1. Понятие процесса. Состояния процесса

Процесс — это программа в состоянии выполнения. Он содержит не только исполняемый код, но и данные, стек, открытые файлы, информацию о текущем состоянии и ресурсах. Состояния процесса:

Готов (Ready) — процесс ожидает запуска.

Выполнение (Running) — процесс в данный момент выполняется.

Ожидание (Waiting) — приостановлен, ждёт ресурс или событие.

Завершён (Terminated) — завершил выполнение.

Новый (New) — только создан, ещё не начал выполняться.

10. Планирование процессов в системах пакетной обработки

В пакетной обработке задачи запускаются одна за другой, пользователь не вмешивается в процесс. Главное — эффективно использовать ресурсы и уменьшить общее время обработки всех заданий. Обычно применяются простые схемы планирования:

FIFO (первым пришёл — первым выполнен)

SJF (сначала короткие задания)

Без вытеснения — процесс не прерывается до завершения.

11. Планирование процессов в интерактивных системах

Интерактивные системы ориентированы на реакцию на действия пользователя. Поэтому ОС должна быстро переключаться между процессами, чтобы интерфейс не "зависал". Для этого применяются вытесняющие алгоритмы, такие как Round Robin, где каждому процессу даётся небольшой отрезок времени. Также может использоваться система приоритетов, где более важные задачи (например, ввод с клавиатуры) получают больше внимания.

12. Планирование процессов в системах реального времени

Здесь важно не просто выполнить процесс, а сделать это в строго определённый момент. В таких системах, например в медицинском оборудовании или в авиации, ошибка по времени может привести к катастрофе. Алгоритмы планирования учитывают крайние сроки выполнения:

RMS (чем чаще задача, тем выше приоритет)

EDF (задача с ближайшим сроком — выполняется первой)

13. Операции над процессами

Операционная система должна уметь управлять процессами. Основные операции включают:

создание нового процесса,

завершение существующего,

приостановку и возобновление,

ожидание завершения другого процесса,

взаимодействие между процессами (например, передача данных).

14. Приоритеты процессов

ОС может давать процессам разные приоритеты, чтобы одни выполнялись чаще или быстрее других. Это особенно важно, когда ресурсов не хватает на всех сразу. Приоритет может быть установлен пользователем или системой и часто меняется динамически — например, если фоновый процесс работает слишком долго, его приоритет могут понизить.

15. Понятие потока. Ресурсы и потоки в операционной системе

Поток — это "лёгкий" процесс, который живёт внутри обычного процесса и делит с ним ресурсы, включая память. Потоки удобны, когда нужна параллельная работа: один поток может выполнять вычисления, а другой — загружать данные. Это улучшает производительность, особенно на многоядерных процессорах.

16. Понятие ресурса. Классификация ресурсов

Ресурсы — это всё, что нужно процессам: процессор, память, файлы, устройства ввода-вывода. Классификация:

Выделяемые и невыделяемые (например, память vs процессорное время),

Восстанавливаемые и невосстанавливаемые (например, файл можно пересоздать, а испорченный принтер — нет).

17. Дисциплины распределения ресурсов на основе очередей

Ресурсы в ОС часто выделяются по принципу очереди. Это может быть:

Обычная очередь (FIFO) — в порядке поступления,

Очередь с приоритетом — более важные задачи идут первыми,

Многоуровневые очереди — задачи делятся по классам (например, системные и пользовательские),

Кольцевое обслуживание (Round Robin) — все получают ресурсы по очереди.

17 (второй). Основные элементы графических интерфейсов

Графический интерфейс пользователя (GUI) состоит из визуальных компонентов: окон, кнопок, полей ввода, меню. Все эти элементы реагируют на действия пользователя — например, щелчок мыши или нажатие клавиши. GUI реализуется с помощью оконных систем (например, Windows API, X11 в Linux), а разработчики используют библиотеки для создания таких интерфейсов.

18. Интерфейс прикладного программирования (API)

API — это набор функций и правил, с помощью которых программы могут взаимодействовать с операционной системой или с другими программами. Например, API позволяет разработчику написать код, который сохраняет файл, отображает окно или получает доступ к интернету, не вникая в низкоуровневые детали. API бывает системным (WinAPI, POSIX) или сетевым (например, REST API для веб-сервисов).