Л.И. Брусиловский, Ю.А. Михайлов

РАЗРАБОТКА МОБИЛЬНОГО МУЛЬТИПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА БАЗЕ СП PASCAL-2

1. Введение

В последнее время все большую популярность завоевывают языки высокого уровня, имеющие средства доступа к "нижнему уровню" и позволяющие создавать сложные пакеты программ, обладающих высокой степенью мобильности. В первую очередь это языки ADA [1], C[2], Modula-2 [3] и диалекты языка Pascal [4]. Сравнение названных языков программирования со стороны разработки мобильного программного обеспечения (кроме Modula-2) приведено в [5].

Язык Modula-2 часто представляется как "Паскаль без недостатков". В то же время в различных реализациях языка Pascal предприняты попытки устранить ряд его "узких мест". В первую очередь это касается мобильной СП Pascal-2 [6,7], реализация которой имеется для таких операционных систем, как RT/11 /TSX-Plus/ SHAREplus, RSX-11M, 1AS, RSTS/E (линия ЭВМ PDP-11), VMS (линия ЭВМ VAX), MS-DOS/PC-DOS (линия ЭВМ IBM PC/XT/AT), VersaDOS (линия ЭВМ на базе микропроцессоров фирмы Motorola) [6-8].

В СП Pascal-2 введены расширения стандартного языка Pascal, которые не только уравнивают его возможности с Modula-2, но в ряде случаев выгодно его отличают. Это, например:

- возможность передачи в качестве параметров многомерных массивов с переменными границами;
- константы с типом;
- работа с файлами прямого доступа;

- возможность обращения к модулям, написанным на других языках программирования.

Кроме того, компилятор Pascal-2 является оптимизирующим, а в СП Pascal-2 имеются мощные средства интерактивной отладки программ на лексике языка высокого уровня и возможности профилирования выполнения программ. Поэтому СП Pascal-2 является реальным конкурентом СП Modula-2 в области создания больших мобильных пакетов программ [9-11].

Тем не менее в СП Pascal-2 нет средств, аналогичных механизму сопрограмм (процессов) в языке Modula-2, с помощью которых легко реализуются такие компоненты операционных систем, как, например, мониторы.

В данной статье предлагаются средства реализации механизма сопрограмм в среде СП Pascal-2.

2. Подпрограммы и сопрограммы

Основное различие между сопрограммами и подпрограммами заключается в том, что выполнение сопрограмм может быть приостановлено с сохранением контекста значений локальных переменных, пока процессор занят выполнением некоторой другой работы, а потом возобновлено с точки останова в старом контексте. При вызове подпрограммы ей передается управление на одну из точек входа. Контекст локальных переменных вызывающей подпрограммы сохраняется, а ее выполнение будет возобновлено со следующего за вызовом оператора после полного завершения работы вызванной подпрограммы, причем контекст вызванной сопрограммы не сохраняется. Повторное обращение к вызываемой подпрограмме вновь передаст управление на одну из точек входа в новом контексте.

Наглядно различие между со- и подпрограммами можно увидеть на примере совместной работы двух модулей, обращающихся друг к другу. В случае подпрограмм такие взаимные обращения приводят к бесконечной рекурсии (что, впрочем, может быть предусмотрено алгоритмом). В случае сопрограмм сохраняется последовательное выполнение. Схематически различие между со- и подпрограммами показано на рис. 1 и рис. 2.

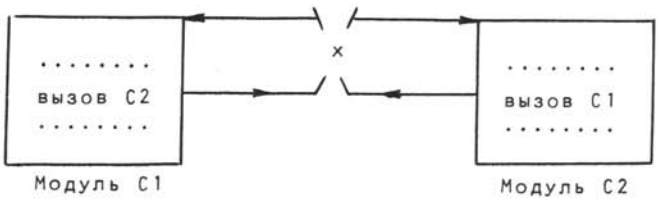

Рис. 1. Бесконечная рекурсия при взаимном вызове двух подпрограмм

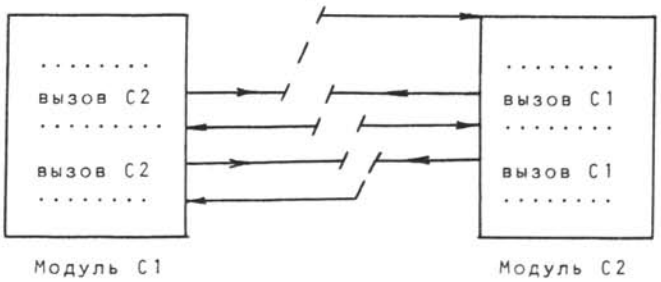

Рис. 2. Передача управления между сопрограммами

В языке Modula-2 работа с сопрограммами реализуется с помощью двух процедур, определенных в модуле SYSTEM (NewProcess и Transfer), и двух процедур и одной функции, определенных в модуле STORAGE (Allocate, Deallocate, Available). Кроме того, в модуле SYSTEM определяются типы данных Address и Process, структура которых является системно-зависимой.

Преобразование процедуры, не имеющей параметров, в сопрограмму (т.е. в Process) выполняется при вызове процедуры NewProcess, заголовок которой обычно имеет вид:

```
Procedure NewProcess (P, STADDR, SIZE : Address;
                      var PROC : Process): external;
```

3. Реализация сопрограмм в СП PASCAL-2

Наличие в СП Pascal-2 средств раздельной компиляции внешних процедур и функций, связи с ассемблером и ряда средств нижнего уровня позволяет реализовать на Pascal-2 механизм сопрограмм, а возможность включения модулей исходного текста на этапе компиляции (директива %include) позволяет записать определения необходимых структур данных и заголовков внешних процедур и функций в некоторый файл, который можно рассматривать как аналог модуля определений в языке Modula-2.

Как уже отмечалось, для реализации работы с сопрограммами в среде Pascal-2 в стиле языка Modula-2 необходимо смоделировать структуру данных Process, процедуры NewProcess, Transfer, Allocate и Deallocate, а также функции Available языка Modula-2.

Аналоги процедур Allocate и Deallocate, а также функции Available имеются в СП Pascal-2 (версии, начиная с 2.1): соответственно это p$inew, p$dispose и space. Для удобства их целесообразно привести к виду, совпадающему с видом в Modula-2:

Фрагмент 1

```
(*$nomain*)
(*$nostacksheck*)
function p$inew(SIZE : integer) integer; external;
procedure p$dispose (PTR, SIZE : integer); external;
function space : integer; external;
procedure Allocate (var PTR : integer, SIZE : integer); external;
procedure Deallocate (P, SIZE : integer); external;
function Available : integer; external;
procedure Allocate;
begin
    PTR:=p$inew(SIZE)
end;
procedure Deallocate;
begin
    p$dispose (P, SIZE)
end;
function Available;
begin
    Available:=space
end;
```

В СП Pascal-2 версии 2.0 и ниже процедуры p$inew и p$dispose отсутствуют. Тем не менее их можно реализовать самостоятельно, используя пакет PASMAC [6, 7]:

Фрагмент 2

```
        .title  p$inew
; function p$inew(SIZE : integer) integer; external;
        func    p$inew,PTR,integer,check=0
        param   SIZE,integer
        begin
        inc     SIZE(sp)
        bic     #1,SIZE(sp)
        mov     SIZE(sp),-(sp)
        jsr     pc,$b70
        .globl  $b70
        mov     (sp)+,PTR(sp)
        endpr
; procedure p$dispose (P, SIZE : integer); external;
        proc    p$dispose,check=0
        param   P,integer
        param   SIZE,integer
        save    <r0>
        begin
        mov     SIZE(sp),r0
        inc     r0
        bic     #1,r0
        mov     P(sp),-(sp)
        jsr     pc,$b72
        .globl  $b72
        endpr
        end.
```

Рассмотрим теперь реализацию процедур NewProcess и Transfer. При создании новой сопрограммы вызовом процедуры NewProcess в структуру данных Process необходимо поместить информацию о стартовом адресе соответствующей процедуры (не имеющей параметров), начальном значении стека (значение, возвращаемое процедурой Allocate, плюс размер рабочей области процесса, т.к. вершина стека при его заполнении смещается сверху вниз) и размере рабочей области (который обычно определяется опытным путем). При создании сопроцесса начальное содержимое регистров r0-r5 для него несущественно. Получение адреса загрузки процедуры в СП Pascal-2 рассмотрено в [9]. Тем не менее, в последующем значения регистров r0-r5 необходимо сохранять и восстанавливать. В [12] предлагается сохранять регистры в стеке, однако это сопряжено с необходимостью отслеживать в стеке как область сохранения сопрограммы, так и локальные переменные. Поэтому сохранять регистры r0-r5 удобнее в самой структуре данных Process. С учетом вышесказанного приводим алгоритм:

! Фрагмент 3 !

```
(*$nomain*)
(*$nostacksheck*)
type
    ADDRESS     = 0..65535;
    Process     = record
        SAVEAREA        : array [0..5] of ADDRESS;
        SP,PC           : ADDRESS
    end;
procedure NewProcess( P,STADDR, SIZE : ADDRESS;
                      var PROS: Process ); external;
procedure NewProcess;
begin
    with PROC do begin
        PC:=P; (* стартовый адрес процедуры P *)
        SP:=STADDR+SIZE-2 (* вершина стека рабочей области *)
    end
end;
```

Компиляция с ключом $ nostacksheck необходима для отмены проверки переполнения стека, поскольку стек сопроцесса берется из хипа программы.

При передаче управления сопрограмме процедура Transfer должна сохранить содержимое регистров r0-r5, убрать из стека аргументы вызова процедуры Transfer и сохранить после этого значение верхушки стека и адрес возврата в вызвавшую процедуру. Реализация процедуры Transfer, с учетом вышесказанного, возможна лишь на ассемблере с использованием пакета PASMAC. Следует учесть, что при входе в процедуру Transfer вершина стека указывает на адрес возврата в вызвавший модуль, а над ней находятся второй и первый параметры вызова процедуры Transfer. Реализация процедуры Transfer может быть такой:

! Фрагмент 4 !

```
        .title Transfer
; рабочие ячейки:
WRK:    .word
WRK1:   .word
; type
; Process   = array [0..7] of integer;
; procedure Transfer( var SOURCE, DEST : Process );
        proc    Transfer,check=0
        param   SOURCE,ADDRESS
        param   DEST,ADDRESS
        begin
        mov     r0,WRK          ; сохранить текущее значение r0
        mov     SOURCE(sp),r0   ; адрес области сохранения SOURCE
; сохранить в SOURCE регистры r0-r5
        mov     WRK,(r0)+
        mov     r1,(r0)+
        mov     r2,(r0)+
        mov     r3,(r0)+
        mov     r4,(r0)+
        mov     r5,(r0)+
; в стеке адрес возврата - запомнить в r1
        mov     (sp),r1
; в стеке за SOURCE аргументы Transfer
        add     #4,r1
```

```
; сохранить правильное значение стека для SOURCE
        mov     r1,(r0)+
        mov     (sp),(r0)       ; адрес возврата
        mov     DEST(sp),r0     ; адрес DEST
; восстановить регистры r0-r5
        mov     (r0)+,WRK
        mov     (r0)+,r1
        mov     (r0)+,r2
        mov     (r0)+,r3
        mov     (r0)+,r4
        mov     (r0)+,r5
        mov     (r0)+,sp
        mov     (r0)+,WRK1      ; точка возобновления процесса
                                ; DEST
        mov     WRK,r0          ; восстановить r0
        jmp     @WRK1           ; возобновить процесс DEST
        endpr
        end.
```

Проверку правильности реализации механизма сопрограмм можно проверить на контрольном примере из [12]:

```
                                        !   Фрагмент 5   !
const
    MEMREQ  = 100;
var
    MAINPRG, P1, P2 : process;
    STACK1, STACK1  : integer;
    I, J            : integer;
%include PROSEC;
procedure TEST1;
begin
    writeln('Первый вход в процедуру TEST1');
    Transfer(P1,MAINPRG);
    J:=7;
    writeln('Второй вход в процедуру TEST1');
    Transfer(P1,MAINPRG);
    writeln('Третий вход в процедуру TEST1');
    Transfer(P1,MAINPRG);
end;
procedure TEST2;
var
    X   : integer;
begin
    writeln('Первый вход в процедуру TEST2');
    X:=1;
    Transfer(P2,MAINPRG);
    writeln('Второй вход в процедуру TEST2, X= ', X, ' J= ', J);
    Transfer(P2,MAINPRG);
end;
begin
    Allocate(STACK1,MEMREQ);
    Allocate(STACK2,MEMREQ);
    NewProcess(TEST1,STACK1,MEMREQ,P1)
    NewProcess(TEST2,STACK2,MEMREQ,P2)
    writeln('Transfer к TEST1');
    J:=5;
    Transfer(MAINPRG, MAINPRG);
    Transfer(MAINPRG,P1); writeln('Назад из TEST1');
    Transfer(MAINPRG,P1); writeln('Transfer к TEST2');
    Transfer(MAINPRG,P1); writeln('Назад из TEST2');
    Transfer(MAINPRG,P1); writeln('Назад из TEST1');
    Transfer(MAINPRG,P1); writeln('В ведущей программе')
end.
```

Файл PROCES.PAS, включаемый во время компиляции, содержит следующие описания:

Фрагмент 6

```
(*$nostacksheck*)
type
    PROCESS = record
        SAVEAREA    : array [0..5] of integer;
        SP          : integer;
        PC          : integer;
    end;
procedure NewProcess(  procedure PROC;
                ADR, SIZE   : integer;
                var P       : Process ); external;
procedure Transfer(  var P1, P2 : Process ); external;
procedure Allocate(  var P : integer; SIZE : integer ); external;
```

Если в программах, использующих сопроцессы, применяется аппаратная арифметика с плавающей точкой (сопроцессор с набором команд FPP), то также необходимо сохранять аккумуляторы плавающей арифметики. Поскольку это, с одной стороны, тривиально, а с другой стороны, используется не часто, то сохранение этих аккумуляторов мы не рассматривали.

4. Реализация сопрограмм в СП Pascal-1

На малых микроЭВМ типа ДВК до настоящего времени активно используется СП Pascal-1, которая отличается небольшой потребностью в дисковых ресурсах, хотя и упрощенной реализацией языка Pascal. Практически все вышеуказанные механизмы введения сопрограмм применимы для СП Pascal-1. Так, содержимое файла PROSEC.PAS остается без изменений. Некоторые изменения должны быть внесены в программные модули, т.к. с СП Pascal-1 регистр r5 используется для указателя на область глобальных переменных и не должен изменяться в процессе работы:

Фрагмент 7

```
(* $e+, $a-, $t-, $c     .title NewProcess *)
type
    ADDRESS     = 0..65535;
    Process     = record
        SAVEAREA        : array [0..5] of ADDRESS;
        SP,PC           : ADDRESS
    end;
procedure NewProcess( P,STADDR, SIZE : ADDRESS;
                      var PROS: Process ); external;
procedure NewProcess;
begin
    with PROC do begin
        PC:=P;  (* стартовый адрес процедуры P *)
        SP:=STADDR+SIZE-2 (* вершина стека рабочей области *)
    end
end;

(* $e+, $a-, $t-, $c     .title Transfer *)
procedure Transfer( var SOURCE, DEST : Process );
type
    Process    = array [0..7] of integer;
begin
(*$c
        mov     r0,WRK          ; сохранить текущее значение r0
        mov     SOURCE(sp),r0   ; адрес области сохранения SOURCE
; сохранить в SOURCE регистры r0-r4
        mov     WRK,(r0)+
        mov     r1,(r0)+
        mov     r2,(r0)+
        mov     r3,(r0)+
        mov     r4,(r0)+
; в вершине стека адрес возврата - запомнить в r1
        mov     sp,r1
; убрать из стека SOURCE аргументы Transfer
        add     #6,r1
```

```
; сохранить правильное значение стека для SOURCE
        mov     r1,(r0)+
        mov     (sp),(r0)       ; адрес возврата
        mov     DEST(sp),r0     ; адрес DEST
; восстановить регистры r0-r4
        mov     (r0)+,WRK
        mov     (r0)+,r1
        mov     (r0)+,r2
        mov     (r0)+,r3
        mov     (r0)+,r4
        mov     (r0)+,sp
        mov     (r0)+,WRK1      ; точка возобновления процесса
                                ; DEST
        mov     WRK,r0          ; восстановить r0
        jmp     @WRK1           ; возобновить процесс DEST
; рабочие ячейки:
WRK:    .word
WRK1:   .word
*)
end;
(* 0e+, 0a-, 0t-, 0c     .title MEMMAN *)
function space : integer; external;
function psinew(SIZE : integer) integer; external;
procedure psdispose (PTR, SIZE : integer); external;
procedure Allocate (var PTR : integer, SIZE : integer);
begin
    PTR:=psinew(SIZE)
end;
procedure Deallocate (P, SIZE : integer);
begin
    psdispose (P, SIZE)
end;
function Available : integer;
begin
    Available:=space
end;

(* 0e+, 0a-, 0t-, 0c     .title  psinew *)
function psinew(SIZE : integer) integer;
var
    PTR : integer;
begin
(*0c
        inc     SIZE(sp)
        bic     #1,SIZE(sp)
        mov     SIZE(sp),-(sp)
        jsr     pc,0b70
        .globl  0b70
        mov     (sp)+,PTR(sp)
*)
    PSINEW:=PTR
end;
procedure psdispose (P, SIZE : integer);
begin
(*0c
        mov     SIZE(sp),r0
        inc     r0
        bic     #1,r0
        mov     P(sp),-(sp)
        jsr     pc,0b72
        .globl  0b72
*)
end.
```

Литература

1. Язык программирования АДА. ГОСТ 27831-88 (ИСО 8652-87).
2. Керниган Б., Ритчи Д., Фьюэр А. Язык программирования СИ. Задачи по языку СИ. М., Финансы и статистика, 1985.
3. Вирт Н. Программирование на языке Модула-2. М., Мир, 1987.
4. Вирт Н., Йенсен К. Паскаль: руководство для пользователя. М., Финансы и статистика, 1989.
5. Языки программирования АДА, СИ, ПАСКАЛЬ. Сравнение и оценка (под ред. Джехани Н.). М., Радио и связь, 1989.
6. Pascal-2. Version 2.0 for RT-11, Oregon MiniSoftware Inc., 1981.
7. Pascal-2. Version 2.1 for RSX-11, Oregon MiniSoftware Inc., 1983.
8. Souter J., Davies M. British Standard Time, Pers. Comput. World, 1988, v. 11, N 6.
9. Брусиловский Л.И., Михайлов Ю.А. Организация взаимодействия межъязыковых модулей (Pascal-2 - Fortran-IV) в операционных системах RSX-11M и RT-11. - В сб. "Компьютерная оптика", вып. 4. М., МЦНТИ, 1989.
10. Брусиловский Л.И., Михайлов Ю.А. Разработка мобильного программного обеспечения для ЭВМ линии PDP-11 и VAX. - В сб. "Компьютерная оптика", вып. 7. М., МЦНТИ, 1990.
11. Михайлов Ю.А. Программирование системных вызовов на Pascal-2 в операционных системах РАФОС и ДОС КП (краткое сообщение). Управляющие системы и машины, № 5, 1989.
12. Beer J., Rojas R. Coroutinen in C und Pascal, mc: Die Mikrocomputer-Zeitschrift, 1987, N 7.