МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

Д.В. Груздев

ОПЕРАЦИОННЫЕ СИСТЕМЫ (2-3 курс)

Учебное пособие

Воронеж
Издательский дом ВГУ
2017

«Операционные системы: практика».

Системное программное обеспечение означает программы и комплексы программ, являющиеся общими для всех, кто совместно использует технические средства компьютера, и применяемые как для автоматизации разработки (создания) новых программ, так и для организации выполнения программ существующих. С этих позиций системное программное обеспечение может быть разделено на следующие пять групп:

1. Операционные системы,
2. Системы управления файлами.
3. Интерфейсные оболочки для взаимодействия пользователя с ОС и программные среды.
4. Системы программирования.
5. Утилиты.

1. Под *операционной системой* (ОС) обычно понимают комплекс управляющих и обрабатывающих программ, который, с одной стороны, выступает как интерфейс между аппаратурой компьютера и пользователем с его задачами, а с другой — предназначен для наиболее эффективного использования ресурсов вычислительной системы и организации надежных вычислений. Любой из компонентов прикладного программного обеспечения обязательно работает под управлением ОС. На рис. 1 изображена обобщенная структура программного обеспечения вычислительной системы.

Видно, что ни один из компонентов программного обеспечения, за исключением самой ОС, не имеет непосредственного доступа к аппаратуре компьютера. Даже пользователи взаимодействуют со своими программами через интерфейс ОС. Любые их команды, прежде чем попасть в прикладную программу, сначала проходят через ОС.

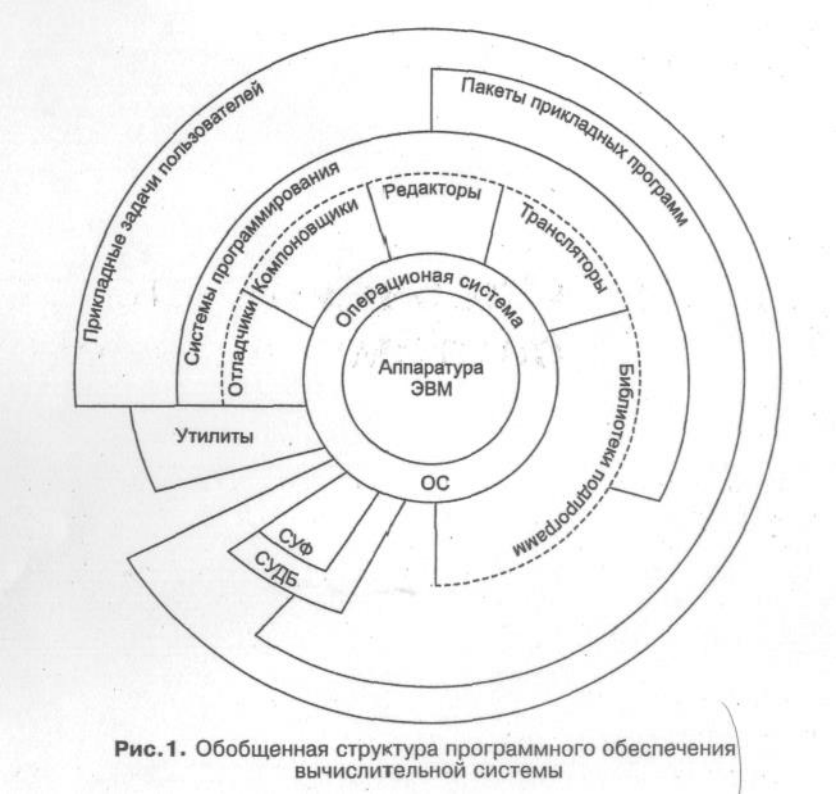


Рис.1. Обобщенная структура программного обеспечения вычислительной системы

Основными функциями, которые выполняет ОС, являются следующие:

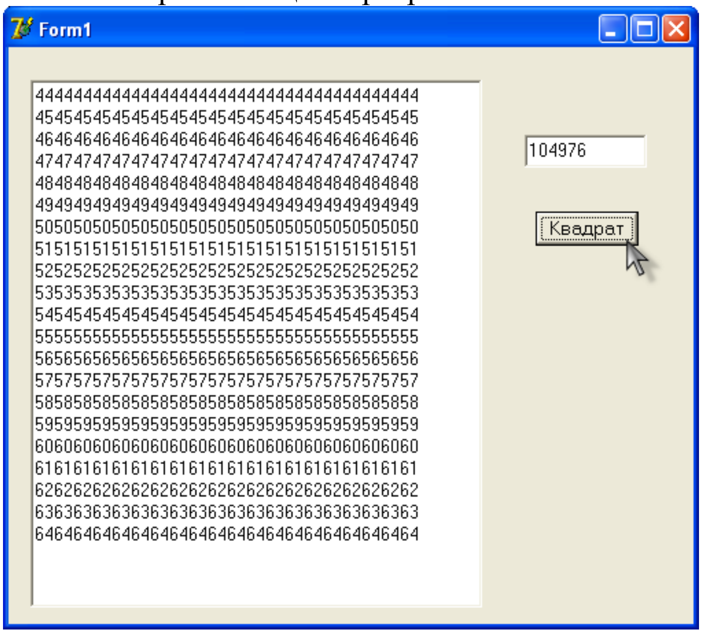
- прием от пользователя (или от оператора системы) заданий или команд, сформулированных на соответствующем языке — в виде директив (команд) оператора или в виде указаний (своеобразных команд) с помощью соответствующего манипулятора (например, с помощью мыши), — и их обработка;
- прием и исполнение программных запросов на запуск, приостановку, остановку других программ;
- загрузка в оперативную память подлежащих исполнению программ;
- инициация программы (передача ей управления, в результате чего процессор исполняет программу);

```
  end;
```

При написании процедуры Execute этот метод можно вызвать следующим образом:

```
    Synchronize(UpdateCaption);
```

Задача. Для иллюстрации приёмов работы с потоками создадим программу, которая будет непрерывно обновлять содержимое поля Memo1 и при этом осуществлять математические вычисления. Окно работающей программы:



Основной поток программы активизируется при щелчке на кнопке **Квадрат**: вначале содержимое расположенного над ней поля Edit1 возводится в квадрат до тех пор, пока отображаемое в нём значение не станет больше 1000000. В этот момент надпись на кнопке меняется на **Корень**, а щелчок на ней вычисляет корень квадратный из величины в поле Edit1.

Дополнительный поток запускается в обработчике события OnActivate главной формы:

```
procedure  TForm1.FormActivate(Sender: TObject);
begin
T1:=MyThread.Create(false)
end;
```

В этом методе объект T1 инициализируется вызовом конструктора MyThread.Create с параметром, который показывает, должен ли вновь созданный поток "спать" (значение true) или он обязан немедленно начать работу (значение false). Программа может в любой момент приостановить работу потока, присвоив его свойству Suspended значение true, и продолжить его выполнение, присвоив этому свойству значение false.

В дополнительном потоке будем непрерывно формировать по 50 строк в поле Memo1 (каждая строка состоит из 20 одинаковых чисел от 0 до 99). Для предотвращения переполнения внутреннего буфера поля Memo оно периодически очищается методом Clear. Чтобы цифры не мелькали слишком быстро, используется процедура sleep(t), задерживающая выполнение программы на t миллисекунд.

Текст основного модуля:

```
unit  Unit1;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Memo1: TMemo;
```

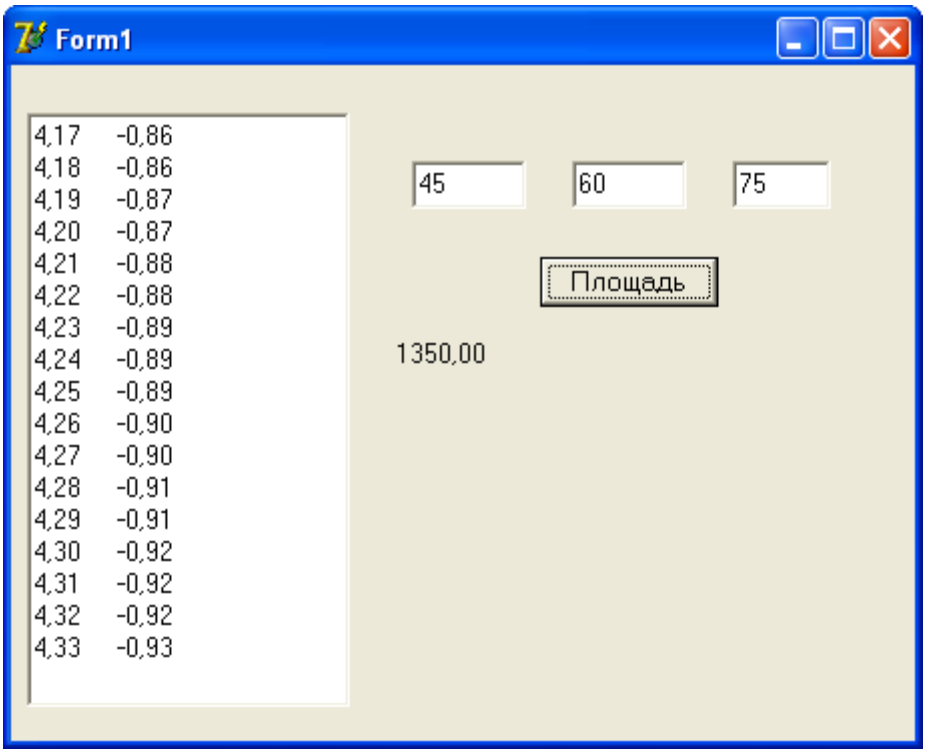
```
    procedure  Button1Click(Sender: TObject);
    procedure  FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var  Form1: TForm1;
implementation
uses unit2;
{$R *.dfm}
procedure   TForm1.Button1Click(Sender: TObject);
begin
if   tag=0 then
 begin
  edit1.Text:=floattostr(sqr(strtofloat(edit1.Text)));
  if  strtofloat(edit1.Text)>1000000  then
  begin   tag:=1;  button1.Caption:='Корень'   end
 end
else
 begin
 edit1.Text:=floattostr(sqrt(strtofloat(edit1.Text)));
 if   strtofloat(edit1.Text)<2   then
 begin   tag:=0;  button1.Caption:='Квадрат'   end
 end
end;
procedure   TForm1.FormActivate(Sender: TObject);
begin
T1:=MyThread.Create(false)
end;
end.
```

Модуль потока:

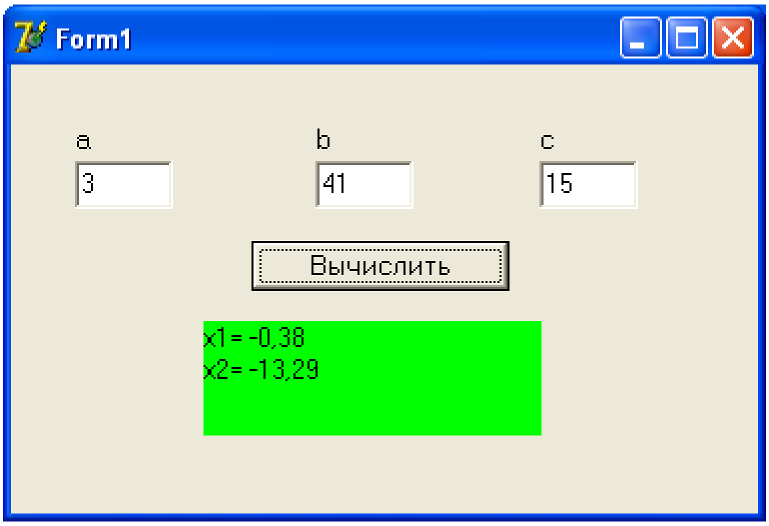
```
unit  Unit2;
interface
uses  Classes;
type
  MyThread = class(TThread)
  private
    { Private declarations }
  protected
    s:string;
    procedure  UpdateMemo;
    procedure  Execute; override;
  end;
var   T1 : MyThread;
implementation
uses   unit1, SysUtils;
procedure  MyThread.Execute;
var   k, j : integer;
begin
repeat
for   k:=0   to   99   do
begin
```

```
s:='';
for  j:=1  to  20  do
s:=s + inttostr(k);  sleep(300);
synchronize(UpdateMemo);
end;
until  false;
end;
procedure  MyThread.UpdateMemo;
begin
with  form1.memo1.Lines  do
if  count>50 then  clear  else  add(s)
end;
end.
```

Задача1. Создать приложение, основной поток которого осуществляет вычисление площади треугольника по трём данным сторонам, а дополнительный поток непрерывно заполняет поле Memo1 значениями синуса угла (данного в радианах) с шагом 0.1 радиан. (В первой колонке значения угла в радианах, во второй – синус этого угла).



Задача2. Создать приложение, основной поток которого осуществляет поиск корней квадратного уравнения $ax^2 + bx + c = 0$, а дополнительный поток
непрерывно меняет цвет метки label1 с зелёного на красный и обратно.



Файловая система.

/bin
Основные программы, необходимые для работы в системе: командные оболочки, файловые утилиты и т.п.
/sbin
Команды для системного администрирования, а также программы, выполняемые в ходе загрузки
/boot
Файлы, необходимые для загрузки системы (образ ядра)
/home
Домашние каталоги пользователей, кроме root
/dev
Файлы устройств
/etc
Файлы настроек: стартовые сценарии, конфигурационные файлы графической системы и различных приложений
/lib
Системные библиотеки, необходимые для основных программ, и модули ядра
/lost+found
Восстановленные после аварийного размонтирования части файловой системы
/media
Сюда обычно монтируются съемные носители: компакт-диски, flash-накопители
/mnt
Временные точки монтирования жестких дисков. Использовать этот каталог необязательно: подмонтировать файловую систему можно к любому другому каталогу
/opt
Дополнительные пакеты программ. Если программа, установленная сюда, больше не нужна, то достаточно удалить ее каталог без обычной процедуры деинсталляции
/proc
Виртуальная файловая система, дающая доступ к информации ядра (например, выведите на экран файл /proc/cpuinfo). Другие файлы в этом каталоге в каждый момент времени содержат информацию о выполняющихся в этот момент программах
/root
Домашний каталог суперпользователя. Домашние каталоги всех остальных могут находиться на отдельном разделе, но /root должен быть в корневой файловой системе, чтобы администратор всегда мог войти в систему для ремонтных работ

/tmp
Временные файлы
/var
Часто меняющиеся данные: системные журналы и протоколы приложений, замки, почтовые ящики, очереди печати и т.п.
/usr Практически все остальное: программы, исходные коды, документация. Сюда по умолчанию устанавливаются новые программы

В ОС UNIX поддерживается три способа указания имен файлов:

- *Краткое имя*. Имя, не содержащее специальных метасимволов косая черта (/), является кратким именем файла. По краткому имени можно сослаться на файлы текущего каталога. Например, команда **ls -l .profile** требует получить полную информацию о файле **.profile** в текущем каталоге.
- *Относительное имя*. Имя, не начинающееся с символа косой черты (/), но включающее такие символы. Оно ссылается на файл относительно текущего каталога. При этом для ссылки на файл или каталог в каком-то другом каталоге используется метасимвол косой черты (/). Например, команда **ls -l ../.profile** требует получить полную информацию о файле **.profile** в родительском каталоге текущего каталога, а команда **vi doc/text.txt** требует открыть в редакторе **vi** файл **text.txt** в подкаталоге **doc** текущего каталога.
- *Полное имя*. Имя, начинающееся с символа косой черты (/). Оно ссылается на файл относительно корневого каталога. Это имя еще называют *абсолютным*, так как оно, в отличие от предыдущих способов задания имени, ссылается на один и тот же файл независимо от текущего каталога. Например, команда **ls -l /home/user01/.profile** требует получить полную информацию о файле **.profile** в каталоге **/home/user01** независимо от того, в каком каталоге выполняется.

Получение информации о текущем каталоге
Команда **pwd** выдает полное имя *текущего* (*рабочего*) каталога. Команда **pwd** не имеет параметров. Вот пример ее использования:
$ pwd
/home/user01
$
Изменение текущего каталога
Для изменения текущего каталога используется команда **cd**:
cd [**каталог**]
Если **каталог** не указан, используется значение переменной среды **$HOME** (обычно это *начальный каталог* пользователя). Чтобы сделать новый каталог текущим (войти в каталог), нужно иметь для него **право на выполнение**. Команда **cd** является встроенной командой интерпретатора и использует для изменения текущего каталога соответствующий системный вызов.
Рассмотрим пример совместного использования команд **cd** и **pwd** для переходов по каталогам файловой системы:
$ pwd
/home/user01
$ cd ..
$ pwd
/home
$ cd user01/tmp
$ pwd
/home/user/tmp
$ cd
$ pwd
/home/user01
Получение информации о файлах
Для просмотра информации о типах (и других атрибутах) файлов в ОС UNIX используется команда **ls** со следующим синтаксисом:
ls [-abCcdeFfgiLlmnopqRrstux1] [**файл** ...]