СИСТЕМА ПРОГРАММИРОВАНИЯ RPS

А. О. Махорин

Сентябрь 2012 г.

1. Введение

Система программирования RPS представляет собой среду разработки компьютерных программ на базе стандартных версий языков программирования Фортран 77 (ANSI Fortran 77) и Си 89 (ANSI C 89). Данная система ориентирована в основном на использование в учебных и учебно-исследовательских проектах, связанных с разработкой и реализацией алгоритмов и численных методов.

Версия системы RPS 1.8 предназначена для использования на компьютере с процессором Intel IA-32 (x86) или совместимым с ним под управлением операционной системы Microsoft Windows XP или в среде Wine[*)] под управлением операционной системы GNU/Linux.

Все программное обеспечение, входящее в состав системы RPS (текстовый редактор, управляющая программа, компиляторы, библиотеки стандартных программ), является свободным (*software libre*).

2. Подготовка системы к работе

Чтобы подготовить систему RPS к работе под управлением MS Windows, необходимо выполнить следующие действия.

1. Вначале следует скопировать дистрибутив системы RPS с интернет-страницы `<http://sourceforge.net/projects/noumenon/files/>`.

Дистрибутивный файл представляет собой zip-архив, который имеет имя `rps-X.Y.zip`, где `X` — номер версии системы, `Y` — номер подверсии, например, `rps-1.8.zip`.

2. Далее дистрибутив системы RPS необходимо распаковать, используя любую подходящую программу-архиватор (WinZip, WinRAR, и т. п.), поменять имя полученного каталога (папки) на `rps`, и переместить этот каталог в рабочий каталог пользователя. (В принципе, каталог RPS может располагаться в любом месте, при условии, что он доступен для чтения и записи.) Ниже предполагается, что каталог системы RPS находится в каталоге My Documents (Мои Документы) пользователя `mao`, а значит, полное имя этого каталога есть `C:\Documents and Settings\mao\My Documents\rps`.

[*)] Wine (Wine Is Not an Emulator) — операционная среда, обеспечивающая выполнение 16- и 32-битовых прикладных программ (приложений), разработанных для MS Windows, под управлением GNU/Linux. Пакет Wine является свободным программным обеспечением.

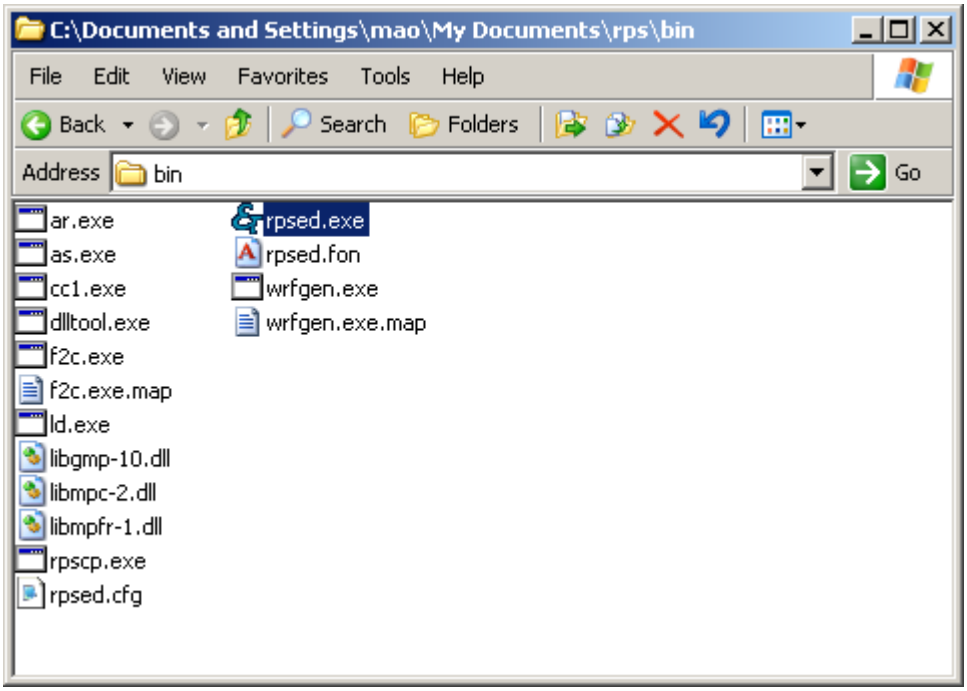

Рис. 1. Первый запуск текстового редактора системы RPS.

3. Теперь необходимо привязать текстовый редактор системы RPS к файлам типа "`.rps`" (файлы заданий для системы RPS). Для этого надо войти в подкаталог `rps\bin`, запустить программу текстового редактора `rpsed.exe` (рис. 1), и после появления окна текстового редактора закрыть его.

4. Чтобы удостовериться, что регистрация привязки прошла успешно, следует войти в подкаталог `rps\examples` и дважды кликнуть левой кнопкой "мыши" по любому файлу, имеющему тип "`.rps`" — это должно вызвать открытие выбранного файла текстовым редактором.

Примечание. Для более удобного вызова текстового редактора рекомендуется создать ссылку (shortcut) на программу `rps\bin\rpsed.exe` и переместить эту ссылку либо в основное меню вызова программ, либо на "рабочий стол" (Desktop).

5. В системе RPS используется динамическая загрузка и исполнение программных модулей (в частности, программных модулей, которые получаются в результате компиляции программ пользователя). Поэтому если на компьютере установлено антивирусное программное обеспечение, которое может блокировать использование указанной возможности, следует отключить такой режим блокировки для программ `rps\bin\rpsed.exe` (текстовый редактор) и `rps\bin\rpscp.exe` (управляющая программа).

6. Если систему RPS необходимо удалить, то для этого достаточно удалить каталог `rps`. Никакие другие действия (в частности, использование управляющей панели Windows) не требуются.

Для подготовки системы RPS к работе под управлением операционной системы GNU/Linux необходимо выполнить следующие действия.

1. Установить пакет Wine. (Этот пакет входит в стандартный дистрибутив GNU/Linux, секция "Cross Platform". Рекомендуется использовать версию не ниже 1.0.1-3.1.)

2. Распаковать дистрибутив системы RPS, поменять имя полученного каталога на `rps` и переместить этот каталог в личный каталог пользователя. (Например, для пользователя с идентификатором `mao` абсолютное имя каталога системы RPS должно быть `/home/mao/rps`.)

3. Установить для файла `rps/bin/rpsed` признак разрешения исполнения. Этот текстовый файл содержит следующую командную процедуру, которую можно использовать для вызова текстового редактора:

```
#!/bin/sh
wine ~/rps/bin/rpsed.exe $1
```

4. Создать ссылку (link) на файл `rps/bin/rpsed` и переместить эту ссылку либо на панель вызова программ, либо на "рабочий стол" (Desktop). Также рекомендуется ассоциировать файлы, имеющие суффикс `".rps"` (файлы заданий для системы RPS) с указанной командной процедурой для автоматического запуска текстового редактора.

3. Работа с системой

Основной единицей обработки в системе RPS является *задание* (*job*). Задание представляет собой обычный текстовый файл, содержащий *управляющие операторы* (*control statements*), исходный код программы на языке Фортран или Си, а также исходные данные для программы.

Для подготовки и выполнения заданий предназначен текстовый редактор RPSED, входящий в состав системы RPS.

Задание состоит из *шагов*, где отдельный шаг определяет программу, которую необходимо вызвать для выполнения определенного действия.

Рассмотрим в качестве примера следующее задание.

```
// job myjob
// exec fortran
      program main
      integer n, sum, i, a(100)
      read *, n
      read (*,*) (a(i), i = 1, n)
      sum = 0
      do 10 i = 1, n
   10 sum = sum + a(i)
      print *, 'sum =', sum
      stop
      end
```

```
// exec link
// exec *
5
12 23 8 19 3
// end
```

Строчки текстового файла, которые начинаются с двух наклонных штрихов (//), являются управляющими операторами задания.

Каждый управляющий оператор имеет следующий формат:

$$// \ \omega \ \ p_1 \ p_2 \ \ldots \ p_n$$

где ω — мнемонический код операции (job, exec, и т. п.), $p_1, p_2, \ldots, p_n$ — дополнительные операнды. Два наклонных штриха, которые являются признаком управляющего оператора, должны находиться в колонках 1 и 2. Колонка 3 должна содержать пробел. Код операции и дополнительные операнды следует записывать, начиная с 4-й колонки и отделяя их друг от друга одним или несколькими пробелами.

Оператор job определяет начало задания и должен быть первым управляющим оператором в задании. После ключевого слова job должно следовать имя задания (от 1 до 8 алфавитно-цифровых символов).

Оператор exec fortran вызывает компилятор Фортрана, который считывает исходный код программы и преобразует его в объектный код. Если второй операнд отсутствует, считается, что исходный код программы следует непосредственно за оператором exec (как в рассматриваемом примере). В остальных случаях второй операнд должен задавать имя текстового файла, содержащего исходный код программы, например:

```
// exec fortran foobar.f
```

Результатом компиляции является объектный модуль, который записывается во временный файл.

Оператор exec link вызывает линкер (редактор связей), который объединяет все объектные модули, созданные на предшествующих шагах компиляции, а также объектные модули из стандартной библиотеки Фортрана, в один загрузочный модуль, содержащий исполнимый код, и записывает этот загрузочный модуль во временный файл.

Оператор exec * вызывает программу (т. е. загружает и исполняет соответствующий загрузочный модуль), созданную линкером на предшествующем шаге задания. По умолчанию считается, что исходные данные (стандартный ввод) для программы следуют непосредственно за оператором exec, а результат выполнения программы (стандартный вывод) должен быть записан в листинг задания (см. ниже).

Оператор end определяет конец задания. Этот оператор должен быть последним управляющим оператором в задании.

Результатом выполнения задания является так называемый *листинг задания* (*job listing*), который содержит управляющие операторы задания, а также стандартный вывод каждой программы, вызванной в процессе выполнения задания. Например, в результате выполнения задания, рассмотренного выше, будет получен следующий листинг:

```
0001 // job myjob
0002 // exec fortran
0013 // exec link
0014 // exec *
 sum = 65
0017 // end
```

(Номера управляющих операторов — это номера соответствующих строчек текстового файла, содержащего задание.)

4. Текстовый редактор

В состав системы RPS входит *текстовый редактор RPSED*, который предназначен как для создания и редактирования текстовых файлов, так и для выполнения заданий.

Текстовый редактор можно вызвать двумя способами:

1. Дважды кликнуть левой кнопкой "мыши" файл, имеющий тип `.rps`. В этом случае соответствующий файл будет автоматически открыт в окне текстового редактора.

2. Использовать ссылку (shortcut) на программу текстового редактора. В этом случае выбор или создание редактируемого файла выполняется средствами текстового редактора после его вызова.

4.1. Редактирование текстовых файлов

Текстовый редактор RPSED имеет интуитивно понятный графический интерфейс и обеспечивает выполнение тех же основных функций, что и большинство других текстовых редакторов (как, например, редактор Notepad, входящий в состав операционной системы MS Windows).

При задании имен текстовых файлов рекомендуется использовать следующие суффиксы (типы) в зависимости от содержимого файла:

Суффикс	Содержимое текстового файла
.rps	задание для системы RPS
.f	программа на языке Фортран
.c	программа на языке Си
.h	хедер (файл заголовка) на языке Си
.txt	простой текстовый файл
.dat	файл с исходными данными
.lst	листинг задания

4.2. Выполнение заданий

Текстовый файл, содержащий задание для системы RPS, должен иметь суффикс (тип) `.rps`.

Чтобы выполнить задание, необходимо открыть (выбрать) соответствующий файл в окне текстового редактора и выбрать пункт 'File/Run' в главном меню или нажать кнопку 'Run' (рис. 2).

В процессе выполнения задания под управлением системы RPS его листинг отображается в окне вывода текстового редактора (рис. 3).

Если необходимо, то для принудительного прекращения выполнения задания (например, в случае зацикливания программы) следует использовать комбинацию клавиш Ctrl+Break.

4.3. Локализация ошибок

При обнаружении ошибок в управляющих операторах или исходном коде программ в листинг задания записываются диагностические сообщения, которые имеют следующий формат:

```
file:nnn: текст сообщения
```

где `file` — имя файла, к которому относится сообщение, `nnn` — порядковый номер строчки в файле (рис. 4). Для быстрого перехода к соответствующей строчке исходного файла нужно дважды щелкнуть левой кнопкой "мыши" по строчке с текстом сообщения об ошибке в области вывода текстового редактора и после появления окна диалога нажать кнопку "Jump to" (рис. 5).

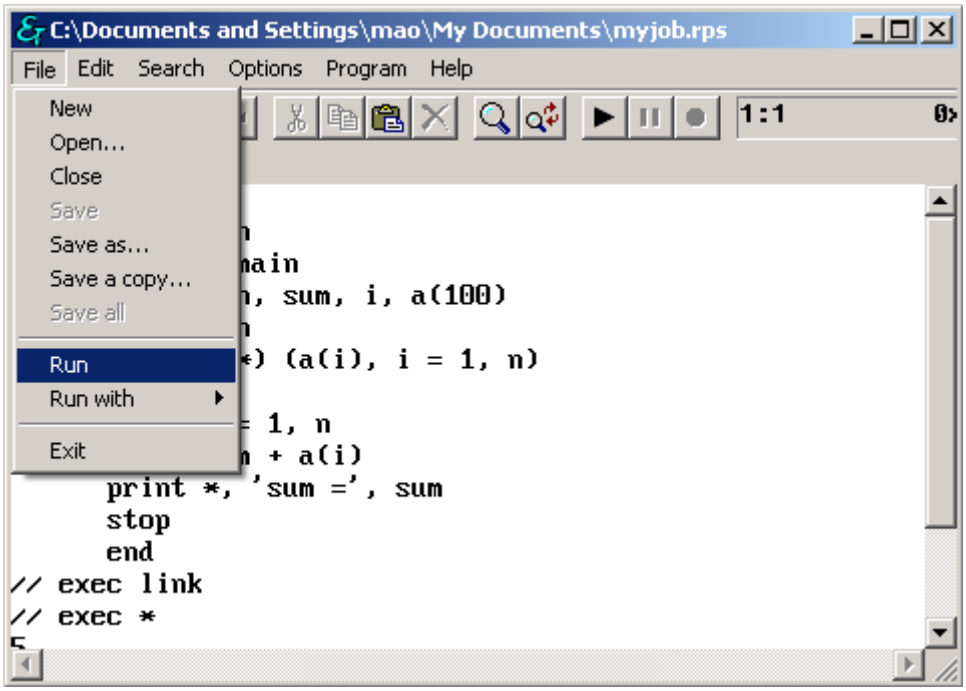

Рис. 2. Запуск задания при работе с текстовым редактором.

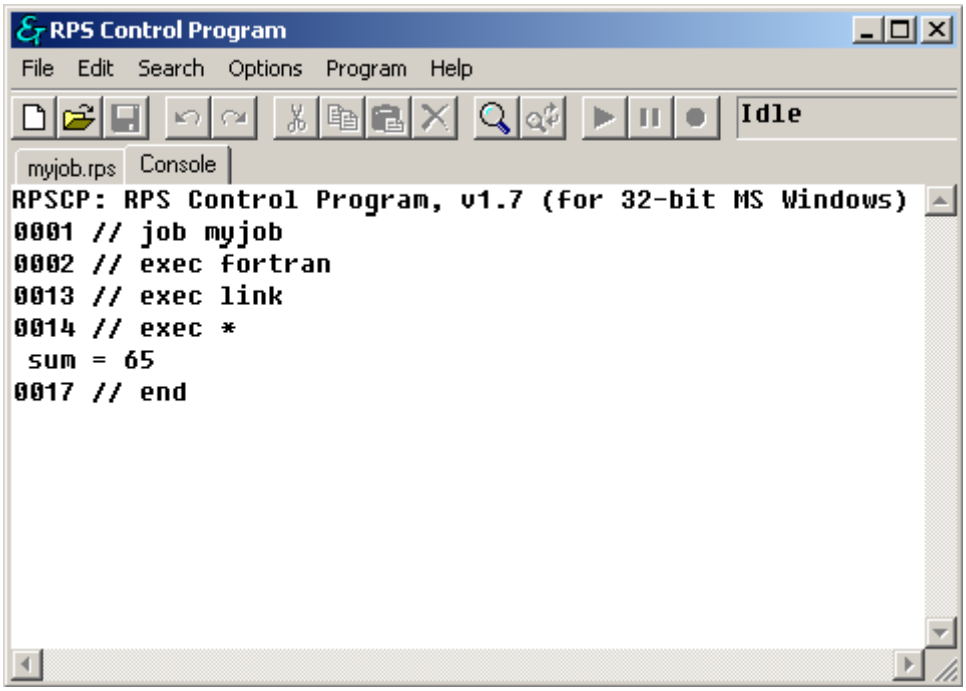

Рис. 3. Листинг задания в окне вывода текстового редактора.

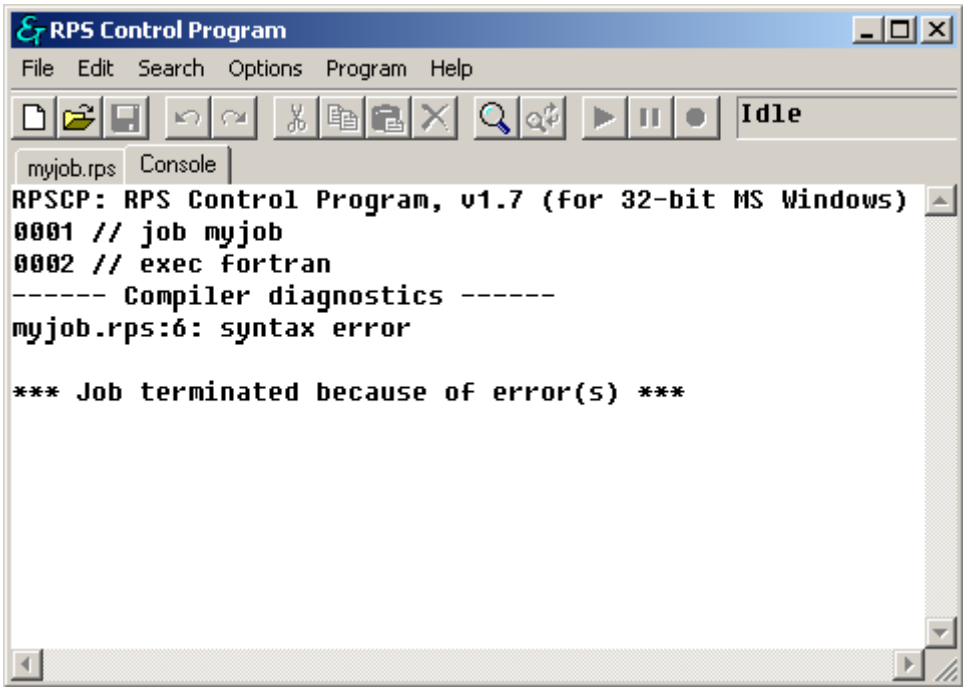

Рис. 4. Листинг задания, содержащий сообщение об ошибке.

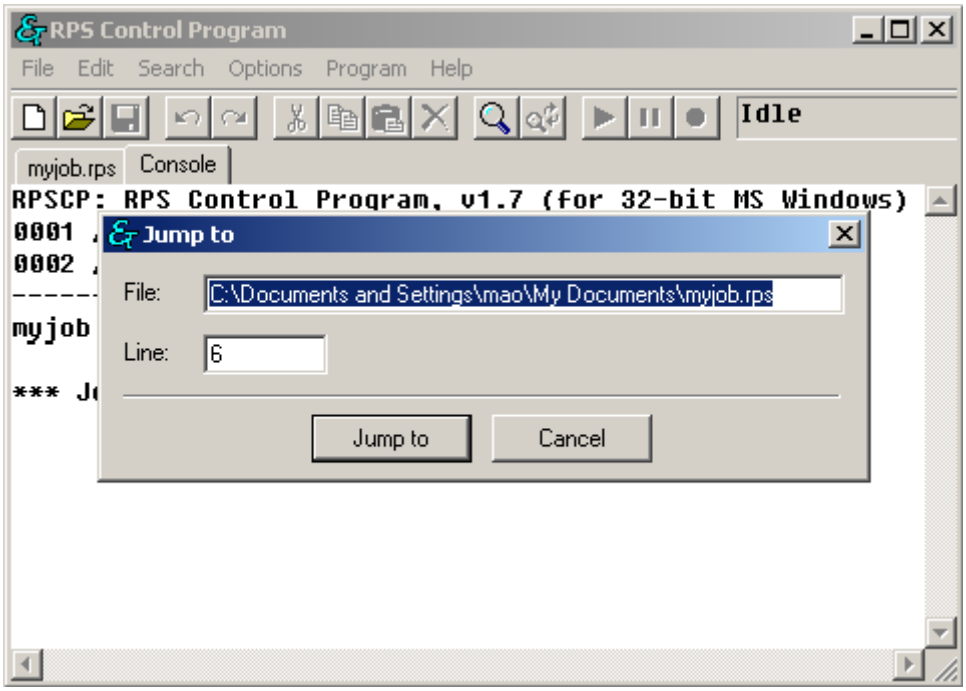

Рис. 5. Быстрый переход к строчке, содержащей ошибку.

5. Примеры типовых заданий для программ на Фортране

Пример 1. Следующее задание включает компиляцию, сборку и выполнение программы, состоящей из одной программной единицы.

```
// job f1
// exec fortran
      program main
      print *, 'Hello, world!'
      stop
      end
// exec link
// exec *
// end
```

Пример 2. Если программа состоит из нескольких программных единиц (подпрограмм), их можно компилировать либо совместно (в одном шаге задания):

```
// job f2a
// exec fortran
      program main
      call sum(3, 4, k)
      end
      subroutine sum(i, j, k)
      k = i + j
      return
      end
// exec link
// exec *
// end
```

либо раздельно (в нескольких шагах задания):

```
// job f2b
// exec fortran
      program main
      call sum(3, 4, k)
      end
// exec fortran
      subroutine sum(i, j, k)
      k = i + j
      return
      end
// exec link
// exec *
// end
```

Раздельную компиляцию удобно применять в тех случаях, когда в программе используются уже отлаженные и (или) библиотечные подпрограммы, которые находятся в отдельных файлах. Так, если исходный код подпрограммы `sum` находится в файле `sum.f`, задание может быть следующим:

```
// job f2c
// exec fortran
      program main
      call sum(3, 4, k)
      end
// exec fortran sum.f
// exec link
// exec *
// end
```

Пример 3. Если для чтения исходных данных программа использует стандартный ввод, то эти исходные данные можно поместить либо непосредственно после управляющего оператора exec:

```
// job f3a
// exec fortran
      program main
      read *, i, j
      print *, 'i=', i, ' j=', j, ' i+j=', i+j
      stop
      end
// exec link
// exec *
3 4
// end
```

либо в отдельном текстовом файле:

```
// job f3b
// exec fortran
      program main
      read *, i, j
      print *, 'i=', i, ' j=', j, ' i+j=', i+j
      stop
      end
// exec link
// assgn stdin f3b.dat
// exec *
// end
```

В последнем случае имя файла, содержащего исходные данные (например, f3b.dat), указывается в управляющем операторе assgn stdin (ключевое слово assgn пишется без буквы "i"), который должен непосредственно предшествовать соответствующему оператору exec.

6. Программирование на языке Си

Составление и выполнение заданий при программировании на языке Си осуществляется аналогично тому, как это было объяснено выше для Фортрана. Единственное отличие состоит в том, что для вызова компилятора Си следует использовать управляющий оператор exec c.

Пример. Следующее задание включает компиляцию, сборку и выполнение программы на языке Си (эта программа эквивалентна Фортран-программе из примера разд. 3):

```
// job sample
// exec c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{     int n, sum, i, a;
      scanf("%d", &n);
      sum = 0;
      for (i = 1; i <= n; i++)
      {  scanf("%d", &a);
         sum += a;
      }
      printf("sum = %d\n", sum);
      return 0;
}
// exec link
// exec *
5
12 23 8 19 3
// end
```