**9. Понятие процесса. Состояния процесса**

**Процесс** — это программа, находящаяся в состоянии выполнения. Каждый процесс имеет:

* собственное адресное пространство,
* регистры процессора,
* стек и кучу,
* дескриптор процесса (PID).

**Состояния процесса** (классическая модель):

1. **Новый (New)** — процесс создан, но ещё не запущен.
2. **Готовность (Ready)** — процесс ожидает выделения CPU.
3. **Выполнение (Running)** — процесс выполняется на CPU.
4. **Ожидание (Waiting/Blocked)** — процесс ждёт внешнего события (ввод/вывод, сигнал).
5. **Завершённый (Terminated)** — процесс завершил выполнение.

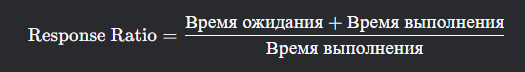
*(Возможны дополнительные состояния, например, "Приостановлен" (Suspended) в многозадачных системах.)*

**10. Планирование процессов в системах пакетной обработки**

**Пакетная обработка** — выполнение задач без интерактивного взаимодействия (например, обработка данных, компиляция).

**Алгоритмы планирования**:

1. **FCFS (First-Come, First-Served)** — в порядке очереди.
   * *Плюсы:* простота.
   * *Минусы:* возможны длительные простои из-за "долгих" задач.
2. **SJF (Shortest Job First)** — сначала выполняются самые короткие задачи.
   * *Плюсы:* минимизирует среднее время ожидания.
   * *Минусы:* требует знания времени выполнения.
3. **HRRN (Highest Response Ratio Next)** — выбирает задачу с наибольшим коэффициентом:



*(В пакетных системах важна****максимальная загрузка CPU****, а не отзывчивость.)*

**11. Планирование процессов в интерактивных системах**

**Интерактивные системы** требуют быстрого отклика (например, ОС для ПК).

**Алгоритмы планирования**:

1. **Round Robin (RR)** — циклическое выделение квантов времени (например, 10-100 мс).
   * *Плюсы:* справедливость, низкая задержка.
   * *Минусы:* возможны частые переключения контекста.
2. **Multilevel Queue (MLQ)** — несколько очередей с разными приоритетами (например, системные процессы → интерактивные → фоновые).
3. **Multilevel Feedback Queue (MLFQ)** — динамическое изменение приоритета (если процесс использует весь квант — понижается приоритет).

*(Критерий эффективности —****время отклика****.)*

**12. Планирование процессов в системах реального времени**

**Системы реального времени (RTOS)** требуют гарантированного времени выполнения.

**Типы**:

* **Жёсткие (Hard RT)** — нарушение сроков недопустимо (управление роботом).
* **Мягкие (Soft RT)** — небольшие задержки допустимы (стриминг).

**Алгоритмы**:

1. **Rate-Monotonic (RM)** — статический приоритет (чем чаще задача, тем выше приоритет).
2. **Earliest Deadline First (EDF)** — динамический приоритет (ближайший дедлайн выполняется первым).

*(Критерий —****предсказуемость****, а не fairness.)*

**13. Операции над процессами**

1. **Создание** (fork() в Unix, CreateProcess() в Windows).
2. **Завершение** (exit()).
3. **Ожидание** (wait() — родительский процесс ждёт завершения дочернего).
4. **Замена образа процесса** (exec() — загрузка новой программы).
5. **Изменение приоритета** (nice в Unix).
6. **Сигналы** (kill — отправка сигналов процессу).

**14. Приоритеты процессов**

Приоритет определяет порядок доступа к CPU.

**Типы**:

* **Статический** (назначается при создании).
* **Динамический** (меняется в runtime, например, в MLFQ).

**Примеры**:

* В Windows: 32 уровня (0-31).
* В Unix: nice от -20 (высокий) до +19 (низкий).

*(Высокоприоритетные процессы могут вызывать****инверсию приоритетов****.)*

**15. Понятие потока. Ресурсы и потоки в ОС**

**Поток (thread)** — легковесный процесс, разделяющий память с другими потоками.

**Ресурсы процесса**:

* **Общие**: память, файлы.
* **Индивидуальные**: регистры, стек.

**Плюсы потоков**:

* Быстрое создание/переключение.
* Эффективное использование CPU (параллелизм).

*(В ОС потоки могут быть:****пользовательскими****(управляются библиотекой) или****ядерными****(управляются ОС).)*

**16. Понятие ресурса. Классификация ресурсов**

**Ресурс** — любой компонент системы (CPU, память, устройства ввода-вывода).

**Классификация**:

1. **По типу**:
   * Аппаратные (CPU, RAM, HDD).
   * Программные (файлы, сокеты, мьютексы).
2. **По возможности разделения**:
   * Разделяемые (RAM, файлы).
   * Неразделяемые (принтер).
3. **По способу выделения**:
   * Статические (выделяются на всё время).
   * Динамические (запрашиваются при необходимости).

**17. Дисциплины распределения ресурсов на основе очередей**

1. **FIFO (First-In, First-Out)** — первым пришёл, первым обслужен.
2. **LIFO (Last-In, First-Out)** — стековая модель.
3. **Приоритетные очереди** — высокоприоритетные задачи обслуживаются первыми.
4. **Round Robin** — циклическое распределение.

*(Используются в планировщиках, диспетчерах ввода-вывода и сетевых очередях.)*

**18. Основные элементы графических интерфейсов**

1. **Окна** (Windows, Tabs).
2. **Меню** (File, Edit).
3. **Кнопки** (Button).
4. **Поля ввода** (Textbox).
5. **Полосы прокрутки** (Scrollbar).
6. **Диалоговые окна** (Modal/Non-modal).
7. **Панели инструментов** (Toolbar).

*(Основаны на****событийно-ориентированной модели****.)*

**19. Интерфейс прикладного программирования (API)**

**API** — набор функций, предоставляемых ОС или библиотекой для взаимодействия с системой.

**Примеры API в ОС**:

* **POSIX** (Unix/Linux).
* **WinAPI** (Windows).
* **Java API** (JVM).

**Типы API**:

1. **Системные вызовы** (read(), write()).
2. **Библиотечные функции** (стандартная библиотека C).
3. **Сетевые API** (HTTP-запросы).

*(API упрощают разработку, абстрагируя низкоуровневые детали.)*