9. Понятие процесса. Состояния процесса.  
Процесс — это экземпляр выполняемой программы, включающий код, данные, регистры процессора, стек, а также информацию о состоянии системы (например, открытые файлы, права доступа и т. д.).

Процесс представляет собой единицу работы в операционной системе (ОС), которой выделяются ресурсы (процессорное время, память, устройства ввода-вывода).

В многозадачной (многопроцессной) системе процесс может находиться в одном из трех основных состояний:

ВЫПОЛНЕНИЕ - активное состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;

ОЖИДАНИЕ - пассивное состояние процесса, процесс заблокирован, он не может выполняться по своим внутренним причинам, он ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого процесса, освобождения какого-либо необходимого ему ресурса;

ГОТОВНОСТЬ - также пассивное состояние процесса, но в этом случае процесс заблокирован в связи с внешними по отношению к нему обстоятельствами: процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса.  
  
Состояния процесса

В процессе выполнения программа может находиться в одном из нескольких состояний. Основные состояния процесса:

Новый (New) - Процесс только создан, но ещё не загружен в оперативную память. ОС выделяет ему идентификатор (PID) и начальные ресурсы.

Готовый (Ready) - Процесс загружен в память и ожидает выделения процессорного времени. Может быть несколько процессов в этом состоянии, ОС выбирает следующий для выполнения с помощью планировщика (scheduler).

Выполняемый (Running) - Процесс выполняется на процессоре. В однопроцессорной системе только один процесс может находиться в этом состоянии в каждый момент времени.

Ожидающий (Waiting / Blocked) - Процесс приостановлен, так как ожидает какого-то события (завершения ввода-вывода, освобождения ресурса, сигнала от другого процесса) Пока процесс ждёт, процессор может выполнять другие задачи.

Завершённый (Terminated / Zombie) Процесс завершил свою работу (нормально или аварийно). ОС освобождает его ресурсы, но запись о процессе может сохраняться до тех пор, пока родительский процесс не прочитает его статус.

Дополнительные состояния (в некоторых ОС)

Приостановленный (Suspended) – процесс временно выгружен из оперативной памяти на диск (например, из-за нехватки памяти).

Готовый, но приостановленный (Ready, Suspended) – процесс готов к выполнению, но находится на диске.

10. Планирование процессов в системах пакетной обработки.

Планирование происходит по определенным алгоритмам по которым определяется приоритет процессов.   
1 «Первым пришёл – первым ушёл»

Самый простой алгоритм планирования. Категория алгоритма – без переключений. Процессам предоставляется доступ к процессору в том порядке, в котором они его запрашивают. Формируется единая очередь процессов. Когда текущий процесс блокируется, запускается следующий в очереди, а когда блокировка снимается, процесс попадает в конец очереди. Недостаток в том, что если существует очередь процессов, в котором есть процессы ограниченные устройствами ввода-вывода (т.е. большую часть времени тратящие на ожидание устройств), то это ожидание будет означать простой процессора.

2 Алгоритм «Кратчайшая задача – первая»

Категория алгоритма – без переключений. Суть алгоритма заключается в следующем: если в очереди есть несколько одинаково важных задач, планировщик выбирает первой самую короткую по времени. Эта схема работает лишь в случае одновременного наличия задач и обычно неактуальна.

3 Алгоритм «Наименьшее оставшееся время выполнения»

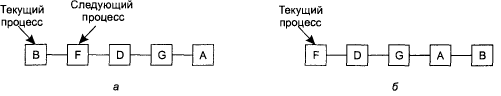
Версия предыдущего алгоритма. В соответствии с этим алгоритмом, планировщик каждый раз выбирает процесс с наименьшим оставшимся временем выполнения. Естественно для таких алгоритмов необходимо знать, сколько времени выполняются процессы, что обычно является сложной задачей.

4 Алгоритм трехуровневого планирования

Системы пакетной обработки позволяют реализовать трехуровневое планирование. По мере поступления в систему новые задачи сначала помещаются в очередь, хранящуюся на диске. Планировщик доступа выбирает задание и передает его системе. Остальные задачи остаются в очереди. Выбор заданий обуславливается установленным приоритетом – по времени выполнения или как-то по другому, например по работе с устройствами ввода-вывода.

11. Планирование процессов в интерактивных системах.

**1) Циклическое планирование.** Каждому процессу предоставляется некоторый интервал времени процессора, так называемый квант времени.Если к концу кванта времени процесс все еще работает, он прерывается, а управление передается другому процессу. Если процесс блокируется или прекращает работу раньше, переход управления происходит в этот момент. Реализация циклического планирования проста. Планировщику нужно всего лишь поддерживать список процессов в состоянии готовности согласно рис. а. Когда процесс исчерпал свой лимит времени, он отправляется в конец списка(рис. б).



Интересным моментом этого алгоритма является длина кванта. слишком малый квант приведет к частому переключению процессов и небольшой эффективности, но слишком большой квант может привести к медленному реагированию на короткие интерактивные запросы. Значение кванта около 20-50 мс часто является разумным компромиссом.

**2) Приоритетное планирование.** каждому процессу присваивается приоритет, и управление передается готовому к работе процессу с самым высоким приоритетом. Чтобы предотвратить бесконечную работу процессов с высоким приоритетом, планировщик может уменьшать приоритет процесса с каждым тактом часов (то есть при каждом прерывании по таймеру). Если в результате приоритет текущего процесса окажется ниже, чем приоритет следующего процесса, произойдет переключение. Возможно предоставление каждому процессу максимального отрезка времени работы. Как только время кончилось, управление передается следующему но приоритету процессу. Приоритеты процессам могут присваиваться статически или динамически. Часто бывает удобно сгруппировать процессы в классы по приоритетам и использовать приоритетное планирование среди классов, но циклическое планирование внутри каждого класса.

**3) Несколько очередей.** Процессам класса с высшим приоритетом выделяется один квант, процессам следующего класса — два кванта, следующего — четыре кванта и т. д. Когда процесс использовал все отведенное ему время, он перемещается на класс ниже.4**) «Самый короткий процесс — следующий**» Если рассматривать выполнение каждой команды как отдельную задачу, можно минимизировать общее среднее время отклика, запуская первой самую короткую задачу. Проблема состоит в том, чтобы понять, какой из ожидающих процессов самый короткий. Один из методов основывается на оценке длины процесса, базирующейся на предыдущем поведении процесса. При этом запускается процесс, у которого оцененное время самое маленькое. **5) Гарантированное планирование.**Система должна отслеживать распределение процессора между процессами с момента создания каждого процесса. Затем система рассчитывает количество ресурсов процессора, на которое процесс имеет право, Теперь можно сосчитать отношение времени, предоставленного процессу, к времени, на которое он имеет право. Полученное значение 0,5 означает, что процессу выделили только половину положенного, а 2,0 означает, что процессу досталось в два раза больше, чем положено. Затем запускается процесс, у которого это отношение наименьшее, пока оно не станет больше, чем у его ближайшего соседа.6**) Лотерейное планировани**е. Когда планировщику необходимо принять решение, выбирается случайным образом лотерейный билет, и его обладатель получает доступ к ресурсу. билетов. лотерейное планирование обладает высокой отзывчивостью. Взаимодействующие процессы могут при необходимости обмениваться билетами.**7) Справедливое планирование**. каждому пользователю достается некоторая доля процессора, и планировщик выбирает процесс в соответствии с этим фактом. Если каждому из пользователей было обещано по 50 % процессора, то им достанется по 50 % процессора, независимоот количества процессов.

12. Планирование процессов в системах реального времени.

2. Требования к планированию в СРВ

Предсказуемость – гарантированное выполнение к заданному сроку.

Детерминизм – время отклика должно быть фиксированным и минимальным.

Приоритетность – критические задачи выполняются в первую очередь.

Минимизация переключений – чтобы снизить накладные расходы.

3. Алгоритмы планирования в СРВ

a) Статическое планирование (Static / Offline Scheduling)

Планировщик заранее строит расписание выполнения задач.

Используется в жёстких СРВ, где все процессы известны заранее.

Пример: Табличное планирование (Time-Triggered Scheduling).

b) Динамическое планирование (Dynamic / Online Scheduling)

Решения принимаются в реальном времени на основе текущей загрузки.

Подходит для мягких СРВ и систем с переменной нагрузкой.

13. Операции над процессами.  
  
подсистема Управление процессомдолжна выполнять следующие операции над процессами:

* Создание.
* .Уничтожение.
* Запуск.
* Блокировка.
* Приостановка.
* Возобновление, а также
* Изменение диспетчерского приоритета процесса (работа с очередями).

Создание состоит из многих этапов:

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

14. Приоритет процессов

Приоритет процесса — это числовое значение, определяющее очередность доступа процесса к ресурсам CPU в многозадачной операционной системе.

15. Понятие потока. Ресурсы и потоки в операционной системе

**1. Понятие потока (thread)**

**Поток** (нить выполнения) — это наименьшая единица обработки, которую может выполнять ОС. В отличие от процесса, поток:

* Использует **общие ресурсы** родительского процесса (память, файлы).
* Имеет **собственный**:
  + Счетчик команд (Program Counter).
  + Регистры процессора.
  + Стек вызовов.

**Многопоточность** — выполнение нескольких потоков в рамках одного процесса для повышения производительности.

16. Понятие ресурса. Классификация ресурсов.

Ресурс — это любой компонент вычислительной системы, который может быть выделен процессу или потоку для выполнения задач. Ресурсы ограничены, поэтому ОС управляет их распределением между конкурирующими процессами.

Классификация:  
По типу: Аппаратные, программные  
По способу использования: Разделяемые, Исключительные  
По времени выделения: статические, динамические  
По управлению: Управляемые ОС, Управляемые процессами  
17. Дисциплины распределения ресурсов на основе очередей.  
FIFO  
LIFO  
Приоритетное обслуживание  
Round Robin (Циклическое)

17. Основные элементы графических интерфейсов.  
Окно, кнопка, радиокнопка, текстовое поле, метка, флажок, выпадающий список, ползунок  
18. Интерфейс прикладного программирования (API).  
API – это набор функций, классов, протоколов и инструментов для взаимодействия двух программных компонентов. Задаёт правила и способы интеграции программных компонентов.  
**API** (Application Programming Interface) - набор правил и инструментов, позволяющих различным программам общаться между собой.