

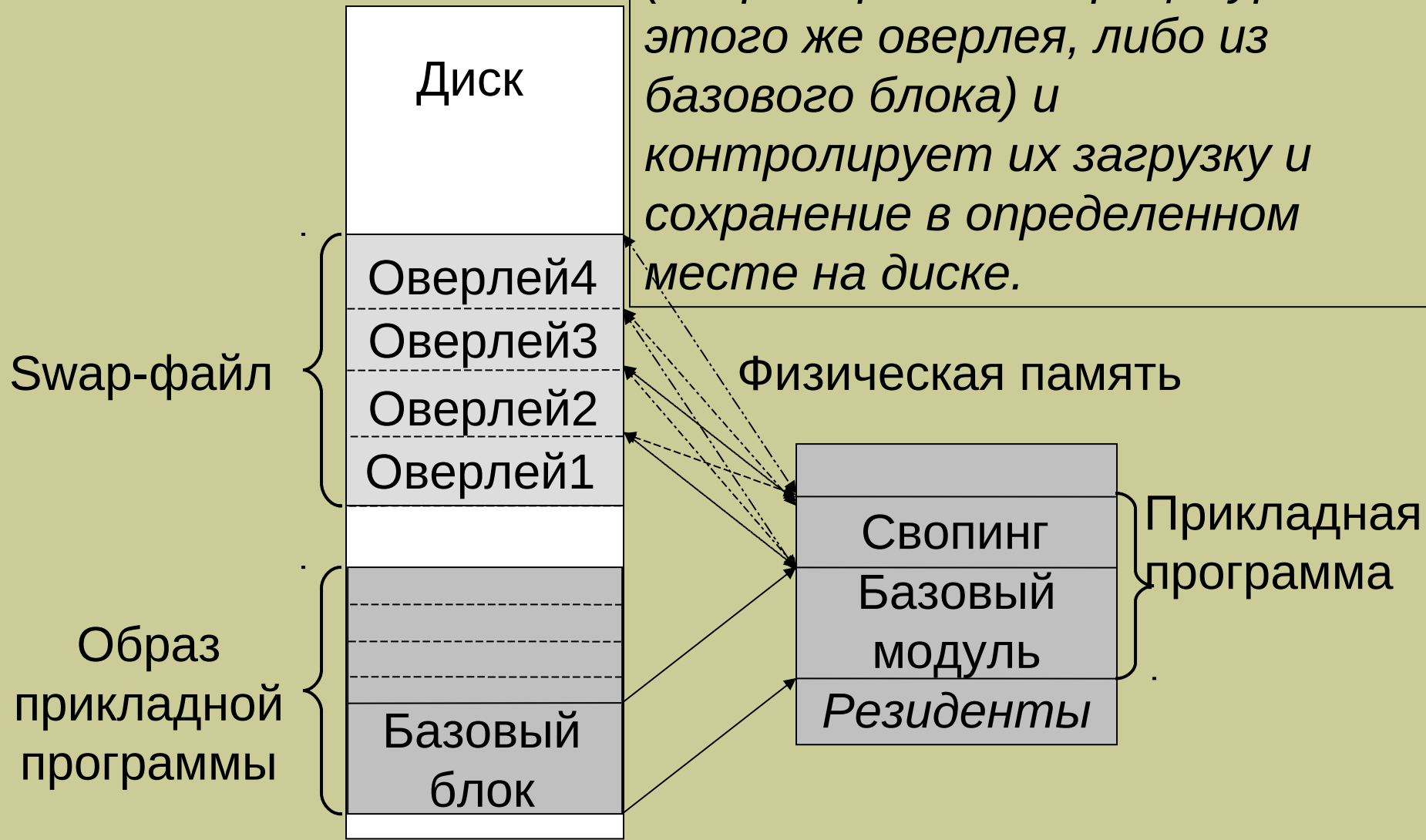
Лекция 6

Содержание

- I. Загрузка и выполнение программы -
 - адресное пространство, виртуальная и физическая адресация, отображение файлов в память
 - механизм виртуальной памяти, таблицы страниц, подкачка страниц.
- II. Получение информации о процессах-
 - интерфейс PSAPI,
 - виртуальная файловая система /proc.

Организация памяти

Оверлейная модель



Виртуальная память.

[Виртуальное] адресное пространство – некоторая последовательность чисел. Код программы может ссылаться на адреса этого числового диапазона.

Существует некоторая схема отображения виртуальных адресов на адреса физической памяти.

Технология страничной организации памяти

Пример: машинный код позволяет адресовать 64К байт памяти, физическая память составляет 4К.

Поделим адресное пространство на 16 областей (страниц) по 4К и установим следующее соответствие:

физический адрес = виртуальный адрес % 4К;

номер области (страницы) = виртуальный адрес / 4К.

При ссылке по виртуальному адресу A

- содержимое физической памяти сохраняется на диске;
- область с номером $A/4K$ загружается в память;
- произойдет обращение по адресу физической памяти $A \% 4K$.

Современные реализации страничной организации памяти

Каждому процессу выделяется адресное пространство (например в 32 разрядной Windows числа от нуля до $0xFFFFFFFF$).

Адресное пространство разбивается на страницы размером, обычно (в зависимости от ОС) от 512 байт до 64К.

Физическая память разбивается на области (*страничные кадры* (фреймы, блоки, слоты)) размером в страницу.

Таблица страниц устанавливает соответствие между страницами и страничными кадрами.

Виртуальная страница	Страничный кадр	Бит присутствия
...
7	0	0
6	3	1
5	4	1
4	0	0
3	2	1
2	0	0
1	0	1
0	1	1

Физическая память	Кадр
Виртуальная страница 5	4
Виртуальная страница 6	3
Виртуальная страница 3	2
Виртуальная страница 0	1
Виртуальная страница 1	0

Отображением виртуальной памяти на физические адреса занимается диспетчер виртуальной памяти – *VMM (Virtual Memory Management)*.

Аппаратной реализацией *VMM* является *MMU (Memory Management Unit)*, расположенный на чипе процессора.

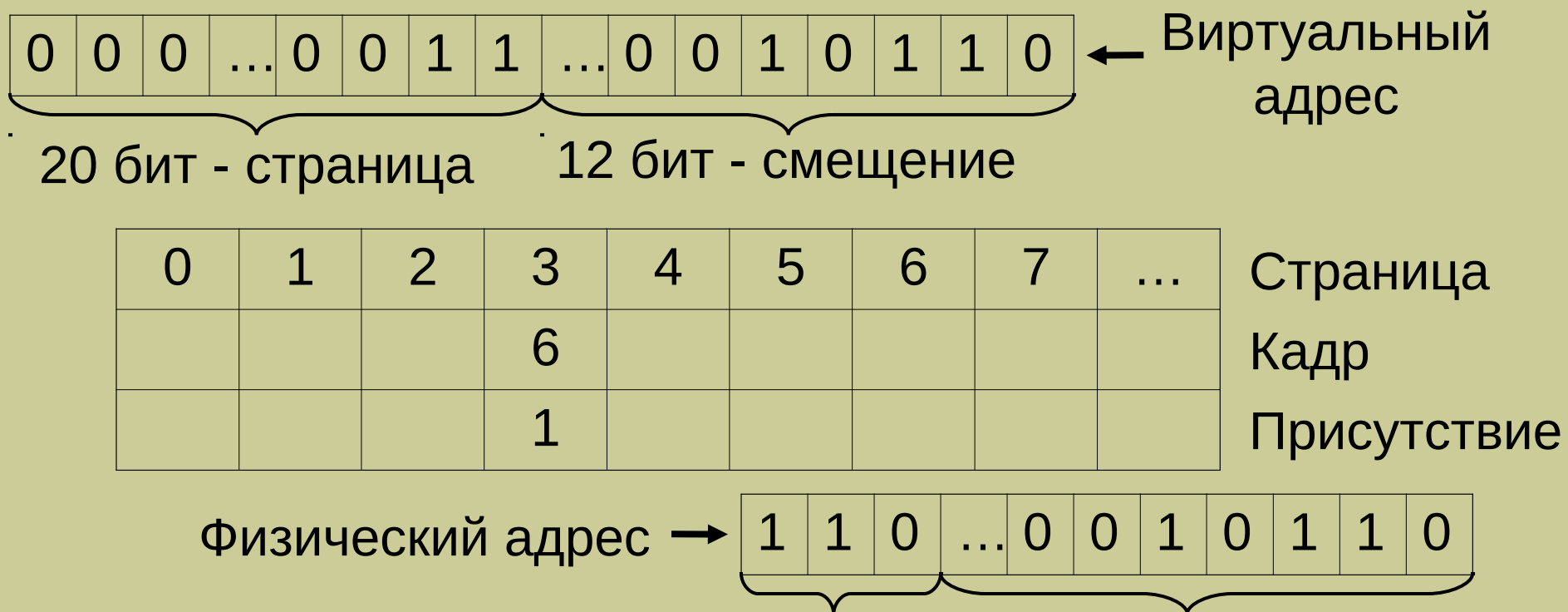


Таблица страниц

Стр	Фр-м	Бит пр.
1	4	1
2	6	1
...
8	...	0

Физическая
память

Стр	Фр-м	Бит пр.
1	7	1
2	1	1
...
8	9	1

Адресное
пространство

Вирт. Стр.1

Вирт. Стр.2

Вирт. Стр.8

Физ. Стр.1

Физ. Стр.4

Физ. Стр.6

Физ. Стр.7

Физ. Стр.9

Вирт. Стр.1

Вирт. Стр.2

Вирт. Стр.8

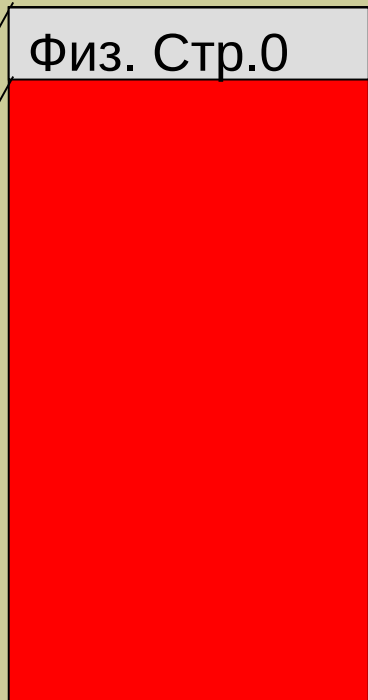
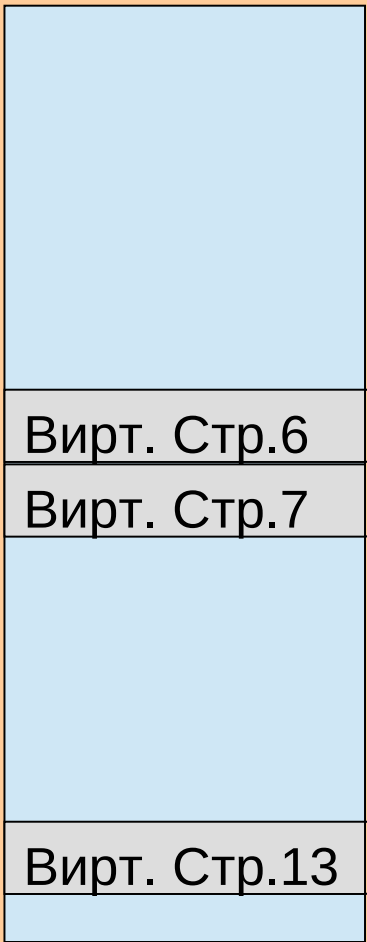
ProcID=19872

ProcID=198055

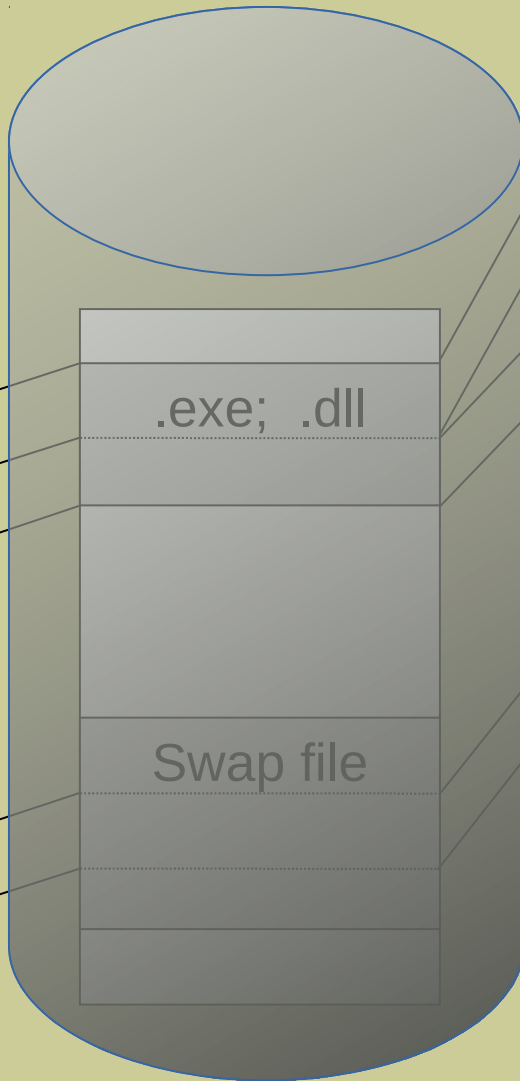
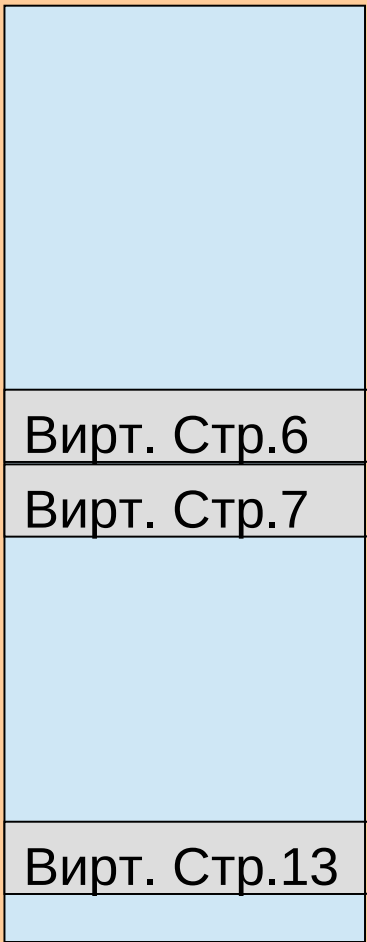
Вызов страниц по требованию. При обращении к адресу страницы, которой нет в основной памяти (бит присутствия 0), генерируется исключение – **ошибка отсутствия страницы** (промах). Обработка этого исключения – считывается нужная страница с диска, в таблице страниц делается соответствующая запись и команда повторяется.

Политика замещения страниц. Существует множество алгоритмов удаления (как правило, с последующим сохранением на диске) страниц из физической памяти. Например: *LRU (Least Recently Used)* – удаляется дольше всего не использовавшаяся страница; *FIFO (First –in First out)* – алгоритм очереди.

Стр	Фр-м	Бит пр.
6	0	1
7	...	0
...
13	...	0



Стр	Фр-м	Бит пр.
6	0	1
7	4	1
...
13	8	1



Карта адресного пространства процесса (*Linux 3.11.6-4, x86_64*)

```
malkov@linux-bx7d:~/WORKSHOP/PROJECTS/Transport/ISSUE> ./sch6
malkov@linux-bx7d:~> ps -aux | grep sch_test6
malkov  17686  0.0  0.0 12952 1328 pts/10  S+  16:23   0:00 ./sch6
malkov@linux-bx7d:~> cd /proc/17686
malkov@linux-bx7d:/proc/17686> cat maps
```

```
00400000-00402000 r-xp 00000000 08:06 7210176 /home/malkov/.../sch6
00601000-00602000 r--p 00001000 08:06 7210176 /home/malkov/.../sch6
00602000-00643000 rw-p 00002000 08:06 7210176 /home/malkov/.../sch6
01f40000-01f82000 rw-p 00000000 00:00 0 [heap]
7f8bc8cec000-7f8bc8e91000 r-xp 00000000 08:05 1179690 /lib64/libc-2.18.so
.....
7f8bc92b2000-7f8bc93b4000 r-xp 00000000 08:05 1179681 /lib64/libm-2.18.so
.....
7f8bc95b5000-7f8bc969f000 r-xp 00000000 08:05 6427589 /usr/lib64/libstdc++.so
.....
7fff1b637000-7fff1b659000 rw-p 00000000 00:00 0 [stack]
.....
```

диапазон адресов области	доступ	смещение в отображении файла	номер устройства
		индексный дескриптор файла	путь к файлу

Адрес и размер области кратен 4К

Получение информации о процессах

Чтение информации из **PCB** (*Process control block*) возможно с помощью интерфейса **PSAPI** (*Process Status API*) , реализация **psapi.dll**, заголовочный файл **psapi.h**

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

```
#include <psapi.h>
```

```
int main(){
```

```
    DWORD pID;
```

```
    HANDLE pHndl;
```

```
    HMODULE* modHndls;
```

```
    DWORD b_alloc=8, b_needed;
```

```
    char modName[MAX_PATH];
```

```
    int i;
```

```
    pID=GetCurrentProcessId();
```

```
    pHndl=OpenProcess(PROCESS_ALL_ACCESS,FALSE,pID);
```

```
while(1){
    modHndls=(HMODULE*)malloc(b_alloc);

    EnumProcessModules(pHndl,modHndls,b_alloc,&b_needed);

    printf("%u  %u\n",pID,pHndl);
    printf("%u  %u\n",b_alloc, b_needed);

    if(b_alloc>=b_needed)
        break;
    else{
        free(modHndls);
        b_alloc=b_needed;
    }
}
```

```
for(i=0;i<b_needed/sizeof(DWORD);i++){  
  
    GetModuleBaseName(pHndl, modHndls[i],  
                      (LPSTR)modName, sizeof(modName));  
  
    printf("%u\t%s", modHndls[i],modName);  
  
    GetModuleFileName(modHndls[i], (LPSTR)modName,  
                      sizeof(modName));  
    printf("\t%s\n",modName);  
}  
  
return 0;  
}
```

Замечание: ...> cl filename.c kernel32.lib psapi.lib

C:\Users\ewgenij\Documents\СибГУТИ\2011-spring\Лекции\
Лекция3\Лаб3>2

4488 16

8 16

4488 16

16 16

4194304 2.exe C:\Users\ewgenij\Documents\СибГУТИ\2011-
spring\Лекции\Лекция3\Лаб3\2.exe

2010251264 ntdll.dll C:\Windows\system32\ntdll.dll

1986002944 kernel32.dll C:\Windows\system32\kernel32.dll

2011561984 PSAPI.DLL C:\Windows\system32\PSAPI.DLL

Упражнение 2: запустите приложение Internet Explorer и с помощью функций EnumProcesses, OpenProcess, EnumProcessModules, GetModuleBaseName и GetModuleFileName определите все модули процесса и места расположения, соответствующих им файлов.

```
BOOL EnumProcesses(  
    DWORD *lpidProcess, // массив идентификаторов процессов  
    DWORD cb,           // размер массива  
    DWORD *cbNeeded // количество необходимых байт  
);
```