

# Въведение в операционните системи

проф. д-р инж. Христо Вълчанов  
([hristo@tu-varna.bg](mailto:hristo@tu-varna.bg))

207- 4E

<http://cs.tu-varna.bg>

Технически университет – Варна  
Катедра “Компютърни науки и технологии”

---

Христо Георгиев Вълчанов

# ОПЕРАЦИОННИ СИСТЕМИ



Ръководство за лабораторни упражнения

---

Варна  
2022

## **Формат на изпита**

**електронен тест - 50 въпроса за 100 минути.**

**Провежда се в системата на ТУ**

**<https://elearning.tu-varna.bg>**

## **Оценка**

**Точките от упражнения (до 50т.) се събират с  
точките, получени от контролните (до 50т.)**

# Семестриален контрол

Всеки студент ще има задачи за домашна работа, чиито решение ще се показва на асистента в следващото занятие след компилиране и изпълнение. Непредставена в следващото занятие решена задача за домашно няма да се оценява повече.

Точките от задачите (до 50) ще формират семестриалния контрол.

Задачи за реализация:

- 1.Задача 4 от тема 4
- 2.Задача 2 от тема 5
- 3.Задача 1 от тема 7
- 4.Задача 3 от тема 7
- 5.Задача 2 от тема 8
- 6.Задача 1 от тема 9
- 7.Задача 2 от тема 9
- 8.Задача 3 от тема 9
- 9.Задача 3 от тема 12
- 10.Задача 1 от тема 13

# Литература

1. *Х. Вълчанов, Операционни системи. Ръководство за лаб. упражнения. ТУ-Варна, 2022. ISBN 978-954-20-0893-2*
2. *B. Ward, How Linux Works, 3rd Edition. 2021, No Starch Press, ISBN-10: 1718500408*
3. *C. Panek, Windows Operating System Fundamentals. 2019, Sybex, ISBN-10: 1119650518*
4. *G. Tomsho. Guide to Operating Systems 5th Edition, Cengage Learning, 2016, ISBN-10: 1305107640*
5. *P. Yosifovich, M. Russinovich, D. Solomon. Windows Internals: System architecture, processes, threads, memory management, and more, Microsoft Press, 2017, ISBN-10: 9780735684188*
6. *W. Stallings. Operating Systems. Internals and Design Principles 9th ed., Prentice Hall, 2017, ISBN-10: 9781292214290*

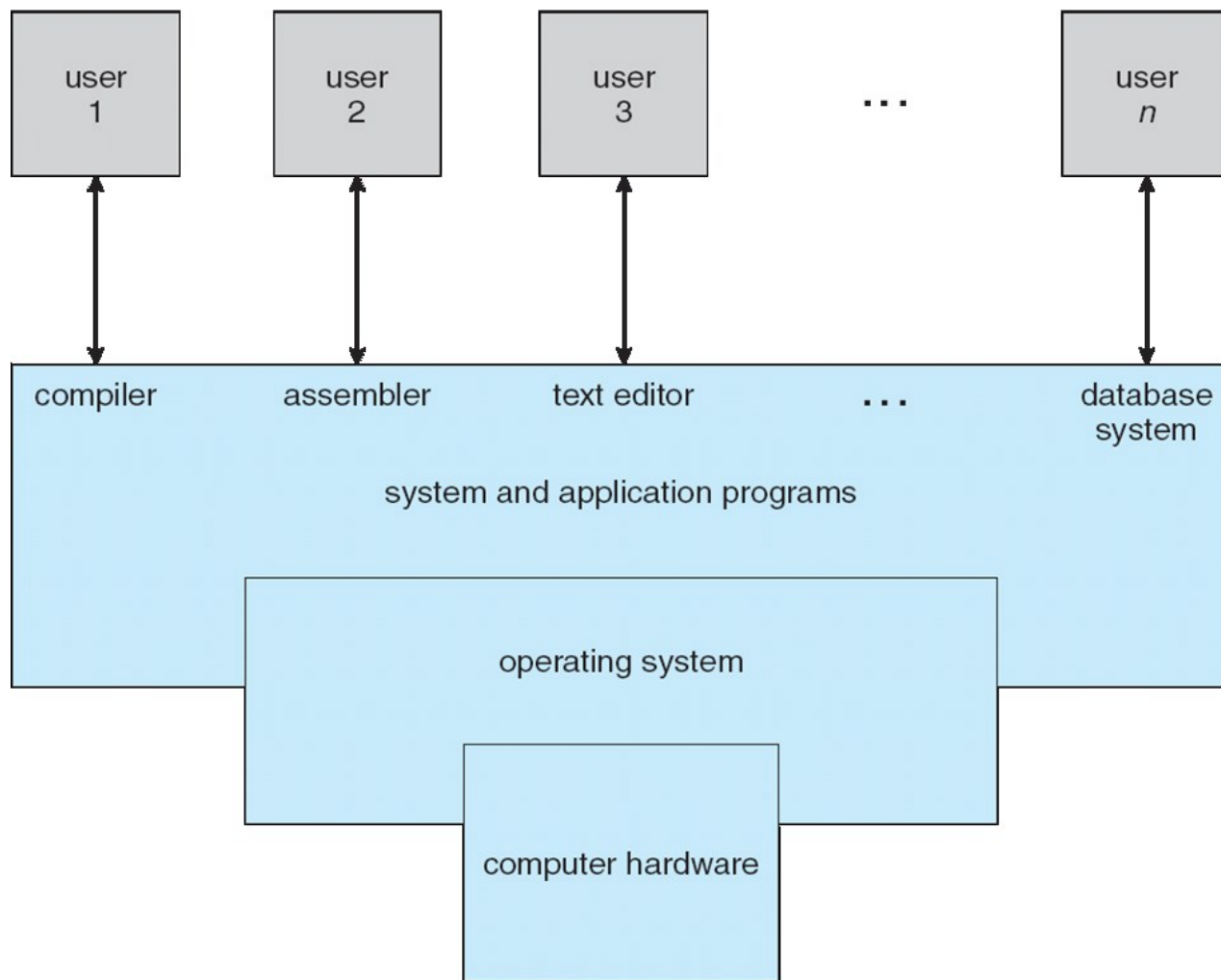
# Операционна система

- Програма, която действа като посредник между потребителя на компютъра и хардуера.
- Цели:
  - Изпълнение на потребителски програми.
  - Удобство при работа с компютърната система.
  - Ефективно използване на хардуера.

# Структура на компютърна система

- Хардуер – предоставя основни изчислителни ресурси (CPU, памет, I/O устройства).
- Операционна система – контролира и координира използването на хардуера от различни приложения и потребители.
- Приложни програми.
- Потребители.

# Компоненти на компютърна система





# Потребителска гледна точка

*Потребителската гледна точка варира в зависимост от интерфейса:*

- **Single user computers** (PC, workstations). Потребителят използва монополно ресурсите. Целта е максимално използване на ресурси от потребител. ОС е проектирана главно за **лесно** използване и **висока** производителност.
- **Multi user computers** (mainframes, computing servers). Потребителите споделят ресурси и обменят информация. ОС е проектирана за ефективно използване на ресурсите.

# Потребителска гледна точка

- **Handheld computers** (смартфони и планшети). Потребителският интерфейс е чрез touch screen. ОС са оптимизирани главно за удобно използване и спрямо живота на батерията.
- **Embedded computers** (компютри в домашни устройства или автомобили). Потребителският интерфейс има най-често числова клавиатура и индикация. ОС са проектирани главно за работа без намеса на потребителя..

# Системна гледна точка

*От гледна точка на компютъра ОС е програма, основно занимаваща се с хардуера. Две различни гледни точки:*

- ОС е **resource allocator**:
  - Управлява всички ресурси.
  - Решава конфликти между заявки за ефикасно и справедливо използване на ресурсите.
- ОС е **control program**:
  - Контролира изпълнението на програмите с оглед избягване на грешки и некоректно използване на компютъра.

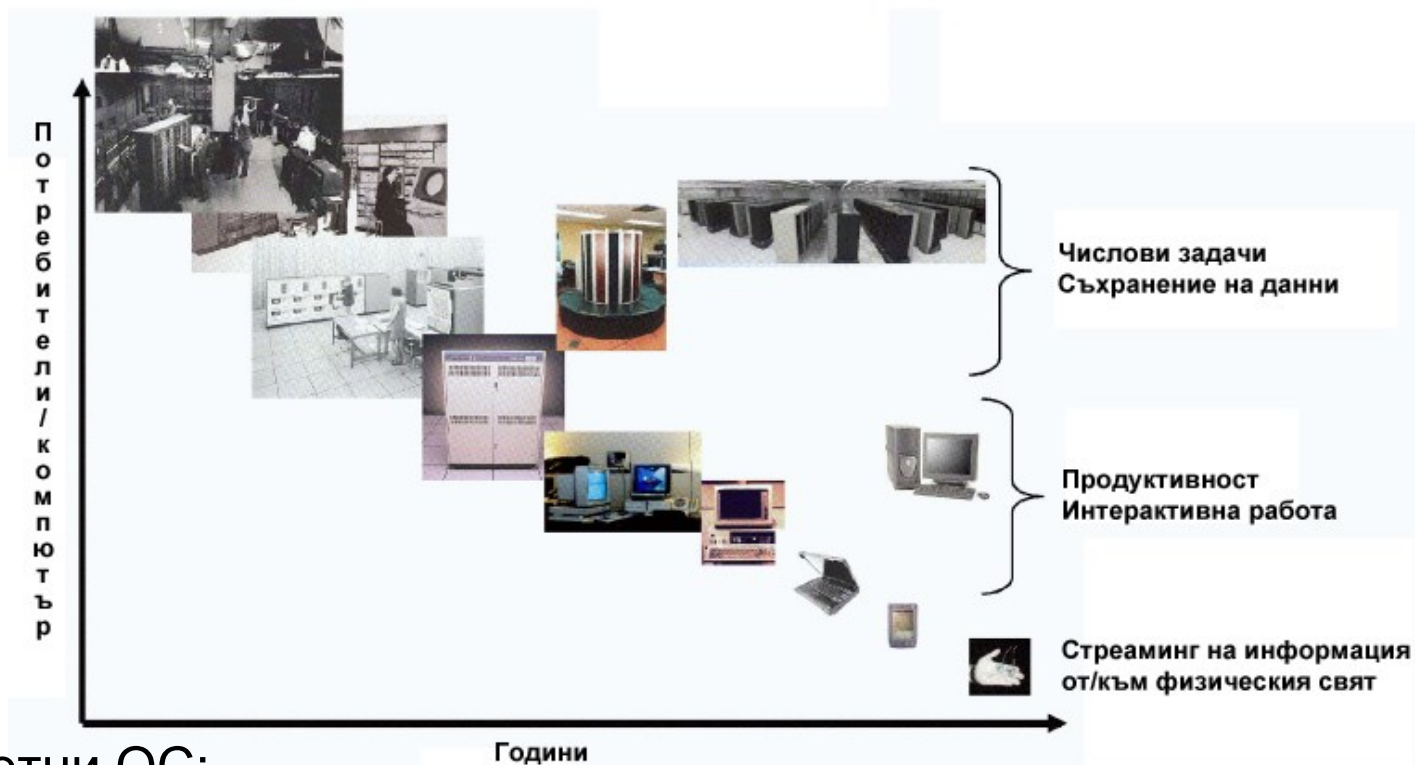
# Предизвикателства към ОС

- Производителност:
  - Време за отговор (latency);
  - Пропускателна способност (throughput);
  - Претоварване (overhead);
  - Справедливост (fairness);
  - Предвидимост (predictability);
- Преносимост:
  - За програми (Application Programming Interface);
  - За ядрата (Hardware Abstraction Layer);

# Използване на ОС



# Историческо развитие на ОС



- Ранни ОС;
- Единични пакетни ОС;
- Мултипрограмни ОС;
- Операционни системи с времеделение;
- Декстоп и мобилни ОС.

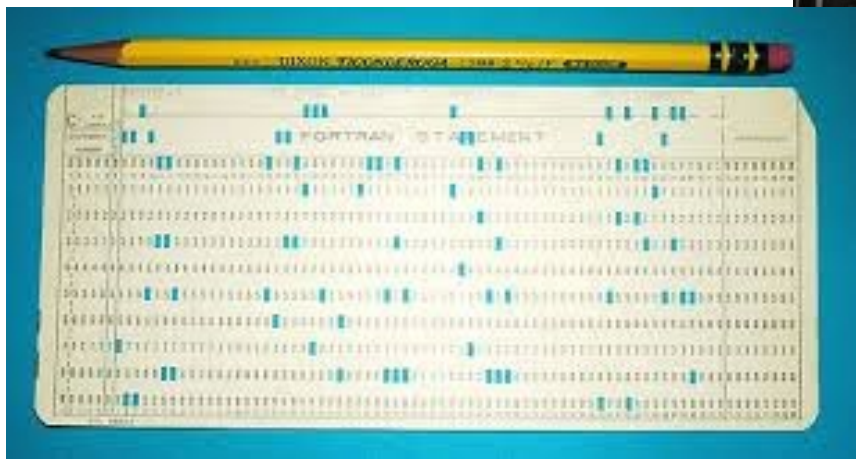
# Ранни ОС (1950)

- Структура
  - Големи машини
  - Работа от конзола
  - Единичен потребител
  - Магнитни ленти, перфокарти
- Софтуер
  - Асемблери
  - Компилатори
  - Свързващи редактори
  - Зареждащи програми
- Сигурно изпълнение
- Неефективно използване на ресурсите



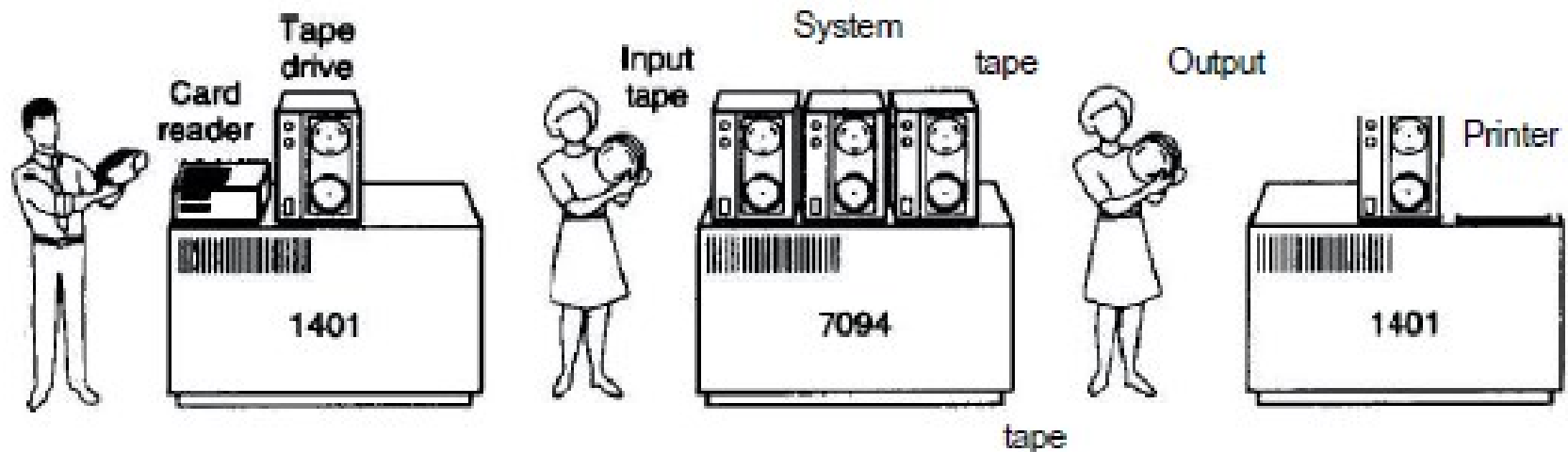
# Пакетни ОС (1960)

- Редуцира се времето за настройка чрез съвместяване на задачи с еднакви изисквания
- Изискват оператор (НЕ Е ПОТРЕБИТЕЛ)
- Бавни I/O устройства (card readers)
- Слабо натоварване на CPU

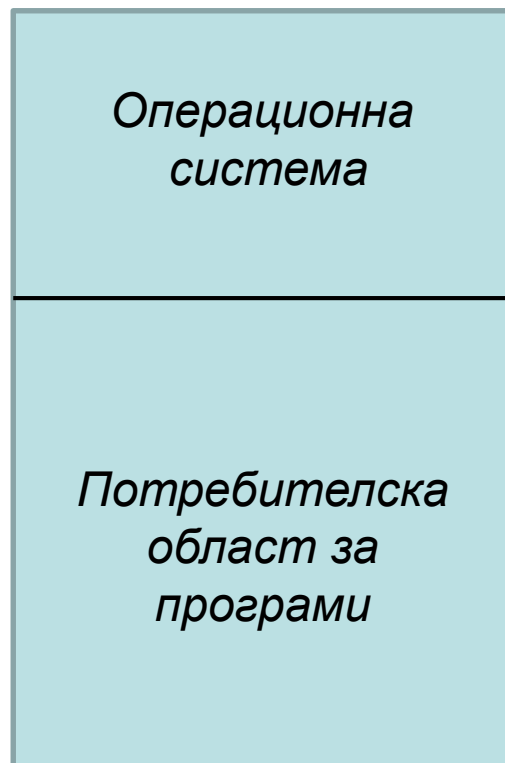




# Изчислителен процес

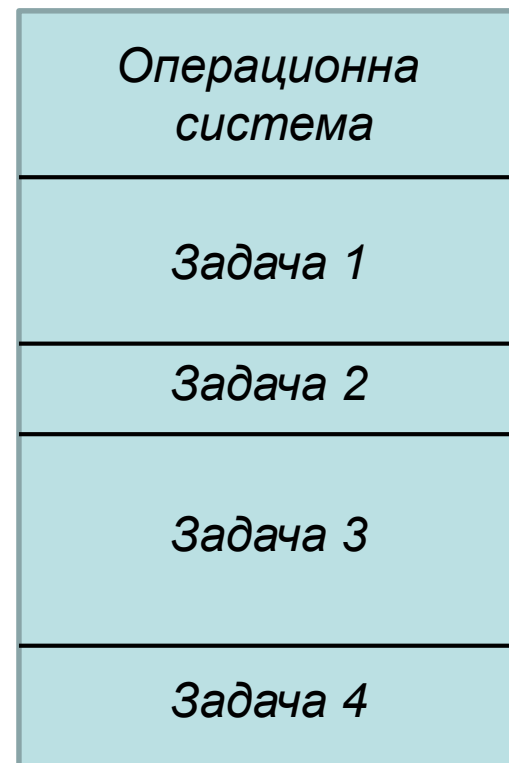


# Използване на паметта при пакетни ОС

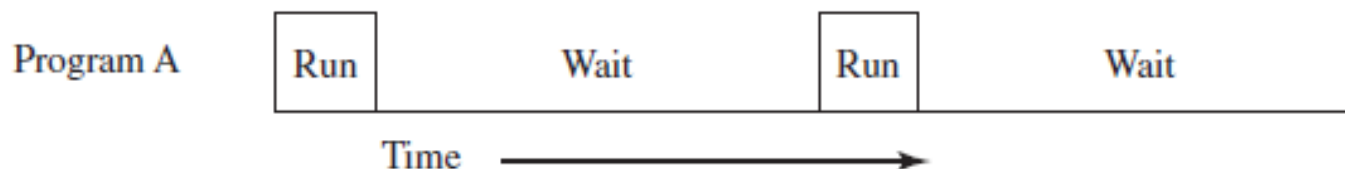


# Мултипрограмни ОС (1960)

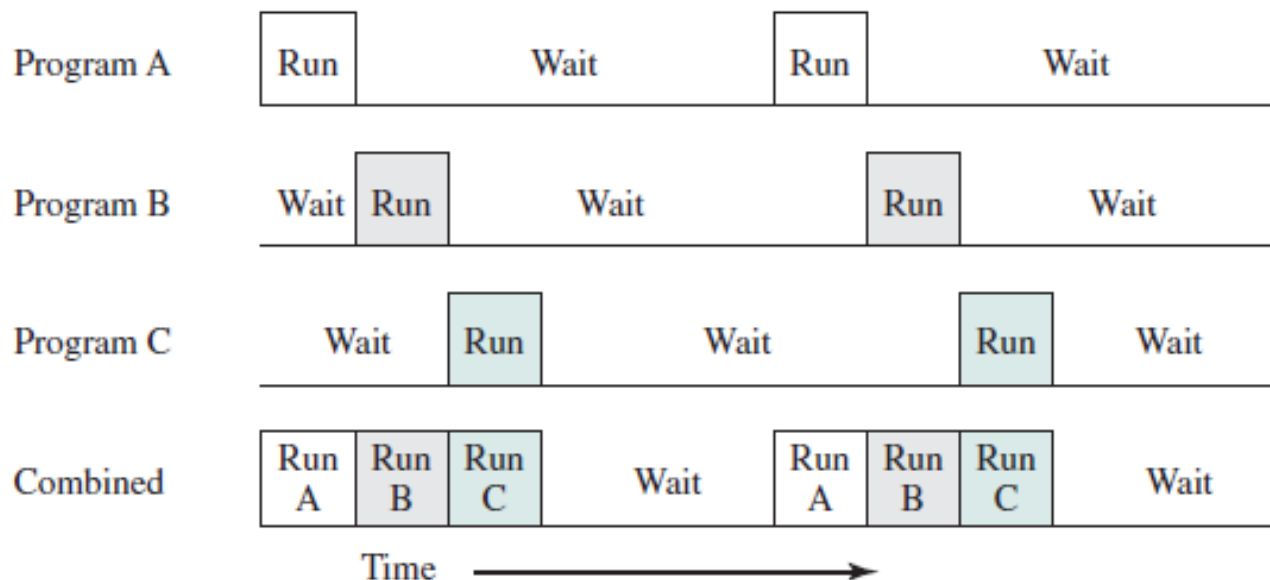
- Съхраняване на няколко задачи едновременно в паметта.
- Изпълнява се задача по задача.
- Ако задачата трябва да изчака някое действие (напр. I/O операция), ОС превключва за изпълнение друга задача.
- Нови изисквания – job scheduling, CPU scheduling, memory management.



# Мултипрограмни ОС



Еднопрограмно изпълнение



Мултипрограмно изпълнение

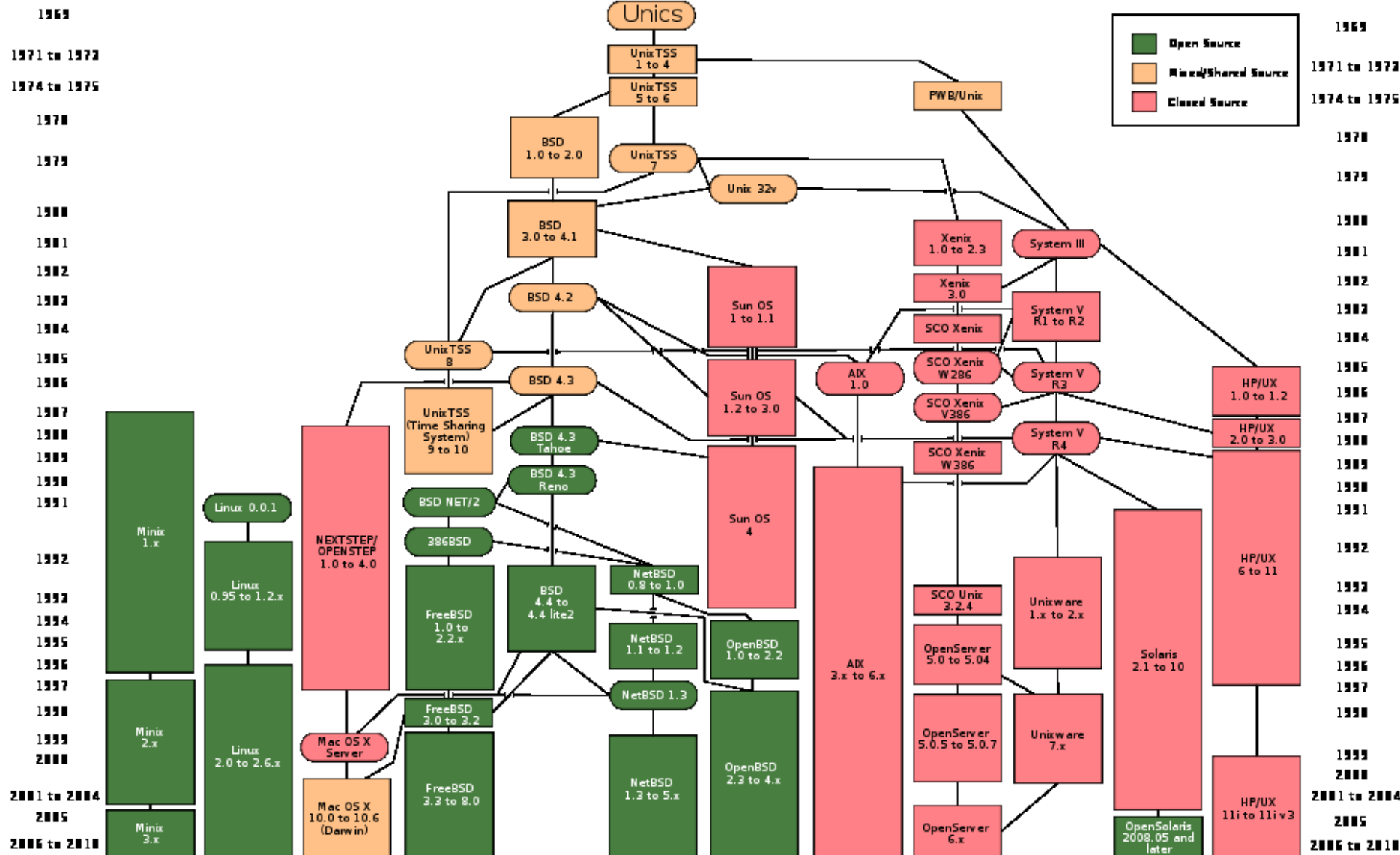
# ОС в режим на времешаделение (time-sharing)

- Интерактивен режим на потребителите – команден интерпретатор.
- Превключване изпълнението между задачите на базата на квант от време.
- Концепция за процес.
- Файлови системи.
- Виртуална памет.

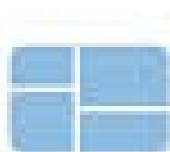
# Десктоп и мобилни ОС (1980)

- Самостоятелни потребителски системи.
- Разнообразни I/O устройства.
- Удобни за потребителя.
- Масово разпространени.

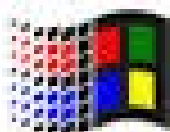
# Евпюіия на Unix



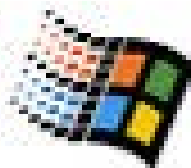
# Еволюция на Windows



Windows 1  
1985



Windows 3.1  
1992



Windows 95  
1995



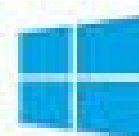
Windows XP  
2001



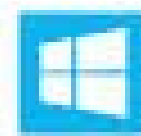
Windows  
Vista 2006



Windows 7  
2009



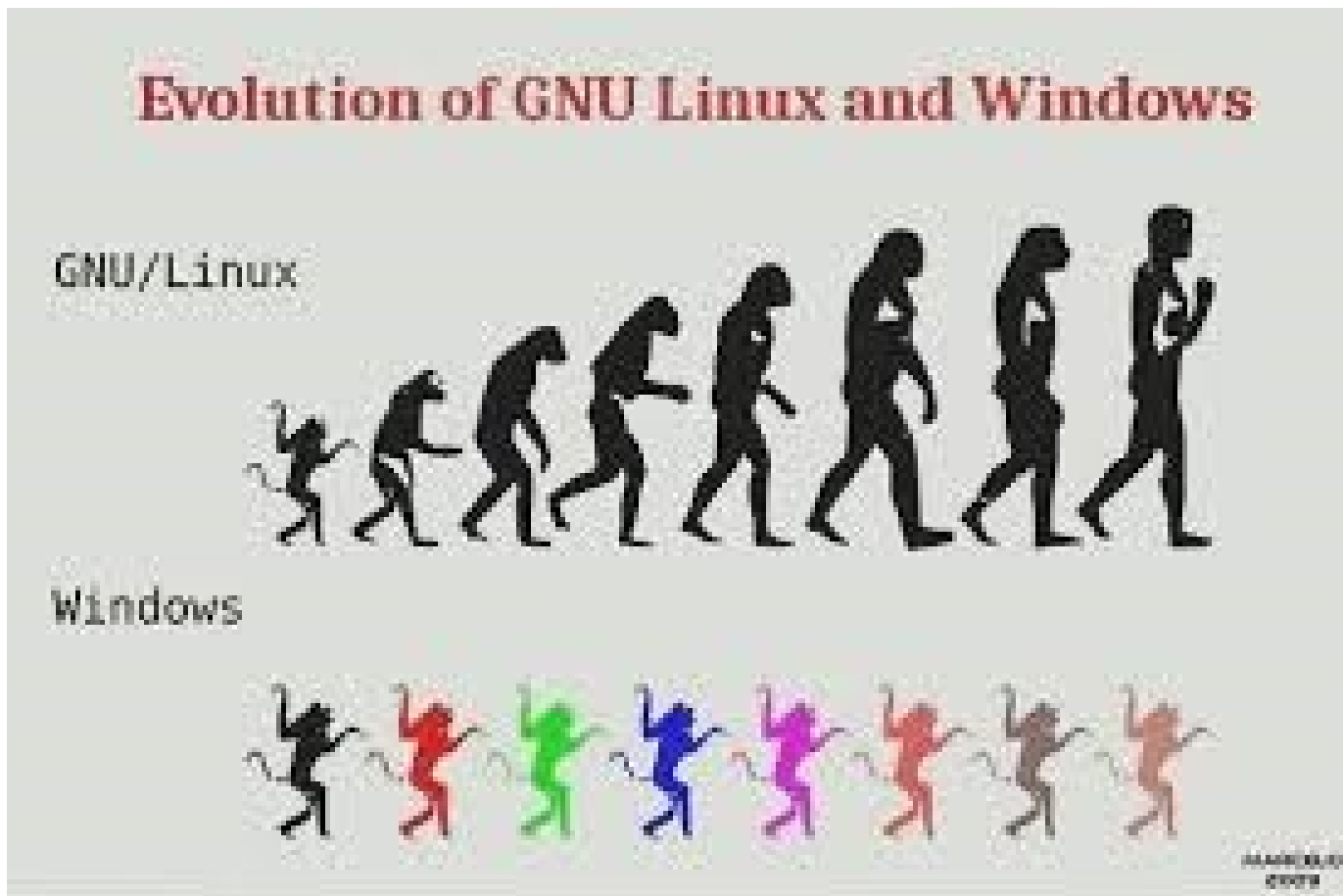
Windows 8  
2012



Windows 10  
2015



# Unix / Windows



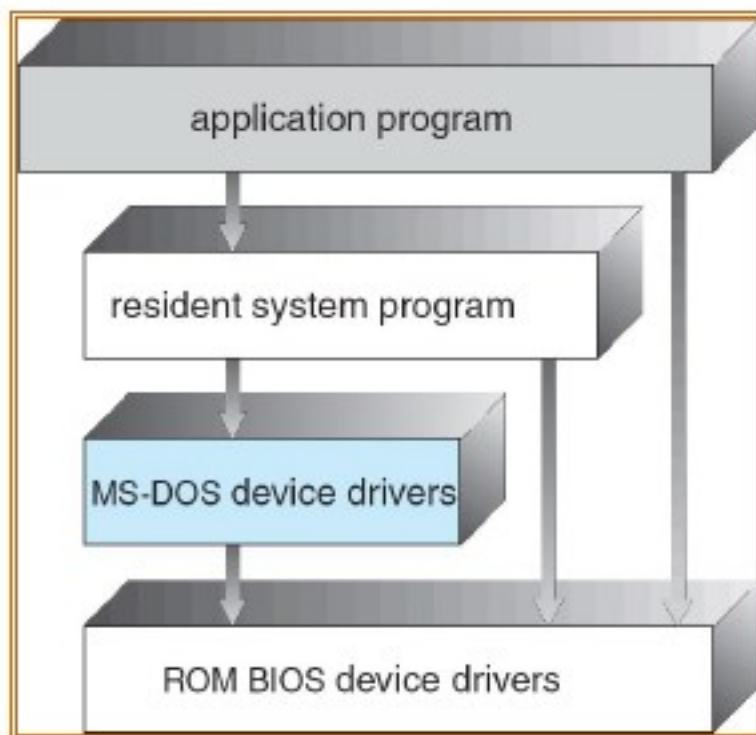
# Windows

Win 3.11 	Win 95 	Win 98 
Win Me 	Win 2000 	Win XP 
Vista 	Win 7 	Win 8 

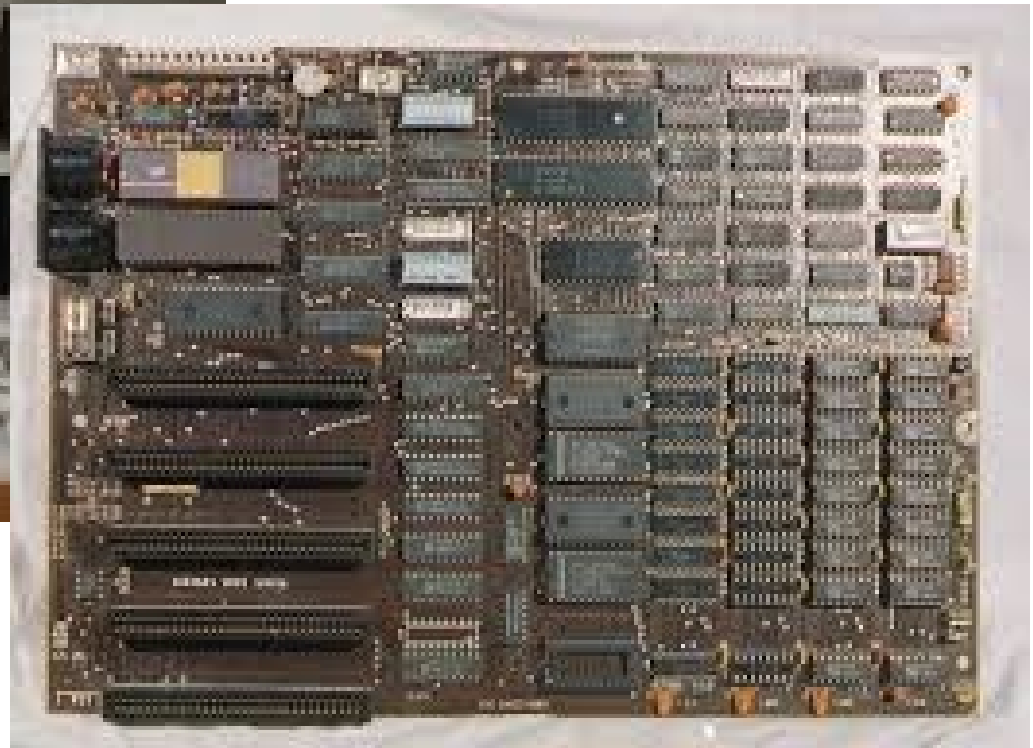


# Структура на ОС – опростен подход

- MS-DOS – функционалност при малък обем памет.

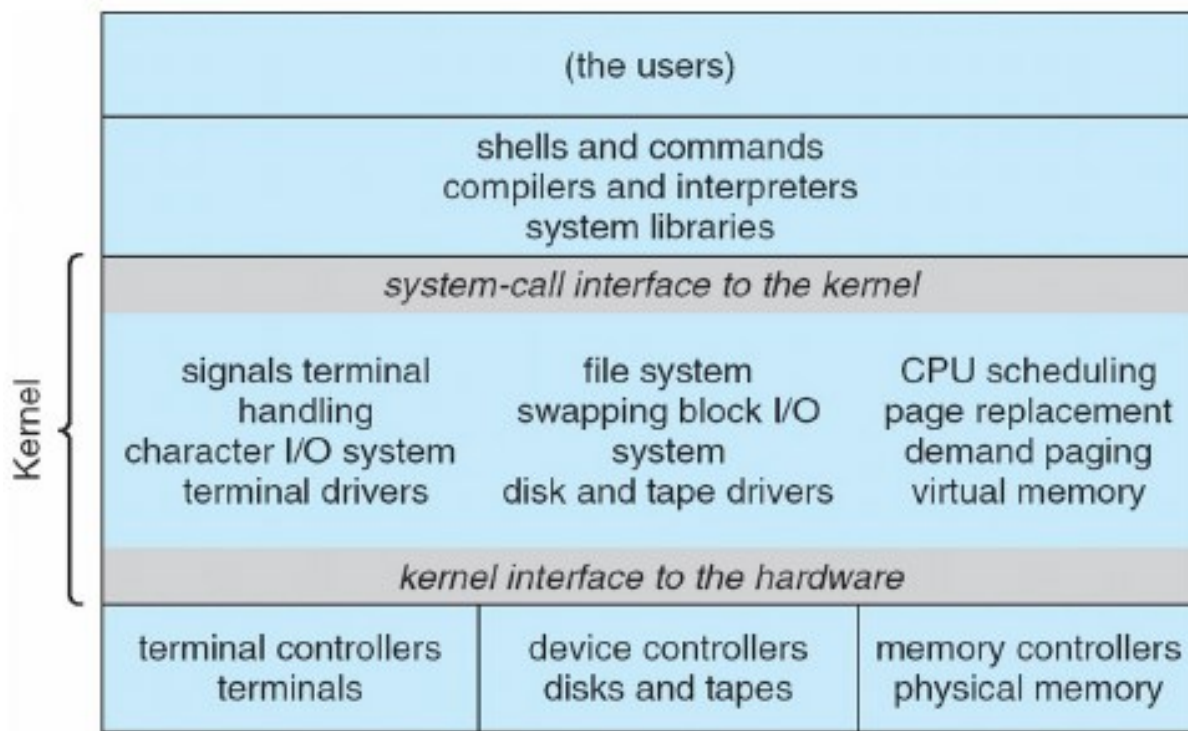


# IBM PC XT



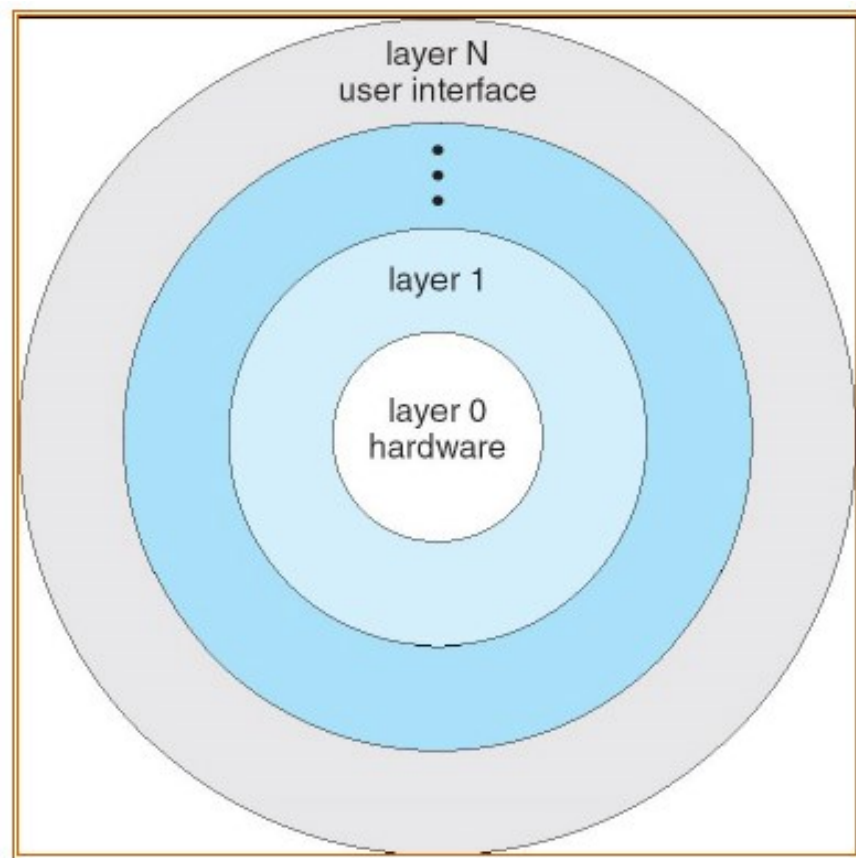
# Структура на ОС – опростен подход

- UNIX – две отделни нива:
  - Системни програми.
  - Ядро (kernel).



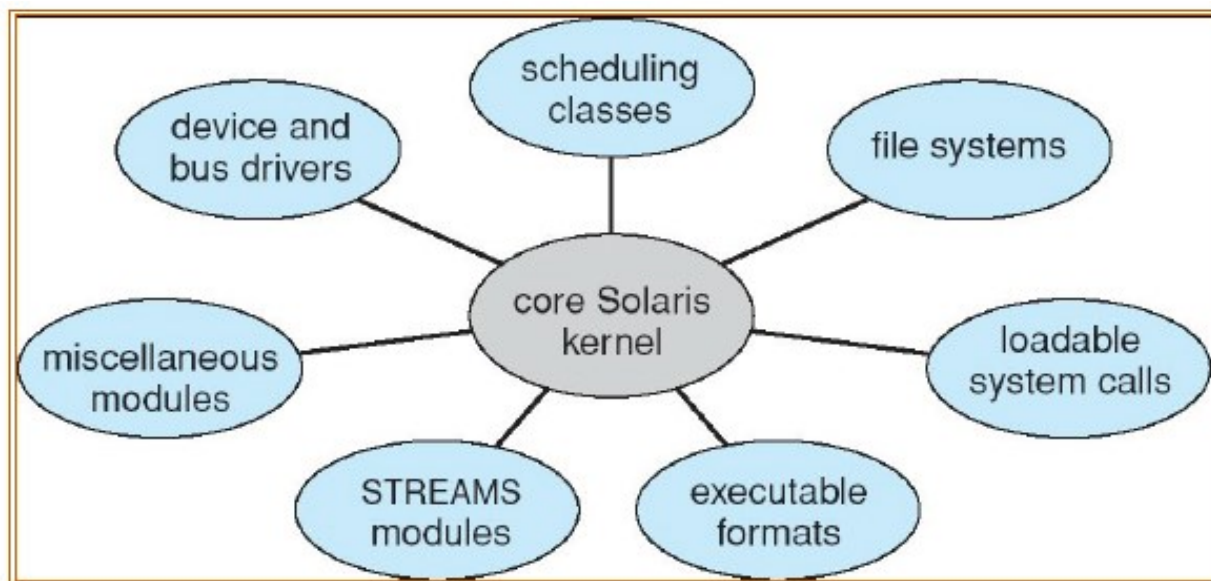
# Структура на ОС – слоеве

- ОС се разделя на отделни слоеве.
- Всеки слой използва функции и услуги само от слоя под него.
- Основно предимство – модулarity



# Модулна структура на ОС

- ОС се състои от множество модули:
  - Всеки компонент на ядрото е реализиран като отделен модул.
  - Компонентите си взаимодействат посредством дефинирани интерфейси.
  - Всеки модул се зарежда към ядрото при необходимост.
- По-голяма гъвкавост.

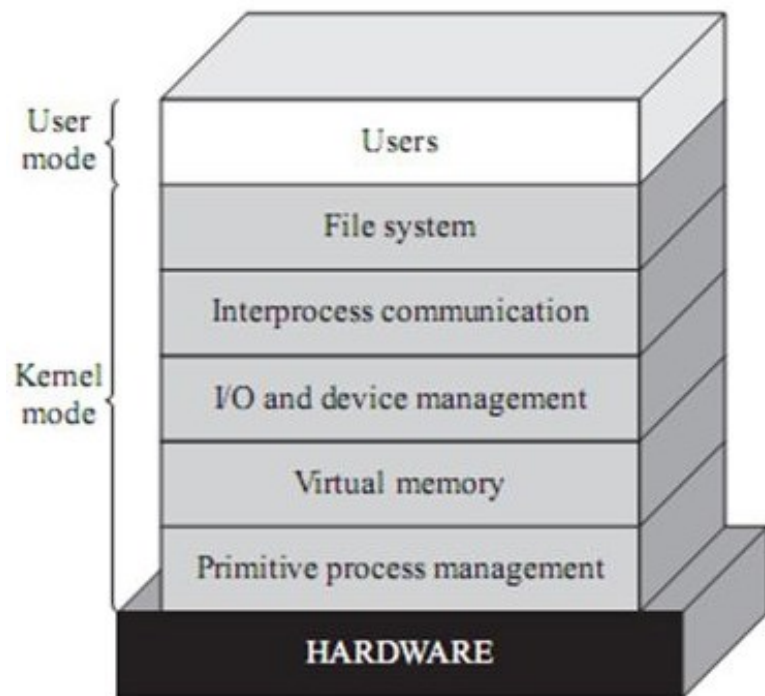


# Microkernel OS

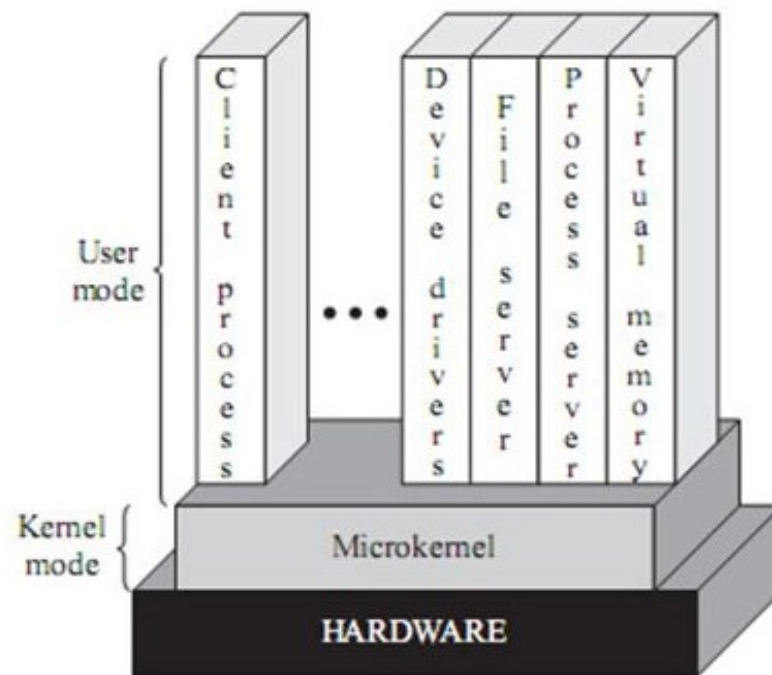
- Microkernel подход (Mach, Mac OS X Server, QNX):
  - Реализиране на някои компоненти на ядрото в т.н. “user mode”.
  - По-малко ядро.
  - Всеки модул се зарежда към ядрото при необходимост.
- Основна цел: ефективна комуникация между клиентски програми и услуги, изпълняващи се в user mode- **message passing**.
- Предимства:
  - Лесно разширяване на микроядрото.
  - Лесно адаптиране към други архитектури.
  - По-надеждно.
  - Изолация на отказите.



# Microkernel OS

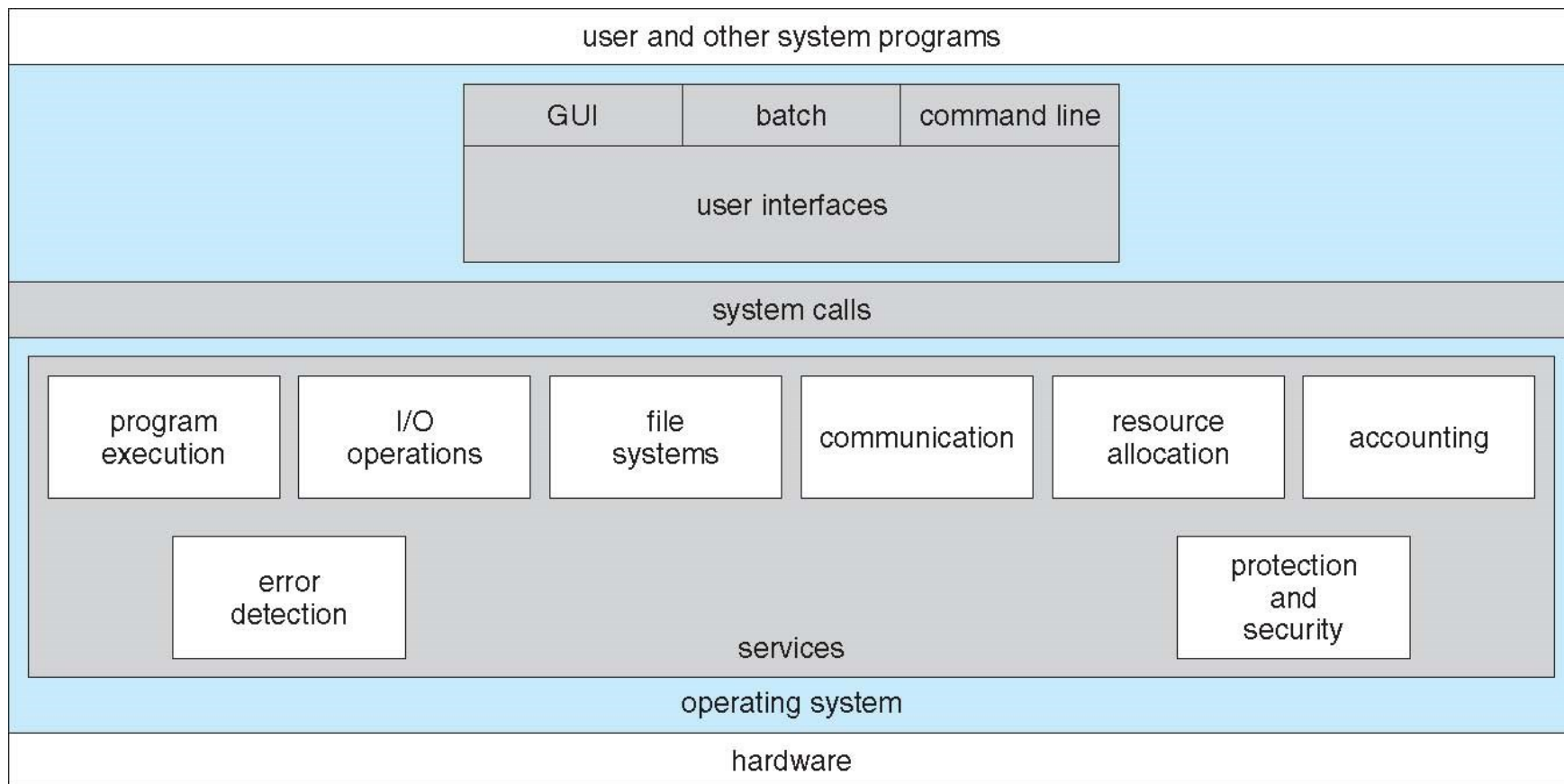


*Слоеста структура*



*Микроядро*

# Функционалност на ОС



# Функции на ОС. Управление на процесите

- Процес – фундаментална концепция в ОС:
  - Процес е инстанция на програма по време на нейното изпълнение.
  - Процесите изискват ресурси – CPU, I/O устройства, памет.
- Управление на процесите:
  - Създаване и унищожаване на процеси.
  - Преустановяване и възстановяване на процеси.
  - Синхронизация между процесите.
  - Междупроцесна комуникация.
  - Взаимодействие между процесите (откриване и предотвратяване на взаимна блокировка)

# Функции на ОС. Управление на паметта

- Основна памет (RAM).
  - Масив от адресируеми байтове.
  - Енергозависима (volatile).
- Управление на паметта:
  - Заемане и освобождаване на памет за процесите.
  - Поддържане на множество процеси в паметта.
  - Споделена памет между процесите.

# Функции на ОС. Управление на I/O устройства

- Необходимост от вторична памет.
  - Управление на дисковото пространство.
- I/O система:
  - Управление на буферирането и кеша.
  - Интерфейси за драйверите на устройствата.
  - Драйвери за специфични устройства.

# Функции на ОС. Файлова система

- **Файл** – логическа последователност от байтове (програми или данни).
- Управление на файловата система:
  - Създаване и изтриване на файлове.
  - Създаване и изтриване на директории.
  - Поддържане на примитиви за манипулиране с файлове.
  - Архивиране на файлове.

# Функции на ОС. Защита и сигурност

- Механизмът на защита контролира достъпа на програмите и потребителите до потребителските и системни ресурси.
- Изисквания:
  - Разграничаване на оторизиран и неоторизиран достъп.
  - Контрол на достъпа до ресурси.
  - Предоставяне на механизми за изпълнение на контрола на достъпа.
- Механизмът на сигурност осигурява защита срещу вътрешни и външни атаки.

# Функции на ОС. Работа в мрежа

- Разпределени системи – съвкупност от процесори, които не споделят обща памет.
- Процесорите са свързани чрез комуникационна мрежа.
- Изисквания:
  - Свързване на процесорите
  - Осигуряване обмена на информация между потребители. и системи.
  - Осигуряване нарастване скоростта на изчисленията.
  - Повишаване надеждността и наличността на информацията.



# Първоначално зареждане на компютър (bootstrap)

- Процеса на зареждане се контролира от BIOS.
- Изпълнява се код на Power On Self Test (POST).
- Стартира се зареждащата програма (bootstrap loader).
- Тя открива ОС, зарежда ядрото и го стартира.
- Ядрото зарежда съответните драйвери.
- Операционната система започва да функционира.

# Прекъсвания

- Два типа прекъсвания:
  - **Хардуерни** -- устройство генерира сигнал към CPU, обикновено чрез системната шина.
  - **Софтуерни** -- програма може да генерира прекъсване чрез изпълнение на специална операция, наречена **system call**.
- Софтуерно генерираните прекъсвания (**trap**, **exception**) се предизвикват или от грешки по време на изпълнение, или от потребителска заявка.
- **ОС е управлявана от прекъсвания (interrupt driven).**

# Прекъсвания

- Прекъсванията предават контрола към специална програма за тяхната обработка (**interrupt routine**).
- Определяне вида на прекъсването:
  - **Polling** – извиква се една и съща програма за обработка за всички прекъсвания. Тя от своя страна запитва всички устройства за да определи причината за прекъсването.
  - **Interrupt Vector Table** – използват се различни програми за обработка на различни прекъсвания.

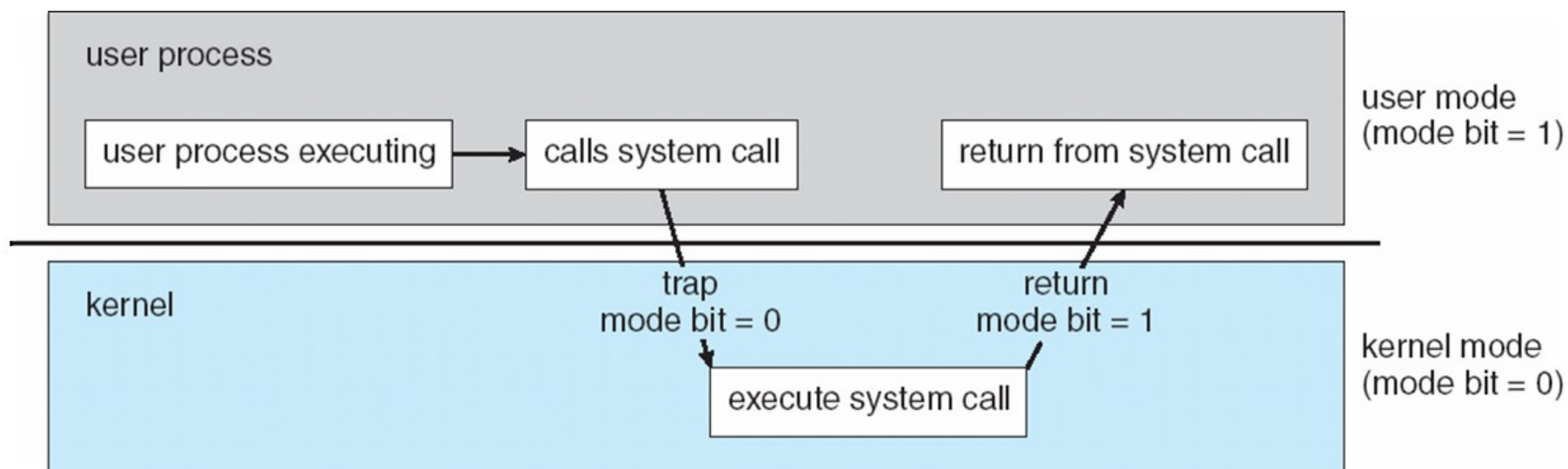
Interrupt Number	Address
0	0003h
1	000Bh
2	0013h
3	001Bh
4	0023h
5	002Bh
6	0033h
7	003Bh
8	0043h
9	004Bh
10	0053h
11	005Bh
12	0063h
13	006Bh
14	0073h
15	007Bh

Interrupt Number	Address
16	0083h
17	008Bh
18	0093h
19	009Bh
20	00A3h
21	00ABh
22	00B3h
23	00BBh
24	00C3h
25	00CBh
26	00D3h
27	00DBh
28	00E3h
29	00EBh
30	00F3h
31	00FBh

# Режими на изпълнение

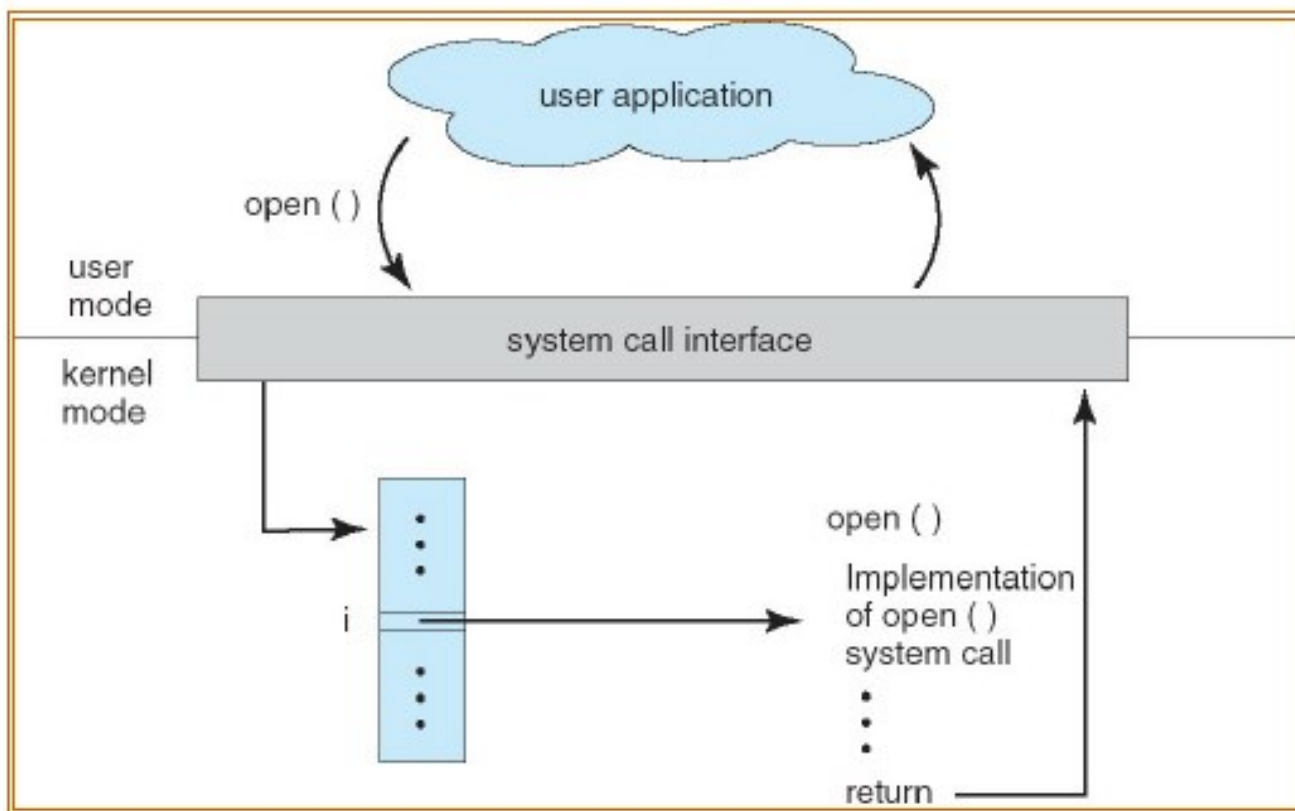
- Механизъм, позволяващ ОС да защити себе си и други системни компоненти.
- Два режима на изпълнение:
  - **User mode** и **kernel mode**
  - **Mode bit (0 or 1)** предоставен от хардуера
    - Разграничаване кога се изпълнява потребителски или код на ядрото.
    - Някои инструкции (**privileged**) могат да се изпълняват единствено в режим на ядро.
    - Системно извикване променя режима в kernel, връщане от него възстановява user режим.
- Все повече CPUs поддържат multi-mode операции
  - Например, **virtual machine manager (VMM)** режим за **guest VMs**.

# Преминаване от потребителски в системен режим



# Системни извиквания

- Интерфейс между изпълняващата се програма и ОС.
  - Инструкции на асемблер.
  - Директно извикване (език C).



# Системни извиквания - пример

```
#include <unistd.h>
```

```
ssize_t read(int fd, void *buf, size_t count)
```



return  
value

function  
name

parameters

# Предаване на параметри към системни извиквания

- Чрез регистрите на процесора (ограничение по брой).
- Чрез блок от паметта (таблица).
- Чрез стека на програмата.



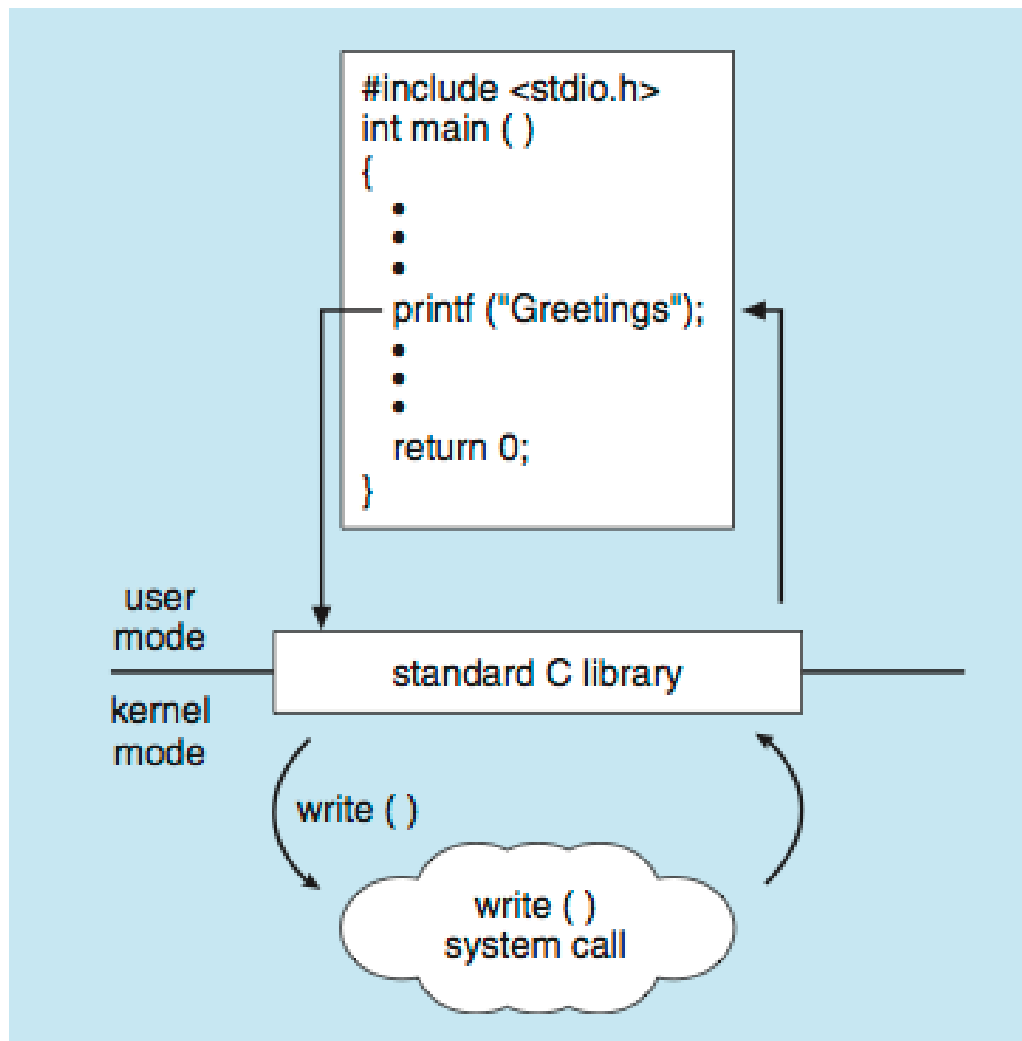
# Видове системни извиквания

- Управление на процеси;
- Манипулиране с файлове;
- Манипулиране с устройства;
- Поддържане на информация;
- Комуникация;
- Защита.

# Примери на системни извиквания

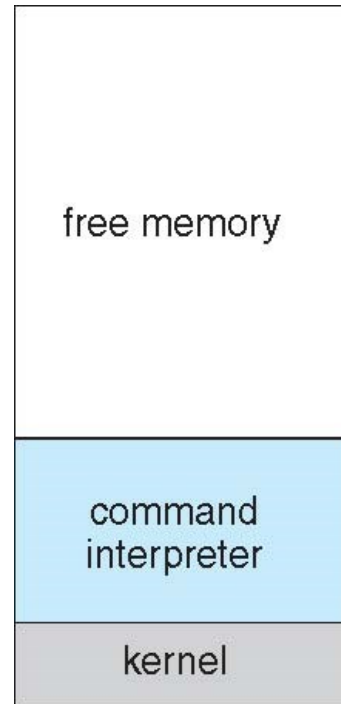
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

# Пример – стандартна C библиотека

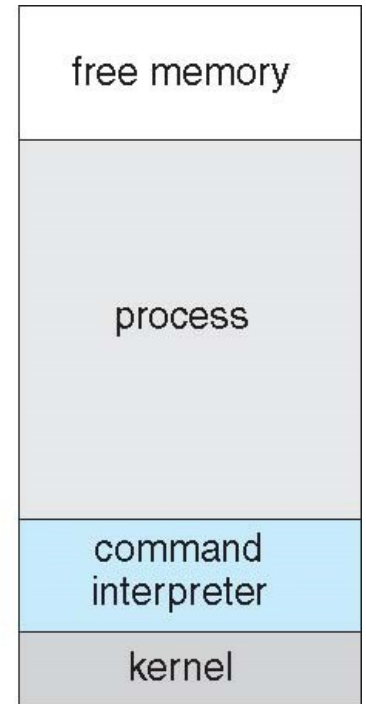


# MS-DOS

- Single-tasking.
- Интерпретаторът се извиква при boot на системата.
- Не се създава процес при стартиране на програма.
- Единично пространство на паметта.
- При зареждане на програма се препокрива всичко без ядрото.
- При завършване на програма се зарежда интерпретатора.



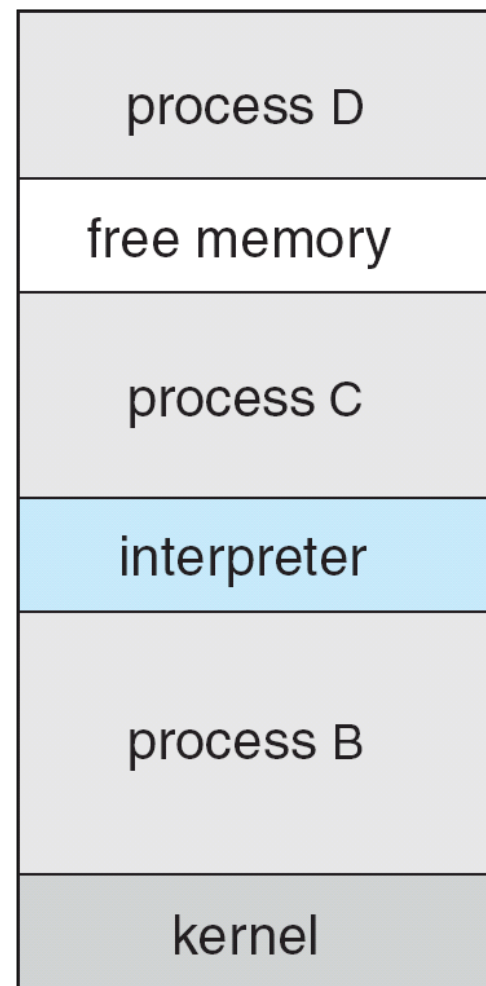
(a)



(b)

# BSD Linux

- Multi-tasking.
- При логване на потребител се извиква интерпретатор по избор.
- За създаване на процес интерпретаторът извика *fork()*.
- Процесът завършва с код:
  - Код = 0 – ОК
  - Код > 0 – Грешка.

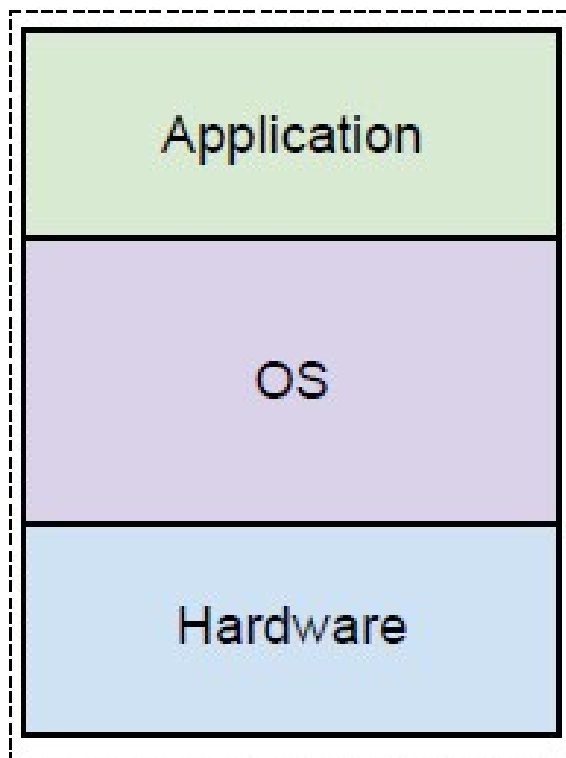


# Виртуализация

- Позволява ОС да стартира приложения в други ОС.
- **Emulation** – когато CPU на източника се различава от този на целевата машина (PowerPC към Intel x86)
  - Бавен метод за изпълнение.
  - Интерпретатори.
- **Virtualization** – ОС е естествено компилирана за CPU, изпълняващ **guest** ОС, също естествено компилирана за него
  - **Hypervisor** - предоставя виртуализационни услуги.

# Виртуализация

## *Physical Machine*

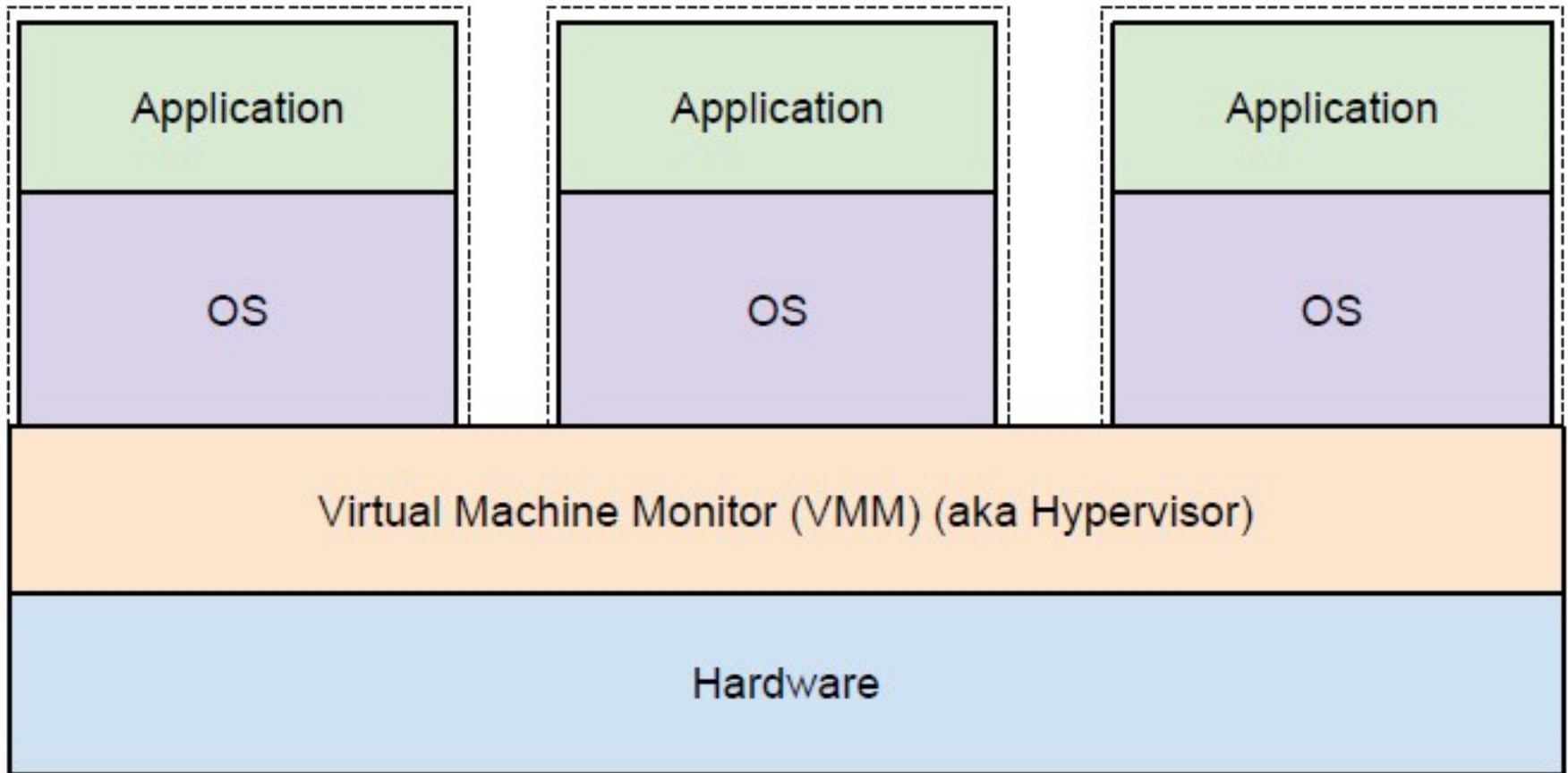


# Виртуализация

***Virtual Machine 1***

***Virtual Machine 2***

***Virtual Machine 3***





# Въпроси?