БИЛЕТ № 10

1) Концепции программирования процесса в многопрограммной ОС.

Система работает в многопрограммном режиме, если в системе находится несколько задач на разной стадии выполнения, и каждая из них может быть прервана другой с последующим восстановлением.

Основной проблемой многопрограммной работы является организация защиты программ от взаимного влияния как на уровне оперативной памяти, так и на разных уровнях внешней памяти. Необходимость организации защиты возникает вследствии того, что различные программы пишутся независимо друг от друга в расчете на единоличное использование ресурсов системы. Это создает опасность взаимного влияния программ друг на друга и должно быть учтено при реализации системы.

Разделение аппаратных и программных ресурсов системы ставит сложные задачи управления перед СУПЕРВИЗОРом, которые он решает благодаря наличию специальных алгоритмов распределения рессурсов системы.

Если количество задач превышает количество процессоров, то возникает проблема определения (планирование) очередности их решения во времени. Планирование во времени — это решение множества задач на ограниченных ресурсах. При организации (планировании) работы параллельной вычислительной ВС и распределении задач между множеством процессоров (ресурсов) планирование осуществляется в пространстве и во времени (рис. 2.4).

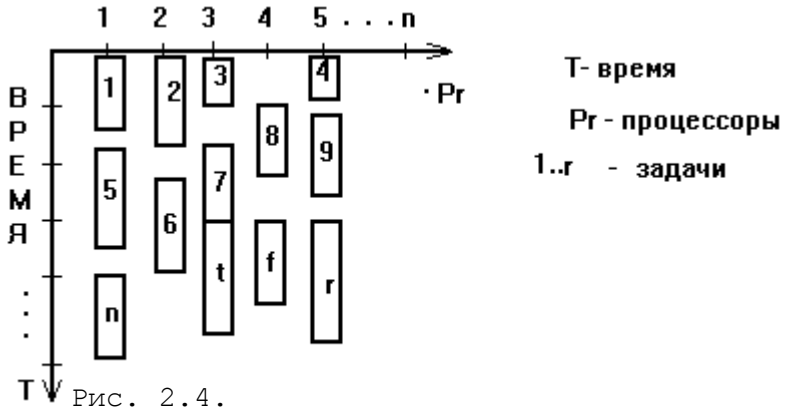


Рис. 2.4.

Как только количество процессоров становится больше, чем количество независимо решаемых задач, надобность во временной координате планирования пропадает.бь

Принято различать пять способов реализации многопрограммного режима работы:

- Классическое мультипрограммирование;
- Параллельная обработка;
- Разделение времени;
- Свопинг.

Классическое мультипрограммирование — это режим истинного совмещения, когда параллельно исполняемые задачи занимают различное оборудование.

Режим мультипрограммирования характеризуется тем, что правила перехода от программы к программе устанавливаются из соображений достижения максимального использования ресурсов машины путем более широкого использования совмещения их действий. Дорогостоящий ЦП все-таки простаивает, когда данные вводятся или выводятся, несмотря на то, что процесс ввода/вывода все время ускоряется. Тогда было предложено другое решение — поместить в память одновременно несколько программ. Память поделена на участки, и каждая программа располагается в своем участке. Теперь, при выполнении ввода/вывода для одной программы, вторая программа может производить вычисления, т.е. процессор переключается на выполнение другой программы (рис. 2.5). Этот подход, при котором несколько программ делят память и по очереди используют ЦП и другие ресурсы, работающие автономно, называется мультипрограммированием.

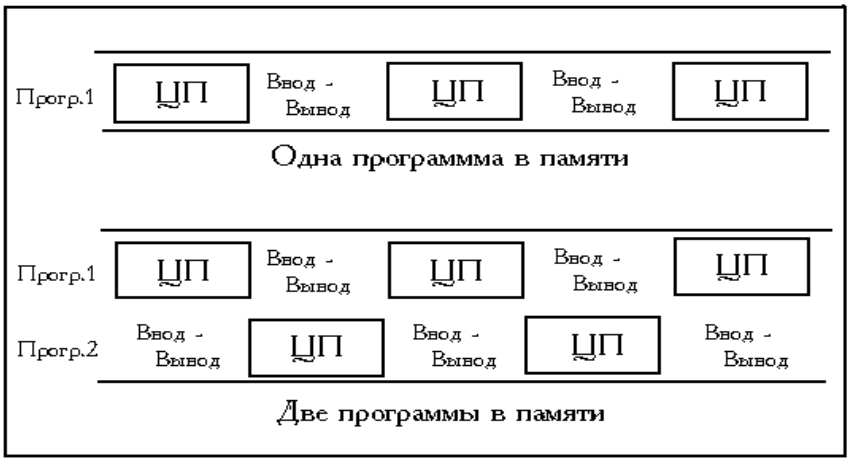


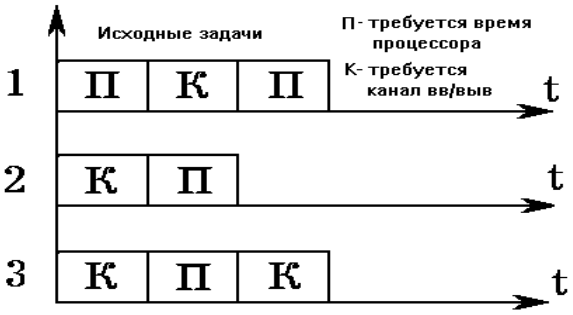
Рис. 2.5. Многозадачный и последовательный подход

При мультипрограммировании улучшение качества обслуживания пользователя непосредственно не предусматривается. Может случится так, что одна из программ, захватив один из ресурсов, например процессор, "заблокирует" выполнение других программ. Кроме того, мультипрограммирование, в чистом виде, не совместимо с непосредственным доступом, так как это один из видов пакетной обработки задач, то есть все пользователи получают результаты одновременно, как и в режиме косвенного доступа. Тем не менее, общее время обработки пакета задач при мультипрограммировании оказывается меньше, чем при последовательной пакетной обработке, за счет лучшего использования оборудования.

Производительность системы, работающей в мультипрограммном режиме, во многом зависит от состава группы программ, исполняемых совместно. Действительно, почти неизбежно возникновение отдельных моментов времен, когда все программы ожидают ввод или вывод данных, а центральное устройство простаивает. Отсюда следует задача целенаправленного формирования пакета программ с целью сокращения непроизводительных простоев оборудования.

На рис. 2.6.б показан пример выполнения трех задач в мультипрограммном режиме. При этом необходимо учитывать приоритеты задач при выборе задачи, которой необходимо дать предпочтение в случае множественных требований к некоторым ресурсам.

При уменьшении абсолютного времени решения, время решения отдельных заявок может увеличиваться по сравнению с однопрограммным режимом работы. рис. 2.6.б



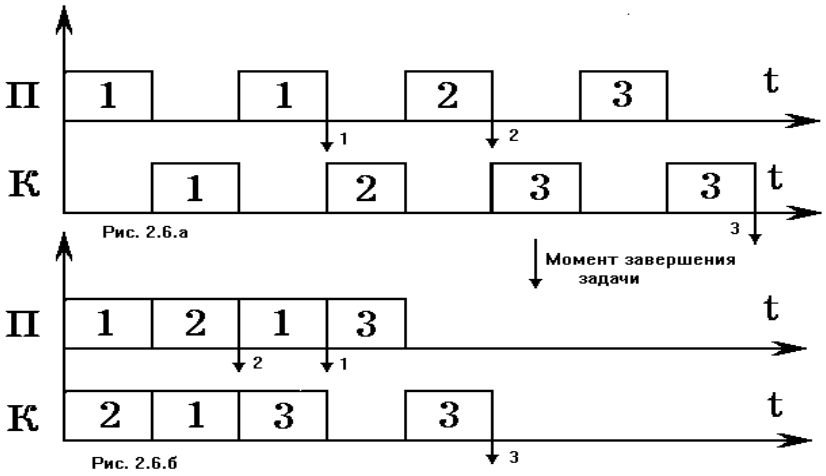


Рис. 2.6.а

Рис. 2.6.б

Если количество однотипных устройств увеличить, то и количество задач, обрабатываемых одновременно, будет увеличиваться. Для такой системы вводится критерий — степень мультипрограммирования. Степень мультипрограммирования, определяет количество заявок при которых система работает наиболее эффективно, а дальнейшее увеличение количества заявок не приводит к повышению эффективности работы вычислительной системы (рис. 2.6.б).

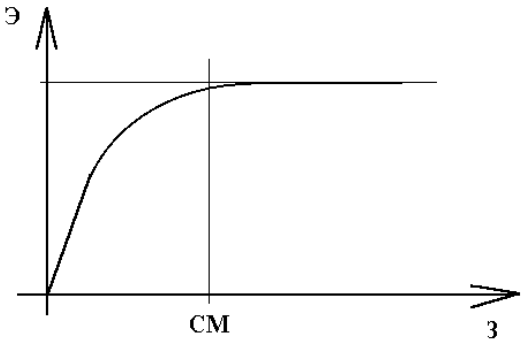


Рис.2.6.б. Э — эффективность, З — задачи,

СМ — степень мультипрограммирования

Под режимом *параллельной обработки* данных нескольких задач понимается такой многопрограммный режим, в котором переход от одной задачи к другой происходит через достаточно короткие промежутки времени (кванты), сравнимые с тактом машины, чтобы создать у пользователя впечатление одновременности исполнения нескольких программ (режим кажущегося совмещения).

Основной целью данного режима является улучшение обслуживания пользователей, выражающееся в том, что у последних создается впечатление отсутствия очереди, так как решение их задач не прерывается на длительные отрезки времени. Кроме того, если пользователю предоставляются некоторые средства прямого доступа (хотя бы для вывода информации), то это впечатление еще более усиливается выдачей результатов по мере их получения. Параллельная обработка основана на значительном отличии времени реакции пользователя и машины вследствии высокой скорости работы последней.

Однако, в общем случае, время ответа при этом режиме не оптимально, так как требуется тонкий механизм учета приоритета задач, учитываемый при определении очередности их обработки. Наибольшее применение получили алгоритмы, основанные на принципе сканирования очереди. Время выполнения пакета задач при параллельной обработке значительно превышает время их выполнения при единоличном использовании машины. Тем не менее, параллельная обработка задач, при которой пользователь может получать результаты, не ожидая конца обслуживания всего пакета, уменьшает время ожидания ответа по сравнению с режимом последовательной обработки.

В большинстве случаев параллельная обработка сочетается с мультипрограммированием, что обеспечивает наличие двух типов прерываний, вызывающих переход к другой программе: естественное прерывание во время ожидания

ввода/вывода (мультипрограммирование); вынужденное прерывание в конце отрезка времени, отводимого каждой программе (параллельная обработка).

Разделение времени — наиболее развитая форма многопрограммной работы, совмещающая мультипрограммирование с непосредственным доступом и обеспечением доступа некоторых привилегированных пользователей к ресурсам системы. Время ответа в такой системе близко к тому, которое было при единоличном использовании машины.

Как и при параллельной работе, задачи выполняются поочередно в течение некоторого отрезка времени (кванта), по завершении которого происходит вынужденное прерывание. Длительность этих отрезков времени (квантов) не является, как правило, фиксированной, а изменяется в зависимости от рассматриваемой задачи, а также от конкретных условий эксплуатации в момент передачи управления от одного пользователя другому. Выбор пользователя, задаче которого будет передано управление, и выделение этой задаче кванта времени осуществляет управляющая программа, являющаяся составной частью программ — диспетчеров, предназначенных для работы с разделением времени. Диспетчер функционирует в соответствии с выбранной для данной системы дисциплиной обслуживания заявок, которые будут рассмотрены ниже.

Существует не менее интересный способ реализации многопрограммного режима работы на однопроцессорной машине — "свопинг". Он используется при недостаточной памяти, для хранения всех параллельно выполняемых программ.

Свопинг характеризуется тем, что после определенного времени (кванта) программы пользователей, находящиеся в оперативной памяти, переписываются на внешний носитель информации (диск) в специально выделенные для этого своп-файлы, т.е. сохраняется состояние этих задач. Следующая совокупность параллельно выполняемых программ занимает освободившееся место в оперативной памяти, и система исполняет их в многопрограммном режиме до завершения следующего кванта.

Приемы, методы повышения эффективности вычислительной машины(за счет организации выч. процесса):
- применение дополнительных уровней памяти(многоуровневая память).
- просмотр команд вперед(подкачка). Буфер – либо одна самая длинна, либо 6 коротких
- секционирование памяти
- конвейерный принцип

2) Особенности планирования процессов в многопрограммной ОС в реальном времени.
Конспект

Работа в реальном времени – система работает в реальном времени, если время ответа системы ограничено внешними факторами.

Если в ОС есть хоть один режим, который нельзя прервать, то система не может работать в реальном времени.

Доп. инфа Разделение времени — наиболее развитая форма многопрограммной работы, совмещающая мультипрограммирование с непосредственным доступом и обеспечением доступа некоторых привилегированных пользователей к ресурсам системы. Время ответа в такой системе близко к тому, которое было при единоличном использовании машины.

Как и при параллельной работе, задачи выполняются поочередно в течение некоторого отрезка времени (кванта), по завершении которого происходит вынужденное прерывание. Длительность этих отрезков времени (квантов) не является, как правило, фиксированной, а изменяется в зависимости от рассматриваемой задачи, а также от конкретных условий эксплуатации в момент передачи управления от одного пользователя другому. Выбор пользователя, задаче которого будет передано управление, и выделение этой задаче кванта времени осуществляет управляющая программа, являющаяся составной частью программ — диспетчеров, предназначенных для работы с разделением времени. Диспетчер функционирует в соответствии с выбранной для данной системы дисциплиной обслуживания заявок.

Приемы, методы повышения эффективности вычислительной машины(за счет организации выч. процесса):
- применение дополнительных уровней памяти(многоуровневая память).
- просмотр канала вперед(подкачка). Буфер – либо одна самая длинна, либо 6 коротких
- секционирование памяти
- конвейерный принцип

3) Особенности выбора размера страниц.

Для того, чтобы правильно выбрать размер страницы, требуется знать хотя бы размер большинства программ. Если размер программ обозначить S, а размер страницы – P, количество страниц – S/P. Если зафиксировать размер записи в таблице страниц E, то количество памяти для таблицы страниц ES/P. Если в каждом кадре пустует пол. Страницы ES/P+P/2 – расход памяти на одном фрагменте.

4) Достоинства и недостатки файловой системы CP\M.

Ответ:

Прямое указание на кластер, где записан файл – с точки зрения надежности – самая приличная система (впервые реализовано в CP/M)

Стоит взглянуть на операционную систему CP/M по нескольким причинам. Во-первых, исторически это была очень важная система, ставшая прямым предшественником системы MS-DOS. Во-вторых, разработчики сегодняшних операционных систем и систем будущего, полагающие, что компьютеру требуется 32 Мбайт памяти, только чтобы загрузить в нее операционную систему, могут поучиться простоте системы, которой вполне хватало 16 Кбайт ОЗУ. В-третьих, в ближайшие десятилетия широкое применение получат встроенные системы. В связи с ограничениями в цене, размерах, весе и потребляемой мощности операционные системы, используемые, например, в часах, видео- и фотокамерах, радиоприемниках и сотовых телефонах, обязательно должны быть компактными, подобно операционной системе CP/M. Конечно, у этих систем не будет 8-дюймовых гибких дисков, но они вполне могут пользоваться электронными дисками во флэш-ОЗУ, на которых несложно организовать файловую систему, подобную CP/M.

Распределение памяти в системе CP/M показано на рис. 6.27. Наверху оперативной памяти (в ОЗУ) располагается базовая система ввода-вывода BIOS, содержащая базовую библиотеку — 17 вызовов ввода-вывода, используемых системой CP/M (в данном разделе мы рассмотрим CP/M 2.2, являвшуюся стандартной версией, когда CP/M была на вершине популярности). Эти системные вызовы предназначены для чтения и записи с гибкого диска, ввода с клавиатуры и вывода на экран.

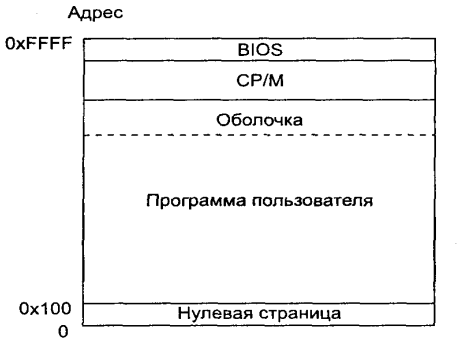


Рис. 6.27. Распределение памяти в системе CP/M

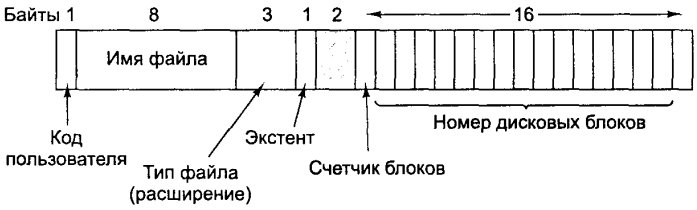


Рис. 6.28. Формат каталоговой записи в системе CP/M

Сразу под BIOS располагается сама операционная система. Для версии CP/M 2.2 ее размер составляет 3584 байт. Невероятно, но факт: вся операционная система занимала менее 4 Кбайт. Под операционной системой размещается оболочка (обработчик командных строк), съедающая еще 2 Кбайт. Остальная память используется для программ пользователя, кроме нижних 256 байт, зарезервированных для векторов аппаратных прерываний, некоторых переменных и буфера для текущей командной строки, в котором к ней могут получить доступ программы пользователя.

Причина, по которой система BIOS отделена от самой операционной системы CP/M (хотя обе системы располагаются в ОЗУ), заключается в переносимости. Операционная система CP/M взаимодействует с аппаратурой только с помощью обращений к BIOS. Для переноса системы CP/M на новую машину нужно всего лишь перенести туда BIOS. Когда это сделано, сама CP/M может быть установлена без изменений.

В файловой системе CP/M всего один каталог, содержащий записи фиксированного размера (32 байт). Размер каталога, фиксированный для данной реализации, может отличаться в других реализациях системы CP/M. В этом каталоге перечисляются все файлы системы. После загрузки система считывает каталог и рассчитывает битовый массив занятых и свободных блоков. Этот битовый массив, размер которого для 180-килобайтного диска составляет всего 23 байта, постоянно хранится в оперативной памяти. После завершения работы операционной системы он не сохраняется на диске. Благодаря такому подходу исчезает необходимость в проверке[1] непротиворечивости файловой системы на диске (вроде той, что выполняет программа *fsck* в UNIX) и сохраняется на диске один блок (в процентном отношении это эквивалентно сохранению 90 Мбайт на современном 16-гигабайтном диске).

Когда пользователь набирает команду, оболочка сначала копирует ее в буфер в нижние 256 байт памяти. Затем она ищет вызываемую программу и загружает ее в память по адресу 256 (над вектором прерываний), после чего передает управление по этому адресу. Программа начинает работу. Она обнаруживает свои параметры в буфере командной строки. Программе разрешается использовать память, занимаемую оболочкой, если ей нужно много памяти. Закончив работу, программа выполняет системный вызов CP/M, сообщая операционной системе, что следует перезагрузить оболочку (если занимаемая ею память использовалась программой) и запустить ее. В двух словах, вот, собственно, и весь рассказ об операционной системе CP/M.

Доп.инфа:

---

5 Семафоры и особенности их использования.

КОНСПЕКТ

Семафор должен быть доступен всем процессам. По этому он должен находиться в адресном пространстве супервизора или (ядро ОС)

И операции с ним производятся в режиме ядра. Помимо собственных задач семафора в структуре семафора храниться идентификатор процесса выполневшего последнюю операцию с семафором, число процессов ожидающих увеличения значения семфора, число процессов, ожид, когда значение семафора будет =0.

Семафоры

В 1965 году Дейкстра (E. W. Dijkstra) предложил использовать целую переменную для подсчета сигналов запуска, сохраненных на будущее [96]. Им был предложен новый тип переменных, так называемые **семафоры**, значение которых может быть нулем (в случае отсутствия сохраненных сигналов активизации) или некоторым положительным числом, соответствующим количеству отложенных активизирующих сигналов.

Дейкстра предложил две операции, down и up (обобщения sleep и wakeup). Операция down сравнивает значение семафора с нулем. Если значение семафора больше нуля, операция down уменьшает его (то есть расходует один из сохраненных сигналов активации) и просто возвращает управление. Если значение семафора равно нулю, процедура down не возвращает управление процессу, а процесс переводится в состояние ожидания. Все операции проверки значения семафора, его изменения и перевода процесса в состояние ожидания выполняются как единое и неделимое **элементарное действие**. Тем самым гарантируется, что после начала операции ни один процесс не получит доступа к семафору до окончания или блокирования операции. Элементарность операции чрезвычайно важна для разрешения проблемы синхронизации и предотвращения состояния состязания.

Операция up увеличивает значение семафора. Если с этим семафором связаны один или несколько ожидающих процессов, которые не могут завершить более раннюю операцию down, один из них выбирается системой (например, случайным образом) и ему разрешается завершить свою операцию down. Таким образом, после операции up, примененной к семафору, связанному с несколькими ожидающими процессами, значение семафора так и останется равным 0, но число ожидающих процессов уменьшится на единицу. Операция увеличения значения семафора и активизации процесса тоже неделима. Ни один процесс не может быть блокирован во время выполнения операции up, как ни один процесс не мог быть блокирован во время выполнения операции wakeup в предыдущей модели.

В оригинале Дейкстра использовал вместо down и up обозначения P и V соответственно. Мы не будем в дальнейшем использовать оригинальные обозначения, поскольку тем, кто не знает датского языка, эти обозначения ничего не говорят (да и тем, кто знает язык, говорят немного). Впервые обозначения down и up появились в языке Algol 68.