19) Последовательность создания процесса. Системные функции создания процесса в Linux (Unix) и Windows .

Для создания нового процесса операционной системе необходимо :

•1. Присвоить новому процессу уникальный идентификатор, т.е. занести новую запись в таблицу процессов;

•2. Выделить пространство для процесса, т.е. выделить адресное пространство для всех элементов образа процесса;

• 3. Загрузить часть кодов и данных процесса в оперативную память.

• 4. Инициализировать управляющий блок процесса;

• 5. Поместить процесс в список “готовых” или “готовых приостановленных процессов”;

* В UNIX/Linux существует только один системный вызов для создания нового процесса — *fork*.
* Этот вызов создает точную копию вызывающего процесса.
* После выполнения системного вызова fork два процесса, родительский и дочерний, имеют одинаковые образ памяти, одинаковые строки описания конфигурации и одни и те же открытые файлы.
* По возвращаемому функцией fork() значению процесс может узнать, дочерний он (fork() возвращает 0) или родительский (fork() возвращает число больше 0).
* Для запуска(загрузки ) программы в созданном системным вызовом fork() процессе применяется системный вызов **exec()**. Вызов  **exec()** заменяет пользовательский контекст текущего процесса (образ процесса)на содержимое некоторого исполняемого файла и устанавливает начальные значения регистров процессора.
* **функция wait() -** приостанавливает родительский процесс до тех пор, пока один из его дочерних процессов не будет остановлен или не завершится, после чего возвращает информацию о том, какой процесс завершился и что стало причиной его завершения.
* Если запущено несколько процессов, то для того чтобы дождаться выполнения конкретного процесса используется функция **waitpid().**
* void exit(int status) прекращает процесс, из которого эта функция была вызвана. Кодом возврата этого процесса становиться значение status.
* В Windows системеым вызовом функции ***CreateProces*s** создается процесс, и в него загружается нужная программа.
* У этого вызова имеется 10 параметров(выполняемую программу, параметры командной строки для этой программы, различные параметры безопасности, биты, управляющие наследованием открытых файлов, информацию о приоритетах, спецификацию окна, создаваемого для процесса (если оно используется), и указатель на структуру, в которой вызывающей программе будет возвращена информация о только что созданном процессе.)
* В дополнение к функции CreateProcess в Win32 имеется около 100 других функций для управления процессами и их синхронизации, а также всем, что с этим связано.

Особенности процесса в Windows:

* + Процессы Windows NT имеют свойства:
    - Процессы Windows NT реализованы в форме объектов - EPROCESS, которые создает менеджер объектов.
    - Объект-процесс, содержит заголовок (PCB), который создает менеджер объектов. Заголовок инициализирует (заполняет ) диспетчер процессов.
    - Менеджер процессов Windows NT не поддерживает между процессами отношений типа «родитель-потомок», т.е родительский процесс не может создать дочерний процесс.
    - Вместо этого в процессе Windows сразу создается начальный поток, в котором выполняется программа процесса.
    - Начальный поток может порождать дочерние потоки

20) Синхронизация процессов (критические секции, блокирующие переменные).

Критические секции

Та часть программы (процесса), в которой используется доступ к общей памяти (общим переменным), называется критической областью или критической секцией.

Для избежания состязаний необходимо, что бы никакие два процесса не находились одновременно в своих критических секциях.

Для задания режима взаимного исключения используется запрещение прерываний. В этом случае запрещаются прерывания по таймеру. И процессор не может переключится по таймеру на другой процесс (недостаток – запрещение прерываний может привести к краху всей системы).

Блокирующие переменный

Когда процессу требуется войти в свою критическую область, сначала он проверяет значение блокирующей переменной.

Если оно равно 0, процесс устанавливает его в 1 и входит в критическую область (т.е. сначала устанавливает, потом входит). Если значение уже равно 1, процесс просто ждет, пока оно не станет равно нулю.

Таким образом, 0 означает, что ни один из процессов не находится в своей критической области, а 1 означает, что какой-то процесс находится в своей критической области

Семафоры

Инициализация семафора (задать начальное значение счётчика):

* + Захват семафора (ждать пока счётчик станет больше 0, после этого уменьшить счётчик на единицу);
  + Освобождение семафора (увеличить счётчик на единицу);
  + Бывает бинарный семафор и семафор счетчик.

Семафор — неотрицательная целая переменная, над которой возможны два вида операций:

**P-операция**. Если перед выполнением P-операции значение семафора было больше 0, то P-операция выполняется без задержек. Если перед выполнением P-операции значение семафора было равным 0, то процесс, выполняющий P-операцию, переводится в состояние ожидания до тех пор, пока значение семафора не станет большим 0.

**V-операция**. Если при этом имеются процессы, задержанные на выполнении P-операции на данном семафоре, один из этих процессов выходит из состояния ожидания и может выполнить свою P-операцию.

21)Основные механизмы межпроцессорного взаимодействия процессов:

Общие файлы

Термином «общие файлы» обычно называют совместный доступ к файлам, преимущественно в контексте операционных систем или служб локальных компьютерных сетей.

Общая и разделяемая память

**Общая память**

В модели программирования с общей памяти для управления доступом к общей памяти используются всевозможные механизмы синхронизации типа семафоров и блокировок процессов. Преимущество этой модели с точки зрения программирования состоит в том, что не нужно явно задавать обмен данными между производителями и потребителями.

**Разделяемая память**

В других средствах межпроцессового взаимодействия обмен информацией между процессами проходит через [ядро](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B), что приводит к [переключению контекста](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%B0) между процессом и ядром, т.е. к потерям производительности.

Техника разделяемой памяти позволяет осуществлять обмен информацией через общий для процессов сегмент памяти без использования системных вызовов ядра. Сегмент разделяемой памяти подключается в свободную часть [виртуального адресного пространства](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8#.D0.B2.D0.B8.D1.80.D1.82.D1.83.D0.B0.D0.BB.D1.8C.D0.BD.D0.B0.D1.8F) процесса.

Сигналы

Сигналы сообщают процессам о возникновении асинхронных событий. Существуют сигналы, которые можно классифицировать следующим образом:

* Сигналы, посылаемые в случае завершения выполнения процесса;
* Сигналы, посылаемые в случае возникновения вызываемых процессом особых ситуаций;
* Сигналы, посылаемые во время выполнения системной функции при возникновении неисправимых ошибок.

Каналы

Каналы – это пары из входного и выходного портов, которые соединяются очередями сообщений. Каналы можно создавать и удалять.

22) Планирование процессов. Виды планирования процессов(среднесрочное, краткосрочное; вытесняющее, невытесняющее), круговое(циклическое) краткосрочное планирование.

Для организации многозадачности ОС осуществляет планирование процессов.

• Долгосрочное планирование – когда будет создан новый процесс. Определяет сколько процессов может работать одновременно (степень мультипрограммирования). Такое событие происходит редко (раз в несколько(десятки) минут);

• Среднесрочное планирование определяет какой из процессов будет загружен в память для выполнения вместо выгруженного на диск процесса (переход новый – готовый, миллисекунды);

• Краткосрочное планирование (диспетчер процессов) это планирование какой процесс будет выполняться на процессоре. (переход готовый – выполняющийся, единицы миллисекунд).

**Краткосрочное планирование процессов – задачи:**

• Определение момента времени для смены выполняемого процесса – когда выбирать;

• Выбор процесса на выполнение из очереди готовых процессов кого выбирать;

• Переключение контекстов "старого" и "нового" процессов.

* 1. Алгоритмы планирования делятся на две группы:

– вытесняющие (может прерываться);

– невытесняющие (не может прерываться).

**Вытесняющие алгоритмы планирования**

* Выполняющийся в настоящий момент процесс может быть прерван и переведен операционной системой в состояние готовности к выполнению.
* Планирование возлагается на операционную систему.

**Не вытесняющие алгоритмы планирования**

* При этом способе планирования, процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление планировщику операционной системы для того, чтобы тот выбрал из очереди другой, готовый к выполнению процесс или пока не окажется в заблокированном состоянии ожидания завершения операции ввода-вывода.
* Планирование возлагается на программу (программиста) и операционную систему.
  1. Наиболее часто встречающиеся алгоритмы (когда выбирать):
* алгоритмы, основанные на квантовании;
* алгоритмы, основанные на приоритетах.

**Алгоритм квантования (Циклический алгоритм. Круговое планирование)**

Алгоритм квантования:

1.Таймер генерирует прерывания через определенные интервалы времени.

1. При каждом прерывании исполняющийся в настоящий момент процесс помещается в очередь готовых к выполнению процессов, и начинает выполняться очередной процесс.
2. Эта методика известна также как квантование времени, поскольку перед тем как оказаться вытесненным, каждый процесс получает квант времени для выполнения.

23)Приоритетное планирование.

* Приоритет может выражаться целыми или дробными, положительным или отрицательным значением. Чем выше привилегии процесса, тем меньше времени он будет проводить в очередях
* Приоритет может назначаться директивно администратором системы в зависимости от важности работы или внесенной платы, либо вычисляться самой ОС по определенным правилам,
* Приоритет может оставаться фиксированным на протяжении всей жизни процесса либо изменяться во времени в соответствии с некоторым законом. В последнем случае приоритеты называются динамическими.
* Приоритет - это число, характеризующее степень привилегированности процесса при использовании ресурсов вычислительной машины: чем выше приоритет, тем выше привилегии.

**Алгоритмы на основе приоритетов**

**1. Первым поступил − первым обслужен**

* простейшая стратегия планирования "первым поступил — первым обслужен" или схема строгой очередности;
* как только процесс становится готовым к выполнению, он присоединяется к очереди готовых процессов. При прекращении выполнения текущего процесса для выполнения выбирается процесс, который находился в очереди дольше других.

1. **Выбор самого короткого процесса**

* для выполнения выбирается процесс с наименьшим ожидаемым временем исполнения;
* основная трудность в применении этого способа состоит в оценку времени выполнения, требующегося каждому процессу.

24)Переключение процессов.

Чтобы управлять выполнением процессов, ядро должно уметь приостанавливать (прерывать) процесс, работающий в данный момент, и возобновлять выполнение другого, ранее приостановленного процесса. Это называется переключение процессов (потоков).

Пример механизма переключений:

1. По истечению кванта времени выполнения процесса процессором возникает прерывание от таймера, по которому выполнение процесса приостанавливается, содержимое регистров процессора (счетчик команд, слово состояния и др.) сохраняется в регистровом контексте процесса в ядре.

2. Планировщик процессов выбирает нужный, ранее прерванный процесс (согласно алгоритма планирования). В регистры процессора из регистрового контекста данного процесса загружается сохраненное ранее содержимое регистров процессора.

3. Выполнение процесса возобновляется с команды программы следующей после последней выполненной процессором в предыдущем цикле работы программы.