Процессы в операционных системах

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

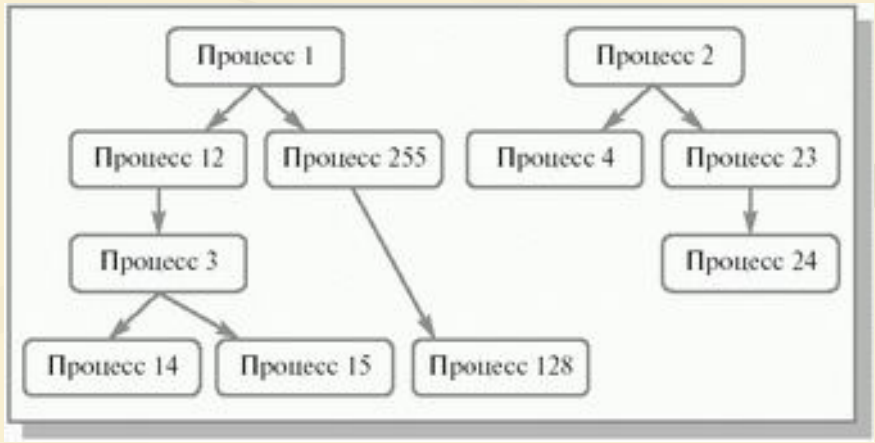
- Для любого *процесса*, находящегося в вычислительной системе, вся информация, необходимая для совершения *операций* над ним, доступна операционной системе.
- Для простоты будем считать,ся что она хранится в одной структуре данных – *PCB* (Process Control Block) или *блоком управления процессом*.
- ***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.).

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.