файлов эмулируются суффиксами имен файлов — расширениями (Windows).» [2]

Посредством оной необходимо получать имена файлов для операций, не зависящих от типа файла, и требующих только его имя:

```
VAR f: Files.File; name: Files.Name;
...
Files.dir.GetName("name", "type", name); f := Files.dir.Old(..., name, ...)
```

Возможная для открытия в Windows запись *f := Files.dir.Old(. . . , "name.type", . . .)* — пример некорректного применения интерфейса *Files*, завязывающего код на особенности платформы.

Проблема возникает при необходимости проанализировать "чистое" имя файла («name») в описателе файла *fi(Files.FileInfo)* для случая (*). Конкретно, реализация *HostFiles* [1] помещает в *fi.name* имя, полученное напрямую из ФС — «filename», которое для компонента Блэкбокс суть результат *Files.dir.GetFileName*, и выделяет из него тип *fi.type*. "Чистое" имя («name») можно получить в данном случае как результат применения обратной к *Files.dir.GetFileName* операции: $(filename, type) \rightarrow name$, которая в интерфейсе *Files* отсутствует. Программирование же этой операции напрямую, требующее знания правил преобразования, нарушает абстрагированность от нижележащей ФС (**), которую должен обеспечивать *Files*.

Если реализацию *Files* для случая (*) выполнять в строгом соответствии с интерфейсом, помещая в *fi.name* "чистое" имя, придётся и строго придерживаться модели типизированности файлов, что может оказаться непрактичным, когда Блэкбокс применяется не как обособленная система, но как система поверх другой системы. Например, для [1] обработка привычного для "всех" случая — "одноимённых" файлов в одном каталоге, имеющих разные «filename» за счёт отличающегося «типа» («расширения»),

```
ivanov.pdf  ivanov.tex  ivanov.png  ...
```

либо становится не возможной (система должна будет трактовать одинаковые «name» в одном каталоге как не допустимую ситуацию), либо потребует от клиента самостоятельного формирования «filename» (выходим на исходную позицию (**)).

Один из вариантов разрешения проблемы (зависимости от платформы компонентов, использующих *Files*) — применение обособленных, дополнительных к интерфейсу *Files* операций *GetFileName* и обратной к ней *GetName* вкупе с программированием их под нужные платформы. Например:

```
PROCEDURE GetFileName* (IN name: Files.Name; IN type: Files.Type;
  OUT filename: Files.Name);
BEGIN
  ASSERT(hook # NIL, 100); ASSERT(name # "", 20);
  hook.GetFileName(name, type, filename)
END GetFileName;
```

```
PROCEDURE GetName* (IN filename: Files.Name; IN type: Files.Type;
  OUT name: Files.Name);
BEGIN
  ASSERT(hook # NIL, 100); ASSERT(filename # "", 20);
  hook.GetName(filename, type, name)
END GetName;
```

TYPE **Hook**
ABSTRACT
Интерфейс разъёма дополнительных (недоступных в *Files*) хост-зависимых ФС-операций.

PROCEDURE (h: Hook) **GetFileName** (IN name: Files.Name; IN type: Files.Type; OUT filename: Files.Name)
NEW, ABSTRACT
Получить имя файла ФС *filename* по имени файла *name* и его типу *type*. То же, что *Files.Directory.GetFileName*; "собственная" реализация требуется для твёрдой гарантии, что данная операция будет обратна к *Hook.GetName*.

Предусловие
гарантируется $name \neq$ ""

PROCEDURE (h: Hook) **GetName** (IN filename: Files.Name; IN type: Files.Type; OUT name: Files.Name)
NEW, ABSTRACT
Получить имя файла *name* из имени файла ФС *filename*, в котором "смешаны" *name* и *type*.

Предусловие
гарантируется $filename \neq$ ""

Постусловие
$name =$ "" *filename* не является "смешением" *name* и *type*
$name \neq$ "" выделенное из *filename* имя

```
PROCEDURE (h: Hook) GetFileName (IN name: Files.Name; IN type: Files.Type;
  OUT filename: Files.Name);
BEGIN
  filename := name + "." + type
END GetFileName;

PROCEDURE (h: Hook) GetName (IN filename: Files.Name; IN type: Files.Type;
  OUT name: Files.Name);
  VAR  i, j: INTEGER;
BEGIN
  i := LEN(filename$); j := LEN(type$);
```

```
  IF j < i THEN
    WHILE (0 < j) & (filename[i - 1] = type[j - 1]) DO
      DEC(j); DEC(i)
    END;
    IF (0 = j) & (filename[i - 1] = ".") THEN
      DEC(i);
      WHILE j < i DO name[j] := filename[j]; INC(j) END;
      name[j] := 0X
    ELSE
      name := ""
    END
  ELSE
    name := ""
  END
END GetName;
```

Список литературы

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

[2] Документация модуля *Files* // Блэкбокс 1.5 — Базовая сборка проекта «Информатика-21» (версия 2012-11-09). URL: `http://www.inr.ac.ru/~blackbox/rsrc/blackbox15i21base.7z`