1. **Поясните понятие «мультизадачная OS с вытеснением».**

Это операционная система, выполняющая несколько задач в один промежуток времени, выделяет этим задачам кванты времени и переключает процессор с выполнения одной задачи на другую по истечению определенного временного интервала.

1. **Поясните понятие «циклическое планирование».**

Это такой вид планирования, при котором каждому процессу назначается определенный интервал времени – квант, - в течение которого ему предоставляется возможность выполнения.

Если процесс к завершению кванта времени все еще выполняется, то ресурс центрального процессора у него отбирается и передается другому процессу, а сам процесс будет помещен в самый конец очереди процессов на выполнение.

Если процесс переходит в заблокированное состояние или завершает свою работу до истечения кванта времени, то переключение центрального процессора на другой процесс происходит именно в этот момент.

1. **Поясните понятие «приоритетное планирование».**

Это такой вид планирования, при котором каждому процессу присваивается значение приоритетности и запускается тот процесс, который находится в состоянии готовности и имеет наивысший приоритет.

1. **Поясните понятие «кооперативное планирование».**

Это такой вид планирования, при котором процесс получает столько процессорного времени, сколько он считает нужным. Таким образом, все процессы делят процессорное время, периодически передавая управление следующей задаче.

1. **Поясните понятие «OS реального времени».**

Система, которая гарантирует фиксированное время для выполнения задачи.

Если операция *должна* быть проведена точно в срок (или в определенный период времени), то мы имеем дело с **системой жесткого реального времени**. Другой разновидностью подобных систем является **система мягкого реального времени**, в которой хотя и нежелательно, но вполне допустимо несоблюдение срока какого-нибудь действия, что не наносит непоправимого вреда.

1. **Поясните понятие «приоритет процесса».**

**Приоритет**[**процесса**](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) — число, ориентируясь на значение которого планировщик процессов может выдавать процессу больше или меньше процессорного [времени](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B2%D1%80%D0%B5%D0%BC%D1%8F).

Это значение, которое влияет на выбор процесса, который будет выполняться следующим.

Приоритет процесса – это числовое значение, определяющее важность и срочность выполнения данного процесса

1. **Поясните выражение «поток уступает процессор другому потоку».**

Это означает, что текущий поток прерывает свое выполнение, освобождая процессорное время другому потоку.

1. **Windows: как поток может уступить процессор?**

Sleep(0) - процесс становится в конец очереди, yield?

1. **Windows: что такое базовый приоритет потока, как он вычисляется и диапазон его изменения?**

базовый приоритет потока - это числовое значение, которое определяет относительную важность потока в контексте планирования выполнения процессов и потоков.

Это приоритет потока, основанный на приоритете процесса и относительном приоритете потока. Он складывается на основании класса приоритета, породившего этот поток процесса и относительного класса приоритета потока. Диапазон изменения от 1 до 31 включительно ([1;31]), вычисляется по специальной таблице (таблицу надеюсь на листик переписывать по памяти не нужно будет...).

по умолчанию – 8



1. **Windows: поясните назначение и принцип применения системного вызова SetThreadIdealProcessor.**

SetThreadIdealProcessor устанавливает предпочтительный процессор для потока, т.е. процессор на котором он будет работать.

Принцип

**SetThreadIdealProcessor**(дикриптор потока, номера выбранного процессора или MAXIMUM\_PROCESSORS)

MAXIMUM\_PROCESSORS, the function returns the current ideal processor without changing it.

Если функция завершается успешно, величина возвращаемого значения - предшествующий привилегированный процессор. Если функция завершается с ошибкой, величина возвращаемого значения равна - (минус) 1.

1. **Windows: поясните назначение и принцип применения системного вызова ResumeThread.**

используется для возобновления выполнения потока (thread), который был приостановлен с помощью системного вызова SuspendThread

ResumeThread уменьшает счетчик приостановки потока на 1. Если счетчик достигает нуля, поток больше не приостановлен, и операционная система может вернуть его в состояние "готов к выполнению".

1. **Windows: поясните назначение и принцип применения системного вызова WaitForSingleObject.**

Приостанавливает выполнение текущего потока до тех пор, пока объект, переданный этой функции, не перейдет в сигнальное состояние или не завершит свое выполнение, но не на больший интервал времени, чем тот, что передан в параметры этого вызова.

1. **Windows: поясните назначение и принцип применения системных вызовов GetProcessPriorityBoost, GetThreadPriorityBoost, SetProcessPriorityBoost, SetThreadPriorityBoost.**

GetProcessPriorityBoost, GetThreadPriorityBoost возвращает true/false в зависимости от того, разрешено ли динамическое изменение приоритета процесса/потока.

SetProcessPriorityBoost, SetThreadPriorityBoost задают это разрешение.

1. **Linux: поясните принцип идентификации процессов и потоков и поясните почему он такой.**

Номер процесса всегда равен номеру главного потока, потому что ядро Линукса не различает потоки и процессы.

1. **Linux: Поясните понятие «планировщик потоков».**

Планировщик потоков в операционной системе Linux - это часть ядра операционной системы, отвечающая за управление выполнением и переключением между потоками в рамках процесса. Он распределяет процессорное времени между различными потоками в системе, учитывая их приоритеты и состояния. Определяет какой поток будет выполняться на центральном процессоре (CPU) в определенный момент времени.

1. **Linux: поясните принцип использования значения nice –процесса, диапазон его изменения, для какого режима работы планировщика это значение применяется?**

Это значение любезности, чем оно меньше, тем выше приоритет у потока.

Диапазон приоритетов[-20, 19], значение по умолчанию = 0; Применяется для вытесняющего режима работы (other) (Планировщик может принудительно забирать управление у потока (например по таймеру или при появлении потока с большим приоритетом))

1. **Linux: перечислите политики планирования, какая действует по умолчанию?**

Политики планирования процесса:

1) стандартная (SCHED\_OTHER, с разделением времени для процессов, работающих не в реальном времени); - по умолчанию

2) FIFO-политика (SCHED\_FIFO, реального времени);

3) карусельная (Round-Robin) политика (SCHED\_RR) реального времени;

4) пакетная политика(BATCH).

1. **Linux: как выяснить действующую политику планирования для процесса с помощью файловой системы proc?**

/proc/PID/sched (в строчке police будет)

0- OTHER

1-FIFO

2-RR

1. **Linux: с помощью какого системного вызова поток может уступить процессор.**

Уступить процессор sched\_yield()

1. **Linux: чем отличается системный вызов nice от вызова setpriority.**

Nice прибавляет к текущему значению любезности nice аргумент, а setpriority устанавливает конкретное значение nice

**nice** изменяет приоритет только для текущего процесса, в котором он был вызван. Он не может изменять приоритет других процессов. **setpriority** может изменять приоритеты процессов по их идентификаторам PID. (он может изменять приоритеты других процессов, а не только текущего процесса)

1. **Linux: поясните понятие «планировщик ввода вывода», каким образом можно выяснить какие планировщики ввода/ вывода доступны?**

Это программная прослойка между блочными устройствами (дисковые устройства) и низкоуровневыми драйверами ввода/вывода. Задача планировщика – оптимизация доступа процесса к дисковому устройству(уменьшение времени поиска данных, обеспечение приоритетности, гарантировать данные за заданное время).

dmesg | grep scheduler

Планировщик ввода-вывода (I/O Scheduler) в операционной системе Linux отвечает за управление запросами на ввод и вывод данных на устройства хранения, такие как жесткие диски (HDD и SSD). Он определяет, в каком порядке эти запросы

Или

**Планировщик ввода/вывода (I/O Scheduler)** - это компонент операционной системы, ответственный за управление запросами на ввод/вывод (I/O requests) и их распределение на физические устройства хранения данных, такие как жесткие диски (HDD) или твердотельные накопители (SSD). Задачей планировщика ввода/вывода является оптимизация процесса I/O для повышения производительности и уменьшения задержек.

1. **Linux: перечислите известные вам планировщики ввода/ вывода, кратко охарактеризуйте их.**

NOOP – простой I/O-планировщик, общая FIFO- очередь read/write-запросов, объединяет однотипные запросы для сокращения операций.

CFQ (Completely Fair Queueing, наиболее справедливая очередь) – у каждого процесса своя очередь, у каждой очереди свой квант времени, планировщик циклически обходит очереди, обслуживает очередь в течении кванта, чтение в очереди имеет приоритет. Поддержка i/o-приоритетов.

BFQ (Budget Fair Queueing) – базируется на CFQ, каждой CFQ-очереди выделяется бюджет, который растет для процессов с интенсивным вводом/выводом.

Deadline - I/O-планировщик пытается выполнить запрос в указанное время, две очереди read и write, read-очередь приоритетнее, запросы объединяются в пакеты по принципу «алгоритма лифта».

MQ-Deadline – модификация Deadline для новых устройств.

Kyber - для работы с быстрыми устройствами, две очереди read и write, read-очередь приоритетнее измеряетcя время завершения каждого запроса и корректирует фактический размер очереди для достижения установленных в настройках задержек.

1. **Linux: каким образом можно выяснить тип планировщика действующего для блокового устройства?**

cat /sys/block/hda/queue/scheduler