

Лекция 1

Содержание

- Цели и задачи курса.
- Краткий обзор курса.
- Структура современной операционной системы.
- Режим ядра и пользовательский режим.
- Системные вызовы. Функции API.
- Windows API. Особенности реализации C/C++ в Microsoft Visual C++.
- Документация MSDN.

Содержание курса

1. Архитектура современных ОС. Структура ОС. Режим ядра и пользовательский режим. Системные вызовы. Интерфейсы прикладного программирования.	1
2. Windows API. Реализация C/C++ Microsoft Visual C++. Объекты ядра. Документация MSDN.	1
3. Модель физической памяти <i>IBM PC</i>. Адресное пространство. Виртуальная память. Страничная организация памяти. Таблица страниц. Преобразование виртуальных адресов в физические адреса. Диспетчер памяти. Свопинг.	2
4. Процессы. Таблица процессов. Создание и завершение процессов. Реализация процессов в <i>MS Windows</i> и <i>GNU/Linux</i> .	4
5. Потоки. Создание и реализация потоков в <i>GNU/Linux</i> и <i>MS Windows</i> . Многопоточные приложения <i>GNU/Linux</i> и <i>MS Windows</i> .	2
6. Архитектура приложения <i>Win32</i>. Оконные приложения. Диалоговые окна.	4
7. Обмен данными между процессами. Библиотеки динамической компоновки. Отображение файлов на память. Анонимные и именованные каналы. Сокеты.	6
8. Синхронизация потоков. Классические задачи синхронизации. Критические области. Семафоры, мьютексы, мониторы. Барьерная синхронизация. Взаимоблокировки. Реализация синхронизации в <i>POSIX</i> и <i>Windows API</i> .	6
9. Файловые системы. Атрибуты файлов. Реализации размещения файлов и доступа к ним. Операции с файлами. Журналируемые файловые системы. Файловые системы <i>MS Windows</i> и <i>GNU/Linux</i> .	2
10. Безопасность. Аутентификация и авторизация пользователей. Типы вирусных атак. Механизмы защиты.	2
11. Дополнительные вопросы. Ловушки <i>MS Windows</i> . Фоновые программы - службы, и их реализация в <i>MS Windows</i> и <i>GNU/Linux</i> . Приложения COM и COM+. Расширения графической оболочки <i>MS Windows</i> .	4

ПОЛЬЗОВАТЕЛИ



ОС

ОБОЛОЧКА

ЯДРО

Управление
процессами

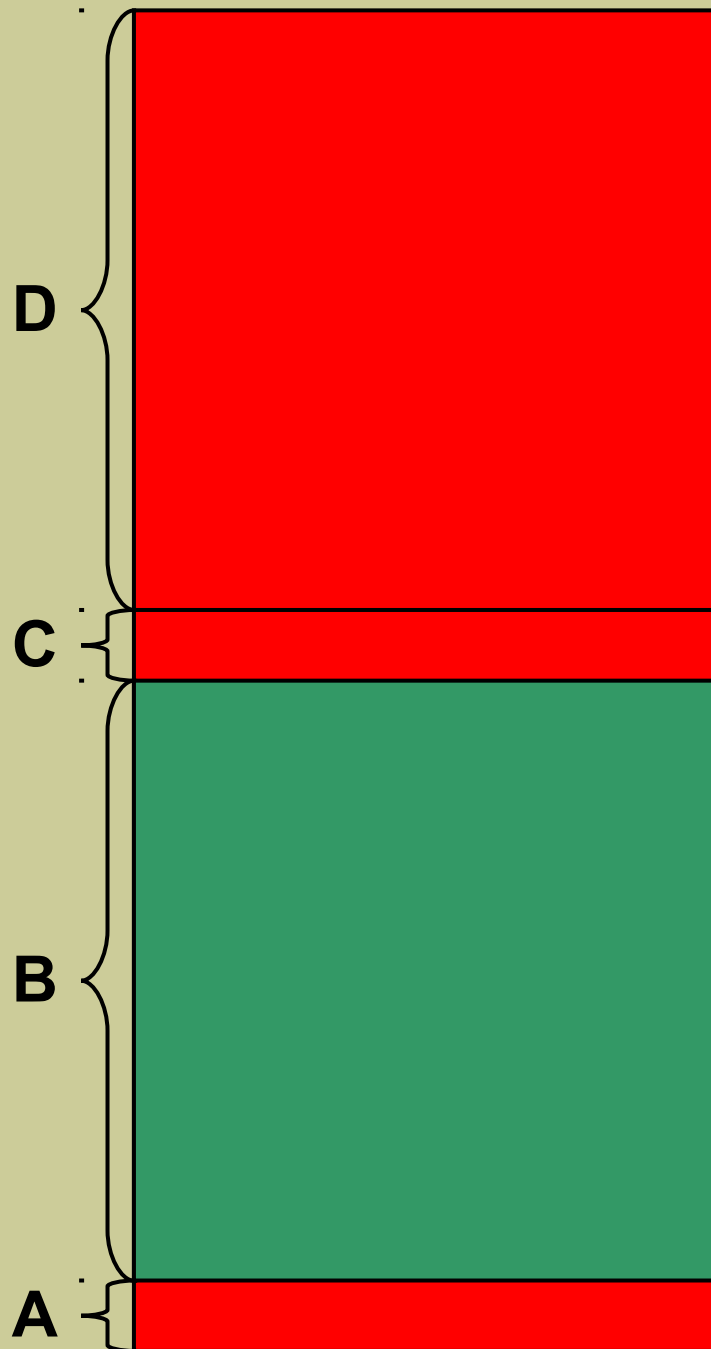
Управление
памятью

Управление
внешними
устройствами



ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

Пример структуры адресного пространства 32-разрядной ОС.



- A. 0x00000000 – 0x0000FFFF; используется для неинициализированных указателей; **недоступно** в пользовательском режиме.
- B. 0x00010000 – 0x7FFEFFFF; адресное пространство процессов, содержит прикладные модули .exe и .dll, win32 (kernel32.dll, user32.dll и т.д.), файлы, отображаемые в память; **доступно** в пользовательском режиме.
- C. 0x7FFF0000 – 0x7FFFFFFF; используется для некорректно инициализированных указателей; **недоступно** в пользовательском режиме.
- D. 0x80000000 – 0xFFFFFFFF; зарезервировано ОС Windows для исполнительной системы, ядра и драйверов устройств; **недоступно** в пользовательском режиме.

Взаимодействие прикладных программ и ОС

Режим ядра (режим супервизора, привилегированный режим):

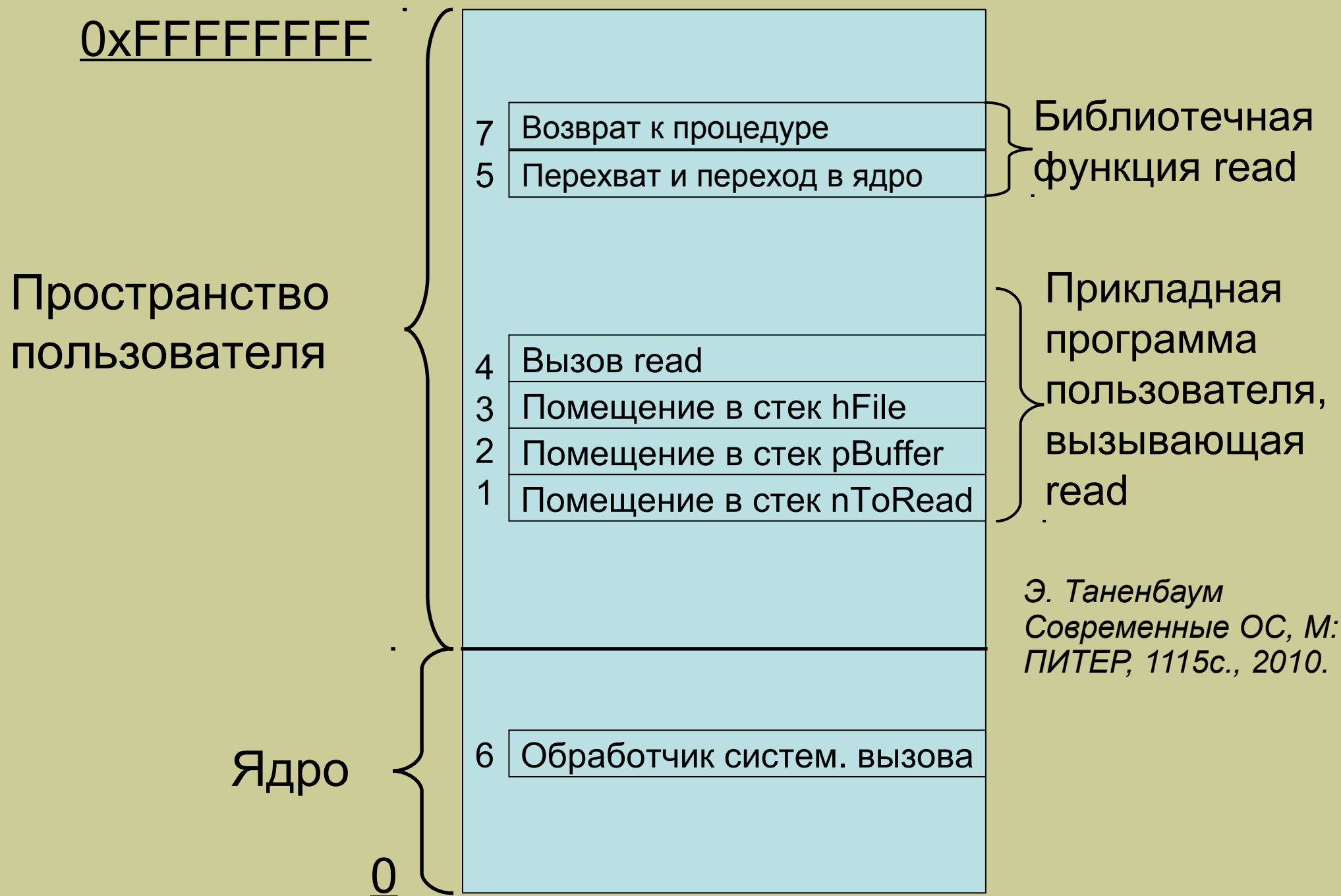
- полный доступ к командам процессора;
- обработка прерываний и исключений;
- доступ к объектам ядра.

Пользовательский режим:

- ограниченный набор команд процессора;
- запрет на вызов обработчиков прерываний.

Интерфейс системных вызовов предоставляет контролируемый доступ прикладных программ к ресурсам компьютера посредством переход из *пользовательского режима в режим ядра*. Пример: *Win32 API*.

`read(hFile, pBuffer, nToRead)` – библиотечная процедура
интерфейса системных вызовов.



Э. Таненбаум
Современные ОС, М:
ПИТЕР, 1115с., 2010.

Реализация системного вызова в Linux

write(int fd, const void *buf, size_t count); - *библиотечная функция*

```
mov    edx, 1      ;сколько байт записать
mov    ecx, hex    ;буфер, откуда писать
mov    ebx, 1      ;куда записывать, 1 - stdout
mov    eax, 4      ;номер системного вызова
int     80h        ;шлюз к ядру
```

(int 2Eh в Windows)

Таблица системных вызовов

%eax	Name	Source	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	kernel/exit.c	int	-	-	-	-
2	sys_fork	arch/i386/kernel/process.c	struct pt_regs	-	-	-	-
3	sys_read	fs/read_write.c	unsigned int	char *	size_t	-	-
4	sys_write	fs/read_write.c	unsigned int	const char *	size_t	-	-
5	sys_open	fs/open.c	const char *	int	int	-	-
6	sys_close	fs/open.c	unsigned int	-	-	-	-

Интерфейс прикладного программирования Windows API*

Особенности реализации языка C компании Microsoft (компилятор *cl*).

Некоторые типы данных, поддерживаемые Microsoft Windows:

DWORD	<code>typedef unsigned long DWORD</code>
BOOL	<code>typedef int BOOL;</code>
BYTE	<code>typedef unsigned char BYTE;</code>
PVOID	<code>typedef void *PVOID;</code>
HANDLE	<code>typedef PVOID HANDLE;</code>

* далее в основном *Win 32 API*

Чтобы обеспечить поддержку типов Microsoft Windows в программе, необходимо включить в нее заголовочный файл *windows.h*.

Этот файл также содержит объявления функций интерфейса системных вызовов *MS Windows Win32 API*.

Пример объявления функции:

```
BOOL GetComputerName(  
    LPTSTR    lpBuffer;  
    LPDWORD  nSize;  
);
```

LPSTR

typedef char *LPSTR

LPDWORD

typedef WORD *LPDWORD

```
#include <windows.h>
```

```
#include <stdio.h>
```

```
int main(){
```

```
    char Buffer[MAX_COMPUTERNAME_LENGTH+1];//[5];
```

```
    int size=sizeof(Buffer);
```

```
    if( !GetComputerName((LPTSTR)Buffer, (LPDWORD)&size) ){
```

```
        printf("System error code: %i\n",GetLastError());
```

```
        return -1;
```

```
    }
```

```
    fprintf(stdout,"The computer name is %s\n",Buffer);
```

```
    return 0;
```

```
}
```

Аварийный выход (при задании размера буфера равным 5):

```
C:\2011-spring\Лекции\Лекция2\Лаб2с>1  
System error code: 111
```

Запись в таблице System Error Codes:

110.....	
111	ERROR_BUFFER_OVERFLOW
112.....	

Нормальное выполнение:

```
C:\2011-spring\Лекции\Лекция2\Лаб2с>1  
The computer name is EWGENIJ-PC
```

Упражнение:

Программно определить пути к системному каталогу Windows и каталогу временных файлов Windows, используя следующие функции Win32 API:

```
UINT GetWindowsDirectory( LPTSTR lpBuffer,  
UINT uSize );
```

```
DWORD GetTempPath( DWORD nBufferLength,  
LPSTR lpBuffer );
```

Замечание. Примеры венгерской нотации:

Префикс	Тип данных
u	беззнаковое целое
lp	дальний указатель (long pointer) (атавизм)
sz	строка, заканчивающаяся нулевым байтом (с-строка)
n	короткое целое