

Операционные системы

Управление виртуальной
памятью в Win32.
Общие принципы.

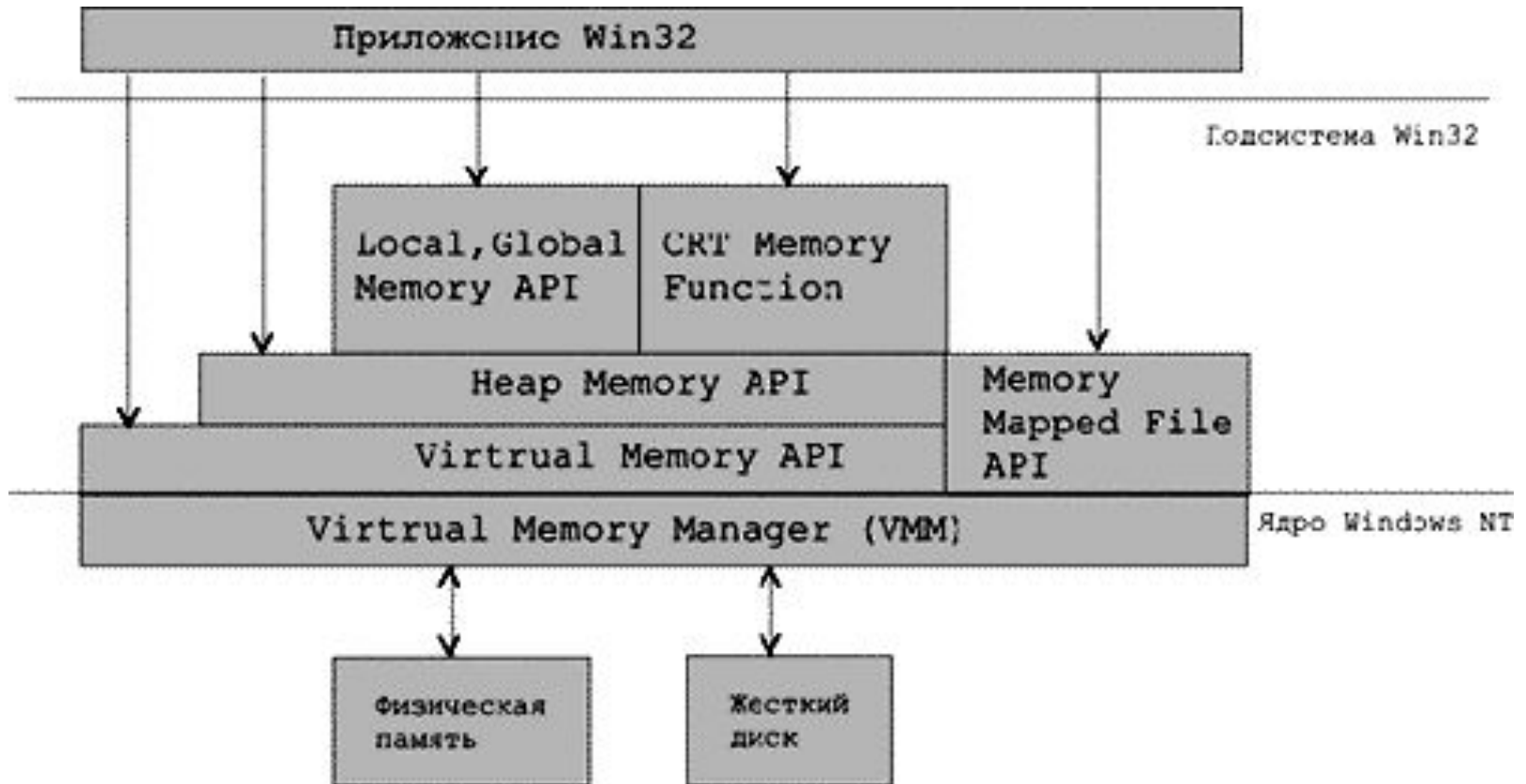
Менеджер виртуальной памяти

- управление виртуальными адресными пространствами процессов;
- разделение памяти между процессами;
- защита виртуальной памяти одного процесса от других процессов.

Менеджер виртуальной памяти

- Менеджер виртуальной памяти (VMM) является составной частью ядра ОС. Приложения не могут получить к нему прямой доступ.
- Для управления памятью прикладным программам предоставляются различные интерфейсы (API).
 - Virtual Memory API - набор функций, позволяющих приложению работать с виртуальным адресным пространством: назначать физические страницы блоку адресов и освобождать их, устанавливать атрибуты защиты.
 - Memory Mapped File API - набор функций, позволяющий работать с файлами, отображаемыми в память. Новый механизм, предоставляемый Win32 API для работы с файлами и взаимодействия процессов.
 - Heap Memory API - набор функций, позволяющих работать с динамически распределяемыми областями памяти (кучами).
 - Local, Global Memory API - набор функций работы с памятью, совместимых с 16-битной Windows. Следует избегать их использования.
 - CRT Memory API - функции стандартной библиотеки языка "С" периода исполнения (runtime).

Архитектура API управления памятью



Адресное пространство процесса



Standart

Enterprise

Средства защиты памяти

- **Объектно-ориентированная защита памяти.** Каждый раз, когда процесс открывает указатель на блок адресов, монитор ссылок безопасности проверяет, разрешен ли доступ процесса к данному объекту.
- **Отдельное адресное пространство для каждого процесса.** Аппаратура запрещает процессу доступ к физическим адресам другого процесса.
- **Два режима работы:** режим ядра, в котором процессам разрешен доступ к системным данным, и пользовательский режим, в котором это запрещено.
- **Страничный механизм защиты.** Каждая виртуальная страница имеет набор признаков, который определяет разрешенные типы доступа в пользовательском режиме и в режиме ядра.
- **Принудительная очистка страниц,** освобождаемых процессами.

Операционные системы

Управление виртуальной
памятью в Win32.
Страничное преобразование.

Страничное преобразование

- Виртуальная память в Windows имеет страничную организацию, принятую во многих современных ОС. Процессоры Intel начиная с Pentium Pro позволяют ОС применять одно-, двух- и трехступенчатые схемы. И даже разрешается одновременное использование страниц различного размера. Эта возможность, конечно, повысила бы эффективность страничного преобразования, будь она внедрена в Windows. Однако Windows возникла раньше и поддерживает только двухступенчатую схему преобразования с фиксированным размером страниц.
- Каждому процессу назначается свой каталог страниц. Именно поэтому адресное пространство каждого процесса изолировано, что очень хорошо с точки зрения защиты процессов друг от друга.

Страничное преобразование

- 32-разрядный виртуальный адрес в ОС Windows разбивается на три части:
 - Старшие 10 разрядов адреса определяют номер одного из 1024 элементов в каталоге страниц, адрес которого находится в регистре процессора CR3. Этот элемент содержит физический адрес таблицы страниц.
 - Следующие 10 разрядов линейного адреса определяют номер элемента таблицы. Элемент, в свою очередь, содержит физический адрес страницы виртуальной памяти.
 - Размер страницы - 4 Кбайт, и младших 12 разрядов линейного адреса как раз хватает ($2^{12} = 4096$), чтобы определить точный физический номер адресуемой ячейки памяти внутри этой страницы.

СВОПИНГ

- Для того, чтобы обеспечить все линейное адресное пространство процесса физическими ячейками памяти, Windows применяет свопинг.
- Организацией свопинга занимается VMM. При генерации системы на диске образуется специальный файл свопинга, куда записываются те страницы, которым не находится места в физической памяти. Процессы могут захватывать память в своем ВАП и, затем, использовать ее. Страница может иметь различные состояния.
- VMM использует локальный алгоритм **LRU** (Least Recently Used) – замещение дольше всех неиспользовавшихся страниц.

Страничное преобразование



Элемент таблицы страниц (**P**age **T**able **E**lement)

Элемент таблицы страниц



- Защита – Win32 API поддерживает ряд значений, в том числе: *PAGE_NOACCESS*, *PAGE_READONLY*, *PAGE_READWRITE*, *PAGE_EXECUTE*.
- Базовый физический адрес страницы в памяти.
- Pagefile – индекс используемого файла подкачки (один из 16 возможных в системе файлов).
- State – состояние страницы в системе:
 - *T (Transition)* – отмечает страницу как переходную;
 - *D (Dirty)* – страница, в которую была произведена запись;
 - *P (Present)* – страница присутствует в ОП или находится в файле подкачки.

Отдельные состояния страниц

- Valid – страница используется процессом. Она реально существует в ОП и помечена в PTE как присутствующая в рабочем множестве процесса ($P=1, D=0,1$).
- Modified – содержимое страницы было изменено ($D=1$). В PTE страница помечена как отсутствующая ($P=0$) и переходная ($T=1$).
- Standby – содержимое страницы не изменялось ($D=0$). В PTE страница помечена как отсутствующая ($P=0$) и переходная ($T=1$).
- Free – страница, на которую не ссылается ни один PTE. Страница свободна, но подлежит обнулению, прежде чем будет использована.
- Zeroed – свободная и обнуленная страница, пригодная к непосредственному использованию любым процессом.
- Bad – страница, которая вызывает аппаратные ошибки и не может быть использована ни одним процессом.

Отдельные состояния страниц

| T | D | P | Page state |
|----------|----------|----------|------------------------------------------------|
| 0 | - | 0 | Invalid page (Free) |
| - | 0 | 1 | Valid page |
| - | 1 | 1 | Valid dirty page |
| 1 | 0 | 0 | Invalid page in transition (Standby) |
| 1 | 1 | 0 | Invalid dirty page in transition (Modified) |

Стратегия управления виртуальной памятью

- ✓ *Стратегия выборки (fetch policy)*
- ✓ *Стратегия размещения (placement policy)*
- ✓ *Стратегия замещения (replacement policy)*

Стратегия выборки

- ✓ *Стратегия выборки (fetch policy):*
 - ✓ Выборка определяет, в какой момент необходимо переписать страницу с диска в ОП.
 - ✓ В Windows используется классическая схема выборки с упреждением: система переписывает в память не только выбранную страницу, но и несколько следующих по принципу пространственной локальности, гласящему: наиболее вероятным является обращение к тем ячейкам памяти, которые находятся в непосредственной близости от ячейки, к которой производится обращение в настоящий момент. Поэтому вероятность того, что будут востребованы последовательные страницы, достаточно высока. Их упреждающая подкачка позволяет снизить накладные расходы, связанные с обработкой прерываний.
- ✓ *Стратегия размещения (placement policy)*
- ✓ *Стратегия замещения (replacement policy)*

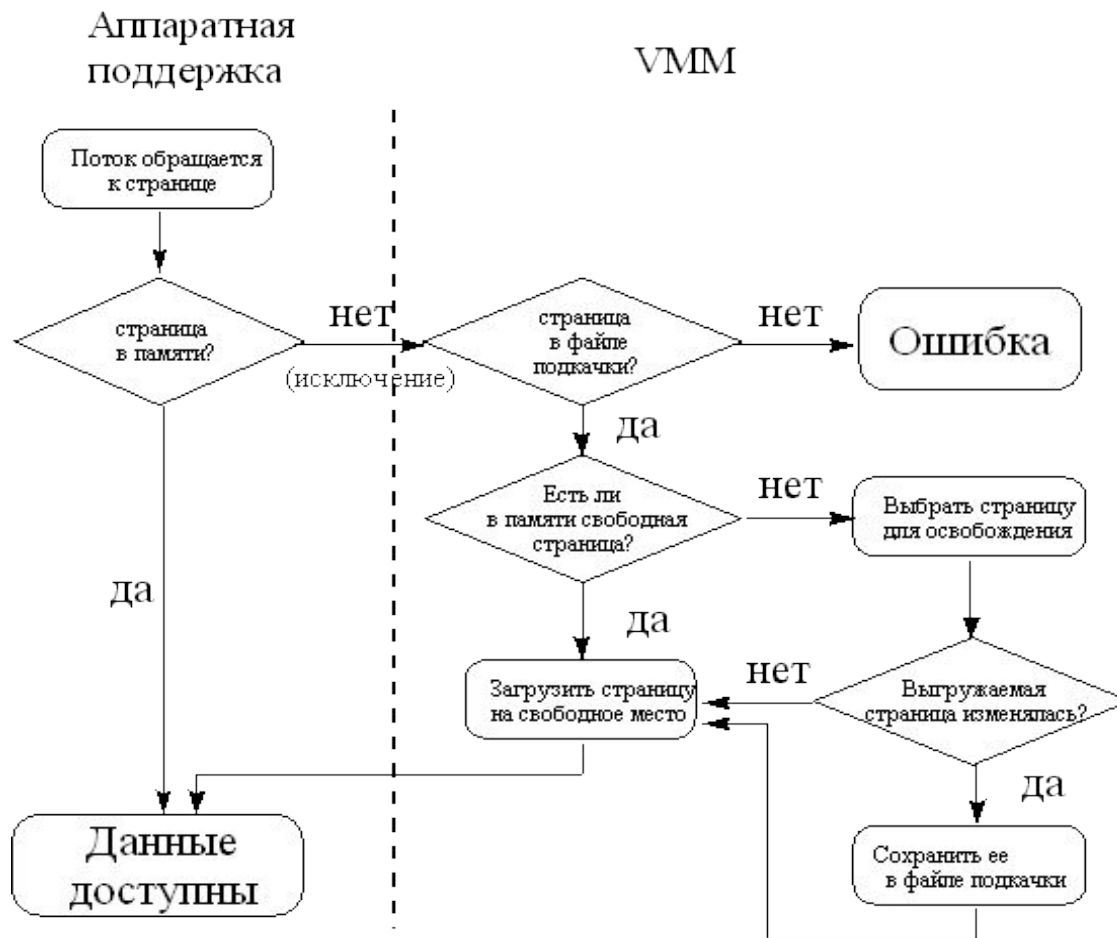
Стратегия размещения

- ✓ *Стратегия выборки (fetch policy)*
- ✓ *Стратегия размещения (placement policy):*
 - ✓ Размещение определяет, в какое место оперативной памяти необходимо поместить подгружаемую страницу.
 - ✓ Для систем со страничной организацией данная стратегия практически не имеет никакого значения, и поэтому Windows выбирает первую попавшуюся свободную страницу.
- ✓ *Стратегия замещения (replacement policy)*

Стратегия замещения

- ✓ *Стратегия выборки (fetch policy)*
- ✓ *Стратегия размещения (placement policy)*
- ✓ *Стратегия замещения (replacement policy):*
 - ✓ Замещение начинает действовать с того момента, когда в оперативной памяти компьютера не остается свободного места для размещения подгружаемой страницы. В этом случае необходимо решить, какую страницу вытеснить из физической памяти в файл подкачки (свопинг).

Реализация свопинга



Реализация алгоритма замещения LRU

- VMM периодически просматривает список страниц с атрибутом Valid и пытается похитить их у процесса (1). Он помечает их как отсутствующие ($P=0$), но на самом деле оставляет их на месте, только переводит в разряд Modified или Standby в зависимости от значения бита D из PTE.
- Если содержимое страницы была изменено в ОП ($D=1$), то VMM выполнит запись страницы на диск (4).
- Если похищенная страница принадлежит рабочему множеству, то к ней в ближайшее время произойдет обращение. Это, конечно, вызовет исключение - ведь страница-то помечена как отсутствующая. Но VMM очень быстро сделает эту страницу вновь доступной процессу, поскольку она реально находится в памяти (2).
- Далее если к странице не будет обращений (страница вне рабочего множества), то она со временем перейдет в состояние Free (5) и станет доступна для замещения страниц в рамках данного процесса (6).
- Затем системный поток обнуляет страницу - Zeroed (7), и она станет доступна другим процессам системы (8).

Переходы между состояниями страниц

