Лекция 01

ПСКП, ПОИТ-3

**ВВЕДЕНИЕ**

1. **ПСКП:** Программирование серверных кроссплатформенных приложений.
2. **ПСКП:** ассистент Дубовик Марина Владимировна
3. **ПСКП:** Учебный план

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Семестр | Лекции  (час) | Лабораторные работы (час) | Курсовой проект(час) | Итого  (час) |
| 3 | 34 | 51 |  | 85 |
| 4 | 54 | 54 | 40 | 148 |
| **Итого** | **88** | **105** | **40** | **233** |

1. **ПСКП:** клиент-серверное приложение = приложение (программа) с клиент-серверной архитектурой: приложение, состоящее из двух компонент – клиента и **сервера**; клиент и сервер взаимодействуют между собой в соответствии с заданными правилами (спецификациями, **протоколами**); для взаимодействия между клиентом и сервером в соответствии с правилами (спецификацией, протоколом) должно быть установлено **соединение**; **инициатором соединения всегда является клиент**.



1. **ПСКП: программирование серверного приложения**: разработка серверной компоненты приложения клиент-серверной архитектуры.
2. **ПСКП: в этом курсе** будут рассматриваться клиент-серверные приложения, клиент и сервер которых **расположены на узлах сети Интернет** (устройствах, подсоединенных к Сети) и взаимодействуют по **протоколу HTTP**.
3. **ПСКП: сеть Интернет:** 1) сеть на основе TCP/IP; 2) стандарты Internet (RFC, STD); 3) службы Интернет (DNS, SMTP/POP3/IMAP, WWW, FTP, Telnet, SSH,…); 4) организации, управляющие сетью Internet (ISOC, IETF, ICANN, IANA, IAB, …).



1. **ПСКП:** клиент-серверное приложение, у которого клиент и сервер взаимодействуют по протоколу HTTP называется **web-приложением**.
2. **ПСКП:** когда говорят о разработке web-приложения, говорят о разработке **frontend** (клиента) и **backend** (сервера)

1. **ПСКП: *курс посвящен*** разработке серверной части web-приложения или иначе ***разработке web-сервера (backend)***.

1. **ПСКП: узел Интернет:** устройство, имеющее IP-адрес и подключенное сети Интернет (обычно к сети Интернет-провайдера). Каждый узел характеризуется своей программно-аппаратной платформой – аппаратурой и операционной системой.

1. **ПСКП: кроссплатформенное приложение**: приложение, способное работать на более чем одной программно-аппаратной (аппаратура + операционная система) платформе. Кроссплатформенность может быть достигнута различными способами: 1) на уровне компилятора (С, С++); 2) на уровне среды (фреймворка) исполнения (Java/JVM, C#/.NET CORE, JS/Node.js/V8).
2. **ПСКП: в этом курсе** будут рассматриваться приложения, **кроссплатформенность** которых обеспечивается **средой исполнения**.
3. **ПСКП: *курс посвящен разработке web-серверов, которые могут работать на более чем одной программно-аппаратной платформе,* или иначе, *разработке кроссплатформенных web-серверов*.**
4. **HTTP**: основные свойства
5. **HTTP:** структура сервера
6. **ПСКП:** технологии для разработки кроссплатформенных web-серверов:

- PHP/Apache, LAMP;

- Java/JVM/Application Server;

- C#/ASP.NET CORE;

- Python/Django;

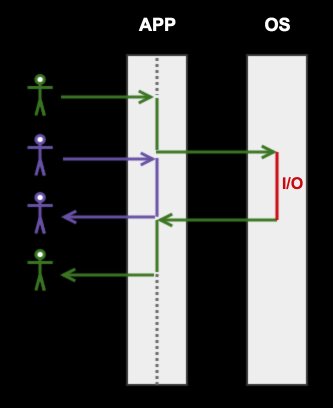
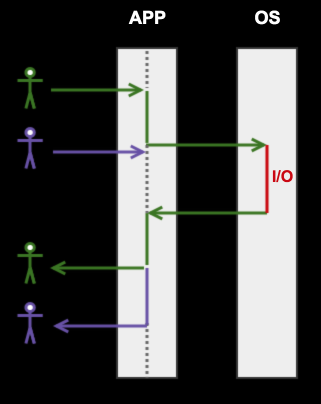
- Ruby on Rails;

- JS/Node.js, ….

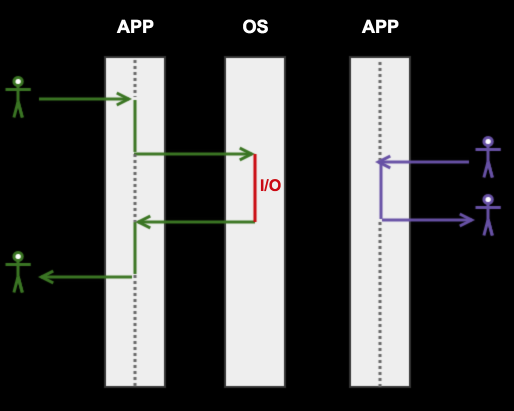
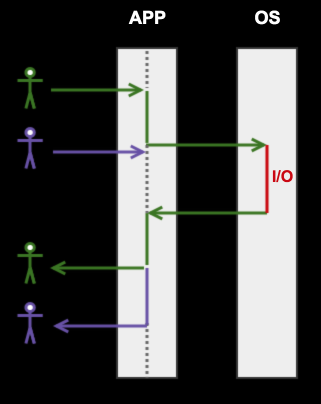
1. **ПСКП: *в этом курсе будет рассматриваться разработка web-серверов с помощью технологии JS/Node.js***
2. **Web-сервер:** ресурсы потребляемые web-сервером, блокирующий ввод/вывод



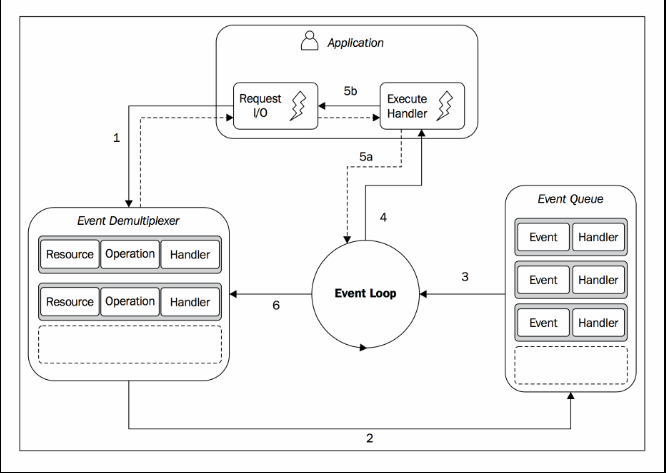
1. **IO-операции**: блокирующие и неблокирующие



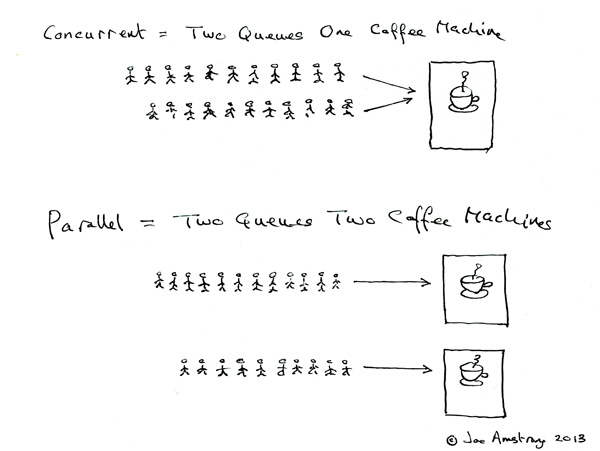
1. **Web-сервер:** 2 подхода для решения проблем блокирующего ввода/вывода: 1) применение многопоточности (ограничение по количеству потоков, каждый поток требует дополнительной памяти); 2) применение паттерна Reactor. Apache – многопоточность, Nginx – Reactor.
2. **Web-сервер: однопоточность** (единственная задача в один момент времени), **многопоточность** (различные задачи в один момент времени.



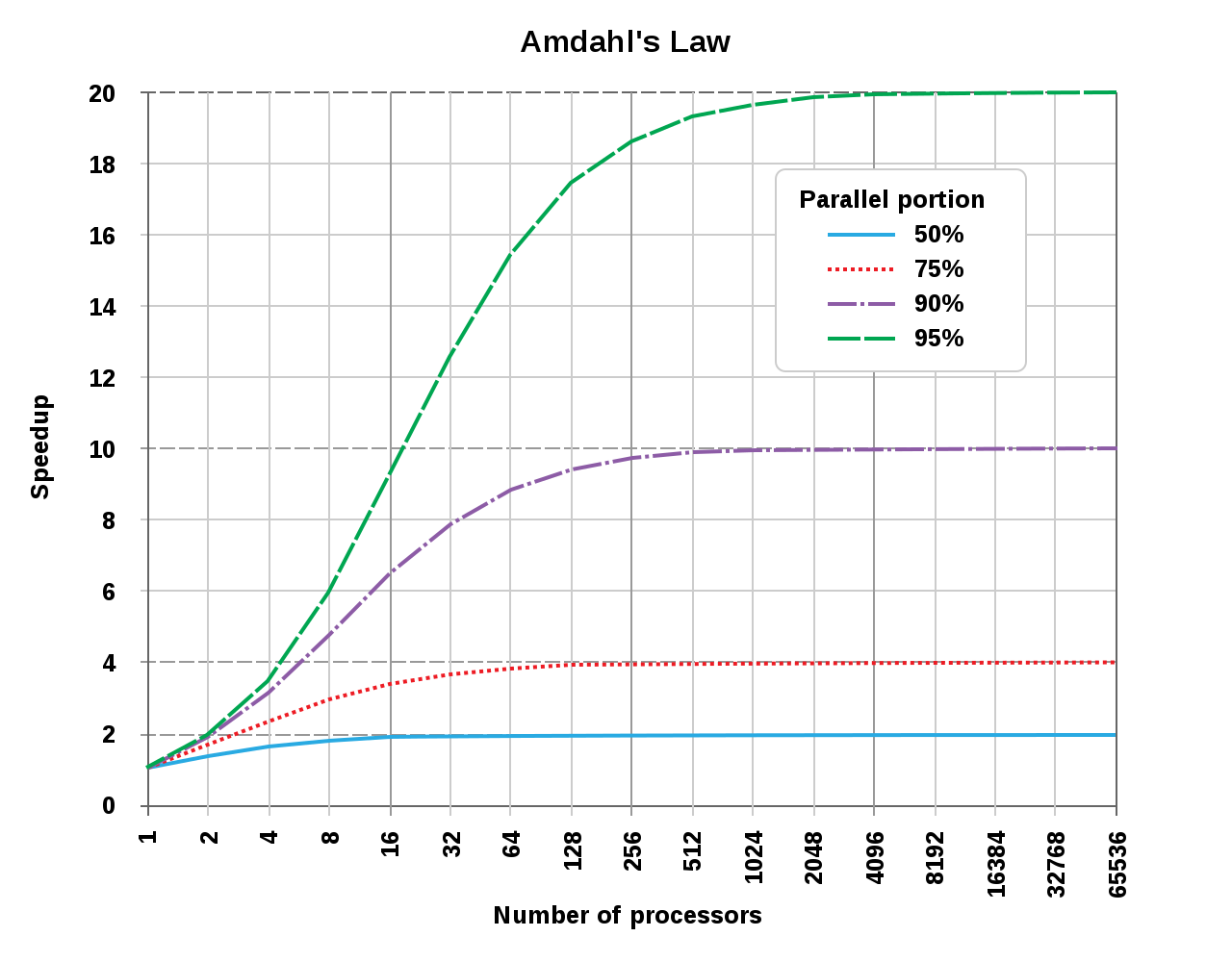
1. **NODEJS:** паттерн Reactor –шаблон проектирования. Используется при обработке параллельных запросов к сервису. Сервисный обработчик разбирает прибывшие запросы и синхронно перенаправляет их на соответствующие обработчики запросов.



1. **NODEJS**: конкурентность и параллельность



1. **NODEJS**: закон Амдала, ограниченность возможностей, speedup – кратность прироста скорости вычисления, parallel portion – степень распараллеливания алгоритма (не все можно распараллелить), number of processors – количество процессоров.



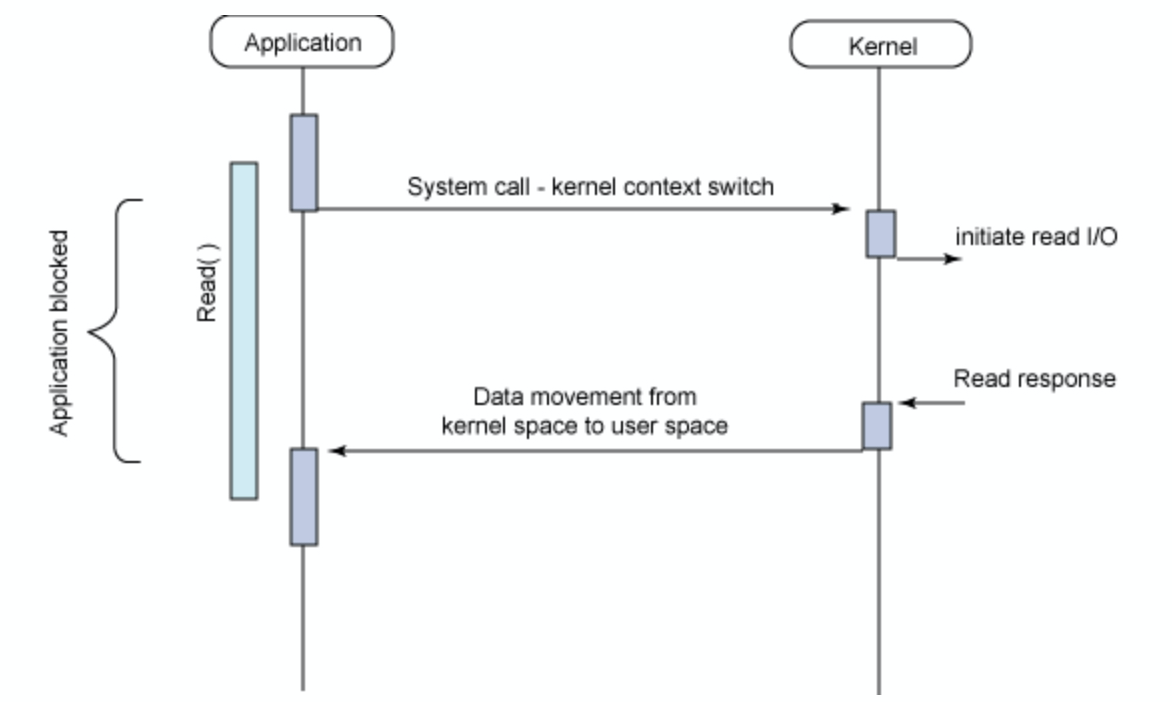
1. **Синхронный http-запрос:** запрос, при котором поток, выдавший http-запрос, блокируется до поступления запроса.
2. **Асинхронные запросы:** запрос, при котором поток, выдавший http-запрос, не блокируется до поступления запроса; для обработки ответа применяется функция обратного вызова.
3. **NODEJS**: при разработке web-приложений используется **асинхронное программирование**.
4. **Понятие асинхронности**: операция называется асинхронной, если ее выполнение осуществляется в 2 фазы: 1) заявка на исполнение; 2) получение результата; при этом участвуют два механизма: A-механизм, формирующий заявку и потом получающий результат; B-механизм, получающий заявку от A, исполняющий операцию и отправляющий результат A; продолжительность исполнения операции B-механизмом, как правило, непредсказуемо; в то время пока B-механизм исполняет операцию, А-механизм выполняет собственную работу. Применение асинхронности не противоречит применению многопоточности.
5. **Асинхронность в программировании**:



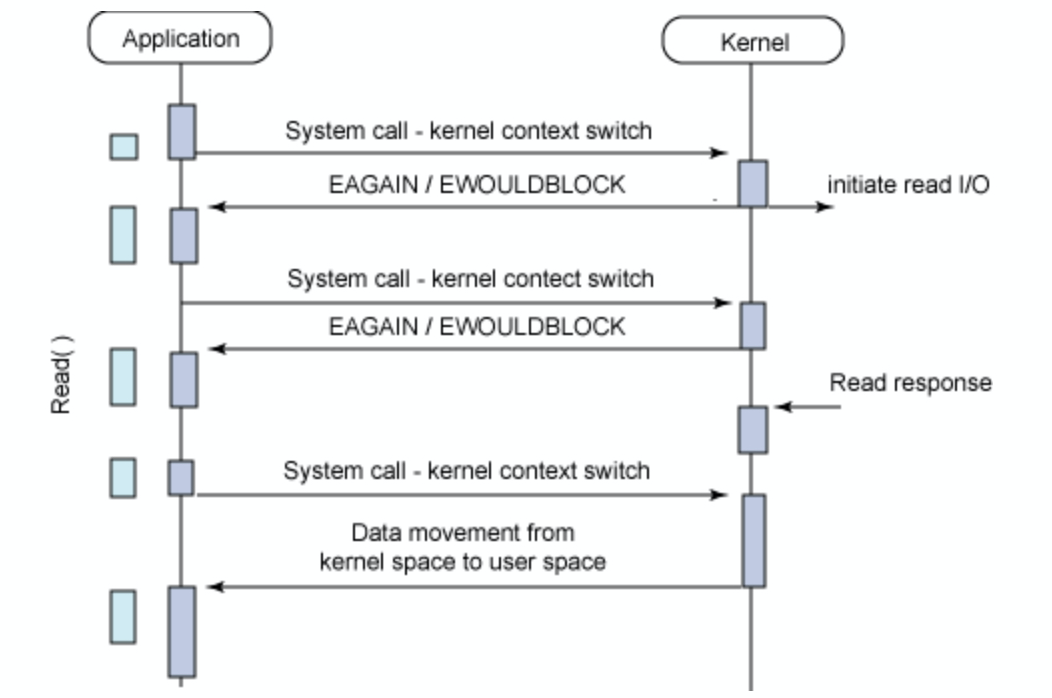
1. **Асинхронные операции ввода вывода в операционной системе:**



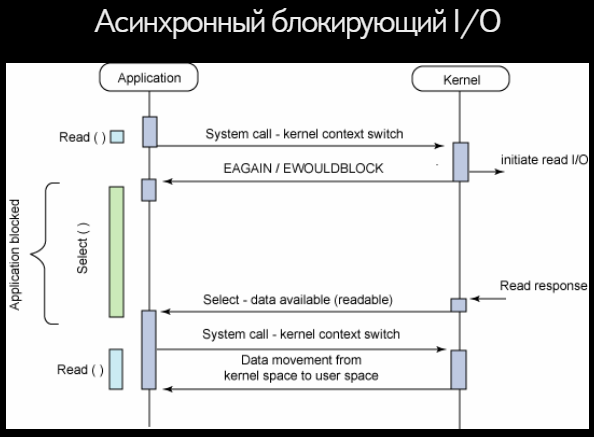
1. **Синхронный блокирующий ввод/вывод:** приложение блокируется до тех пор, пока операция ввода/вывода не будет завершена.



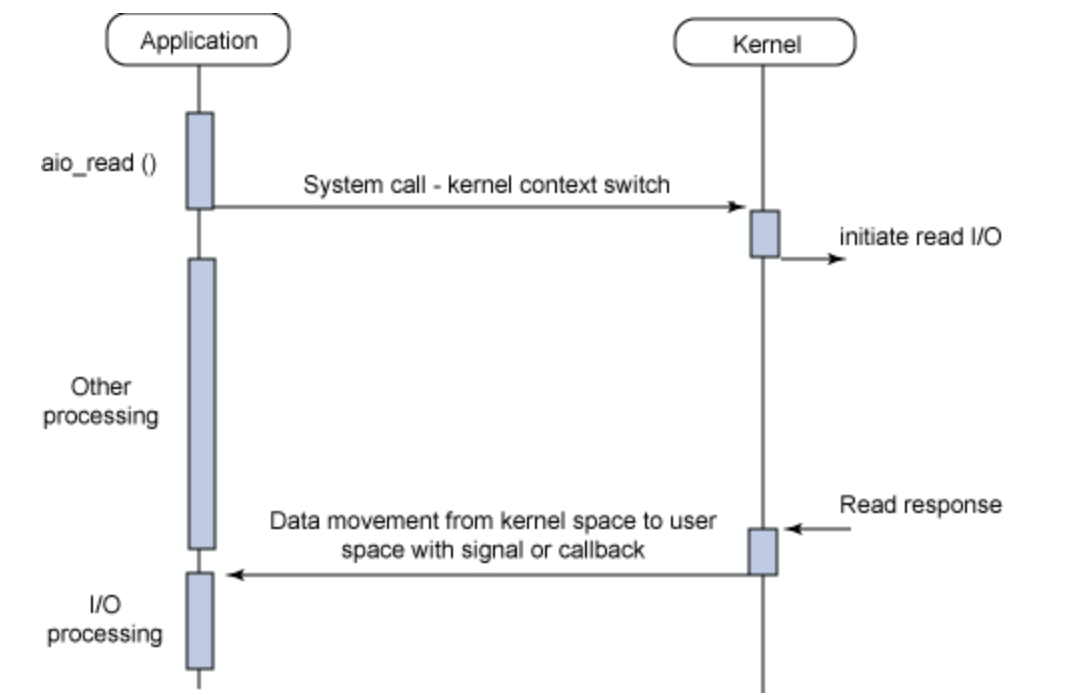
1. **Синхронный неблокирующий ввод/вывод:** приложение может сделать что-то еще вместо ожидания ввода/вывода; увеличивает количество переключений контекста между ядром и пользовательским пространством.



1. **Асинхронный блокирующий ввод/вывод:** как только файловый дескриптор, переданный вызову select (), готов к работе, вызов read() разблокируется, и приложение сможет извлечь и обработать данные.



1. **Асинхронный неблокирующий ввод/вывод:** приложение может выполнять вычисления, пока фактическое чтение выполняется в фоне ядром; при готовности ответа генерируется сигнал или обратный вызов для завершения ввода-вывода.



1. **AJAX: Asynchronous JavaScript and XML** – асинхронный JavaScript and XML – методология (подход) построения динамических приложений, при которых не осуществляется полная перезагрузка html-страниц.
2. **Асинхронные http-запросы**:



