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() - процесс становится в конец очереди, SwitchToThread()?

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. вычисляется по специальной таблице



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

SetThreadIdealProcessor устанавливает предпочтительный процессор для потока. Если функция завершается успешно, величина возвращаемого значения - предшествующий предпочтительный процессор. Если функция завершается с ошибкой то -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.

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 A value of TRUE indicates that dynamic boosting is disabled. A value of FALSE indicates normal behavior

SetProcessPriorityBoost, SetThreadPriorityBoost задают это разрешение. If this parameter is TRUE, dynamic boosting is disabled. If the parameter is FALSE, dynamic boosting is enabled.

Возвращают 0 \

– при ошибке, не ноль, если выполнено успешно

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

Номер процесса всегда равен номеру главного потока, потому что 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)

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 policy 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 устанавливает приоритет при запуске программы, а setpriority позволяет изменять приоритет во время работы

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.

dmesg | grep scheduler

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

NOOP

CFQ (Completely Fair Queueing)

BFQ (Budget Fair Queueing)

Deadline

Kyber

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

cat /sys/block/<device>/queue/scheduler

device – имя блочного устройства