Процессы в ОС

Что такое процесс?

– это абстракция, представляющая собой выполняющуюся программу. Это не просто статический файл с кодом, а динамическая сущность, которая в данный момент времени использует ресурсы компьютера для выполнения определенных задач.

Характеристики процесса

- Независимость
- Динамичность
- Ресурсы
- Идентефикатор

Зачем нужны процессы

- Многозадачность
- Изоляция
- Управление

Пример

Когда вы запускаете веб-браузер, операционная система создает новый процесс для этого приложения. Этот процесс получает выделенную память, процессорное время и другие ресурсы, необходимые для отображения веб-страниц, обработки запросов и т.д. Если вы откроете несколько вкладок в браузере, для каждой вкладки будет создан отдельный процесс.

Аналогия

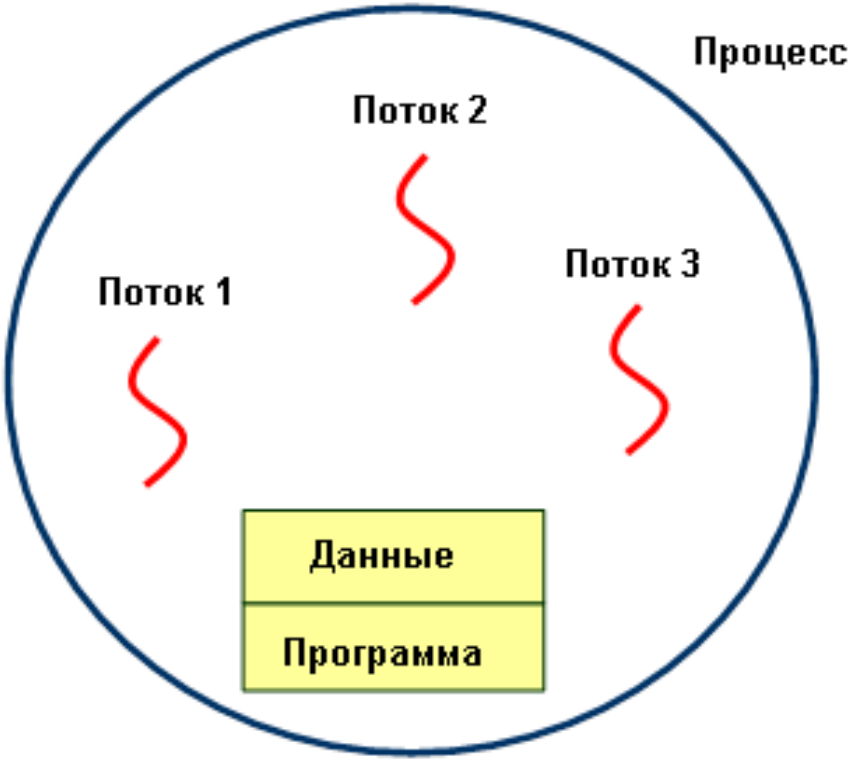
Представьте, что каждый процесс – это отдельный рабочий на фабрике. Каждый рабочий имеет свою рабочую станцию, инструменты и задание. Рабочие могут работать одновременно, но не мешают друг другу. Мастер-участник (операционная система) координирует работу всех рабочих, распределяет задания и ресурсы.

Дополнительные понятия

Поток: Более мелкая единица выполнения внутри процесса. Несколько потоков могут выполняться в контексте одного процесса, разделяя общую память.

Контекст процесса: Совокупность данных, характеризующих состояние процесса в определенный момент времени.

Дополнительные понятия



Дополнительные понятия

Планировщик процессов: Компонент операционной системы, отвечающий за распределение процессорного времени между процессами.

Зачем это знать вообще

Понимание процессов важно для:

- Разработки программного обеспечения
- Администрирования систем
- Понимания принципов работы операционных систем

Создание процесса

– это фундаментальная операция в любой операционной системе, позволяющая запускать новые программы и выполнять задачи параллельно. Этот процесс происходит по определенным алгоритмам и включает в себя несколько ключевых этапов.

Причины создания процесса

- Запуск программ
- Выполнение фоновых задач
- Разделение задач

Этапы создания процесса

1. Системный вызов

2. Выделение ресурсов

- адресное пространство

-описатели файлов

- сигналы

- идентификатор процесса

3. Копирование контекста

4. Инициализация

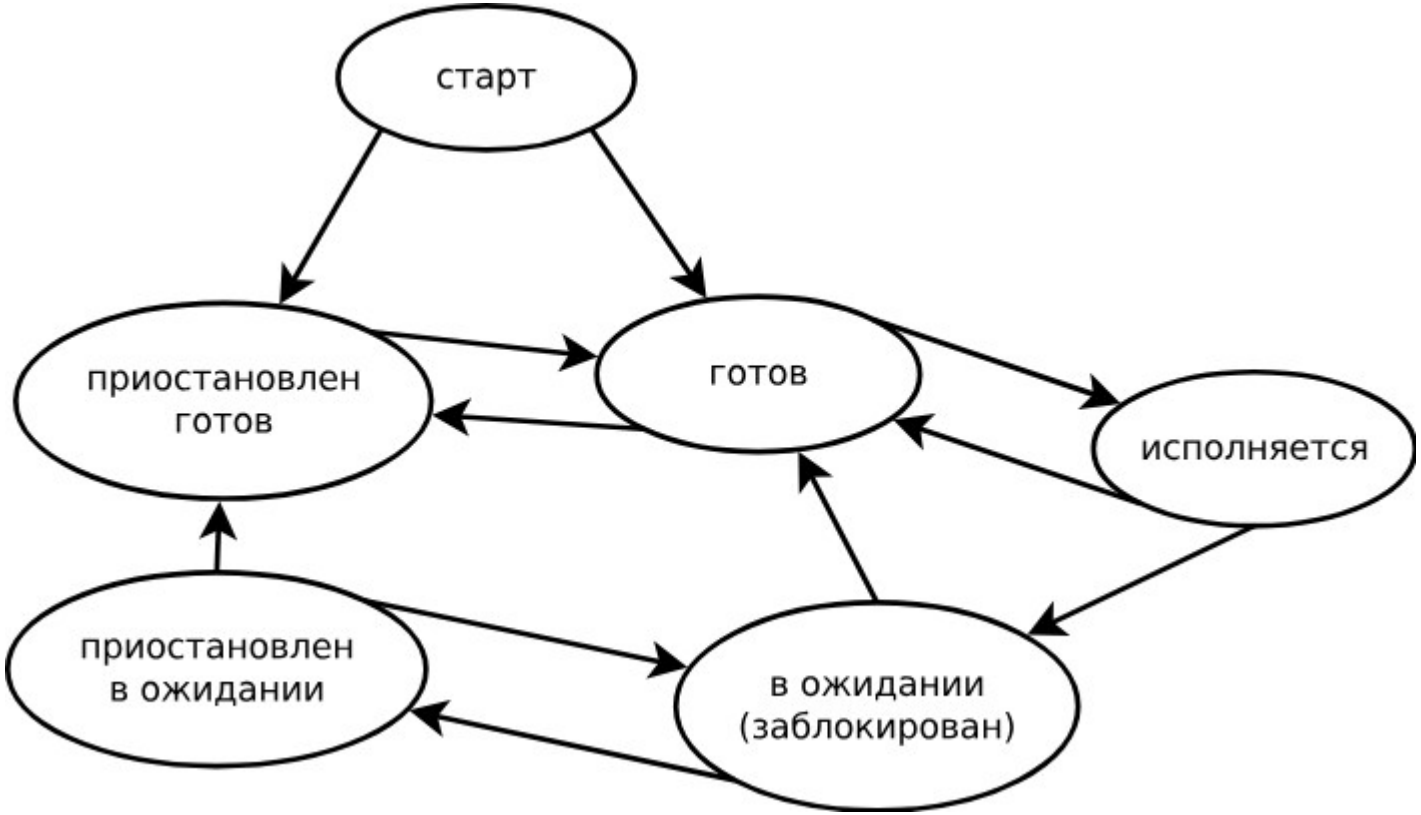
Каким бывает создание процесса

- Создание дочернего процесса
- Запуск программы
- Инициализация системы

Состояния процесса

- **Готовность:** Процесс готов к выполнению, но ожидает своей очереди.
- **Выполнение:** Процесс активно использует процессор.
- **Ожидание:** Процесс приостановлен в ожидании какого-либо события (например, завершения ввода-вывода).

Диаграмма состояний процесса



Завершение процесса

– это естественный этап жизненного цикла любой программы. Это происходит, когда программа успешно выполнила свою задачу, столкнулась с ошибкой или была принудительно остановлена.

Причины завершения процесса

- Нормальное завершение
- Ошибка
- Сигнал
- Принудительное завершение

Этапы завершения процесса

1. Удаление из очередей планирования
2. Освобождение ресурсов
3. Уведомление родительского процесса
4. Удаление записей о процессе

Способы завершения процесса

- Вызов системной функции
- Сигнал
- Принудительное завершение

Коды завершения процесса

Процесс может возвращать код завершения, указывающий на результат его работы. Например, 0 обычно означает успешное завершение, а ненулевое значение – ошибку.

Важно отметить

- Зомби-процессы: Если родительский процесс не собирает информацию о завершении дочернего процесса, то дочерний процесс переходит в состояние зомби. Это может привести к утечке ресурсов.

- Иерархия процессов: Процессы организованы в иерархическую структуру, где каждый процесс имеет родительский процесс.

Что такое сигналы

- это асинхронное уведомление процесса о каком-либо событии. Это как прерывание, но на программном уровне. Сигналы позволяют процессам взаимодействовать между собой, а также реагировать на различные события, такие как ошибки, системные события или запросы от других процессов.

Зачем нужны

- Межпроцессное взаимодействие
- Обработка ошибок
- Управление процессами

Как работают

1. Генерация сигнала
2. Доставка сигнала
3. Обработка сигнала

Что делает обработчик

- Завершение процесса
- Игнорирование сигнала
- Выполнение какого-либо действия

Типы сигналов

- SIGINT: Прерывание от клавиатуры (обычно генерируется при нажатии Ctrl+C).
- SIGTERM: Запрос на завершение процесса.
- SIGKILL: Принудительное завершение процесса.
- SIGSEGV: Ошибка сегментации (некорректный доступ к памяти).
- SIGALRM: Сигнал таймера.

Пример на C++

```c
#include <signal.h>

void my_handler(int sig) {
    printf("Получен сигнал %d\n", sig);
    exit(1);
}

int main() {
    signal(SIGINT, my_handler);
    while (1); // Бесконечный цикл
    return 0;
}
```

Пример на C++

В этом примере мы устанавливаем обработчик для сигнала SIGINT. Когда пользователь нажмет Ctrl+C, будет вызван обработчик, который выведет сообщение и завершит программу.

Иерархия процессов

– это древовидная структура, которая отражает взаимосвязи между процессами в системе. Каждый процесс, за исключением первого (инициализационного) процесса, имеет родительский процесс, который его создал. Этот родительский процесс, в свою очередь, может иметь своих дочерних процессов и так далее.

Почему это важно

- Управление процессами
- Наследование ресурсов
- Межпроцессное взаимодействие
- Завершение процессов

Пример

Представьте, что вы запустили терминал. Терминал – это процесс. Затем вы запустили в терминале текстовый редактор. Текстовый редактор будет дочерним процессом терминала. Если вы откроете несколько файлов в текстовом редакторе, каждый файл может быть представлен отдельным процессом, который будет дочерним процессом текстового редактора.

Инициализационный процесс

В каждой операционной системе есть один специальный процесс, который запускается при загрузке системы. Этот процесс называется инициализационным процессом (init). Все остальные процессы в системе являются потомками этого процесса, напрямую или косвенно.

Инициализационный процесс

В каждой операционной системе есть один специальный процесс, который запускается при загрузке системы. Этот процесс называется инициализационным процессом (init). Все остальные процессы в системе являются потомками этого процесса, напрямую или косвенно.

Дополнительные понятия

- PID (Process ID): Уникальный идентификатор процесса.

- PPID (Parent Process ID): Идентификатор родительского процесса.

- Орбита процесса: Множество всех потомков процесса.

Как работать с процессами в ОС

Для просмотра иерархии процессов в Unix-подобных системах можно использовать команду ps aux. В Windows для этого можно использовать диспетчер задач.