Операционные системы, среды и оболочки

Процессы в операционных системах

Многозадачность ОС

- Организация рационального использования ресурсов компьютера – одна из важнейших функций ОС.
- **Многозадачность (мультипрограммирование) –** способ организации вычислительного процесса, при котором на одном (или нескольких) процессоре (-ах) выполняется несколько задач. Данные задачи совместно используют не только ресурсы процессора, но также оперативную и внешнюю память, устройства ввода-вывода и т.д.
- Организация вычислительного процесса характеризуется несколькими критериями эффективности:
  - *Пропускная способность* – количество выполняемых задач в единицу времени;
  - *Удобство работы пользователя*
  - *Реактивность системы* – способность системы выдерживать промежутки времени между запуском программы и получением результата.

Классификация многозадачных операционных систем

- Операционные системы по типу организации многозадачности делятся на:
  - *Системы пакетной обработки*
  - *Системы разделения времени*
  - *Системы реального времени.*

Системы пакетной обработки

- Для повышения пропускной способности вычислительной системы и минимизации простоев центрального процессора, вызванных ожиданием приложением ввода данных или другими причинами, используется организация вычислительного процесса в виде очереди (пакета) исполняемых приложений.
- Схема работы системы имеет следующий вид:
  - В начале работы формируется пакет заданий, причем каждое задание содержит требования к системным ресурсам.
  - Из пакета формируется *мультипрограммная смесь*, то есть множество одновременно выполняемых задач (как правило, выбираются наборы задач, предъявляющие разные требования к ресурсам).
- Область применения систем пакетной обработки – системы с интенсивными вычислениями (например, научные и технические расчеты).

Системы разделения времени

- В данных системах пользователю (или пользователям) предоставляется возможность интерактивной работы сразу с несколькими приложениями.
- Для решения данной задачи ОС должна **принудительно приостанавливать работу приложений** и **переключать работу процессора** на обработку другого приложения.
- Всем приложениям попеременно выделяется некоторый **квант** процессорного времени. Пользователь получает возможность поддерживать диалог с запущенным приложением.
- **Достоинство** данных систем – удобство работы пользователя. Каждый пользователь получает терминал для управления работой приложений.
- **Недостаток** – меньшая пропускная способность операционных систем.

Системы реального времени

- Системы реального времени представляют собой вариант систем с разделением времени выполнения в случае, если задача обработки информации должна быть выполнена в течение некоторого предельно допустимого времени.
- Мультипрограммная смесь в системах реального времени – фиксированный набор заранее разработанных программ. Выбор задачи осуществляется по прерываниям или в соответствии с расписанием плановых работ.
- Способность ОС к быстрому ответу зависит от скорости переключения с одной задачи на другую, в частности, от скорости обработки прерываний.
- В системах реального времени при планировании загрузки устройств устанавливается некоторый запас вычислительной мощности системы, для того, чтобы, в случае пиковой нагрузки, система могла осуществить прерывание работы приложения.
- Примера таких систем – управление техническими объектами или технологическими процессами. В таких системах критерий эффективности – способность выдержать заданные интервалы времени.

Мультипроцессорная обработка

- **Мультипроцессорная обработка** – способ организации вычислительного процесса в системах с несколькими процессорами (разделение процедуры обработки на разных процессорах).
- При мультипроцессорной обработке усложняется задача управления ресурсами – планирование процессов для нескольких процессоров, использующих общую оперативную и внешнюю память, другие устройства. Необходимо планировать средства блокировки при доступе к разделяемым информационным структурам ядра.
- Современные серверные ОС включают поддержку мультипроцессорной обработки – Sun Solaris, IBM OS/2, Novell NetWare, Microsoft Windows.

Способы мультипроцессорной обработки

- **Симметричная архитектура** мультипроцессорной системы – система включает набор однородных процессоров (как правило, 2-4-8-16 и т.д.), единообразно включенных в общую схему. Симметричные мультипроцессорные системы разделяют общую оперативную память между всеми процессорами. Масштабирование в симметричных системах ограничено.
- **Асимметричная архитектура** – система включающая разные типы процессоров (как по характеристикам, так и функциональной роли). Функциональная неоднородность асимметричных систем приводит к структурным отличиям во фрагментах системы. Масштабирование ассиметричных систем выполняется по *горизонтали*. Каждое устройство называется *кластером*, а система – *кластерной*.

Потоки и процессы

- Основная задача ОС – распределение ресурсов между процессами и потоками.
  - **Процесс**, связан с программным кодом исполняемого модуля и рассматривается ОС как заявка на потребление всех видов ресурсов, кроме процессорного времени. Для изоляции процессов друг от друга ОС обеспечивает каждый процесс отдельным виртуальным адресным пространством. Один процесс не может получить прямого доступа к командам и данным другого процесса.
  - **Поток** представляет собой процесса, способ распараллеливания вычислений. Поток – последовательность команд, выполняемых процессором.
- ОС распределяет процессорное время между потоками. Процессу назначается адресное пространство и набор ресурсов, которые используются всеми его потомками.

Понятие процесса

- **Процесс** характеризуется 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 текущего момента его выполнения (значения регистров, программного счетчика, состояние стека и значения переменных), находящуюся под управлением операционной системы.
- Процесс находится под управлением операционной системы, поэтому в нем может выполняться часть кода ее ядра, как в случаях, специально запланированных авторами программы (например, при использовании системных вызовов), так и в непредусмотренных ситуациях (например, при обработке внешних прерываний).

Состояния процесса

- Для мультипрограммных вычислительных систем псевдопараллельная обработка нескольких процессов достигается с помощью переключения процессора с одного процесса на другой.
- Пока один процесс выполняется, остальные ждут своей очереди.
- Каждый процесс может находиться как минимум в двух состояниях: процесс исполняется и процесс не исполняется.

Модель процессов

- Всякий новый процесс, появляющийся в системе, попадает в состояние готовности. Операционная система, пользуясь каким-либо алгоритмом планирования, выбирает один из готовых процессов и переводит его в состояние исполнение.
- В состоянии исполнения происходит непосредственное выполнение программного кода процесса. Выйти из состояния выполнения процесс может по трем причинам:
  - операционная система прекращает его деятельность;
  - он не может продолжать свою работу, пока не произойдет некоторое событие, и операционная система переводит его в состояние ожидание;
  - в результате возникновения прерывания в вычислительной системе (например, прерывания от таймера по истечении предусмотренного времени выполнения) его возвращают в состояние готовность.
- Модель должна описывать поведение процессов не только во время их существования, но и при появлении процесса в системе и его исчезновении. Для полноты картины вводится еще два состояния процессов: рождение и закончил исполнение

Состояния процессов

- При рождении процесс получает в свое распоряжение **адресное пространство**, в которое загружается программный код процесса;
- Процессу выделяются **стек** и **системные ресурсы**; устанавливается начальное значение программного счетчика этого процесса и т. д. Родившийся процесс переводится в состояние готовность. При завершении своей деятельности процесс из состояния исполнение попадает в состояние закончил исполнение.
- В операционных системах состояния процесса могут быть еще более детализированы, могут появиться некоторые новые варианты переходов из одного состояния в другое.
- Так, например, модель состояний процессов для операционной системы Windows NT содержит 7 различных состояний, а для операционной системы Unix – 9.

Набор операций над процессами

- Процесс не может перейти из одного состояния в другое самостоятельно. Изменением состояния процессов занимается ОС, совершая операции над ними.
- Операции можно объединить в три пары:
  - **создание процесса – завершение процесса**;
  - **приостановка процесса** (перевод из состояния исполнение в состояние готовность) – **запуск процесса** (перевод из состояния готовность в состояние исполнение);
  - **блокирование процесса** (перевод из состояния исполнение в состояние ожидание) – **разблокирование процесса** (перевод из состояния ожидание в состояние готовность).
- Операции создания и завершения процесса являются **одноразовыми**.
- Все остальные операции, связанные с изменением состояния процессов, будь то запуск или блокировка, как правило, являются **многоразовыми**.

Управление процессами

- Для того чтобы операционная система могла выполнять операции над процессами, каждый процесс представляется в ней некоторой структурой данных.
- Эта структура содержит информацию, специфическую для данного процесса:
  - состояние, в котором находится процесс;
  - программный счетчик процесса или, другими словами, адрес команды, которая должна быть выполнена для него следующей;
  - содержимое регистров процессора;
  - данные, необходимые для планирования использования процессора и управления памятью (приоритет процесса, размер и расположение адресного пространства и т. д.);
  - учетные данные (идентификационный номер процесса, какой пользователь инициировал его работу, общее время использования процессора данным процессом и т. д.);
  - сведения об устройствах ввода-вывода, связанных с процессом (например, какие устройства закреплены за процессом, таблицу открытых файлов).

Блок управления процессом

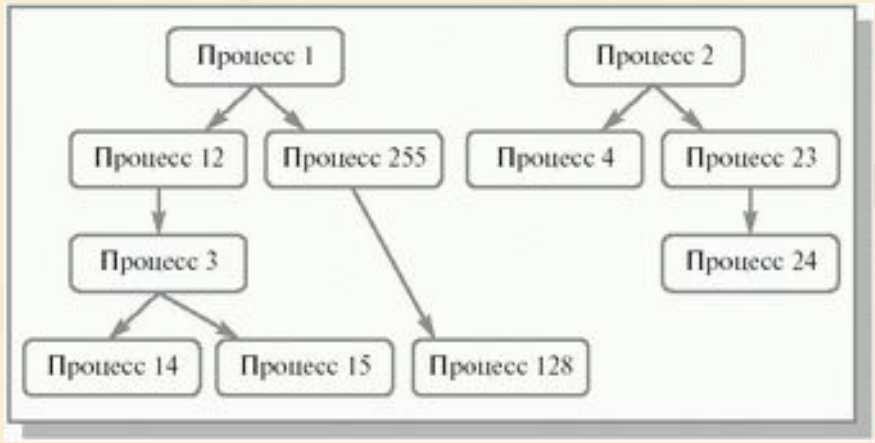
- Для любого *процесса*, находящегося в вычислительной системе, вся информация, необходимая для совершения *операций* над ним, доступна операционной системе.
- Для простоты будем считать,ся что она хранится в одной структуре данных – *PCB* (Process Control Block) или *блоком управления процессом*.
- ***Блок управления процессом*** является моделью *процесса* для операционной системы. Любая *операция*, производимая операционной системой над *процессом*, вызывает определенные изменения в *PCB*. В рамках принятой модели *состояний процессов* содержимое *PCB* между *операциями* остается постоянным.

Контексты процесса

- Информацию, для хранения которой предназначен *блок управления процессом*, можно разделить на две части.
- Содержимое всех регистров процессора (включая значение программного счетчика) называться *регистровым контекстом процесса*, а все остальное – *системным контекстом процесса*.
- Знания *регистрового* и *системного контекстов процесса* достаточно для того, чтобы управлять его работой в операционной системе, совершая над ним *операции*. Однако этого недостаточно для того, чтобы полностью охарактеризовать *процесс*.
- Операционную систему не интересует, какими именно вычислениями занимается *процесс*, т. е. какой код и какие данные находятся в его адресном пространстве. С точки зрения пользователя, наоборот, наибольший интерес представляет содержимое адресного пространства *процесса*, возможно, наряду с *регистровым контекстом* определяющее последовательность преобразования данных и полученные результаты. Код и данные, находящиеся в адресном пространстве *процесса*, называется его *пользовательским контекстом*.
- Совокупность *регистрового*, *системного* и *пользовательского контекстов процесса* для краткости принято называть просто *контекстом процесса*. В любой момент времени *процесс* полностью характеризуется своим *контекстом*.

Одноразовые операции. Рождение процессов

- Любая ОС, поддерживающая концепцию *процессов*, обладает средствами для их *создания*. В очень простых системах все *процессы* могут быть порождены на этапе старта системы. Более сложные операционные системы создают *процессы* динамически, по мере необходимости.
- Инициатором рождения нового *процесса* после старта операционной системы может выступить либо *процесс* пользователя, совершивший специальный системный вызов, либо сама операционная система, то есть, в конечном итоге, тоже некоторый *процесс*.
- *Процесс*, инициировавший *создание* нового *процесса*, принято называть процессом-родителем (parent process), а вновь созданный *процесс* – процессом-ребенком (child process). Процессы-дети могут в свою очередь порождать новых детей и т. д., образуя, в общем случае, внутри системы набор генеалогических деревьев *процессов* – генеалогический лес.



Одноразовые операции. Завершение процессов

- После того как *процесс* завершил свою работу, операционная система переводит его в *состояние* закончил исполнение и освобождает все ассоциированные с ним ресурсы, делая соответствующие записи в *блоке управления процессом*.
- При этом сам *PCB* не уничтожается, а остается в системе еще некоторое время. Подобная информация сохраняется в *PCB* отработавшего *процесса* до запроса процесса-родителя или до конца его деятельности, после чего все следы завершившегося *процесса* окончательно исчезают из системы. В операционной системе Unix *процессы*, находящиеся в *состоянии* закончил исполнение, принято называть процессами-зомби.
- В ряде ОС (например, в VAX/VMS) гибель процесса-родителя приводит к *завершению* работы всех его «детей».
- В других операционных системах процессы-дети продолжают свое существование и после окончания работы процесса-родителя. При этом возникает необходимость изменения информации в *PCB* процессов-детей о породившем их *процессе* для того, чтобы генеалогический лес *процессов* оставался целостным.

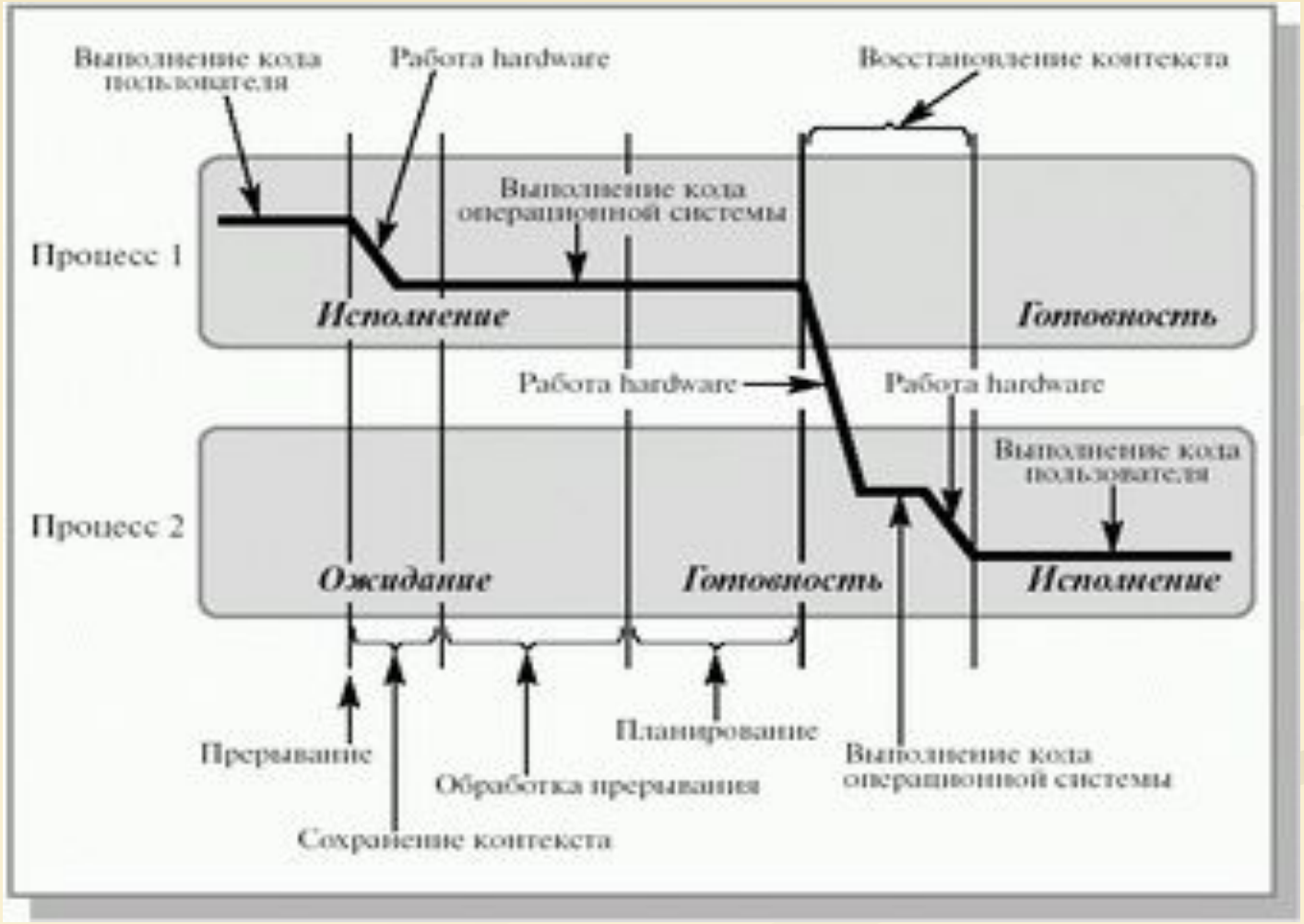
Многоразовые операции

- ***Приостановка процесса***. Работа *процесса*, находящегося в *состоянии* исполнение, приостанавливается в результате какого-либо прерывания. Процессор автоматически сохраняет счетчик команд и, возможно, один или несколько регистров в стеке исполняемого *процесса*, а затем передает управление по специальному адресу обработки данного прерывания. На этом деятельность hardware по обработке прерывания завершается. По указанному адресу обычно располагается одна из частей операционной системы. Она сохраняет динамическую часть *системного* и *регистрового контекстов процесса* в его *PCB*, переводит *процесс* в *состояние* готовность и приступает к обработке прерывания, то есть к выполнению определенных действий, связанных с возникшим прерыванием.
- ***Блокирование процесса***. *Процесс* блокируется, когда он не может продолжать работу, не дождавшись возникновения какого-либо события в вычислительной системе. Для этого он обращается к операционной системе с помощью определенного системного вызова. Операционная система обрабатывает системный вызов (инициализирует операцию ввода-вывода, добавляет *процесс* в очередь *процессов*, дожидающихся освобождения устройства или возникновения события, и т. д.) и, при необходимости сохранив нужную часть *контекста процесса* в его *PCB*, переводит *процесс* из *состояния* исполнение в *состояние* ожидание.
- ***Разблокирование процесса***. После возникновения в системе какого-либо события операционной системе нужно точно определить, какое именно событие произошло. Затем операционная система проверяет, находился ли некоторый *процесс* в *состоянии* ожидание для данного события, и если находился, переводит его в *состояние* готовность, выполняя необходимые действия, связанные с наступлением события (инициализация *операции* ввода-вывода для очередного ожидающего *процесса* и т. п.).

Переключение контекста

- Мультипрограммная операционная система выполняет цепочку *операций*, выполняемых над различными *процессами*, и переключая процессор с одного *процесса* на другой.
- *Операция разблокирования процесса*, ожидающего ввода-вывода может быть представлена следующим образом.
- При исполнении процессором некоторого *процесса* (на рисунке – *процесс* 1) возникает прерывание от устройства ввода-вывода, сигнализирующее об окончании *операций* на устройстве. Над выполняющимся *процессом* производится *операция приостановки*.
- Далее операционная система разблокирует *процесс*, инициировавший запрос на ввод-вывод (на рисунке – *процесс* 2) и осуществляет *запуск* приостановленного или нового *процесса*, выбранного при выполнении планирования (на рисунке был выбран *разблокированный процесс*).

Переключение контекста



Переключение контекста

- Для корректного переключения процессора с одного *процесса* на другой ОС должна сохранить *контекст* исполнявшегося *процесса* и восстановить *контекст процесса*, на который будет переключен процессор.
- Такая процедура сохранения/восстановления работоспособности *процессов* называется ***переключением контекста***.
- Время, затраченное на *переключение контекста*, не используется вычислительной системой для совершения полезной работы и представляет собой накладные расходы, снижающие производительность системы. Оно меняется от машины к машине и обычно колеблется в диапазоне от 1 до 1000 микросекунд.
- Существенно сократить накладные расходы в современных операционных системах позволяет расширенная модель *процессов*, включающая в себя понятие threads of execution (нити исполнения или просто нити).

Планирование и диспетчеризация процессов

- Для реализации многозадачности ОС выполняет планирование и диспетчеризацию потоков.
- **Планирование** – определение момента времени для смены текущего потока, а также выбор нового потока для выполнения.
- **Диспетчеризация** – реализация найденного в ходе планирования решения, т.е. организация переключения с одного потока на другой.
- Планирование может быть *динамическим* – решения принимаются системой на основе анализа текущей ситуации (характерен для универсальных ОС) или *статическим* – если потоки запускаются на основе заранее разработанного расписания (характерен для ОС реального времени).

Динамическое планирование

- В процессе динамического планирования реализуются различные классы алгоритмов:
  - Вытесняющие и невытесняющие алгоритмы:
    - Вытесняющие алгоритмы – способ планирования потоков, при котором решение о переключении на выполнение другого потока принимается ОС;
    - Невытесняющие алгоритмы – способ, при котором активный поток выполняется до того момента, пока сам не передаст управления ОС.
  - Алгоритмы квантования:
    - В соответствии с концепцией квантования каждому потоку поочередно предоставляется ограниченный непрерывный промежуток времени – *квант*. Смена происходит в случае, если:
      - Поток завершился
      - Произошла ошибка
      - Поток перешел в состояние ожидания
      - Исчерпан квант времени

Динамическое планирование (продолжение)

- В процессе динамического планирования реализуются различные классы алгоритмов:
  - Приоритетные алгоритмы:
    - В основе данного класса лежит понятие *приоритета* – специальное число, характеризующее привилегированность данного потока при использовании вычислительных ресурсов.
    - Приоритет потока связан с приоритетом процесса и изначально устанавливается при запуске процесса.
    - В течение работы, приоритет может быть изменен ОС (динамические приоритеты). В зависимости от установленного приоритета меняется и процессорное время, выделяемое диспетчером ОС.

Система прерываний ОС

- Система прерываний – средство, позволяющее ОС реагировать на внешние события, происходящие асинхронно вычислительному процессу:
  - Сигналы готовности устройства ввода-вывода;
  - Аварийные сигналы аппаратуры вычислительной системы;
  - Информация о завершении потока;
  - др.
- В зависимости от источника прерывания делятся на три класса:
  - **Внешние прерывания**, связанные с сигналами от внешних устройств;
  - **Внутренние прерывания**, возникающие в результате ошибок вычислений;
  - **Программные прерывания**, представляющие удобный механизм вызова процедур операционной системы.

Механизм прерываний

- Для реализации механизм прерываний должен поддерживаться аппаратными средствами компьютера и программными средствами ОС.
- Для упорядочивания процессов обработки прерываний все источники прерываний делятся по нескольким приоритетным уровням, а роль арбитра выполняет диспетчер прерываний ОС.

Способы выполнения прерываний

- Существует два основных способа выполнения прерывания:
  - **Векторный** – в процессор передается номер вызываемой процедуры обработки прерываний
  - **Опрашиваемый** – процессор последовательно опрашивает потенциальные источники запроса прерываний.

Системные вызовы

- **Системные вызовы** предназначены для обеспечения возможности обслуживания приложений со стороны операционной системы. Системные вызовы функционируют на основе механизма прерываний.
- Системные вызовы могут выполнятся *синхронно*, когда поток приостанавливается до завершения системного вызова, или, *асинхронно*, когда поток продолжает работу параллельно с системной процедурой, реализующей вызов.
- Реализация системных вызов должна удовлетворять следующим требованиям:
  - Обеспечить переключение в привилегированный режим;
  - Обеспечить высокую скорость вызова процедур ОС;
  - Обеспечить единообразное обращение к системным вызовам для всех аппаратных платформ, на которых работает ОС;
  - Допускать расширение набора системных вызовов;
  - Обеспечить контроль со стороны ОС за корректным использованием системных вызовов.

Синхронизация процессов и потоков

- Для обеспечения синхронизации процессов и потоков, выполняющих общие задачи и совместно использующих общие ресурсы, в операционных системах используются специальные механизмы:
  - Критические секции;
  - Сигналы;
  - Семафоры;
  - События;
  - Таймеры и др.

Проблемы при отсутствии синхронизации

- При отсутствии таких механизмов возможны нежелательные последствия, такие как гонки и тупики.
  - Гонка – ситуация, при которой два или более потоков обрабатывают разделяемые данные и конечный результат зависит от соотношения скоростей потоков.
  - Тупик (взаимная блокировка) – возможность ситуации при которой два и более потоков взаимно блокируют действия друг друга.

Управление памятью

- Оперативная память – важнейший ресурс вычислительной системы, требующий управления со стороны ОС. Причина – процессы и потоки хранятся и обрабатываются в оперативной памяти.
- Память распределяется между приложениями и модулями самой операционной системы.
- Функции ОС по управлению оперативной памятью:
  - Отслеживание наличия свободной и занятой памяти;
  - Вытеснение кодов и данных из оперативной памяти на диск, когда размеров памяти недостаточно для размещения всех процессов, и возвращение их обратно;
  - Настройка адресов программы на конкретную область физической памяти;
  - Защита выделенных областей памяти процессов от взаимного вмешательства.

Представление потоков в оперативной памяти

- Для идентификации переменных и команд программы используются разные типы адресов:
  - **Символьные** (имена переменных, функций и т. п.);
  - **Виртуальные** – условные числовые значения, вырабатываемые компиляторами;
  - **Физические** – адреса фактического размещения в оперативной памяти.

Виртуальное пространство

- Совокупность виртуальных адресов называется *виртуальным адресным пространством*. Диапазон возможных адресов виртуального пространства у всех процессов одинаков.
- Совпадение виртуальных адресов различных процессов не должно приводить к конфликтам и операционная система отображает виртуальные адреса различных процессов на разные физические адреса.
- Разные ОС по разному организуют виртуальное адресное пространство:
  - Линейная организация – пространство представляется непрерывной линейной последовательностью адресов (по другому плоская структура адресного пространства).
  - Сегментная организация – пространство разделяется на отдельные части. В этом случае, помимо линейного адреса, может быть использован виртуальный адрес (сегмент, смещение).

Виртуальное адресное пространство

- В виртуальном адресном пространстве выделяют две непрерывные части:
  - Системная – для размещения модулей общих для всей системы (размещаются коды и данные ядра ОС, другие служебные модули);
  - Пользовательская – для размещения кода и данных пользовательских программ.
- Системная область включает в себя область, подвергаемую страничному вытеснению, и область, на которую страничное вытеснение не распространяется.
- Системные процессы, требующие быстрой реакции или постоянного присутствия в памяти размещаются в области на которую не распространяется вытеснение.
- Остальные сегменты подвергаются вытеснению, как и пользовательские приложения.

Алгоритмы распределения памяти

Кэширование данных

- Для ускорения доступа к данным используется принцип кэширования. В вычислительных системах существует иерархия запоминающих устройств:
  - нижний уровень занимает емкая, но относительно медленная дисковая память;
  - оперативная память;
  - верхний уровень – сверхоперативная память процессорного кэша.
- Каждый уровень играет роль кэша по отношению к нижележащему.

Кэширование данных

- Каждая запись в кэш-памяти об элементе данных включает в себя:
  - Значение элемента данных;
  - Адрес, который этот элемент данных имеет в основной памяти;
  - Дополнительную информацию, которая используется для реализации алгоритма замещения данных в кэше и включает признак модификации и актуальности данных.