Процессы и потоки

Лекция 2 (из 16) по курсу Операционные системы на кафедре КИМ КФУ им. В.И. Вернадского

Зуев С.В. 17-02-2025

Модель процесса

запуск

- **Процесс** - это экземпляр выполняемой программы вместе с текущими значениями счетчика команд, регистров и переменных.

У каждого процесса - свой виртуальный процессор. В реальности - они делят один. Это - многозадачный режим работы

- **Процессы создаются**: при инициализации системы, при создании дочернего процесса, по запросу пользователя, при инициализации пакетного задания.
- **Системные процессы** делятся на **высокоприоритетные** и **демоны**. Демоны основную часть времени находятся в ожидании или осуществляют служебные функции (можете называть последние джиннами).
- **Отображение запущенных процессов**: в UNIX/Linux: ps, в Windows - диспетчер задач.
- В UNIX/Linux запущенный пользователем **процесс работает** в том же окне, где запущен; в Windows - сам создает под себя окно или окна.
- В UNIX/Linux есть только один **вызов** для создания дочернего процесса - fork() - который делает полную копию родительского процесса и только потом они начинают жить по-разному. В Windows функция CreateProcess создает процесс и в него загружается нужная программа (у этого вызова много параметров).

Модель процесса

завершение и иерархии

- Процесс завершается: обычным выходом (добровольно), выходом при ошибке (добровольно), из-за фатальной ошибки (принудительно), уничтожением другим процессом (принудительно).
- Системные вызовы окончания процесса: exit (UNIX/Linux), ExitProcess (Windows).
- Когда дочерний процесс сам создает процесс, формируется **иерархия процессов**.

В загрузочном образе UNIX есть процесс init. В начале работы он считывает файл, в котором указано количество терминалов и разветвляется, порождая по одному процессу на каждый терминал. Они ждут пока кто-нибудь не зарегистрируется в системе. Успешная регистрация порождает оболочку для приема команд. Эти команды порождают другие процессы и т.д. Все процессы в итоге принадлежат одному дереву, в корне которого стоит процесс init.

- В Windows нет иерархии процессов - там все процессы равнозначные. У родительского процесса есть маркер (дескриптор) для управления дочерним процессом - вот и вся иерархия. Этот маркер еще и передавать можно.

Модель процесса

состояния процессов

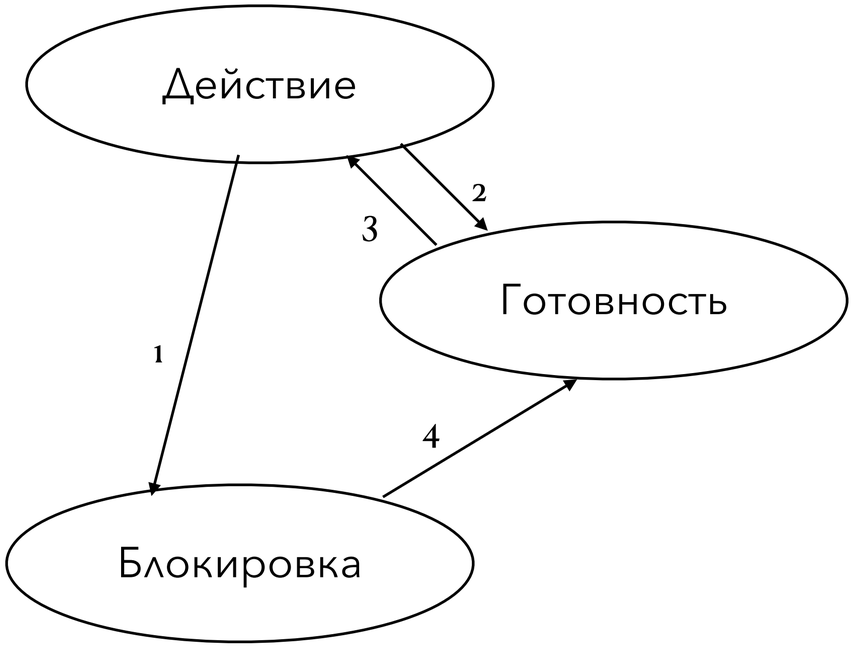
- Состояния отображены на диаграмме

1 - Процесс заблокирован в ожидании ввода (процесс выполняться не может по причине, не связанной с доступностью процессора).

2 - Диспетчер выбрал другой процесс

3 - Диспетчер выбрал данный процесс

4 - Входные данные стали доступны (произошло событие, которое сделало процесс доступным для исполнения).



Реализация процессов

управляющая таблица

- Процессы реализуются при помощи таблицы процессов, в которой каждая запись соответствует отдельному процессу;
- В таблице процессов собраны все параметры процесса, которые нужно сохранять при переходе из состояния выполнения в состояние готовности или блокировки, чтобы при возобновлении процесса ничего не выдало то, что он стоял.
- В таблицу процессов входят:
  - регистры, счетчики команд, слово состояния, указатель стека, состояние процесса, приоритет, параметры планирования, идентификатор процесса, родительский процесс, группы процесса, сигналы, время запуска процесса, использованное время процессора, время использованное дочерними процессами, время следующего аварийного сигнала (это про управление процессом);
  - указатель на информацию о текстовом сегменте, указатель на информацию о сегменте данных, указатель на информацию о сегменте стека (это про управление памятью);
  - корневой каталог, рабочий каталог, дескрипторы файлов, идентификатор пользователя, идентификатор группы (это про управление файлами).

Реализация процессов

прерывания

- Фиксированная область памяти, связанная с каждым классом устройств ввода-вывода, содержащая адрес процедуры, обслуживающей прерывание, - это **вектор прерывания**.
- Когда машина переходит на адрес, указанный в векторе прерывания, начинается процедура обслуживания прерывания.

Все прерывания сначала сохраняют состояния регистров (используя таблицу процессов). Затем стек очищается и указатель стека переставляется на временный стек, используемый обработчиком прерывания. Эти действия не могут быть выражены на языках высокого уровня (C) и выполняются небольшой подпрограммой на ассемблере, одной и той же для всех прерываний.

- После сохранения состояния регистров и очистки стека подпрограммой на ассемблере, вызывается C-процедура, которая делает всё остальное по этому прерыванию.

См. следующий слайд

Реализация прерываний

Схема работ низшего уровня ОС при прерывании

1. Оборудование перемещает в стек счетчик команд и т.п.
2. Оборудование загружает новый счетчик команд из вектора прерываний.
3. Процедура на ассемблере сохраняет регистры.
4. Процедура на ассемблере устанавливает указатель на новый стек.
5. Запускается процедура на языке С, обслуживающая прерывание (как правило, она считывает входные данные и помещает их в буфер).
6. Планировщик принимает решение какой процесс запускать следующим.
7. Процедура на языке С возвращает управление коду на ассемблере.
8. Процедура на ассемблере запускает новый текущий процесс

Процесс может быть прерван тысячи раз, но каждый раз он возвращается в точности к тому состоянию, в котором он был до прерывания.

Многозадачность

и ее степень

- Если в системе 4 процесса и вычисления по каждому занимают 25% времени пребывания в памяти, то CPU всегда загружен? Только в идеале - когда не случается ситуации, что все 4 процесса ждут окончания ввода-вывода.
- Если процессов n, то вероятность того, что все они ожидают ввода-вывода обозначим через p^n. Тогда время, в которое процессор задействован, вычисляется по формуле

$$T = 1 - p^n$$

- Это - функция от n (степени многозадачности) и она возрастает от 0 до 1, никогда не достигая 1. Чем меньше доля p времени ожидания ввода вывода, тем быстрее возрастает эта функция.