Л.И. Брусиловский, Ю.А. Михайлов

ОРГАНИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ МЕЖЪЯЗЫКОВЫХ МОДУЛЕЙ (PASCAL-2 - FORTRAN-4) В ОПЕРАЦИОННЫХ СИСТЕМАХ RSX-11M И RT-11

1. Введение

Средства вычислительной техники (СВТ) являются основной технической базой для создания любой комплексной системы автоматизации (КСА), характерным примером которой может служить комплексная система автоматизации компьютерной оптики (КСА КО) [1]. Развитое системное обеспечение большинства современных компьютеров позволяет выбрать операционные системы (ОС) в зависимости от места используемых СВТ в КСА. При этом проблема мобильности программного обеспечения решается, как правило, за счет использования в различных ОС единого языка высокого уровня. Так, для распространенного при решении задач КСА семейства 16-разрядных машин типа PDP-11 и совместимых с ними примерами таких языков могут служить FORTRAN-4 и PASCAL-2 [2-4].

Наличие компиляторов с PASCAL-2 в обеих ОС позволяет разрабатывать программы, эффективно использующие память машин: память под локальные объекты не включается в загрузочный модуль, а выделяется динамически по мере надобности в процессе работы, что крайне важно в силу ограниченности адресного пространства программ, работающих под управлением этих ОС. Однако достаточно большая часть существующего прикладного ПО в ОС RSX-11M и RT-11 написана на языке FORTRAN-4 (например, большое число различных библиотек научно-технических и инженерных расчетов). Поэтому вопрос об организации взаимодействия модулей, написанных на PASCAL-2 и FORTRAN-4, имеет важное практическое значение в рамках рассматриваемых

ОС. Такая необходимость может быть еще вызвана тем обстоятельством, что входящий в системы программирования PASCAL-2 обеих ОС пакет PASMAC, облегчающий интерфейс с ОС на ассемблере, не поддерживает автоматическую адресацию глобальных переменных, что может усложнить программирование и сопровождение разрабатываемых программ. В силу этого может оказаться целесообразным использовать входящий в состав обеих ОС интерфейс, оформленный в виде библиотеки системных подпрограмм, вызываемых с использованием обычной схемы FORTRAN (оператор CALL).

Хотя входной язык компилятора PASCAL-2 допускает обращение к внешним модулям, написанным на FORTRAN-4, тем не менее существует ряд ограничений, препятствующих непосредственному использованию имеющегося в PASCAL-2 механизма вызова непаскальных модулей с описанием NONPASCAL.

Во-первых, оба компилятора генерируют код формирования среды для работы программ, написанных на соответствующих языках. В первую очередь это касается организации управляющих структур для организации ввода/вывода (в/в) и инициализации стека (через регистр R6). Из-за различия организации в/в все операции по в/в, использующие встроенные в язык программирования средства, должны выполняться в модулях, написанных на том же языке, что и главная программа. (Тем не менее в модуле, написанном на языке, отличном от языка главной программы, возможна организация в/в на физическом уровне посредством обращения к системным функциям ОС.)

Во-вторых, компиляторы используют различный механизм вызова внешних модулей: PASCAL-2 организует передачу параметров через стек (использование описателя у внешней процедуры/функции NONPASCAL заставляет компилятор генерировать при ее вызове последовательность кодов, аналогичную генерируемой компилятором FORTRAN-4, то есть загружать в регистр R5 адрес списка адресов фактических аргументов). Однако даже при соблюдении стандартной последовательности вызова при организации взаимодействия модулей PASCAL-FORTRAN возможно возникновение проблем: передача имени внешнего модуля в качестве фактического параметра (непосредственно передать имя внешнего модуля, написанного на PASCAL-2, в модуль, написанный на языке FORTRAN, нельзя в силу особенностей генерации кода при передаче в качестве параметра имени внешней процедуры), доступ в фортрановском модуле к глобальным переменным PASCAL и т.д. и, наоборот, доступ к переменным блоков COMMON в модулях, написанных на PASCAL-2.

Непосредственный перенос двоичных файлов, содержащих записи переменной длины, превышающие 512 байт, из ОС RSX-11M в ОС RT-11 невозможен (системная утилита FLX ОС RSX-11M, предназначенная для переноса файлов между этими системами, обрабатывает записи длиной не более 512 байт). Ниже будет показано, как при помощи двух простых программ на PASCAL-2 легко решить эту проблему.

В данной статье будут рассмотрены практические приемы взаимодействия межъязыковых модулей (PASCAL-2 - FORTRAN-4) в ОС RSX-11M и RT-11, а также организации работы с файлами переменной длины на PASCAL-2/RT-11 и методы переноса двоичных записей, длина которых превышает 512 байт, из ОС RSX-11M в ОС RT-11.

2. Организация взаимодействия PASCAL-FORTRAN

2.1. ПОЛУЧЕНИЕ АДРЕСА ЗАГРУЗКИ ВНЕШНЕГО МОДУЛЯ

При программировании на языке FORTRAN данная проблема легко решается с помощью вызова специальной функции, входящей в состав системы программирования FORTRAN-4 обеих ОС (например, для RT-11 это функция IADDR). Напомним, что для машин семейства PDP-11 адрес - это целое число в диапазоне 0 - 65535. Получить адрес внешнего модуля в процедуре, написанной на языке PASCAL, можно, например, с помощью следующей вспомогательной раздельнокомпилируемой (для исключения проверки компилятором согласования типов фактического и формального параметров) функции:

```
(*$nomain*)
type
    ADDRESS : 0..65535;

function ADDR( I : ADDRESS )* ADDRESS; external;
function ADDR; (* тело функции ADDR *)
begin
    ADDR:=I
end;
```

При вызове функции ADDR в качестве фактического параметра указывается имя внешнего модуля (описанного как EXTERNAL), адрес которого необходимо получить. Например, из программы на языке PASCAL в подпрограмму на языке FORTRAN (FORT) необходимо передать имя внешнего модуля EXT в качестве фактического параметра. Схема вызова будет следующей:

```
.....
type
    ADDRESS : 0..65535;
.....
procedure EXT; external;
function ADDR( procedure F ) : ADDRESS; external;
procedure FORT( I : ADDRESS); nonpascal;
.....
    FORT(ADDR(EXT));
.....
```

Отметим, что, независимо от того, на каком языке написан модуль EXT, его необходимо описать как EXTERNAL, а не NONPASCAL. Не важно, имеет ли он формальные параметры, а также то, процедура (подпрограмма) это или

функция. Важно то, что в данном случае параметр (адрес точки загрузки внешнего модуля) должен передаваться в фортрановский модуль по значению.

Рассмотрим организацию модуля, написанного на PASCAL-2, вызываемого из модуля, написанного на FORTRAN-4. При вызове внешнего модуля компилятор FORTRAN-4 заносит в регистр R5 адрес списка адресов фактических параметров. В младшем байте первого слова списка содержится число фактических параметров, а адреса фактических параметров начинаются со второго слова списка. Допустим, из модуля, написанного на языке FORTRAN, в модуль, написанный на PASCAL-2, передаются три параметра: целое число, вещественный массив и вещественное число. Для отмены генерации кода проверки согласования длины массива модуль, написанный на PASCAL-2, должен компилировать с ключом /NOCHECK. Имитация списка формальных параметров может иметь вид:

```
(*$nomain*)
procedure EXT; external;
procedure EXT;
type
    VECTOR = array [1..100] of real;
    (* список адресов параметров *)
    PARAMETER_LIST = record
        NP : integer;   (* число параметров *)
        AI : ^integer;  (* адрес 1-го параметра *)
        AV : ^VECTOR;   (* адрес 2-го параметра *)
        AR : ^real;     (* адрес 3-го параметра *)
        end;
var
    (* адрес и содержимое R5 *)
    R5 origin 177058 : ^PARAMETER_LIST;
.....
begin
    K:=R5^.AI^; (* K = значению 1-го фактического параметра *)
.....
```

Если бы в данном случае схема вызовов была "PASCAL-FORTRAN-PASCAL", то в приведенном выше модуле при необходимости можно было бы обращаться к глобальным переменным.

2.2. ДОСТУП К РАЗДЕЛЯЕМЫМ ПЕРЕМЕННЫМ

Теперь рассмотрим вопросы, связанные с доступом к глобальным переменным модулей, написанных на PASCAL-2, из модулей, написанных на FORTRAN-4.

Компилятор PASCAL-2 размещает глобальные переменные в программной секции с именем GLOBAL (или с именем соответствующего модуля при использовании ключа компиляции /OWN) со смещением в 4 байта относительно ее начала. Следовательно, для получения доступа к таким переменным в модуле, написанном на FORTRAN-4, соответствующие переменные необходимо описать в блоке COMMON с именем GLOBAL (или соответствующим) со смещением в 4 байта относительно его начала:

```
var  (* глобальные переменные *)
     A : real;
     B : array [1..200] of boolean;
     C : record
           AX : integer;
           ST : array [1..100] of char
         end;
         SUBROUTINE F1( ПАРАМЕТРЫ, ЕСЛИ ЕСТЬ )
         COMMON/GLOBAL/DUMMY,A,B,AX,ST
         REAL A
         BYTE B(200),ST(100)
         INTEGER AX
.....
```

Здесь DUMMY - фиктивная переменная, наличие которой обеспечивает смещение в 4 байта относительно начала блока COMMON с именем GLOBAL. Для доступа к переменным блока COMMON в модуле на PASCAL-2 аналогичный метод непригоден, поскольку исполнительная система PASCAL-2 использует первые 4 байта программной секции.PSECT, отводимой для размещения глобальных переменных, и, следовательно, помещенная в них информация будет потеряна. В этом случае метод доступа к переменным блока COMMON будет следующим: необходимо на языке FORTRAN-4 написать подпрограмму, которая возвращала бы адреса (в виде 16-разрядных целых чисел) точек загрузки блоков COMMON (например, в RT-11 с помощью системной функции IADDR). В программе на PASCAL-2 этим адресам необходимо поставить в соответствие переменные ссылочного типа, базовый тип которых соответствует описанию блоков COMMON. Например, пусть необходимо получить доступ к переменным, размещенным в общих блоках COM1, COM2 и COM3, в процедуре, написанной на PASCAL-2.

Тогда подпрограмма на FORTRAN-4 может иметь вид:

```
      SUBROUTINE CADDR(ADDR)
      INTEGER ADDR(1)
      COMMON/COM1/A(100),I(10)
      COMMON/COM2/B(200),J(300),F
      COMMON/COM3/Z
      ADDR(1)=IADDR(A)          ! или ADDR(1)=IADDR(COM1)
      ADDR(2)=IADDR(B)          ! или ADDR(2)=IADDR(COM2)
      ADDR(3)=IADDR(Z)          ! или ADDR(3)=IADDR(COM3)
      RETURN
      END
```

а вызывающая процедура на PASCAL-2:

```
type
     COMADDR = array [1..3] of integer;
     CM1 = record
             A : array [1..100] of real;
             I : array [1..10] of integer;
           end;
     CM2 = record
             B : array [1..200] of real;
             I : array [1..300] of integer;
           end;
     CM3 = real;
     ACOM1 = ^CM1;
     ACOM2 = ^CM2;
     ACOM3 = ^CM3;
.............
var
     COM1 : ACOM1;
     COM2 : ACOM2;
     COM3 : ACOM3;
     ADDR : COMADDR;
.............
procedure CADDR( var A : COMADDR ); nonpascal;
.............
begin
.............
     CADDR(ADDR);
     COM1:=loophole(ACOM1,ADDR[1]);
     COM2:=loophole(ACOM2,ADDR[2]);
     COM3:=loophole(ACOM3,ADDR[3]);
.............
     K:=COM1^.I[L]+COM2^.J[1];
.............
```

---

3. Работа с файлами

3.1. ЗАПИСИ ПЕРЕМЕННОЙ ДЛИНЫ НА PASCAL

В PASCAL-2 все записи в нетекстовых файлах рассматриваются как записи фиксированной длины, определяемой типом файла. Работу с записями переменной длины на языке PASCAL можно организовать следующим образом, используя механизм раздельной компиляции модулей. Файл, содержащий записи переменной длины, необходимо описать как

```
type FBYTE = file of integer
```

и весь в/в записей переменной длины организовать с помощью двух процедур, написанных на PASCAL-2: процедуры WVL (вывод) и RVL (ввод):

```
(*$nomain*)
type
    FBYTE = file of integer;
    ABYTE = array [1..10] of integer;
procedure WVL(var F:FBYTE; var X:ABYTE; K:integer); external;
procedure WVL;
var
    I : integer;
begin
    write( F,K );          (* записать длину записи *)
    (* запись структуры *)
    for I:=1 to (K+1)div2 do write(F,X[I])
end;
procedure RVL(var F:FBYTE; var X:ABYTE; K:integer); external;
procedure RVL;
var
    I : integer;
begin
    read( F,K );           (* читать длину записи *)
    (* читать запись *)
    for I:=1 to (K+1)div2 do read(F,X[I])
end;
```

Пример вызова процедуры WVL для записи:

```
type
    FBYTE = file of integer;
    VREC1 = record ... end;
    VREC2 = record ... end;
.................
var
    F1, F2 : FBYTE;
    X : VREC1;
    Y : VREC2;
    I : integer;
.................
procedure WVL(var F:FBYTE; I,LEN:integer);external;
.................
procedure RVL(var F:FBYTE; I:integer; var LEN:integer);external;
.................
begin
.................
    WVL(F1,loophole(integer,ref(X)),size(VREC1));
.................
    RVL(F2,loophole(integer,ref(Y)),I));
.................
end.
```

Получающаяся структура записей будет идентична структуре записей переменной длины в ОС RSX-11M.

3.2. ОБРАБОТКА СИМВОЛЬНЫХ СТРОК ПЕРЕМЕННОЙ ДЛИНЫ

Жесткий контроль типов, присущий языку программирования PASCAL-2, создает определенные проблемы, связанные с обработкой текста. Входящий в систему программирования PASCAL-2 пакет для работы со строками, не

обладает гибкостью, поскольку все строки в этом случае должны иметь одинаковую максимальную длину, независимо от конкретных случаев. Используя системную объектную библиотеку SYSLIB, содержащую подпрограммы работы со строками, и механизм задания строк переменной длины (конец строки определяется первым вхождением нулевого байта), эту проблему можно легко решить. В качестве фактических параметров системных подпрограмм необходимо указывать не сами строки, а их адреса (в виде целых чисел): при этом эти адреса должны передаваться по значению. В качестве примера приведен вызов функции определения длины строки:

```
.................
var S: packed array[1..81] of char;
.................
function LEN( I: integer): integer; nonpascal;
.................
    K:=LEN(loophole(integer, ref(S)));
.................
```

3.3. ПЕРЕНОС ЗАПИСЕЙ ПЕРЕМЕННОЙ ДЛИНЫ ИЗ ОС RSX-11M В ОС RT-11 ПРИ ДЛИНЕ ЗАПИСЕЙ БОЛЬШЕ 512 БАЙТ

Данная проблема возникает при попытке переноса файлов, созданных операторами бесформатного ввода FORTRAN ОС RSX-11M, в ОС RT-11, когда длина записей переносимого файла превышает 512 байт. При ее решении мы рекомендуем следующий алгоритм:

- программой PIP/RSX-11M сделать переносимый файл непрерывным (при помощи ключа /CO программы PIP);
- с помощью программы DMP/RSX-11M распечатать заголовок файла и определить адрес 1-го блока файла на диске (виртуальный блок 0).

Далее под управлением ОС RT-11 необходимо проделать следующее:

- командой

COPY/DEV/FIL DV1:/START:STARTADDRESS/END:ENDADDRESS DV2:FILNAM.EXT

скопировать файл с диска формата RSX-11 на диск формата RT-11, где ENDADDRESS - адрес 0-го блока файла на диске, полученный программой DMP + длина переносимого файла;

- если данный файл необходимо читать из модулей, написанных на PASCAL-2, то больше ничего делать не надо; этот файл уже можно читать приведенной выше процедурой RVL. Если данный файл необходимо обрабатывать операторами последовательного бесформатного ввода FORTRAN, необходимо выполнить две следующие программы (первая написанная на PASCAL-2 читает файл и формирует промежуточный выходной файл; вторая программа, написанная на FORTRAN-4, читает промежуточный файл и формирует бесформатные записи FORTRAN-4/RT-11).

```
type
    F1: file of integer;
    F2: text;
    I,J,K: integer;
begin
    reset( F1, ИМЯ ): rewrite( F2, ИМЯ );
    while not eof(F1) do begin
        (* Длина записи *)
        read(F1,I); write(F2,I);
        for J:=1 to (I+1)div2 do begin
            read(F1,K);
            writeln(F2,K)
        end
    end
end.
```

```
        BYTE M(МАКСИМАЛЬНАЯ ДЛИНА В БАЙТАХ)
        INTEGER N(МАКСИМАЛЬНАЯ ДЛИНА В СЛОВАХ)
        EQUIVALENCE(M,N)
        OPEN(UNIT=1, NAME='ИМЯПРОМЕЖФАЙЛА'),DISPOSE='DELETE')
        OPEN(UNIT=2, NAME='ИМЯВЫХФАЙЛА'),FORM='UNFORMATTED')
1       READ(1,*,END=3) L ! ДЛИНА В СЛОВАХ
        DO 2 I=1,(L+1)/2
2       READ(1,*) N(I)
        WRITE (2)(M(I),I=1,L)
        GOTO 1
3       STOP
        END
```

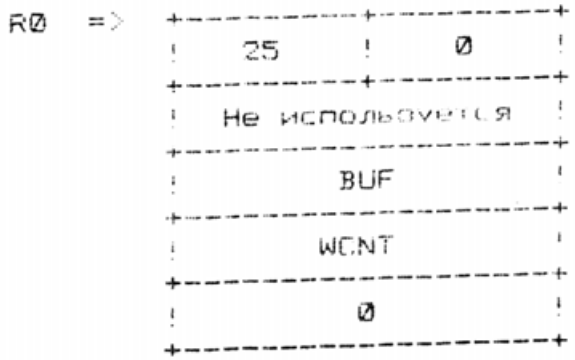
---

4. Обращение к монитору из программы, написанной на PASCAL-2

Для обращения к монитору непосредственно из программы, написанной на PASCAL-2, можно использовать встроенную (предопределенную) процедуру EMT. Обращение в программе к этой процедуре эквивалентно выдаче команды EMT с кодом, равным значению параметра процедуры EMT. Например, вызов EMT (375B) равносилен наличию инструкции:

EMT 375

Заметим, что перед обращением к процедуре EMT необходимо в регистр R0 загрузить адрес блока аргументов EMT. Допустим, из программы на PASCAL-2 необходимо передать данные другому заданию с помощью системного вызова SDATW. Формат таблицы аргументов EMT имеет следующий вид:



где

BUF - адрес буфера для передачи данных;

WCNT - длина буфера в словах.

Фрагмент программы на PASCAL-2 может иметь вид:

```
const   BUFLEN=1000B;
type    BUFFER=array [1..BUFLEN] of integer;
        ARGBLOCK=record
            CODE: packed array[1..2] of 0..255;
            DUMMY: integer;
            BUF: ^BUFFER;
            WCNT: integer;
            ENDA: integer;
        end;
var     BUFF: BUFFER;
        EMTARG: ARGBLOCK;
        R0 origin 177700B : ^ARGBLOCK; (* это для ЭВМ типа PDP-11/40 *)

begin
.................
        with  EMTARG do begin
                CODE[1]:=0;
                CODE[2]:=25B;
                BUFF:=ref(BUFF);
                WCNT:=BUFLEN;
                ENDA:=0;
        end;
        R0:=ref(EMTARG);
        EMT(375B);
```

Литература

1. Березный А.Е., Брусиловский Л.И. и др. Проект системы автоматизации проектирования, создания, исследования и применения элементов плоской оптики (версия 1). - В сб.: Компьютерная оптика, вып. 2. Автоматизация проектирования и технологии. М.: МЦНТИ, 1987.

2. PASCAL-2: Version 2.0 for RT-11. S.L., 1981.

3. PDP-11 PASCAL/RSX-11 Programmer's Guide. Maynard, Digital Equipment Corp., 1983.

4. PDP-11 Fortran Language Reference Manual. Maynard, Digital Equipment Corp., 1979.