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

Это операционная система, выполняющая несколько задач в один промежуток времени, которая выделяет этим задачам кванты времени и принимает решения о переключении процессора 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 и по истечению, которого ресурс ЦП будет отобран у него отобран и передан другому процессу, а сам процесс будет помещен в самый конец очереди процессов на выполнение.

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».**

Это операционная система, реагирующая в определенный как можно более короткий промежуток времени на непредсказуемое появление (внешних) событий.

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

Это величина, определяющая, как часто данный процесс, по сравнению с другими процессами, стоящими в очереди на выполнение процессора, будет исполняться процессором.

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?**

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...).



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

SetThreadIdealProcessor устанавливает предпочтительный процессор для потока, т.е. процессор на котором он будет работать. Если функция завершается успешно, величина возвращаемого значения - предшествующий привилегированный процессор или MAXIMUM\_PROCESSORS, если поток не имеет такового. Если функция завершается с ошибкой, величина возвращаемого значения равна - (минус) 1.

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

ResumeThread уменьшает счет времени приостановки работы потока, вызванной системным вызовом SuspendThread. Когда счет времени приостановки работы уменьшается до нуля, выполнение потока продолжается. ResumeThread проверяет счет времени приостановки работы подчиненного потока. Если счет времени приостановки работы равен 0, поток в настоящее время не приостановлен. Иначе, счет времени приостановки работы подчиненного потока уменьшается. Если итоговое значение - 0, то выполнение подчиненного потока продолжается.

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 Linux не знает ничего о потоках, для ядра поток – это процесс (LWP); для ядра нет разницы между двумя изолированными процессами и двумя потоками в рамках одного процесса; два потока в одном процессе для ядра – это два процесса использующих общие ресурсы ядра.

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

Это часть ядра, алгоритм, который определяет порядок выполнения потоков. Планировщик потоков ядра отвечает за то, какие потоки выполняются на процессорах системы.

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

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

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