РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ОТКРЫТЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ
МИНИСТЕРСТВА ПУТЕЙ СООБЩЕНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

20/32/2

Одобрено кафедрой
«Вычислительная техника»

ОПЕРАЦИОННЫЕ СИСТЕМЫ

**Задание на контрольную работу №1
для студентов V курса**

специальностей

220100 ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ, КОМПЛЕКСЫ, СИСТЕМЫ И СЕТИ (ЭВМ)

071900 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ (ИСЖ)

**Задание на контрольную работу №2
для студентов V курса
с методическими указаниями**

специальности

220100 ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ, КОМПЛЕКСЫ, СИСТЕМЫ И СЕТИ (ЭВМ)

Москва – 2004

Р е ц е н з е н т — канд. техн. наук, доц. Л.Г. КОПТЕВА

ОБЩИЕ УКАЗАНИЯ

Изучение дисциплины «Операционные системы» студентами-заочниками специальности ИСЖ на V курсе сопровождается выполнением контрольной работы №1, а студентами-заочниками V курса специальности ЭВМ – контрольных работ №1 и №2.

В контрольных работах вариант задачи выбирается по последней цифре учебного шифра.

ЗАДАНИЕ НА КОНТРОЛЬНУЮ РАБОТУ №1

Дать ответ на теоретические вопросы, указанные в следующих вариантах.

Вариант 0

1. Опишите отличия в работе процессора в привилегированном и пользовательском режимах.
2. Выясните, может ли процесс в мультипрограммном режиме выполняться быстрее, чем в монопольном.

Вариант 1

1. Указать этапы, которые включает разработка варианта мобильной ОС для новой аппаратной платформы.
2. Поясните, что такое виртуальная память.

Вариант 2

1. Опишите порядок взаимодействия приложений с ОС, имеющей микроядерную структуру.
2. Укажите, как величина файла подкачки влияет на производительность системы.

Вариант 3

1. Опишите отличия выполнения системного вызова в микроядерной ОС и ОС с монопольным ядром.
2. Поясните, почему размер страницы выбирается равным степени двойки, можно ли принять такое ограничение для сегмента, на что влияет размер страницы, каковы преимущества и недостатки большого размера страниц.

Вариант 4

1. Поясните определение операционной системы как расширенной машины.
2. Укажите, какие характеристики содержит таблица сегментов и таблица страниц при сегментно-страничной организации памяти.

Вариант 5

1. Назовите абстрактно сформулированные задачи ОС по управлению любыми типами ресурса. Конкретизируйте эти задачи применительно к процессору памяти, внешним устройствам.
2. Поясните, как обеспечивается согласование данных в кэше с помощью методов обратной и сквозной записи.

Вариант 6

1. Поясните, может ли процесс в мультипрограммном режиме выполняться быстрее, чем в монопольном.
2. Укажите, какие функции выполняет менеджер ввода-вывода.

Вариант 7

1. Объясните потенциально более высокую надежность операционных систем, в которых реализована вытесняющая многозадачность.
2. Поясните, какие устройства позволяют распараллелить ввод-вывод даже в однопроцессорных системах.

Вариант 8

1. Опишите средства синхронизации процессов, которыми располагает современная ОС.
2. Укажите, каким из двух типов (блок-ориентированным или байт-ориентированным) драйверов обслуживается диск.

Вариант 9

1. Опишите механизм обработки прерываний в Windows NT.
2. Укажите, какие события вызывают перепланирование процессов (потоков).

ЗАДАНИЕ НА КОНТРОЛЬНУЮ РАБОТУ №2

Разработать программу, позволяющую осуществить запуск на исполнение и закрытие внешнего дочернего процесса. После создания процесса вывести значения идентификаторов процесса в соответствии с вариантом задания, который представлен в Таблице вариантов и определяется по последней цифре учебного шифра студента.

В контрольной работе привести задание, таблицу идентификаторов, окно программы в стадии выполнения с информацией об идентификаторах, листинг программы.

Таблица вариантов

вариант	Внешнее приложение	Действие, выполняемое при запуске приложения	Идентификаторы потока	Приоритет процесса	Способ закрытия приложения
0	MS Word	Открытие заданного документа Word	Дескриптор созданного процесса, глобальный идентификатор процесса	Выполняется во время простоя системы	Сообщение WM_CLOSE
1	MathCad (или любой другой математический пакет)	Открытие заданного документа MathCad	Дескриптор первого потока, глобальный идентификатор потока	Высокий	Сообщение WM_CLOSE
2	Просмотрщик презентаций MS Power Point	Запуск на просмотр заданной презентации	Дескриптор первого потока, глобальный идентификатор процесса	Нормальный	Функция TerminatePr ocess
3	Delphi (или другая используемая среда программирования)	Открытие заданного проекта Delphi	Дескриптор созданного процесса, глобальный идентификатор	Нормальный	Сообщение WM_CLOSE
4	Microsoft Access	Открытие заданной базы данных.	Дескриптор созданного процесса, дескриптор первого потока	Критическая задача	Функция Terminate-Process

Окончание табл.

5	Windows Media Player (или любой другой media player)	Просмотр заданного видео-файла	Дескриптор созданного процесса, глобальный идентификатор процесса	Высокий	Функция Terminate-Process
6	MS Excel	Открытие заданного документа Excel	Дескриптор первого потока, глобальный идентификатор потока	Выполняется во время простоя системы	Сообщение WM_CLOSE
7	MS Internet Explorer	Выход на заданный ресурс Internet (или локальный Web-ресурс)	Дескриптор первого потока, глобальный идентификатор процесса	Нормальный	Функция Terminate-Process
8	Paint (или любой другой графический редактор)	Открытие для редактирования графического файла	Дескриптор созданного процесса, глобальный идентификатор потока	Высокий	Сообщение WM_CLOSE
9	Kaspersky Anti-Virus Scaner (или любой другой антивирусный сканер)	Проверка дискеты на наличие вируса	Дескриптор созданного процесса, дескриптор первого потока	Критическая задача	Функция Terminate-Process

Методические указания к решению задачи

Задача выполняется в среде программирования Delphi 7.

Для создания нового процесса используется функция *CreateProcessA*.

```
function CreateProcessA(lpApplicationName: PAnsiChar; lpCommandLine: PAnsiChar; lpProcessAttributes, lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL; dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PAnsiChar; const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL
```


Функция порождает новый дочерний процесс и его первый поток (нить). В рамках этого процесса выполняется указанный файл **lpApplicationName** скомандной строкой **lpCommandLine**. Параметр **lpApplicationName** может быть равен **nil**, а имя выполняемого модуля в этом случае должно быть первым элементом командной строки, задаваемой параметром **lpCommandLine**. Сам выполняемый модуль может быть любого вида: 32-разрядным приложением Windows, приложением MS-DOS, OS/2 и т.п.

Однако, если из приложения Windows создается процесс MS-DOS, то параметр **lpApplicationName** должен быть равен **nil**, а имя файла и его командная строка включаются в **lpCommandLine**.

Поэтому, чтобы не ошибиться, проще всегда задавать **lpApplicationName = nil**, и помещать всю информацию в **lpCommandLine**.

Рассмотрим некоторые параметры функции (более подробную информацию можно получить в справке Delphi).

Параметры **lpProcessAttributes**, **lpThreadAttributes**, **lpEnvironment**, **bInhe-ritHandles** определяют наследование дочерним процессом свойств родительского процесса. По умолчанию, можно первые три из этих параметров задавать равными **nil**, а последний — **false**.

Параметр **lpCurrent-Directory** указывает на строку, определяющую текущий каталог и диск дочернего процесса. Это используется в приложениях-оболочках, выполняющих различные приложения с различными рабочими каталогами. Если параметр равен **nil**, текущий каталог совпадает с родительским.

Параметр **dwCreationFlags** определяет флаги, задающие характеристики создаваемого процесса. Эти флаги определяют тип процесса (например, **CREATE_NEW_CONSOLE** — создание нового консольного приложения), характер взаимоотношения с родительским процессом и класс приоритета нового процесса:

HIGH_PRIORITY_CLASS	Указывает на процесс как на критическую задачу, которая должна выполняться немедленно
IDLE_PRIORITY_CLASS	Все потоки процесса выполняются только во время простоя системы. Пример — хранители экрана. Все наследники такого процесса будут иметь тот же класс приоритета
NORMAL_PRIORITY_CLASS	Нормальный приоритет процесса
REALTIME_PRIORITY_CLASS	Высокий приоритет, превышающий приоритеты других процессов, включая приоритеты процессов операционной системы

Параметр **lpStartupInfo** указывает на структуру типа **TStartupInfo**, определяющую основное окно дочернего процесса. Из всех полей этой структуры обязательным для заполнения является только **cb** — размер в байтах данной структуры. Остальные можно не заполнять, что обеспечит вид окна по умолчанию.

Параметр **lpProcessInformation** указывает на структуру **TProcessInformation**, из которой родительское приложение может получать информацию о выполнении нового процесса. Ее поля обозначают следующее:

hProcess	Возвращает дескриптор созданного процесса. Используется во всех функциях, осуществляющих операции с объектом процесса
hThread	Возвращает дескриптор первого потока (нити) созданного процесса. Используется во всех функциях, осуществляющих операции с объектом потока
dwProcessId	Возвращает глобальный идентификатор процесса. Значение доступно с момента создания процесса и до момента его завершения
dwThreadId	Возвращает глобальный идентификатор потока. Значение доступно с момента создания потока и до момента его завершения

Если функция **CreateProcess** успешно выполнена, она возвращает ненулевое значение (**true**). Если произошла ошибка — возвращается 0 (**false**). Тогда информацию об ошибке можно получить, вызвав функцию **GetLastError**.

Закрывает процесс и все его потоки функция **ExitProcess.** Однако эта функция может закрыть процесс, если она вызывается из потока, созданного этим процессом. В более общем случае для немедленного завершения работы внешнего процесса используется функция **TerminateProcess**.

```
function TerminateProcess(hProcess: THandle; uExitCode: UINT): BOOL;
```

Параметр **hProcess** - дескриптор завершамого процесса, его значение можно взять из одноименного поля структуры **TProcessInformation** , которая храниться в процессе-родителе.

Параметр **uExitCode** содержит код завершения процесса. Для его определения используется функция **GetExitCodeProcess.**

```
function GetExitCodeProcess(hProcess: THandle; var lpExitCode: DWORD): BOOL;
```

Параметр **hProcess** - дескриптор завершамого процесса.

Параметр **lpExitCode** собственно и содержит код завершения процесса.

Типы UNIT и DWORD являются целыми типами Windows, пришедшими из C++, они переопределены в Delphi как тип LongWord, поэтому при написании программы можно в этих функциях использовать переменную знакомого вам типа LongWord.

Замечание. Отметим, что функцией **TerminateProcess** работа приложения завершается немедленно, при этом возможны потери данных. Например, если в окне «Блокнота» вы производили редактирование текста, то после выполнения функции **TerminateProcess** внесенные изменения будут потерены. Корректно завершить работу приложения, как вы уже знаете, можно послав окну приложения сообщение **WM_CLOSE**.

Ниже приводится листинг программы, содержащей необходимые комментарии.

```
{Модуль Project1.dpr}
program Project1;

uses
 Forms,
 Unit1 in “Unit1.pas” {Form1};

{$R *.RES}

begin
 Application.Initialize;
 Application.CreateForm(TForm1, Form1);
 Application.Run;
end.
```

{Модуль Unit1.pas}

{Эта программа дает возможность запустить программу Блокнот с загрузкой в нее заданного файла Zapiska.txt, а затем закрыть эту пр-мму Блокнот. Приведенная программа после своего запуска дает экран формы Form1, а на нем - шесть компонентов, - Edit1, Edit2, Label1, Label2, Button1, Button2.

Здесь Label1 и Label2 служат для подсказок, а в поле Edit1 надо ввести путь к файлу запускаемой программы Блокнот (Notepad.exe), (например, C:\Windows\), а в поле Edit2 надо ввести путь к загружаемому в этот редактор текст.файлу Zapiska.txt (например, C:\My Documents\). После этого щелчком на кнопке Button1 («Запустить Блокнот») запускаем Блокнот с загрузкой в него указанного текстового файла. Обеспечиваем приоритет процесса - «Критическая задача». (Если такого файла по указанному пути не окажется, то программа даст возможность его создать заново) Для закрытия Блокнота следует щелкнуть кнопку Button2 («Завершить Блокнот»). После создания процесса помещаем в поля Edit1 и Edit2 значения дескриптора созданного процесса и глобального идентификатора процесса. Т.е. используем прежние компоненты Edit1, Edit2, а заодно и Label1, Label2 (для подсказки к этим новым значениям полей Label1, Label2). }


```
unit Unit1;

interface

uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
 Forms, Dialogs, StdCtrls; //Не опускать элементы списка

type
 TForm1 = class(TForm)
 Button1: TButton;
 Edit1: TEdit;
 Label1: TLabel;


 Button2: TButton;
 Edit2: TEdit;
 Label2: TLabel;
 procedure Button1Click(Sender: TObject);
 procedure Button2Click(Sender: TObject);
 procedure FormActivate(Sender: TObject);
 private
 { Private declarations }
 public
 { Public declarations }
 end;

var
 Form1: TForm1;
 pInfo: TProcessInformation; //Структура,
 //содержащая информацию о созданном процессе
implementation

{$R *.DFM}

{Обработка события активации Form1 (On Activate)}
procedure TForm1.FormActivate(Sender: TObject);
begin
 Form1.Caption := “Запуск программы Блокнот” +
 “ с загрузкой Zapiska.txt и завершение ее работы”;
```

```
 Label1.Caption := “Введите путь к Notepad.exe”;
 Label2.Caption := “Введите путь к Zapiska.txt”;
 Button1.Caption := “Запустить Блокнот”;
 Button2.Caption := “Завершить Блокнот”;
end;

{Обработка щелчка на кнопке Button1.
Открываем программу Блокнот и загружаем
в нее файл Zapiska.txt. Выводим дескриптор
созданного процесса и глобальный идентификатор
процесса }
procedure TForm1.Button1Click(Sender: TObject);
 var
 SInfo : TStartupInfo; //Структура, заполняемая
 // при создании процесса
begin


 FillChar(SInfo, SizeOf(SInfo), #0);
 //Очистили поля структуры SInfo
 SInfo.cb := SizeOf(SInfo);
 //Заполнили в структуре SInfo поле cb
 //размером этой структуры в байтах
 If Not CreateProcess(nil,
 PChar(Trim(Edit1.Text) + “Notepad.exe” +
 “ “ + Trim(Edit2.Text) + “Zapiska.txt”),
 //Выше стоит командная строка запуска Блокнот с
 //загрузкой заданного текст.файла Zapiska.txt
 nil,
 nil,
 False,
 HIGH_PRIORITY_CLASS,
 // Приоритет процесса - «Критическая задача»
 nil,
 nil,
 SInfo,
 pInfo)
 Then
 //Выдать сообщение об ошибке
 ShowMessage(IntToStr(GetLastError));
```



```
 //Вывести дескриптор созд. процесса
 //и глоб.идентиф-р процесса
 Label1.Caption := “Дескриптор созданн. процесса”;
 Label2.Caption := “Глобальн. идентиф-тор процесса”;
 Edit1.Text := IntToStr(pInfo.hProcess);
 Edit2.Text := IntToStr(pInfo.dwProcessId);
end;

{Обработка щелчка на кнопке Button2. Закрываем
программу Блокнот с помощью функции TerminateProcess.
Если даже текст в редакторе был изменен, то закроет его,
не предложив сохранить внесенные изменения и их
не сохранит}
procedure TForm1.Button2Click(Sender: TObject);
var
 uExitCode: LongWord;
begin
 if not GetExitCodeProcess(pInfo.hProcess, uExitCode)
 //Определили код выхода процесса uExitCode


 then ShowMessage(“Ошибка завершения процесса N” +
 IntToStr(GetLastError));
 TerminateProcess(pInfo.hProcess,uExitCode);
 //Процедура завершила процесс
 end;
```

{ Альтернативный вариант закрытия программы Блокнот с помощью функции PostMessage. Эта функция дает возможность сохранить изменения текста в редакторе. DES - дескриптор окна вызываемой программы, функция FindWindow получает в качестве вх.параметров имя класса окна и имя окна (отображаемое в строке заголовка окна). Для отыскания имени класса окна надо заранее в Windows запустить программу Пуск\Программы\Delphi6(или 7)\WinSight32 и затем выполнить команду Spy\Find Window и найти в появившемся списке строку Overlapped с именем нужного окна, в ней также будет стоять в фигурных скобках и имя класса окна.

```
procedure TForm1.Button2Click(Sender: TObject);
var DES: hWnd;
 begin
 DES:=FindWindow(“Notepad”,’Zapiska.txt - Блокнот’);
 PostMessage(DES,WM_Close,0,0);
 end;
}

end.
```

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. О л и ф е р В.Г., О л и ф е р Н.А. Сетевые операционные системы. —М.: Изд-во «Питер», 2002.

2. А р х а н г е л ь с к и й А.Я. Программирование в Delphi. —М.: «Изд-во Бином», 2003.

канд. физ.-мат. наук, доц. Е.А. НОСИЛОВСКИЙ,
доц. И.М. УКОЛОВ,
канд. техн. наук, доц. В.П. ФОМЧЕНКОВ

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Задания на контрольные работы №1 и №2
с методическими указаниями

Редактор *Г.В. Тимченко*
Компьютерная верстка *Ю.А. Варламова*

Тип. зак.	Изд. зак. 266	Тираж 1000 экз.
Подписано в печать	Гарнитура Times.	Офсет
Усл. печ. л. 0,75		Формат $60\times90^{1}/_{16}$

Издательский центр РГОТУПСа,
125993, Москва, Часовая ул., 22/2

Типография РГОТУПСа, 125993, Москва, Часовая ул., 22/2