**Тезаурусный словарь**

Микроядро (micro-kernel) – один из важнейших принципов разработки ОС, который заключается в переносе максимально возможного числа модулей из системного в пользовательское "пространство", т.е. ОС разрабатывается таким образом, что большинство ее модулей выполняются в пользовательском режиме. Коммуникация выполняется между пользовательскими модулями с помощью передачи сообщений.

Виртуальные машины – другой распространенный подход к разработке операционных систем и их пользовательских интерфейсов

Виртуализация – один из наиболее современных принципов развития и использования программного обеспечения.

Система виртуальных машин – хорошая основа для исследования и разработок в области ОС. Разработка систем выполняется над виртуальной машиной, а не на физической машине, и не нарушает нормального функционирования системы

**Ментальная карта**

**Память без использования абстракций**

Адресация физической памяти

Запуск программы с несколькими потоками.

# Память без использования абстракций

Простейшей абстракцией памяти можно считать полное отсутствие какой-либо абстракции. Ранние универсальные машины (до 1960 года), ранние мини-компьютеры (до 1970 года) и ранние персональные компьютеры (до 1980 года) не использовали абстракции памяти. Каждая программа просто видела физическую память. Когда программа выполняла команду

MOV REGISTER1,1000

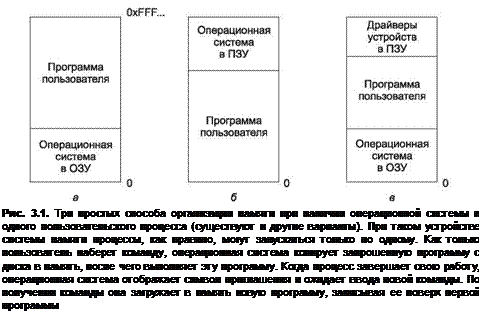
компьютер просто перемещал содержимое физической ячейки памяти 1000 в REGISTER1. Таким образом, модель памяти, предоставляемая программисту, была простой физической памятью, набором адресов от 0 до некоторого максимального значения, где каждый адрес соответствовал ячейке, содержащей какое-то количество бит (обычно 8).

При таких условиях содержание в памяти сразу двух работающих программ не представлялось возможным. Если первая программа, к примеру, записывала новое значение в ячейку 2000, то она тем самым стирала то значение, которое сохраняла там вторая программа. Работа становилась невозможной, и обе программы практически сразу же давали сбой.

Даже в условиях, когда в качестве модели памяти выступает сама физическая память, возможны несколько вариантов использования памяти. Три из них показаны на рис. 3.1. Операционная система может (рис. 3.1, а) размещаться в нижней части адресов, в оперативном запоминающем устройстве (ОЗУ), или, по-другому, в памяти с произвольным доступом — RAM (Random Access Memory). Она может размещаться также в постоянном запоминающем устройстве (ПЗУ), или, иначе, в ROM (ReadOnly Memory), в верхних адресах памяти (рис. 3.1, б). Или же драйверы устройств могут быть в верхних адресах памяти, в ПЗУ, а остальная часть системы — в ОЗУ, в самом низу (рис. 3.1, в). Первая модель прежде использовалась на универсальных машинах и мини-компьютерах, а на других машинах — довольно редко. Вторая модель использовалась на некоторых КПК и встроенных системах. Третья модель использовалась на ранних персональных компьютерах (например, на тех, которые работали под управлением MS-DOS), где часть системы, размещавшаяся в ПЗУ, называлась базовой системой ввода-вывода — BIOS (Basic Input Output System). Недостаток моделей, изображенных на рис. 3.1, а и в, заключается в том, что ошибка в программе пользователя может затереть операционную систему, и, возможно, с весьма пагубными последствиями.

Единственный способ добиться хоть какой-то параллельности в системе, не имеющей абстракций памяти, — это запустить программу с несколькими потоками. Поскольку предполагается, что все имеющиеся в процессе потоки видят один и тот же образ памяти, их принудительное применение проблемы не составляет. Хотя эта идея вполне работоспособна, она не нашла широкого применения, поскольку людям зачастую необходим одновременный запуск не связанных друг с другом программ, а этого абстракция потоков не обеспечивает. Более того, столь примитивные системы, не обеспечивающие абстракций памяти, вряд ли могут предоставить абстракцию потоков.

Тем не менее даже в отсутствие абстракций памяти одновременный запуск нескольких программ вполне возможен. Для этого операционная система должна сохранить все содержимое памяти в файле на диске, а затем загрузить и запустить следующую



программу.

Поскольку одновременно в памяти присутствует только одна программа, конфликтов не возникает. Эта концепция называется заменой данных (или свопингом) и будет рассмотрена чуть позже.

При наличии некоторого специального дополнительного оборудования появляется возможность параллельного запуска нескольких программ даже без использования свопинга. На ранних моделях IBM 360 эта проблема решалась следующим образом: память делилась на блоки по 2 Кбайт, каждому из которых присваивался 4-битный защитный ключ, содержащийся в специальных регистрах за пределами центрального процессора. Машине с объемом памяти 1 Мбайт нужно было иметь лишь 512 таких 4-битных регистров, и все хранилище ключей занимало в итоге 256 байт памяти. Слово состояния программы (Program Status Word (PSW)) также содержало 4-битный ключ. Аппаратное обеспечение IBM 360 перехватывало любую попытку запущенного процесса получить доступ к памяти с ключом защиты, отличающимся от ключа PSW. Поскольку изменить ключи защиты могла только операционная система, пользовательские процессы были защищены от вмешательства в работу друг друга и в работу самой операционной системы.

Тем не менее это решение имело серьезный недостаток, показанный на рис. 3.2. Здесь изображены две программы, каждая из которых имеет объем 16 Кбайт. Они показаны на рис. 3.2, а и б. Первая из них закрашена, чтобы показать, что у нее иной ключ памяти, нежели у второй. Первая программа начинается с перехода на ячейку памяти с адресом 24, в которой содержится команда MOV. Вторая программа начинается с перехода на ячейку памяти с адресом 28, в которой содержится команда CMP. Команды, не имеющие отношения к рассматриваемому вопросу, на рисунке не показаны. Когда две программы загружаются друг за другом в память, начиная с ячейки с адресом 0, мы получаем ситуацию, показанную на рис. 3.2, в. В этом примере мы предполагаем, что операционная система находится в верхних адресах памяти и поэтому не показана.

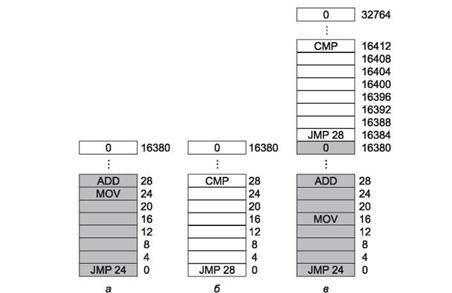


Рис. 3.2. Иллюстрация проблемы перемещения: а — программа объемом 16 Кбайт; б — еще одна программа объемом 16 Кбайт; в — две программы, последовательно

После загрузки программы могут быть запущены на выполнение. Поскольку у них разные ключи памяти, ни одна из них не может навредить другой. Но проблема имеет иную природу. При запуске первой программы будет выполнена командаJMP 24 и, как и ожидалось, осуществлен переход к другой команде. Эта программа будет успешно работать.

Но после того как первая программа проработает достаточно долго, операционная система может принять решение на запуск второй программы, которая была загружена над первой программой, начиная с адреса 16 384. Первой исполняемой командой будет JMP28, осуществляющая переход к команде ADD первой программы вместо того, чтобы перейти к предполагаемой команде CMP. Скорее всего, это приведет к сбою программы на первой же секунде.

В данном случае суть проблемы состоит в том, что обе программы ссылаются на абсолютный адрес физической памяти, что совершенно не соответствует нашим желаниям. Нам нужно, чтобы каждая программа ссылалась на занимаемый ею набор адресов. Давайте вкратце рассмотрим, как это достигается. На IBM 360, например, в процессе загрузки буквально на лету осуществлялась модификация второй программы, при этом использовалась технология, известная как статическое перемещение. Она работала следующим образом: когда программа загружалась с адреса 16 384, в процессе загрузки к каждому адресу в программе прибавлялось постоянное значение 16 384 (следовательно, инструкция «JMP 28» становилась инструкцией «JMP 16412» и т. д.). При всей исправности работы этого механизма он был не самым универсальным решением, и к тому же замедлял загрузку. Более того, это решение требовало дополнительной информации обо всех исполняемых программах, сообщающей, в каких словах содержатся, а в каких не содержатся перемещаемые адреса. В результате чего 28 на рис. 3.2, б должно было подвергнуться перемещению, а инструкция вроде

МОУ 1\*Еб1БТЕ1и,28

которая помещает число 28 в ЯЕШБТЕШ, должна была остаться нетронутой. Нужен был какой-нибудь способ, позволяющий сообщить загрузчику, какое из чисел относится к адресу, а какое — к константе.

И наконец, как отмечалось в главе 1, в компьютерном мире истории свойственно повторяться. Хотя прямая адресация физической памяти, к сожалению, осталась в прошлом, в устаревших устройствах памяти универсальных компьютеров, миникомпьютеров, настольных компьютеров, ноутбуков и смартфонов дефицит абстракций памяти — вполне обычное явление во встроенных системах и смарт-картах. В наши дни устройства вроде радиоприемников, стиральных машин и микроволновых печей заполнены программным обеспечением (в ПЗУ), и в большинстве случаев их программы используют адресацию к абсолютной памяти. Все это неплохо работает, поскольку все программы известны заранее, и пользователи не могут запускать на бытовых устройствах какие-нибудь собственные программы.

Сложные операционные системы присутствуют только в высокотехнологичных встроенных устройствах (например, в смартфонах), а более простые устройства их не имеют. В некоторых случаях у них тоже есть операционная система, но это всего лишь библиотека, связанная с прикладной программой, которая предоставляет системные вызовы для выполнения операций ввода-вывода и других общих задач. Простым примером популярной операционной системы, представляющей собой библиотеку, может послужить е-еоБ.