Живехонький Pascal

Рыская по Интернету в поисках подходящих средств обучения, новичок порой натыкается на советы такого рода: «к чему тратить время на «мертвый» Паскаль? Учите C и C++, – на них программируют все!».

Простим этим «советчикам» их заблуждение! Откуда им знать, что в серьезных технических сферах – оборонке, космосе, авиации – для разработки встроенных систем предпочитают надежные языки: Ada, Modula-2, Oberon. Все они – «дети» Паскаля. Но и «папаша» не отстает! На этом универсальном языке можно написать все, что угодно, даже операционную систему!

Встроенные системы

Вокруг полно умных устройств, и мы привыкли жить в их окружении. Мобильники, бытовая техника, автомобили – все это напичкано чипами и хитрыми программами. Встроенные системы – вот общее название этой начинки. К встроенным системам относятся и системы управления самолетами, ракетами, электростанциями, опасными производствами. Такие системы отличает работа в режиме жесткого реального времени, при этом они выполняют одновременно много задач. Так, например, авиационная система управления принимает сигналы сотен датчиков, обрабатывает их и выдает сигналы на органы управления самолетом, обеспечивая при этом достаточно быструю реакцию на события.

Однажды мне случилось создавать небольшую встроенную систему на базе PC-совместимого одноплатного контроллера, – это вроде персонального компьютера величиной с ладошку. В моем распоряжении была только система MS-DOS, – а вы знаете, что она однозадачная. Но в приложении требовалось выполнять несколько параллельных независимых процессов (задач). Тратить время на разработку новой операционки я не стал. Я сделал многозадачную надстройку над MS-DOS, оставив за последней файловые операции. Так появилась программа, с которой вы сейчас ознакомитесь.

Своим инструментом я выбрал Borland Pascal 7.0. Не вдаваясь в подробности проекта, покажу лишь основные идеи, подтверждающие безграничные возможности Паскаля.

Многозадачность

Основное понятие операционных систем реального времени – задача или процесс. Здесь я эти слова использую как синонимы. Так что же такое процесс? Это часть программы, которая выполняется параллельно с другими похожими частями и независимо от них. Наблюдателю порой кажется, что каждый процесс выполняется на отдельном процессоре, но это не так. «Одновременность» эта мнима, и достигается быстрым переключением процессора с одной задачи на другую.

Возьмем тот же самолет, и рассмотрим несколько задач его системы управления. Одна из них – поддержание постоянных оборотов двигателя. Компьютер с заданной периодичностью (например, через 0,1 секунды) опрашивает датчики двигателя и формирует сигнал управления на его топливную систему. Если на самолете четыре двигателя, то в управляющей программе работают четыре таких процесса.

Другая задача – поддержание нужной высоты, скорости и курса. Этот процесс опрашивает (с другой периодичностью) свои датчики и формирует сигналы, управляющие рулями направления, высоты и элеронами. Наконец, третья задача отвечает за поддержание климата в салоне, – здесь работает свой циклический алгоритм. Как видите, эти процессы никак не связаны между собой. Или мало связаны. Так, например, процессу управления высотой и скоростью может потребоваться изменить обороты двигателей, и тогда он отправит сообщения об этом процессам управления двигателями. Но не будет напрямую вмешиваться в работу этих процессов. Так осуществляется межзадачное взаимодействие в системах.

На Паскале типовой процесс можно выразить следующей процедурой:

```
Procedure Process_A;

begin

 Инициализация_Цикла;

 repeat

      сделать_что-нибудь_полезное;

      выдержать_паузу или дождаться_события;

 until false;

end;
```

Таким образом, процесс – это бесконечный цикл. На базе одной такой процедуры можно создать несколько процессов, работающих одновременно сходным образом (вспомните об управлении двигателями). В демонстрационном примере вы увидите это.

Демонстрационный пример

Основной модуль, поддерживающий многозадачность моей программы, назван Figaro (почему? догадайтесь сами). В нем сосредоточены функции по управлению процессами, здесь же обрабатываются прерывания, и выполняется управление часами.

Для тестирования, исследования и демонстрации многозадачности я разработал программу на базе библиотеки Turbo Vision (модули FigaroTV, FigaroIO, FigaroDG). В результате получилось нечто похожее на MS-Windows – каждый процесс программы выполняется в отдельном окне. Есть даже свой диспетчер задач!

В прилагаемом архиве находятся два исполняемых файла:

- Demo_Win.exe – для исполнения под MS-Windows;
- Demo_Dos.exe – для исполнения под MS-DOS.

Первый из них приспособлен для работы в Windows, второй – в настоящей MS-DOS.

В архиве исходников вы найдете все файлы проекта. Головной файл проекта – Demo.pas. Для компиляции вам нужно либо сделать его текущим, либо установить в

качестве первичного через пункт меню «Compile → Primary File». Вдобавок перед компиляцией надо сбросить флаги компилятора:

```
[ ]  Stack checking

[ ]  Overflow checking
```

Вы можете откомпилировать демонстрационный пример и сразу запустить его.

Появится знакомое окно, похожее на окно IDE Borland Pascal. В нижней строке (строке статуса) отображаются:

- Mem = NNNNN – объем свободной памяти в куче;
- Max = NNNNN – размер максимального блока в куче;
- Scan: NNN – частота переключения процессов (1/c);
- Real: NNN – частота прерываний таймера (1/c);

Через пункт меню «Process» можно запустить несколько процессов. Так, на рисунке 1 показаны шесть запущенных процессов: три терминальных и три процесса «About». Каждый терминальный процесс периодически выводит в окно свое имя и время, прошедшее с начала работы программы. В окнах «About» также периодически отображается информация об авторе программы.

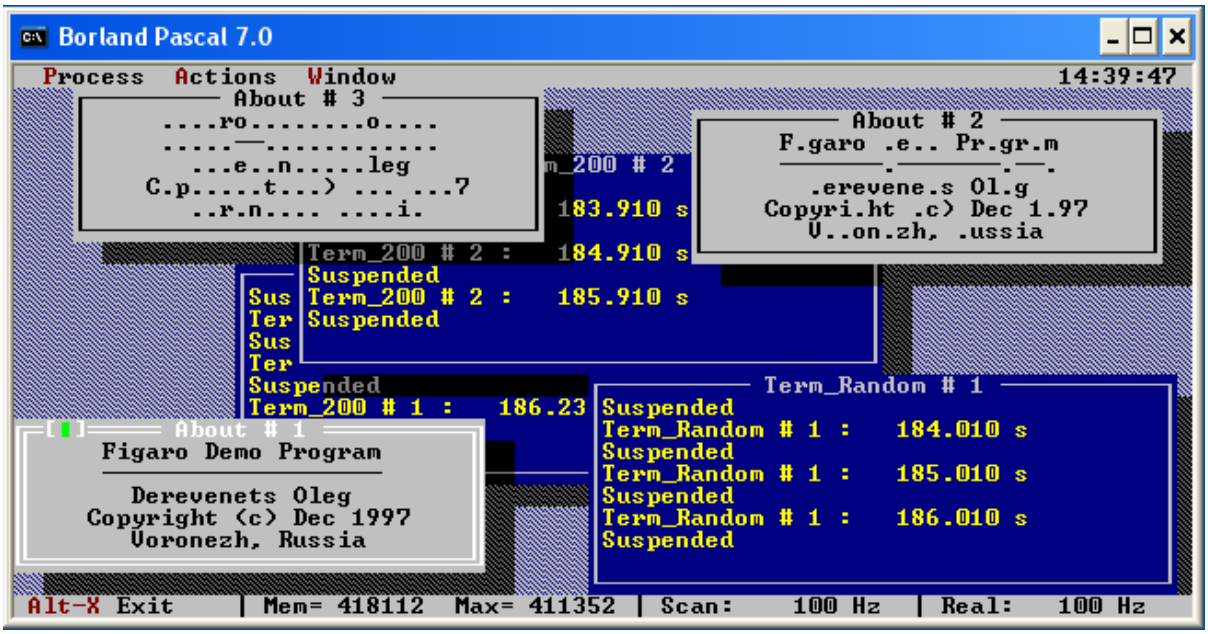

Рисунок 1 - Окна шести процессов

Процесс System_Table (рисунок 2) подобен диспетчеру задач в MS-Windows, он запускается при старте программы (и через меню). Здесь можно наблюдать все выполняемые в программе процессы (включая и сам System_Table). Процессы с именами «...Mouse» и «...KeyBoard» обеспечивают обработку событий от мыши и клавиатуры.

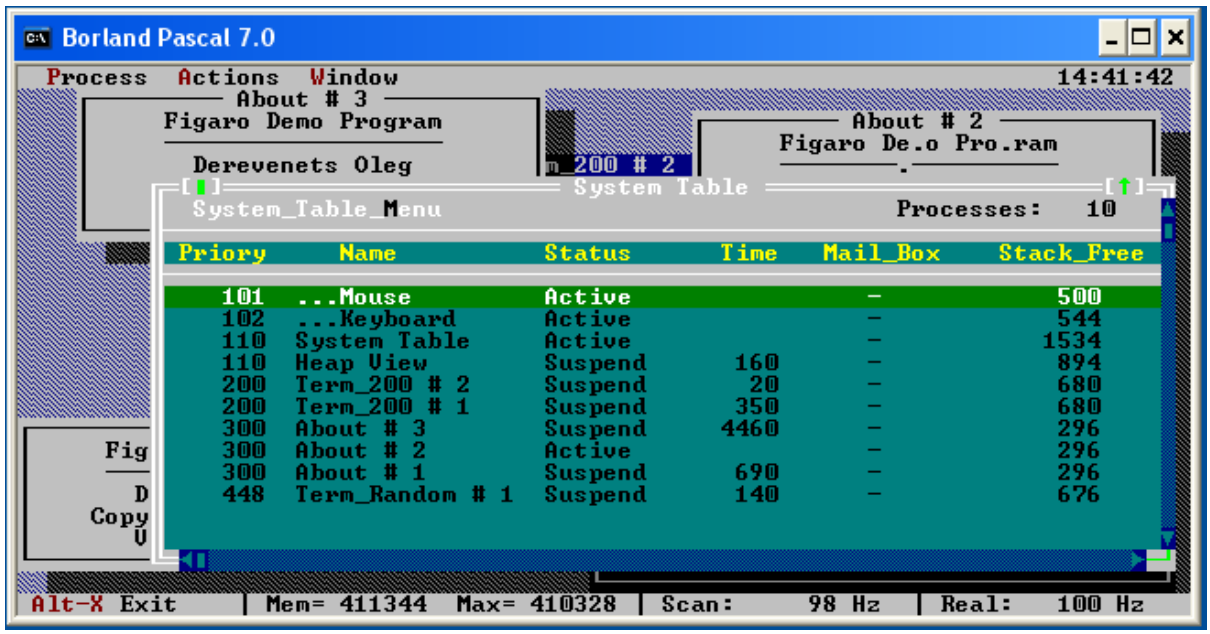

Рисунок 2 – Диспетчер задач «System Table»

Еще два процесса – «Text Copy» и «Stream Copy» – показывают возможности параллельной обработки файлов. При запуске этих процессов вам придется задать исходный и конечный файлы для копирования. В качестве конечного файла можно использовать NUL-устройство (дабы не засорять винчестер).

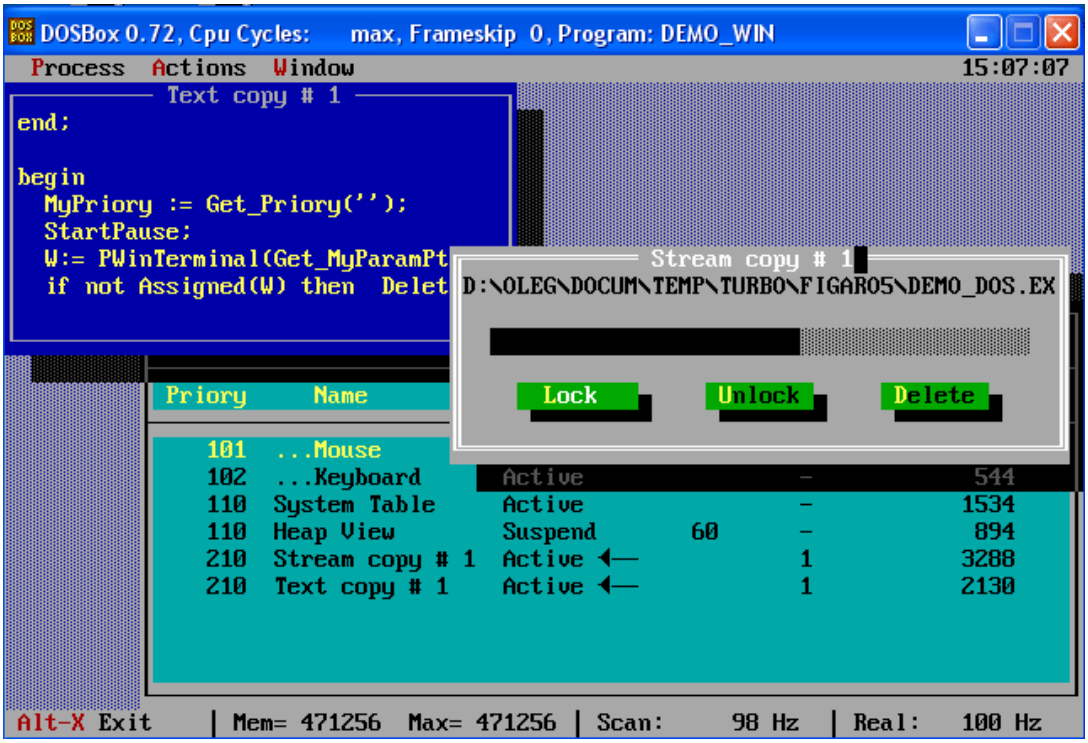

Рисунок 3 - Процессы копирования файлов

Разумеется, если в распоряжении программиста есть подходящая многозадачная операционная система, он не будет «изобретать велосипед». Этим примером я лишь показываю вам: тому, кто владеет Паскалем, нет никаких преград!