Лекция 1

Содержание

- Цели и задачи курса.
- Краткий обзор курса.
- Структура современной операционной системы.
- Режим ядра и пользовательский режим.
- Системные вызовы. Интерфейс прикладного программирования (API).

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 *IBM PC*. 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 MS Windows* и *GNU/Linux*.	4
5. Потоки. Создание и реализация потоков в *GNU/Linux* и *MS Windows*. Многопоточные приложения *GNU/Linux* и *MS Windows*.	2
6. Архитектура приложения *Win32*. 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 *POSIX* и *Windows API*.	6
9. Файловые системы. Атрибуты файлов. Реализации размещения файлов и доступа к ним. Операции с файлами. Журналируемые файловые системы. Файловые системы *MS Windows* и *GNU/Linux*.	2
10. Безопасность. Аутентификация и авторизация пользователей. Типы вирусных атак. Механизмы защиты.	2
11. Дополнительные вопросы. Ловушки *MS Windows*. Фоновые программы - службы, и их реализация в *MS Windows* и *GNU/Linux*. Приложения COM и COM+. Расширения графической оболочки *MS Windows*.	4

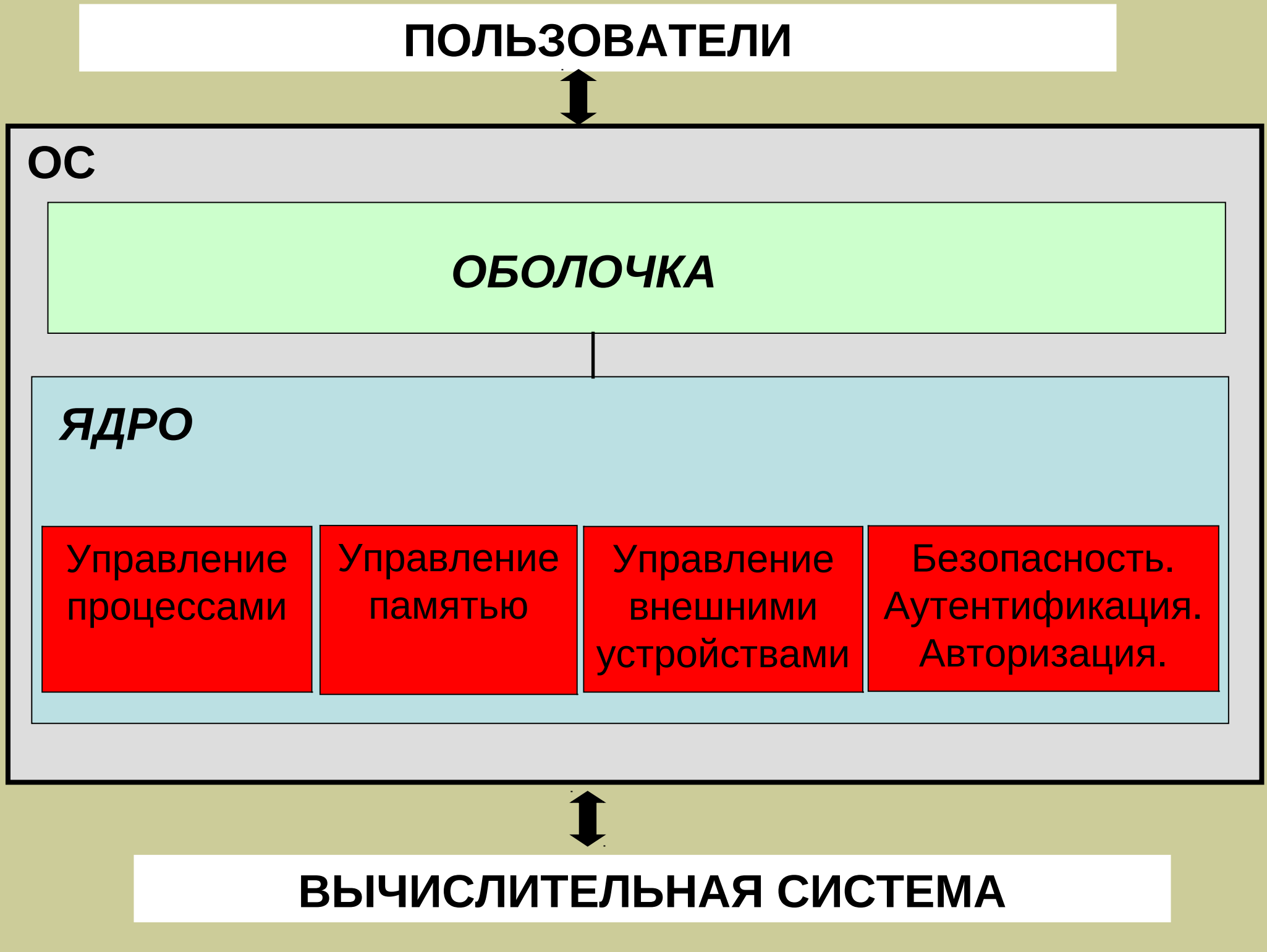
Краткий обзор ОС

Windows

Microsoft

macOS

Google


ПОЛЬЗОВАТЕЛИ
ОС
ОБОЛОЧКА
ЯДРО
Управление процессами
Управление памятью
Управление внешними устройствами
Безопасность. Аутентификация. Авторизация.
ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

Взаимодействие прикладных программ и ОС

Режим ядра (*режим супервизора, привилегированный режим*):

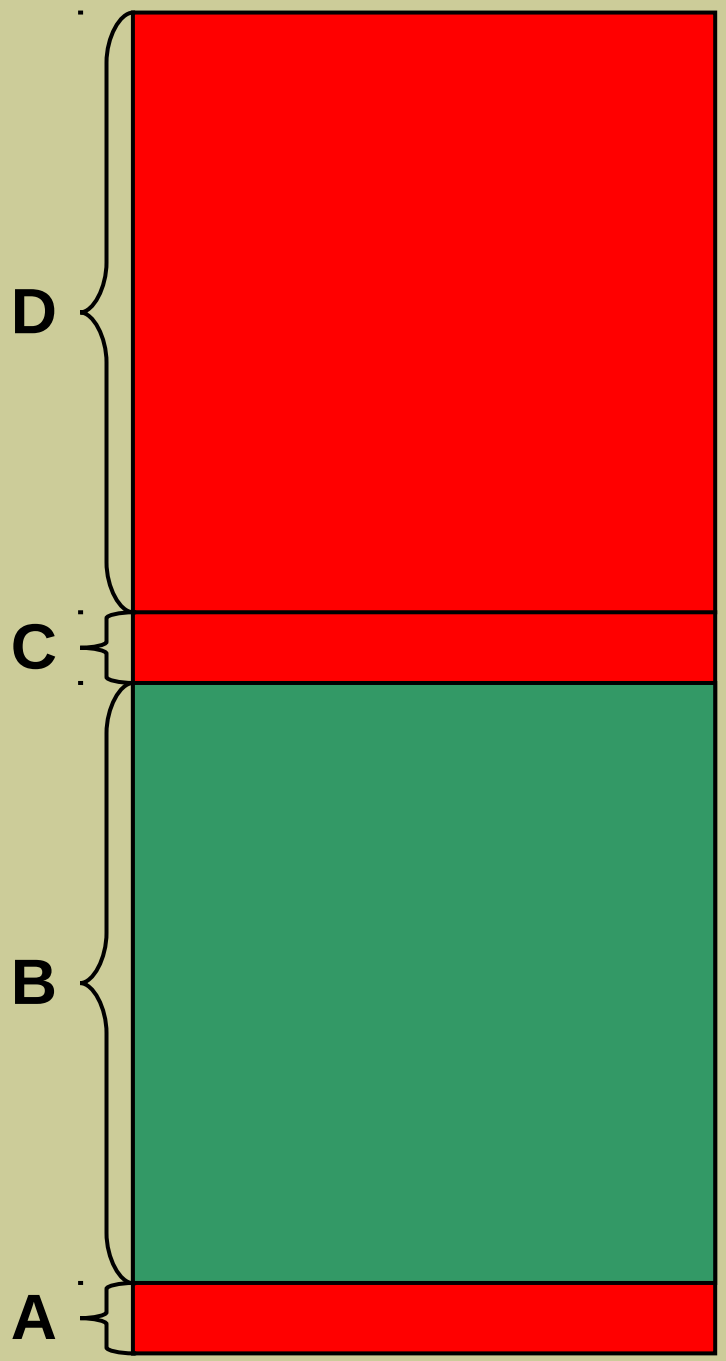
- полный доступ к командам процессора;
- обработка прерываний и исключений;
- доступ к объектам ядра.

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 структуры адресного пространства 32-разрядной ОС.



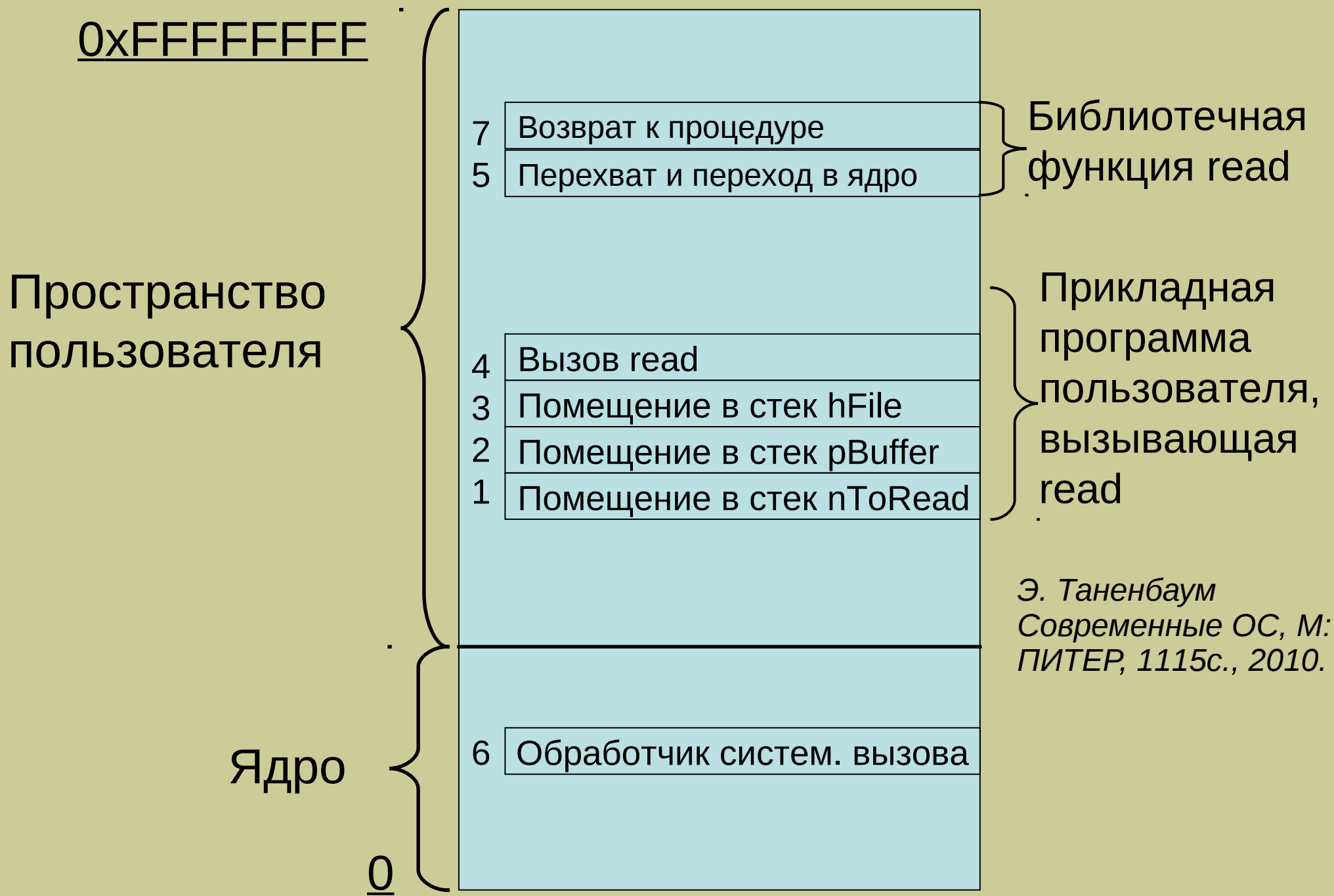
A. 0x00000000 – 0x0000FFFF; используется для неинициализированных указателей; ***недоступно*** в пользовательском режиме.

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.

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	-	-	-	-