Оглавление

1. Какие основные компоненты входят в обобщенную структуру вычислительной системы?

Рис. 1. Пользователь и обобщенная структура вычислительной системы

2. Что такое техническое и программное обеспечение ЭВМ?

Аппаратное или техническое обеспечение (англ. hardware) – это оборудование, то есть процессоры, память, мониторы, дисковые устройства, накопители на магнитных лентах, сетевая коммуникационная аппаратура, принтеры и т. д., объединенные магистральным соединением (шиной).

В программном обеспечении (ПО) ВС выделяют две части – системное и прикладное.

Системное ПО – это набор программ, которые управляют компонентами ВС, такими как процессор, коммуникационные и периферийные устройства, и предназначены для обеспечения функционирования и работоспособности системы в целом.

Прикладное ПО – это набор программ, которые напрямую решают проблемы пользователя, предназначены для выполнения определенных пользовательских задач и рассчитаны на непосредственное взаимодействие с пользователем.

3. В чем отличие системного и прикладного программного обеспечения?

Системное ПО – это набор программ, которые управляют компонентами ВС, такими как процессор, коммуникационные и периферийные устройства, и предназначены для обеспечения функционирования и работоспособности системы в целом. Большинство из них отвечают непосредственно за контроль и объединение в единое целое различных компонентов аппаратного оборудования ВС, обеспечение работы компьютера самого по себе и выполнение различных прикладных программ.

Системное ПО противопоставляется **прикладному** ПО, которое напрямую решает проблемы пользователя и предназначено для выполнения определенных пользовательских задач и рассчитано на непосредственное взаимодействие с пользователем. К прикладному ПО, как правило, относят разнообразные вспомогательные программы (игры, текстовые процессоры и т. п.)

Следует отметить, что деление на прикладное и системное ПО является отчасти условным и зависит от того, кто осуществляет такое деление.

4. Какие основные цели преследуют разработчики ОС?

- Главными целями разработчиков операционных систем являются следующие:
- Эффективное использование всех компьютерных ресурсов.
- Повышение производительности труда программистов.
- Простота, гибкость, эффективность и надежность организации вычислительного процесса.
- Обеспечение независимости прикладного ПО от аппаратного ПО.

5. Какие определения операционной системы вам известны?

1. Операционная система (ОС) – это программа, которая обеспечивает возможность рационального использования оборудования компьютера удобным для пользователя образом.

2. ОС– базовый комплекс компьютерных программ, обеспечивающий управление аппаратными средствами компьютера, работу с файлами, ввод и вывод данных, а также выполнение прикладных программ и утилит.

Кроме различных определений ОС, два из которых приведены выше, пользователи выделяют ряд различных «точек зрения» на ОС:

- ОС как виртуальная машина;
- ОС как система управления ресурсами;
- ОС как защитник пользователей и программ;
- ОС как постоянно функционирующее ядро.

6. Что понимают под ОС как виртуальной машиной?

Использование архитектуры персонального компьютера на уровне машинных команд является крайне неудобным для применения прикладных программ. В связи с этим необходимо обеспечить интерфейс между пользователем и компьютером, скрывая лишние подробности за счет использования относительно простых и высокоуровневых абстракций.

Например, представлять информационное пространство диска как набор файлов, которые можно открывать для чтения или записи, использовать для получения или сброса информации, а затем закрывать, создавать иллюзию неограниченного размера операционной памяти, числа процессоров и прочее. Обеспечением такого высокоуровневого абстрагирования занимается ОС, что позволяет представлять ее пользователю в виде виртуальной машины, с которой проще иметь дело, чем непосредственно с оборудованием компьютера.

7. Что понимают под ОС как системой управления ресурсами?

В случае, если несколько программ, работающих на одном компьютере, будут пытаться одновременно осуществлять вывод на принтер, то можно получить «мешанину» строчек и страниц. ОС должна предотвращать такого рода хаос за счет буферизации подобной информации и организации очереди на печать. Не менее актуальная проблема – проблема управления ресурсами для многопользовательских компьютеров.

Таким образом, ОС как менеджер ресурсов осуществляет упорядоченное и контролируемое распределение процессоров, памяти и других ресурсов между различными программами.

8. Что понимают под ОС как «защитника» пользователей и программ?

Если в вычислительной системе требуется обеспечение совместной работы нескольких пользователей, то возникает проблема организации их безопасной деятельности. Так, необходимо обеспечить:

- сохранность информации на диске, защиту от повреждения или
- удаления файлов;
- разрешение программам одних пользователей произвольно вмешиваться в работу программ других пользователей;
- пресечение попыток несанкционированного использования вычислительной системы.

Эти задачи, как правило, возложены на ОС как организатора безопасной работы пользователей и их программ.

9. Что понимают под ОС как постоянно функционирующее ядро?

Можно говорить об ОС как о программе(программах), постоянно работающей на компьютере и взаимодействующей с множеством прикладных программ. Очевидно, что такое определение верно лишь отчасти, т. к. во многих современных ОС постоянно работает на компьютере лишь часть ОС, которую принято называть ее ядром.

10. Какие этапы эволюции вам известны? В чем их суть?

Первое поколение (1940–50-е гг.):

- Первые ламповые вычислительные устройства и принцип программы, хранящейся в памяти машины (Джон фон Нейман, 1945 г.);
- Одна и та же группа людей участвует в проектировании, эксплуатации и программировании ВМ;
- ОС отсутствуют.
- ВС выполняет одновременно только одну операцию (ввод-вывод или вычисления);
- Первое системное ПО – прообразы компиляторов (Fortran) в 1951-1952 гг., Ассемблер в 1954;
- Выполнение программ строго последовательно;
- Высокая стоимость ВС при их малом количестве и низкой эффективности использования.

Второй период (1950–60-е гг.):

Новая техническая база – полупроводниковые элементы. Повышение надёжности, снижение энергопотребления. Бурное развитие алгоритмических языков. Разделение на программистов и операторов, специалистов по эксплуатации и разработчиков. Первые системы пакетной обработки – прообразы современных ОС. Часть машинного времени тратится на выполнение системной управляющей программы. Программа, получившая доступ к процессору, обслуживается до её завершения (наличие простоев).

Третий период (1960-70-е гг.):

- Переход к ИС;
- Появление spooling (сокращение от Simultaneous Peripheral Operation On Line), или подкачки-откачки данных;
- Разработка аппарата прерываний для совмещения операций ввода-вывода одного задания с выполнением другого;
- Использование магнитных дисков, возможность выбора очередного выполняемого задания при обработке пакета заданий, планирование заданий;
- Реализация идеи мультипрограммирования;
- Реализация идей совместимых компьютеров и стандартизации ОС (серии IBM/360 с ОС OS/360);
- Появление первых ОС реального времени;

Четвёртый период (1970-1980-е гг.):

- Появление вытесняющей многозадачности;
- Использовании концепции баз данных для хранения больших объёмов информации;
- Введение приоритетного планирования и систем разделения времени;
- Использование механизма виртуальной памяти (неполное нахождение программы в оперативной памяти) для создания иллюзии неограниченной оперативной памяти ЭВМ;

Пятый период (Середина 1980-х – наше время):

- Появление микропроцессоров;
- Виртуализация ресурсов ЭВМ;
- Возникновение сетевых компьютеров, все данные получающих через компьютерную сеть;
- Развитие технологии «клиент-сервер».

11. В чем преимущества пакетной обработки заданий? Что такое spooling?

Пакетная обработка заданий направлена против простоев процессора, вызванных сменой входных ресурсов при выполнении одного и того же задания. Системы пакетной обработки автоматизируют запуск одной программы из пакета за другой и тем самым увеличивают коэффициент загрузки процессора.

Spooling (сокращение от Simultaneous Peripheral Operation On Line), или подкачка-откачка данных – выполнение действительных операций ввода-вывода в режиме онлайн, на том же компьютере, который производит вычисления.

12. В чем заключается основная идея мультипрограммирования?

Идея мультипрограммирования заключается в следующем: пока одна программа выполняет операцию ввода-вывода, процессор не простаивает, как это происходило при однопрограммном режиме, а выполняет другую программу. Когда операция ввода-вывода заканчивается, процессор возвращается к выполнению первой программы. При этом каждая программа загружается в свой участок оперативной памяти, называемый разделом, и не должна влиять на выполнение другой программы.

13. Какие изменения потребовало мультипрограммирование в строении вычислительной системы? В чем их суть?

Наиболее существенные особенности аппаратной поддержки мультипрограммирования:

- Реализация защитных механизмов. Программы не должны иметь самостоятельного доступа к распределению ресурсов, что приводит к появлению привилегированных и непривилегированных команд. Привилегированные команды, например, команды ввода-вывода, могут исполняться только операционной системой. Говорят, что она работает в привилегированном режиме. Переход управления от прикладной программы к ОС сопровождается контролируемой сменой режима. Кроме того, это защита памяти, позволяющая изолировать конкурирующие пользовательские программы друг от друга, а ОС – от программ пользователей.
- Наличие прерываний. Внешние прерывания оповещают ОС о том, что произошло асинхронное событие, например, завершилась операция ввода-вывода. Внутренние прерывания (сейчас их принято называть исключительными ситуациями) возникают, когда выполнение программы привело к ситуации, требующей вмешательства ОС, например, деление на ноль или попытка нарушения защиты.
- Развитие параллелизма в архитектуре. Прямой доступ к памяти и организация каналов ввода-вывода позволили освободить центральный процессор от рутинных операций.

14. Какие основные операции при организации мультипрограммирования реализуются ОС?

- организация интерфейса между прикладной программой и ОС при помощи системных вызовов;
- организация очереди из заданий в памяти и выделение процессора одному из заданий потребовали планирования использования процессора;
- переключение с одного задания на другое требует сохранения содержимого регистров и структур данных, необходимых для выполнения задания, иначе говоря, контекста для обеспечения правильного продолжения вычислений;
- поскольку память является ограниченным ресурсом, нужны стратегии управления памятью, то есть требуется упорядочить процессы размещения, замещения и выборки информации из памяти;
- организация хранения информации на внешних носителях в виде файлов и обеспечение доступа к конкретному файлу только определенным категориям пользователей;
- поскольку программам может потребоваться произвести санкционированный обмен данными, необходимо их обеспечить средствами коммуникации;
- для корректного обмена данными необходимо разрешать конфликтные ситуации, возникающие при работе с различными ресурсами и предусмотреть координацию программами своих действий, т. е. снабдить систему средствами синхронизации.

15. Что такое операционная система реального времени? Какое ее предназначение и основные характеристики?

Операционная система реального времени применяется в тех случаях, когда существует предельно допустимое время, в течение которого должна быть выполнена та или иная программа, управляющая объектом.

Характерным для ОСРВ является обеспечение заранее заданных интервалов времени реакции на предусмотренные события для получения управляющего воздействия – реактивность. Поскольку в технологических процессах промедление может привести к нежелательным и даже опасным последствиям, это реализовано за счет проектирования ОСРВ для работы в наихудших условиях.

16. Что такое системы разделения времени?

Системы, при использовании которых процессор переключается между задачами не только на время операций ввода-вывода, но и через определённые интервалы времени. Эти переключения происходят так часто, что пользователи могут взаимодействовать со своими программами во время их выполнения, то есть интерактивно. В результате появляется возможность одновременной работы нескольких пользователей на одной компьютерной системе.

17. Какие функции ОС относят к интерфейсным, а какие к внутренним

Обзор этапов развития вычислительных и операционных систем позволяет все функции ОС условно разделить на две различные группы – интерфейсные и внутренние. К интерфейсным функциям ОС относят:

- управление аппаратными средствами;
- управление устройствами ввода-вывода;
- поддержку файловой системы;
- поддержку многозадачности (разделение использования памяти, времени выполнения);
- ограничение доступа, многопользовательский режим работы, планирование доступа пользователей к общим ресурсам;
- интерфейс пользователя (команды в MS DOS, Unix; графический интерфейс в ОС Windows);
- поддержку работы с общими данными в режиме коллективного пользования;
- поддержку работы в локальных и глобальных сетях.

К внутренним функциям ОС, которые выделились в процессе эволюции вычислительных и операционных систем, следует отнести:

- реализацию обработки прерываний;
- управление виртуальной памятью;
- планирование использования процессора;
- обслуживание драйверов устройств.

18. В каких архитектурах реализуют операционные системы?

Монолитное ядро, микроядерная архитектура, многоуровневые системы, ВМ

19. Чем характеризуется ОС, реализованная в архитектуре «монолитное ядро»?

- каждая процедура может вызвать любую другую;
- все процедуры работают в привилегированном режиме;
- все части монолитного ядра работают в одном адресном пространстве;
- ядро «совпадает» со всей ОС;
- сборка (компиляция) ядра осуществляется отдельно для каждого компьютера, при установке, добавлении или исключении отдельных компонент требуется перекомпиляция;
- старейший способ организации ОС.

20. В чем отличие работы аппаратуры компьютера в пользовательском режиме и в режиме ядра?

Пользовательский режим - наименее привилегированный режим, поддерживаемый NT; он не имеет прямого доступа к оборудованию и у него ограниченный доступ к памяти.

Режим ядра - привилегированный режим. Те части NT, которые исполняются в режиме ядра, такие как драйверы устройств и подсистемы типа Диспетчера Виртуальной Памяти, имеют прямой доступ ко всей аппаратуре и памяти.

21. Какие недостатки вызывает использование ОС в архитектуре монолитное ядро? Что означает модульное ядро?

Монолитность ядер усложняет их отладку, понимание кода ядра, добавление новых функций и возможностей, удаление кода, унаследованного от предыдущих версий. «Разбухание» кода монолитных ядер также повышает требования к объёму оперативной памяти, требуемому для функционирования ядра ОС. Старые монолитные ядра требовали перекомпиляции при любом изменении состава оборудования.

Большинство современных ядер позволяют во время работы динамически подгружать модули, выполняющие части функции ядра. Такие ядра называются модульными ядрами.

22. Чем характеризуется ОС, реализованная в микроядерной архитектуре?

При разработке ОС используют подход, при котором значительную часть системного кода переносят на уровень пользователя с одновременной минимизацией ядра. Построение ядра ОС осуществляется так, что большинство составляющих ОС являются самостоятельными программами, а взаимодействие между ними обеспечивает специальный модуль ядра – **микроядро**, работающее в привилегированном режиме и обеспечивающее взаимодействие между программами, планирование использования процессора, первичную обработку прерываний, операции ввода-вывода и базовое управление памятью.

23. В чем заключаются достоинства и недостатки ОС, реализованных в микроядерной архитектуре?

Преимущества:

- упрощенное добавление и отладка компонентов ОС без необходимости перезапуска системы за счет высокой степени модульности ядра;
- возможность без прерывания работы системы загружать и выгружать новые драйверы, файловые системы и т. д.;
- возможность отладки компонентов ядра с помощью обычных программных средств;
- повышенная надежность системы (ошибка на уровне непривилегированной программы менее опасна, чем отказ на уровне режима ядра).

К недостаткам построения ОС в данной архитектуре относят:

- дополнительные накладные расходы, связанные с передачей сообщений и возникающие за счет частого переключения из защищенного режима ядра в незащищенный, в котором функционируют остальные модули;
- усложнение процесса проектирования при попытке снижения возможных накладных расходов (требуется «аккуратное» проектирование, разбиение системы на компоненты, минимизация взаимодействия между ними).

24. Чем характеризуется многоуровневая ОС? В чем ее достоинства и недостатки?

Обеспечивая строгую структуризацию, можно представить всю вычислительную систему в виде ряда уровней с хорошо определенными связями между ними. При этом объекты уровня N могут вызывать только объекты уровня N - 1. Чем ниже уровень, тем более привилегированные команды и действия может выполнять модуль, находящийся на этом уровне.

В качестве достоинства многоуровневых систем отмечают:

- простоту реализации (за счет того, что при использовании операций нижнего слоя не нужно знать, как они реализованы, нужно лишь понимать, что они делают);
- простоту тестирования (отладка осуществляется послойно и при возникновении ошибки всегда легко локализовать ошибку);
- простоту модификации (при необходимости можно заменить лишь один слой, не трогая остальные).

К недостаткам относят:

- сложность разработки (непросто верно определить порядок и состав каждого из слоев);
- меньшую по сравнению с монолитными системами эффективность за
- счет необходимости прохождения целого ряда слоев (например, для выполнения операций ввода-вывода программе пользователя придется последовательно проходить все слои от верхнего до нижнего).

25. Какова обобщенная структура виртуальной машины? Назовите примеры современных виртуальных машин.

Программа пользователя	Программа пользователя	Программа пользователя
MS-DOS	*Linux*	*Windows NT*
Виртуальное hardware	Виртуальное hardware	Виртуальное hardware
Реальная операционная система		
Реальное аппаратное обеспечение		

На виртуальную машину, можно инсталлировать ОС, у виртуальной машины может быть BIOS, оперативная память, жёсткий диск (выделенное место на жёстком диске реального компьютера), могут эмулироваться периферийные устройства. На одном компьютере может функционировать несколько виртуальных машин.

Примеры современных ВМ: Bochs, DOSBox, Virtual PC, Parallels Workstation, QEMU, VirtualBox, VMware Fusion, VMware Workstation.

26. Что такое смешанные (гибридные) ОС? Какие ОС реализованы в этой архитектуре?

В большинстве случаев современные ОС используют различные комбинации подходов, реализуя смешанные (гибридные) ОС. Например, ядро ОС *Linux* представляет собой монолитную систему с элементами микроядерной архитектуры. Системы 4.4*BSD* и *MkLinux* – монолитные ОС, работающие на микроядре *Mach* (микроядро обеспечивает управление виртуальной памятью и работу низкоуровневых драйверов; остальные функции, в том числе взаимодействие с прикладными программами, осуществляется монолитным ядром). Совместно элементы микроядерной архитектуры и элементы монолитного ядра используются в ядре *Windows NT*:

- компоненты ядра *Windows NT* располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений, как и положено в микроядерных ОС;
- все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных.

27. По каким признакам классифицируют ОС?

Прежде всего, традиционно различают ОС **общего** и **специального** назначения. Системы специального назначения, в свою очередь, подразделяются на ОС для носимых микрокомпьютеров и различных встроенных систем.

По *режиму обработки* задач различают ОС, обеспечивающие **однопрограммный** и **мультипрограммный** (мультизадачный, многозадачный) режимы. Также многозадачные ОС подразделяют на различные типы в соответствии с использованными при их разработке критериями эффективности:

- *системы пакетной обработки* (например, *ЕС*, критерий – коэффициент загрузки процессора);
- *системы разделения времени* (*Unix, VMS*, критерий – удобство и эффективность работы пользователей при одновременном выполнении нескольких пользовательских приложений);
- *системы реального времени* (*QNX, RT*/11, критерий – реактивность).

Также ОС классифицируют по архитектуре, в которой они реализованы (монолитное ядро, микроядерная архитектура, многоуровневые системы, ВМ).

28. В чем отличие мультипрограммного режима работы ОС от мультизадачного?

Принципиальное отличие этих понятий заключается в том, что мультипрограммный режим обеспечивает параллельное выполнение нескольких приложений, и при этом программисты, создающие эти программы, не должны заботиться о механизмах организации их параллельной работы (эти функции берет на себя сама ОС).

Мультизадачный режим, наоборот, предполагает, что забота о параллельном выполнении и взаимодействии приложений ложится как раз на прикладных программистов.

29. Что означают понятия вытесняющая и невытесняющая многозадачность?

Невытесняющая многозадачность (*NetWare*, *Windows* 3.x). В этом случае активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление ОС для того, чтобы та выбрала из очереди другой готовый к выполнению процесс.

Вытесняющая многозадачность (*Windows NT, OS*/2, *Unix*). При вытесняющей многозадачности решение о переключении процессора с одного процесса на другой принимается ОС, а не самим активным процессом.

30. Что означает симметричная и ассиметричная мультипроцессорная обработка?

Асимметричная ОС целиком выполняется только на одном из процессоров системы, распределяя прикладные задачи по остальным процессорам.

Симметричная ОС полностью децентрализована и использует весь пул процессоров, разделяя их между системными и прикладными задачами.

31. В чем отличие сетевых и распределенных ОС?

Сетевая ОС характеризуется тем, что наделена развитыми функциями работы с сетью, а также контроля доступа к файлам (систему прав доступа). К сетевым ОС относят как системы для рабочих мест (*Novell for DOS, MS Windows, GNU/Linux*), так и серверные ОС (*GNU/Linux*, семейство *BSD*-систем, серверные версии *MS Windows, NetWare*), а также специализированные ОС сетевого оборудования (*Cisco IOS5*).

При использовании **распределенной ОС** пользователь не знает, на локальной или удаленной машине хранятся его файлы и выполняются его программы, он может не знать, подключен ли его компьютер к сети. Внешне распределенная ОС выглядит как обычная автономная система, а ее внутреннее строение имеет существенные отличия от автономных систем.

32. Каковы характерные черты мультипрограммирования в современных ОС?

Мультипрограммирование – это режим обработки данных, при котором ресурсы вычислительной системы предоставляются каждому процессу из группы процессов обработки данных, находящихся в вычислительной системе, на интервалы времени, длительность и очередность предоставления которых определяется управляющей программой этой системы с целью обеспечения одновременной работы в интерактивном режиме.

Суть - пока одна программа ожидает завершения очередной операции ввода-вывода, другая программа может быть поставлена на решение. Это позволяет более полно использовать имеющиеся ресурсы и уменьшить общее время, необходимое для решения множества задач.

33. Какие критерии используются для определения эффективности мультипрограммных вычислительных систем?

Наиболее характерными критериями эффективности мультипрограммных вычислительных систем являются:

- Пропускная способность – количество задач, выполняемых вычислительной системой в единицу времени;
- Удобство работы пользователей, заключающееся, в частности, в том, что они имеют возможность интерактивно работать одновременно с несколькими приложениями на одной машине;
- Реактивность системы– способность системы выдерживать заранее заданные (возможно, очень короткие) интервалы времени между запуском программы и получением результата.

34. Как формируется «мультипрограммная смесь» в системах пакетной обработки?

В начале работы формируется пакет заданий, каждое из которых содержит требование к системным ресурсам. Из пакета заданий формируется мультипрограммная «смесь», то есть множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие разные требования к ресурсам, так, чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины. Например, в мультипрограммной смеси желательно одновременное присутствие вычислительных задач и задач с интенсивным вводом-выводом.

36. Какие особенности имеет мультипрограммирование в системах пакетной обработки?

Основное предназначение систем пакетной обработки – решать задачи вычислительного характера, не требующие быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки является пропускная способность, то есть решение числа задач в единицу времени.

В таких системах выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, то есть выбирается «выгодное» в некотором смысле задание. Следовательно, в вычислительных системах, работающих под управлением пакетных ОС, невозможно гарантировать выполнение того или иного задания в течение определенного периода времени.

Кроме того, переключение процессора с выполнения одной задачи на выполнение другой происходит по инициативе самой активной задачи, например, когда она отказывается от процессора из-за необходимости выполнить операцию ввода-вывода. Поэтому существует высокая вероятность того, что одна задача может надолго занять процессор и выполнение интерактивных задач станет невозможным. Такой подход повышает эффективность функционирования аппаратуры, но снижает эффективность работы пользователя.

37. Какие критерии к «мультипрограммной смеси» применяют в системах разделения времени?

Критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя. Системы разделения времени призваны исправить основной недостаток систем пакетной обработки – изоляцию пользователя-программиста от процесса выполнения его задач. Каждому пользователю предоставляется терминал, с которого он может вести диалог со своей программой. В связи с тем, что в системах разделения времени каждой задаче выделяется только квант процессорного времени, ни одна задача не занимает процессор надолго, и время ответа оказывается приемлемым.

38. Какие особенности имеет мультипрограммирование в системах разделения времени?

Системы разделения времени обладают меньшей пропускной способностью, чем системы пакетной обработки, так как на выполнение принимается каждая запущенная пользователем задача, а не та, которая «выгодна» системе. Кроме того, производительность системы снижается из-за возросших накладных расходов вычислительной мощности на более частое переключение процессора с задачи на задачу.

Это вполне соответствует тому, что критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя. Вместе с тем мультипрограммное выполнение интерактивных приложений повышает пропускную способность компьютера (пусть и не в такой степени, как пакетные системы). Аппаратура загружается более эффективно, поскольку в то время, пока одно приложение ждет сообщения пользователя, другие приложения могут обрабатываться процессором.

39. Какие критерии к «мультипрограммной смеси» применяют в системах реального времени?

Критерием эффективности ОСРВ является способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата (управляющего воздействия). Это время называется временем реакции системы, а соответствующее свойство системы – реактивностью.

40. Какие особенности имеет мультипрограммирование в системах реального времени?

В ОСРВ «мультипрограммная смесь» представляет собой фиксированный набор заранее разработанных программ, а выбор программы на выполнение осуществляется по прерываниям (исходя из текущего состояния объекта) или в соответствии с расписанием плановых работ.

В ОСРВ не стремятся максимально загружать все устройства, наоборот, при проектировании программного управляющего комплекса обычно закладывается некоторый «запас» вычислительной мощности на случай пиковой нагрузки. Статистические аргументы о низкой вероятности возникновения пиковой нагрузки, основанные на том, что вероятность одновременного возникновения большого количества независимых событий очень мала, не применимы ко многим ситуациям в реальных системах управления. Если ОСРВ не спроектирована для поддержки пиковой нагрузки, то может случиться так, что система не справится с работой именно тогда, когда она нужна в наибольшей степени.

41. Чем характеризуется мультипроцессорная обработка? На какие виды ее разделяют?

Мультипроцессорная обработка – это способ организации вычислительного процесса в системах с несколькими процессорами, при котором несколько задач (процессов, потоков) могут одновременно выполняться на разных процессорах системы.

Мультипроцессорные системы разделяют на симметричные и несимметричные (асимметричные). При этом следует четко разделять, к какому аспекту мультипроцессорной системы относится эта характеристика – к типу архитектуры или к способу организации вычислительного процесса.

42. Что означает симметричная и несимметричная мультипроцессорная обработка с точки зрения архитектуры?

Симметричная архитектура мультипроцессорной системы предполагает однородность всех процессоров и единообразие включения процессоров в общую схему системы. Традиционные симметричные мультипроцессорные конфигурации разделяют одну большую память между всеми процессорами и используют одну схему отображения памяти. Они могут очень быстро обмениваться данными так, что обеспечивается достаточно высокая производительность для тех приложений (например, при работе с базами данных), в которых несколько задач должны активно взаимодействовать между собой. В настоящее время функции поддержки симметричной мультипроцессорной обработки данных имеются во всех популярных ОС.

Возможность наращивания числа процессоров (масштабируемость) в симметричных системах ограничена вследствие того, что все они пользуются одной и той же оперативной памятью и, следовательно,

должны располагаться в одном корпусе. Такая конструкция, называемая масштабируемой по вертикали, практически ограничивает число процессоров до четырех или восьми.

В ***асимметричной архитектуре*** разные процессоры могут отличаться как своими характеристиками, так и функциональной ролью, которая поручается им в системе. Функциональная неоднородность в асимметричных архитектурах влечет за собой структурные отличия во фрагментах системы, содержащих разные процессоры системы.

Масштабирование в асимметричной архитектуре реализуется иначе, чем в симметричной. В связи с тем, что требование «единого корпуса» отсутствует, система может состоять из нескольких устройств, каждое из которых содержит один или несколько процессоров – масштабирование по горизонтали. Каждое такое устройство называется кластером, а вся мультипроцессорная система– кластерной.

43. Что означает симметричная и несимметричная мультипроцессорная обработка с точки зрения организации вычислительного процесса?

Асимметричное мультипроцессирование является наиболее простым способом организации вычислительного процесса в системах с несколькими процессорами. Этот способ иногда условно называют «ведущий-ведомый». Функционирование системы по принципу «ведущий–ведомый» предполагает выделение одного из процессоров в качестве «ведущего», на котором работает ОС и который управляет всеми остальными «ведомыми» процессорами. В этом случае «ведущий» процессор берет на себя функции распределения задач и ресурсов, а «ведомые» процессоры работают только как обрабатывающие устройства и никаких действий по организации работы вычислительной системы не выполняют. Учитывая то, что ОС работает исключительно на одном процессоре и функции управления централизованы, то такая система по сложности схожа с ОС однопроцессорной системы.

Следует отметить, что асимметричная организация вычислительного процесса может быть реализована как для симметричной мультипроцессорной архитектуры, в которой все процессоры аппаратно неразличимы, так и для несимметричной, для которой характерна неоднородность процессоров, их специализация на аппаратном уровне. В архитектурно-асимметричных системах на роль ведущего может быть назначен наиболее надежный и производительный процессор. Специализация процессоров снижает надежность системы в целом, так как процессоры не являются взаимозаменяемыми.

Симметричное мультипроцессирование как способ организации вычислительного процесса (процессоры работают с общими устройствами и разделяемой основной памятью) может быть реализовано исключительно в системах с симметричной мультипроцессорной архитектурой. Симметричное мультипроцессирование реализуется ОС, общей для всех процессоров.

При симметричной организации все процессоры равноправно участвуют и в управлении вычислительным процессом, и в выполнении прикладных задач. Разные процессоры могут в какой-то момент одновременно обслуживать как разные, так и одинаковые модули общей ОС, однако при этом ОС должна обладать свойством реентерабельности. Следует отметить, что большим преимуществом симметричных систем перед асимметричными системами является то, что в случае отказа одного из процессоров симметричные системы, как правило, значительно легче реконфигурируются.

44. Что означает масштабирование в многопроцессорной архитектуре по вертикали? По горизонтали?

Масштабируемость по вертикали заключается в наращивании количества процессоров, располагающихся в одном корпусе. Это единственный вид масштабируемости, доступный для архитектурно-симметричных систем вследствие того, что все процессоры таких систем пользуются одной и той же оперативной памятью и, следовательно, должны располагаться в едином корпусе. Такая конструкция практически ограничивает число процессоров до четырех или восьми.

Масштабирование по горизонтали применяется для архитектурно-асимметричных систем. В связи с тем, что требование «единого корпуса» отсутствует, такая система может состоять из нескольких устройств, каждое из которых содержит один или несколько процессоров. Каждое такое устройство называется кластером, а вся мультипроцессорная система – кластерной.

45. Что такое «кластерная мультипроцессорная система»?

Кластерной называется мультипроцессорная система, состоящая из нескольких кластеров. Каждый кластер – это устройство, содержащее один или несколько процессоров и располагающееся в отдельном корпусе. Масштабирование мультипроцессорной системы путём увеличения количества кластеров называется «масштабированием по горизонтали» и применяется обычно для архитектурно-асимметричных систем.

46. Какова роль прерываний при мультипрограммировании?

Прерывания представляют собой механизм, позволяющий координировать параллельное функционирование отдельных устройств вычислительной системы и реагировать на особые состояния, возникающие при работе процессора, то есть прерывание – это принудительная передача управления от выполняемой программы к системе (а через нее – к соответствующей программе обработки прерывания), происходящая при возникновении определенного события. Основная цель введения прерываний – реализация асинхронного режима функционирования и распараллеливание работы отдельных устройств вычислительного комплекса.

47. На какие классы делят прерывания? В чем отличия этих классов между собой?

В зависимости от источника все прерывания делят на два класса:

- аппаратные (внешние и внутренние);
- программные.

Аппаратные прерывания – события от периферийных устройств или события в микропроцессоре, возникающие вследствие подачи некоторой аппаратурой электрического сигнала, который передается на специальный вход прерывания процессора. При этом внешними будут прерывания, инициированные периферийными устройствами (например, нажатия клавиш клавиатуры, движение мыши, сигнал от таймера, сетевой карты или дискового накопителя), а внутренними – те, что происходят в микропроцессоре.

Внешние прерывания являются асинхронными по отношению к потоку инструкций прерываемой программы. Аппаратура процессора работает так, что асинхронные прерывания возникают между выполнением двух соседних инструкций, при этом система после обработки прерывания продолжает выполнение процесса, уже начиная со следующей инструкции.

Внутренние прерывания, называемые также исключениями, происходят в микропроцессоре и инициируются синхронно выполнению программы при появлении аварийной ситуации (например, деление на ноль или обращение по несуществующему адресу) в ходе исполнения некоторой инструкции программы – возникают непосредственно в ходе выполнения тактов команды («внутри» выполнения).

Программные прерывания возникают при исполнении особой команды процессора, которая имитирует прерывание, т. е. переход на новую последовательность инструкций. Этот механизм был специально введен для того, чтобы переключение на системные программные модули происходило не просто как переход на подпрограмму, а точно таким же образом, как и обычное прерывание. Этим прежде всего обеспечивается автоматическое переключение процессора в привилегированный режим с возможностью исполнения любых команд ОС.

48. Какова пошаговая последовательность реализации прерываний в вычислительной системе? Каковы при этом главные функции механизма прерываний?

Механизм обработки прерываний независимо от архитектуры вычислительной системы подразумевает выполнение некоторой последовательности шагов.

- Шаг 1. Установление факта прерывания (прием сигнала запроса на прерывание) и идентификация прерывания (в ОС идентификация прерывания иногда осуществляется повторно, на шаге 4).
- Шаг 2. Запоминание состояния прерванного процесса вычислений.
- Состояние процесса выполнения программы определяется, прежде всего, значением счетчика команд (адресом следующей команды), содержимым регистров процессора, и может включать также спецификацию режима (например, режим пользовательский или привилегированный) и другую информацию.

- Шаг 3. Управление аппаратно передается на подпрограмму обработки прерывания. В простейшем случае в счетчик команд заносится начальный адрес подпрограммы обработки прерываний, а в соответствующие регистры – информация из слова состояния.
- Шаг 4. Сохранение информации о прерванной программе, которую не удалось спасти на шаге 2 с помощью аппаратуры. В некоторых процессорах предусматривается запоминание довольно большого объема информации о состоянии прерванных вычислений.
- Шаг 5. Выполнение программы, связанной с обработкой прерывания. Эта работа может быть выполнена той же подпрограммой, на которую было передано управление на шаге 3, но в ОС достаточно часто она реализуется путем последующего вызова соответствующей подпрограммы.
- Шаг 6. Восстановление информации, относящейся к прерванному процессу (этап, обратный шагу 4).
- Шаг 7. Возврат на прерванную программу.

Следует отметить, что шаги 1–3 реализуются аппаратно, а шаги 4–7 – программно.

Главные функции механизма прерываний– это:

- распознавание или классификация прерываний;
- передача управления соответствующему обработчику прерываний;
- корректное возвращение к прерванной программе

49. Что такое «обработчик прерывания»? В чем его функция?

Обработчик прерываний – специальная процедура, вызываемая по прерыванию для выполнения его обработки. Обработчики прерываний могут выполнять множество функций, в зависимости от причины, вызвавшей прерывание. Вид функции, исполняемой обработчиком в каждом конкретном случае, определяется в соответствии с таблицей векторов прерываний, размещаемой в определенном месте адресного пространства.

50. В какие моменты при обработке прерывания система должна отключать возможности прерывания?

Возможности прерывания должны отключаться в первой секции подпрограммы обработки прерываний перед сохранением контекста прерванных вычислений, а также в последней перед восстановлением контекста. После сохранения или восстановления контекста возможность прерываний снова включается.

51. Для чего в ОС используется супервизор прерываний? Каково его место в общей схеме обработки прерываний?

Первые секции подпрограмм обработки прерываний выделяются в специальный системный программный модуль – супервизор прерываний.

Супервизор прерываний сохраняет в дескрипторе текущей задачи рабочие регистры процессора, определяющие контекст прерываемого вычислительного процесса. Далее он определяет ту подпрограмму, которая должна выполнить действия, связанные с обслуживанием настоящего (текущего) запроса на прерывание. Наконец, перед тем, как передать управление на эту подпрограмму, супервизор прерываний устанавливает необходимый режим обработки прерывания (осуществляется маскирование, определяющее запрет некоторых сигналов прерывания).

52. Что такое «приоритет прерывания»? Обработка прерываний каких компонентов вычислительной системы обладает более высоким, а каких – более низким приоритетом?

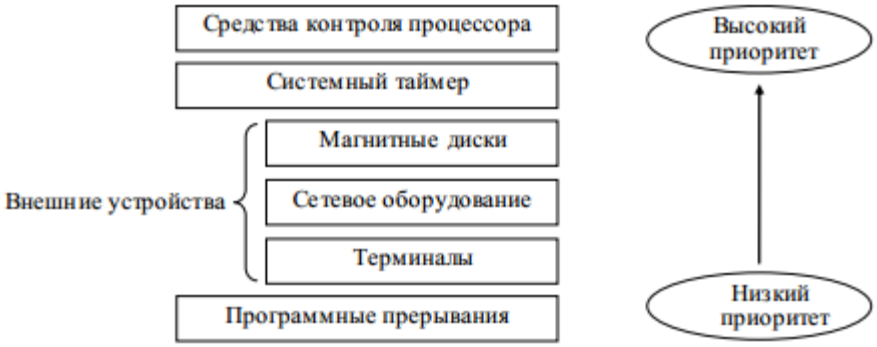


Рис. 8. Распределение прерываний по уровням приоритета

Сигналы, вызывающие прерывания, формируются вне процессора или в процессоре, и могут возникать одновременно. В связи с этим еще одним неотъемлемым элементом обработки прерываний является приоритет прерываний, с помощью которого они ранжируются по степени важности и срочности. О прерываниях, имеющих одинаковое значение приоритета, говорят, что они относятся к одному уровню приоритета прерываний. Так, со всей очевидностью, прерывания от схем контроля процессора должны обладать наивысшим приоритетом, а, например, программные прерывания – самым низким. Например, если приоритет запрашиваемого прерывания ниже приоритета средства контроля процессора, то в этом случае прерывания не произойдет.

Учет приоритета может быть встроен в технические средства, а также определяться ОС, т. е. кроме аппаратно реализованных приоритетов прерывания большинство вычислительных машин и комплексов допускают программно-аппаратное управление порядком обработки сигналов прерывания. Второй способ, дополняя первый, позволяет применять различные дисциплины обслуживания прерываний.

53. В чем отличие между терминами «программа» или «задание» и термином «процесс»?

Термины **программа** и задание правомерно применять для описания некоторых статических, неактивных объектов. Для описания динамических объектов используют термин процесс.

Процесс характеризует некоторую совокупность набора исполняющихся команд, ассоциированных с ним ресурсов (выделенная для исполнения память или адресное пространство, стеки, используемые файлы и устройства ввода-вывода и т. д.) и информации о текущем моменте его исполнения (значения регистров, программного счетчика, состояние стека и значения переменных), находящуюся под управлением ОС.

54. Что такое поток или нить исполнения? В чем заключаются основные отличия нити от процесса?

Поток выполнения (тред; от англ. *thread* — нить) — наименьшая единица обработки, исполнение которой может быть назначено ядром операционной системы.

Другими словами, понятию **поток** соответствует последовательный переход процессора от одной команды программы к другой, при этом ОС распределяет процессорное время между потоками. В то же время **процессу** ОС назначает адресное пространство и набор ресурсов, которые совместно используются всеми его потоками.

55. Каковы основные предпосылки появления потоков?

Для того чтобы процессы не могли вмешаться в распределение ресурсов, а также не могли повредить коды и данные друг друга, важнейшей задачей ОС является изоляция одного процесса от другого. Для этого ОС обеспечивает каждый процесс отдельным виртуальным адресным пространством, так что ни один процесс не может получить прямого доступа к командам и данным другого процесса. При необходимости взаимодействия процессы обращаются к ОС, которая, выполняя функции посредника, предоставляет им средства межпроцессной связи – конвейеры, разделяемые секции памяти и др.

В то же время приложение, выполняемое в рамках одного процесса, может обладать внутренним параллелизмом, который в принципе мог бы позволить ускорить его работу. Если, например, в программе предусмотрено обращение к внешнему устройству, то на время этой операции можно не блокировать выполнение всего процесса, а продолжить вычисления по другой ветви программы

56. Какие основные этапы создания процесса?

Создание процесса состоит из нескольких этапов:

- присвоение идентификатора процессу;
- включение его в список активных процессов, известных системе;
- формирование блока управления процессом;
- выделение процессу начальных ресурсов.

В многопоточной системе при создании процесса ОС создает для каждого процесса как минимум один поток. При создании потока, так же как и при создании процесса, ОС генерирует специальную информационную структуру – описатель потока.

57. Какую информацию содержит описатель потока?

- идентификатор процесса (Process Identifier, PID);
- тип (или класс) процесса, который определяет для супервизора некоторые правила предоставления ресурсов;
- приоритет процесса, в соответствии с которым супервизор предоставляет ресурсы (в рамках одного класса процессов в первую очередь обслуживаются более приоритетные процессы);
- переменную состояния, которая определяет, в каком состоянии находится процесс (готов к работе, выполняется, ожидает устройства ввода-вывода и т. д.);
- контекст задачи, то есть защищенную область памяти (или адрес такой области), в которой хранятся текущие значения регистров процессора, когда процесс прерывается, не закончив работы;
- информацию о ресурсах, которыми процесс владеет и/или имеет право пользоваться (указатели на открытые файлы, информация о незавершенных операциях ввода-вывода и др.);
- место (или его адрес) для организации общения с другими процессами;
- параметры времени запуска (момент времени, когда процесс должен активизироваться, и периодичность этой процедуры);
- в случае отсутствия системы управления файлами адрес задачи на диске в ее исходном состоянии и адрес на диске, куда она выгружается из оперативной памяти (ОП), если ее вытесняет другая задача (последнее справедливо для диск-резидентных задач, которые постоянно находятся во внешней памяти на системном магнитном диске и загружаются в ОП только на время выполнения)

58. Какие элементы выделяют на диаграмме состояний процессов? Как при этом реализуется механизм смены состояний процессов?

Каждая диаграмма состояний описывает все возможные состояния одного экземпляра определенного класса и возможные последовательности его переходов из одного состояния в другое, то есть моделирует все изменения состояний объекта как его реакцию на внешние воздействия.

Процесс, находящийся в состоянии процесс исполняется, может через некоторое время завершиться или быть приостановлен ОС и снова переведен в состояние процесс не исполняется. Приостановка процесса может произойти, например, по следующим причинам: для его дальнейшей работы потребовалось возникновение какого-либо события (например, завершения операции ввода-вывода) или истек временной интервал, отведенный ОС для работы этого процесса.

После этого ОС по определенному алгоритму выбирает для исполнения один из процессов, находящихся в состоянии процесс не исполняется, и переводит его в состояние процесс исполняется. Новый процесс, появляющийся в системе, первоначально помещается в состояние процесс не исполняется.

59. В чем заключается планирование использования процессора? Какие задачи предполагаются к решению при планировании?

Планирование – это работа по определению того, в какой момент времени прервать выполнение одного процесса и какому процессу предоставить возможность выполняться. Планирование использования процессора впервые возникает в мультипрограммных вычислительных системах, где в состоянии готовность могут одновременно находиться несколько процессов. Планирование процессов включает в себя решение следующих задач:

- определение момента времени для смены выполняемого процесса;
- выбор процесса на выполнение из очереди готовых процессов;
- переключение контекстов «старого» и «нового» процессов.

60. В чем заключается диспетчеризация процессов? В чем ее отличие от планирования? Какие основные задачи при этом должны быть решены?

Диспетчеризация – это реализация решения, найден ного в результате планирования. Задачами диспетчеризации являются:

- сохранение контекста текущего потока;
- загрузка контекста нового потока;
- запуск нового потока на выполнение.

61. Какие выделяют уровни планирования? Чем характеризуется каждый из уровней?

При построении алгоритмов планирования выделяют три различных уровня, которые обуславливают особенности работы этих алгоритмов:

- долгосрочное;
- краткосрочное;
- среднесрочное.

Долгосрочное планирование характеризуется тем, что:

- отвечает за порождение новых процессов в системе, определяя ее степень мультипрограммирования;
- осуществляется достаточно редко (между появлением процессов могут проходить минуты и даже десятки минут);
- оказывает влияние на функционирование вычислительной системы на протяжении достаточно длительного времени;
- в некоторых ОС сведено к минимуму или отсутствует совсем.

Краткосрочное планирование характеризуется тем, что:

- применяется при планировании использования процессора (например, при обращении исполняющегося процесса к устройствам ввода-вывода или по завершении определенного интервала времени);
- осуществляется не реже одного раза в 100 миллисекунд;
- оказывает влияние на функционирование системы до наступления очередного аналогичного события.

Среднесрочное планирование, в свою очередь, характеризуется тем, что применяется в вычислительных системах для повышения производительности при Временном удалении какого-либо частично выполнившегося процесса из оперативной памяти на диск, а позже – его возвращение для дальнейшего выполнения («swapping»)

62. Какие основные цели и свойства алгоритмов планирования?

- **Справедливость.** Гарантировать каждому заданию или процессу определенную часть времени использования процессора в компьютерной системе, стараясь не допустить «захвата» процессора одним процессом.
- **Эффективность**. Постараться занять процессор на 100 % рабочего времени, не позволяя простаивать ему в ожидании процессов, готовых к исполнению. В реальных вычислительных системах загрузка процессора составляет 40…90 %.
- Сокращение **полного времени выполнения**. Обеспечить минимальное время между стартом процесса или постановкой задания в очередь для исполнения и его завершением.
- Сокращение **времени ожидания**. Сократить время, которое проводят процессы в состоянии готовность в очереди для исполнения.
- Сокращение **времени отклика**. Минимизировать время, которое требуется процессу в интерактивных системах для ответа на запрос пользователя.

Кроме целей планирования, которые необходимо достигнуть, желательно также, чтобы алгоритмы обладали следующими свойствами:

- **Предсказуемость**. Одно и то же задание должно выполняться приблизительно за одно и то же время.
- **Минимальные накладные расходы**. По сути означает, что T исполнения процесса >> Твыбора процесса. Другими словами, если на каждые 100 миллисекунд, выделенные процессу для

использования процессора, будут приходиться 200 миллисекунд на определение того, какой именно процесс получит процессор в свое распоряжение, и на переключение контекста, то такой алгоритм, очевидно, применять не стоит.

- **Равномерная загрузка** ресурсов вычислительной системы, отдавая предпочтение тем процессам, которые будут занимать малоиспользуемые ресурсы.
- **Масштабируемость**. Рост количества процессов в системе в два раза не должен приводить к увеличению полного времени выполнения процессов на порядок.

63. Что такое параметры планирования? Какие виды параметров выделяют?

Среди параметров планирования вычислительной системы выделяют следующие:

- Статические (не изменяемые в ходе функционирования) – предельные значения ресурсов системы: размер оперативной памяти,максимальное количество памяти на диске для осуществления свопинга, количество подключенных устройств ввода-вывода и т. п.);
- Динамические (изменяемые в ходе функционирования) – значения ресурсов системы на текущий момент.

К статическим параметрам процессов относятся характеристики, как правило, присущие заданиям уже на этапе загрузки:

- Пользователь, запустивший процесс;
- Приоритетность выполнения поставленной задачи;
- Соотношение процессорного времени и времени, необходимого для осуществления операций ввода-вывода;
- Номенклатура (оперативная память, устройства ввода-вывода, специальные библиотеки и системные программы и т. д.) и величина необходимых заданию ресурсов вычислительной системы.

Алгоритмы долгосрочного планирования процессов используют в своей работе:

- Параметры вычислительной системы (статические и динамические);
- Статические параметры процессов (динамические параметры процессов на этапе загрузки заданий еще не известны);

64. Какие виды планирования выделяют?

Существует два основных вида алгоритмов планирования процессов – **невытесняющие**, (применяются в ОС NetWare) и **вытесняющие** (применяются в ОС Unix, Windows, OS/2, VMS).

- **Невытесняющая многозадачность** – это способ планирования процессов, при котором активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление планировщику ОС для того, чтобы тот выбрал из очереди другой, готовый к выполнению процесс.
- **Вытесняющая многозадачность** – это такой способ планирования, при котором решение о переключении процессора с выполнения одного процесса на выполнение другого принимается планировщиком ОС, а не самим активным процессом.

Алгоритмы планирования могут быть:

- Основаны на квантовании;
- Основаны на приоритетах.

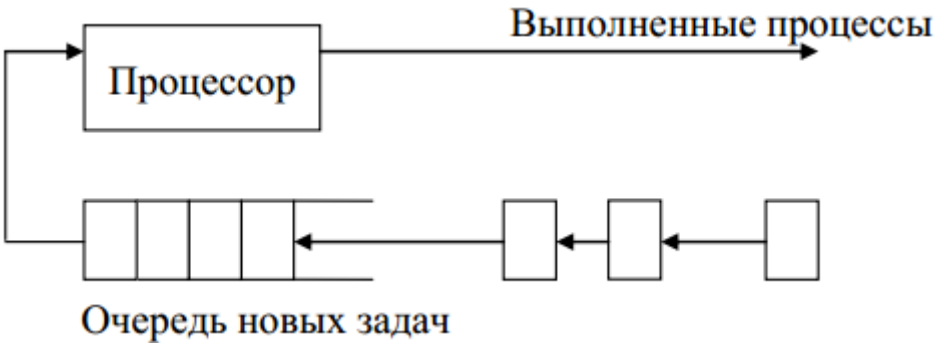
65. Какие разновидности приоритетного планирования выделяют?

Выделяют две разновидности приоритетного планирования: обслуживание с относительными приоритетами и обслуживание с аб- солютными приоритетами. В обоих случаях выбор потока на выполнение из очереди готовых осуществляется одинаково – выбирается поток, имеющий наивысший приоритет. Отличие заключается в определении момента смены активного потока. В системах с относительными приоритетами

активный поток выполняется до тех пор, пока он сам не покинет процессор, перейдя в состояние ожидания (или произойдет ошибка, или поток завершится). В системах с абсолютными приоритетами выполнение активного потока прерывается, если в очереди готовых потоков появился поток, приоритет которого выше приоритета активного потока

66. Какие существуют алгоритмы планирования? В чем заключаются их достоинства и недостатки?

FCFS. Простейшим алгоритмом планирования – First Come, First Served (первым пришел, первым обслужен). *Преимуществом* алгоритма FCFS является легкость его реализации, *недостатками* – среднее время ожидания и среднее полное время выполнения для этого алгоритма существенно зависят от порядка расположения процессов в очереди



Round Robin. Модификацией алгоритма FCFS. По сути, это алгоритм FCFS, только реализованный в режиме вытесняющего планирования (очередной процесс передается на исполнение по таймеру по истечении определенного кванта времени). На производительность алгоритма RR существенно влияет величина кванта времени – *при очень больших величинах* кванта алгоритм RR вырождается в алгоритм FCFS, *при очень малых* – создается иллюзия того, что каждый из n процессов работает на собственном виртуальном процессоре с производительностью ~ 1/n от производительности реального процессора

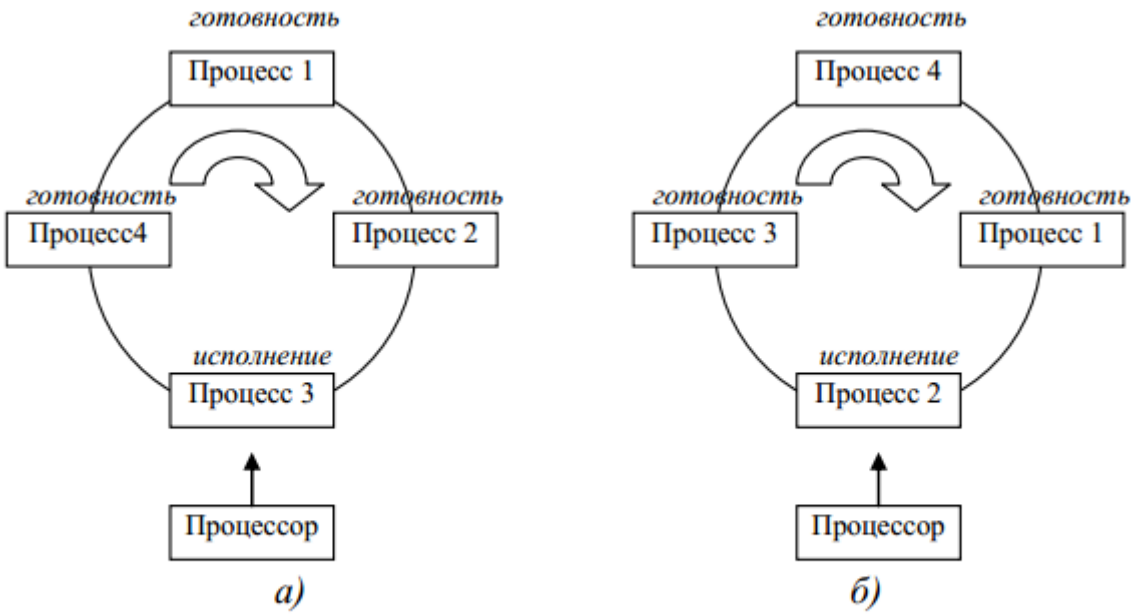


Рис. 16. Процессы «на карусели»:
а – момент времени t_1; б – момент времени t_2

Shortest Job First. Если выбирать процесс не по порядку (как в FCFS и RR), а основываясь на его минимальном времени непрерывного использования процессора, то это позволит повысить производительность алгоритма планирования использования процессора. Описанный алгоритм получил название «кратчайшая работа первой». *Основную сложность* при реализации алгоритма SJF представляет невозможность точно знать в каждом случае время исполнения очередного процесса.

67. Что понимают под параллельно действующими процессами? В чем отличие независимых и взаимодействующих процессов?

Параллельными называют такие последовательные вычислительные процессы, которые одновременно находятся в каком-нибудь активном состоянии. Два параллельных процесса могут быть независимыми либо взаимодействующими.

Независимыми являются процессы, множества переменных которых не пересекаются. Под переменными в этом случае понимают файлы данных, а также области оперативной памяти, сопоставленные промежуточным и определенным в программе переменным. Независимые процессы не влияют на результаты работы друг друга, так как не могут изменять значения переменных друг у друга.

Взаимодействующие процессы используют совместно некоторые (общие) переменные, и выполнение одного процесса может повлиять на выполнение другого.

68. В чем заключается обеспечение синхронизации процессов?

Синхронизация процессов – приведение двух или более процессов к такому их протеканию, когда определенные стадии разных процессов совершаются в определенном порядке, либо одновременно.

Потребность в синхронизации возникает только в мультипрограммной ОС и связана с совместным использованием аппаратных и информационных ресурсов вычислительной системы. Синхронизация необходима в любых случаях, когда параллельно протекающим процессам (потокам) необходимо взаимодействовать – для исключения гонок и тупиков при обмене данными между потоками, разделении данных, при доступе к устройствам ввода-вывода.

Для ее организации используются средства межпроцессного взаимодействия (Inter Process Communications, IPC), что отражает историческую первичность понятия «процесс» по отношению к понятию «поток». Среди наиболее часто используемых средств синхронизации – сигналы, сообщения, семафоры и мьютексы.

69. Какие механизмы ОС относят к средствам синхронизации процессов и потоков?

Для синхронизации потоков прикладных программ программист может использовать как собственные средства и приемы синхронизации, так и средства ОС.

Например, два потока одного прикладного процесса могут координировать свою работу с помощью доступной для них обоих глобальной **логической переменной**, которая устанавливается в единицу при осуществлении некоторого события, например выработки одним потоком данных, нужных для продолжения работы другого. Однако во многих случаях более эффективными или даже единственно возможными являются средства синхронизации, предоставляемые ОС в форме **системных вызовов**. Так, потоки, принадлежащие разным процессам, не имеют возможности вмешиваться каким-либо образом в работу друг друга.

70. Какие примеры необходимости обеспечения синхронизации процессов вам известны?

Предположим, что в некоторый момент процесс R решил распечатать свой файл, для этого он прочитал значение переменной NEXT, значение которой для определенности предположим равным 4. Процесс запомнил это значение, но поместить имя файла не успел, так как его выполнение было прервано (например, в следствие исчерпания кванта). Очередной процесс S, желающий распечатать файл, прочитал то же самое значение переменной NEXT, поместил в четвертую позицию имя своего файла и нарастил значение переменной на единицу. Когда в очередной раз управление будет передано процессу R, то он, продолжая свое выполнение, в полном соответствии со значением текущей свободной позиции, полученным во время предыдущей итерации, запишет имя файла также в позицию 4, поверх имени файла процесса S. Таким образом, процесс S никогда не увидит свой файл распечатанным.

71. Каков механизм синхронизации процессов с использованием блокирующей переменной, семафоров, мониторов, сигналов? Какие характерные особенности каждого из этих методов, достоинства и недостатки?

Блокирующие переменные. Для синхронизации потоков одного процесса прикладной программист может использовать глобальные блокирующие переменные. С этими переменными, к которым все потоки процесса имеют прямой доступ, программист работает, не обращаясь к системным вызовам ОС. Каждому набору критических данных ставится в соответствие двоичная переменная, которой поток присваивает значение 0, когда он входит в критическую секцию, и значение 1, когда он ее покидает.

Плюсы: Блокирующие переменные могут использоваться не только при доступе к разделяемым данным, но и при доступе к разделяемым ресурсам любого вида. Если все потоки разработаны с учетом вышеописанных соглашений, то взаимное исключение гарантируется. При этом потоки могут быть прерваны ОС в любой момент и в любом месте, в том числе в критической секции. Однако следует заметить, что одно ограничение на прерывания все же имеется – нельзя прерывать поток между выполнением операций проверки и установки блокирующей переменной, т. к. это нарушит принцип взаимного исключения.

Минусы: в течение времени, когда один поток находится в критической секции, другой поток, которому требуется тот же ресурс, получив доступ к процессору, будет непрерывно опрашивать блокирующую переменную, бесполезно растрачивая выделяемое ему процессорное время, которое могло бы быть использовано для выполнения какого-нибудь другого потока. Для устранения этого недостатка во многих ОС предусматриваются специальные системные вызовы (аппарат событий) для работы с критическими секциями.

Семафор. Если ресурс занят, то процесс не выполняет циклический опрос, а вызывает системную функцию WAIT(D), здесь D обозначает событие, заключающееся в освобождении ресурса D. Функция WAIT(D) переводит активный процесс в состояние ожидание и делает отметку в его дескрипторе о том, что процесс ожидает события D. Процесс, который в это время использует ресурс D, после выхода из критической секции выполняет системную функцию POST(D), ОС просматривает очередь ожидающих процессов и переводит процесс, ожидающий события D, в состояние готовность.

Плюсы:

- Пассивное ожидание (постановка в очередь и автоматическая выдача ресурсов).
- Возможность управления группой однородных ресурсов.

Минусы: некорректное использование операций на семафоре может привести к нарушению работоспособности параллельных систем.

Мониторы. Мониторы представляют собой тип данных, обладающий собственными переменными, определяющими его состояние. Значения этих переменных извне могут быть изменены только с помощью вызова функций-методов монитора. Функции-методы могут использовать в работе только данные, находящиеся внутри монитора, и свои параметры. Важной особенностью мониторов является то, что в любой момент времени только один процесс может быть активен, т. е. находиться в состоянии готовность или исполнение, внутри данного монитора

Плюсы: исключение входа нескольких процессов в монитор реализуется не программистом, а компилятором

Минусы: Реализация мониторов требует использования специальных языков программирования и компиляторов для них

Сигналы. Это некоторое значимое событие (например, прерывание), источником которого может быть ОС или иная составляющая вычислительной системы. Сигналы вызывают прерывание процесса, выполнение заранее предусмотренных действий.

Сигналы могут вырабатываться как результат работы самого процесса (т. е. вырабатываться синхронно), а могут быть направлены процессу другим процессом (т. е. вырабатываться асинхронно). Синхронные сигналы чаще всего приходят от системы прерываний процессора и свидетельствуют о действиях процесса, блокируемых аппаратурой, например деление на нуль, ошибка адресации, нарушение защиты памяти и т. д.

Особенности: Поскольку посылка сигнала предусматривает знание идентификатора процесса, то взаимодействие посредством сигналов возможно только между родственными процессами, которые могут получить данные об идентификаторах друг друга.

72. В чем заключается и когда возникает взаимная блокировка процессов (тупик)? Какие подзадачи требуют разрешения для решения проблемы тупиков? Какие условия необходимы для возникновения тупиков и как их избежать? Какие существуют пути восстановления системы после тупиков?

При конкуренции нескольких процессов за обладание конечным числом ресурсов может возникнуть ситуация, когда запрашиваемый процессом ресурс недоступен, и ОС переводит данный процесс в состояние ожидания. В то же время, если этот же ресурс удерживается другим ожидающим процессом, то первый процесс не сможет сменить свое состояние. Необходимым условием возникновения тупика является потребность потока сразу в нескольких ресурсах.

Разрешение проблемы тупиков может быть осуществлено путем:

- распознавания тупиков;
- предотвращения тупиков;
- восстановления системы после тупиков;

- игнорирования.

Существуют формальные, программно реализованные методы распознавания тупиков, основанные на ведении таблиц распределения ресурсов и таблиц запросов к занятым ресурсам, анализ которых позволяет обнаружить взаимные блокировки

Предотвращение тупиков. Тупики могут быть предотвращены на стадии проектирования и разработки программного обеспечения, чтобы тупик не мог возникнуть ни при каком соотношении взаимных скоростей процессов.

В качестве необходимых условий возникновения тупиков называют следующие:

- Условие взаимоисключения. Одновременно использовать ресурс может только один процесс.
- Условие ожидания ресурсов. Процессы удерживают ресурсы, уже выделенные им, и могут запрашивать другие ресурсы.
- Условие «неперераспределяемости». Ресурс, выделенный ранее, не может быть принудительно забран у процесса до его завершения. Освобождены они могут быть только процессом, который их удерживает.
- Условие кругового ожидания. Существует кольцевая цепь процессов, в которой каждый процесс ждет доступа к ресурсу, удерживаемому другим процессом цепи.

При возникновении можно:

- снять только часть из них, при этом освобождая ресурсы, ожидаемые остальными процессами;
- вернуть некоторые процессы в область «свопинга»;
- совершить «откат» некоторых процессов до некоторой контрольной точки (т. е. места, где возможен тупик), в которой запоминается вся информация, необходимая для восстановления выполнения программы с данного места.

73. Какие механизмы ОС относят к классу средств межпроцессного взаимодействия? В чем заключаются основные особенности практического использования каждого из этих средств?

Канал - однонаправленный механизм передачи данных (неструктурированного потока байтов) между процессами без необходимости создания файла на диске. Канал представляет собой буфер в оперативной памяти, поддерживающий очередь байт согласно FIFO. Для программиста, использующего канал, этот буфер выглядит как безымянный файл, в который можно писать и читать, осуществляя тем самым обмен данными.

Обычный канал получил развитие, результатом которого стало появление именованного канала или именованного конвейера – зарегистрированного в системе канала (по сути – запись в каталоге файловой системы), который разрешено использовать различным процессам или потокам (не обязательно родственным). Реализуется это путем создания одним, а чтения – другим процессом (потоком) файла типа FIFO с одним и тем же указанным в процессах именем.

Очереди - Механизм очередей сообщений в целом схож с механизмом каналов, но позволяет процессам и потокам обмениваться структурированными сообщениями. При этом синхронизация осуществляется по сообщениям, то есть процесс, пытающийся прочитать сообщение, переводится в состояние ожидания в том случае, если в очереди нет ни одного полного сообщения. Применяется при невозможности применения средств синхронизации. Работа с очередями сообщений имеет ряд отличий от работы с каналами:

- Очереди сообщений предоставляют возможность использовать несколько дисциплин обработки сообщений (FIFO, LIFO, приоритетный доступ, произвольный доступ).
- Если при чтении сообщения оно удаляется из конвейера, то при чтении сообщения из очереди этого не происходит, и сообщение может быть прочитано несколько раз.
- В очередях присутствуют не непосредственно сами сообщения, а их адреса в памяти и размер. Эта информация размещается системой в сегменте памяти, доступном для всех задач, общающихся с

помощью данной очереди. Каждый процесс, использующий очередь, должен предварительно получить разрешение на доступ в общий сегмент памяти с помощью системных запросов API, ибо очередь – это системный механизм, и для работы с ним требуются системные ресурсы и, соответственно, обращение к самой ОС.

Разделяемая память. Разделяемая память представляет собой сегмент физической памяти, отображенной в виртуальное адресное пространство двух или более процессов. Механизм разделяемой памяти поддерживается подсистемой виртуальной памяти, которая настраивает таблицы отображения адресов для процессов, запросивших разделение памяти, так что одни и те же адреса некоторой области физической памяти соответствуют виртуальным адресам разных процессов.

После того как сегмент разделяемой памяти подключен к виртуальной памяти процесса, процесс может обращаться к соответствующим элементам памяти с использованием обычных машинных команд чтения и записи, не прибегая к дополнительным системным вызовам.