Operating Systems

Introduction

MSc CO502
Autumn Term Weeks 7-11

Morris Sloman & Anandha Gopalan

m.sloman@imperial.ac.uk
Room 575

Course Objectives

What is an operating system, and how it supports the implementation of software on a computer.

Understand the features and mechanisms that underlie operating systems, including:

- process and thread management and synchronization
- memory management
- security
- input-output
- file systems

Linux characteristics as a case study

Outline

Morris Sloman (13 lectures/tutorials)

- Overview: function and structure
- Processes and Threads: concepts and scheduling
- Process synchronization
- Deadlocks

Anandha Gopalan(13 lectures/tutorials)

- Memory Management: allocation and virtual memory
- Input/Output: device drivers, disk management & scheduling
- File Systems: files and directory structures

Course Structure

Six lectures/tutorials per week (Weeks 7 – 11)

Times: Mondays 2-4pm, Wednesdays 11-1pm,
Fridays 11-1pm

Course slides are on Cate

Acknowledgements:

Slides based on material by Peter Pietzuch, Cristian Cadar and Julie McCann

Recommended Books

1. **Modern Operating Systems: Global Edition**, A. Tanenbaum, H. Bos, 4th edition, Pearson, 2015
2. **Operating Systems – Internals and Design Principles**, W. Stallings, 8th Edition, Pearson, 2014
3. **Operating System Concepts**, A. Silberschatz, P. Galvin, G. Gagne, 8th Edition, John Wiley & Sons, 2014

Note: Earlier editions of these are OK and may be more readily available

☛ Important: Do not just rely on these slides!

OS Overview

Computer Architecture Overview

Processor

- Controls computer hardware
  Executes instructions and programs

Memory

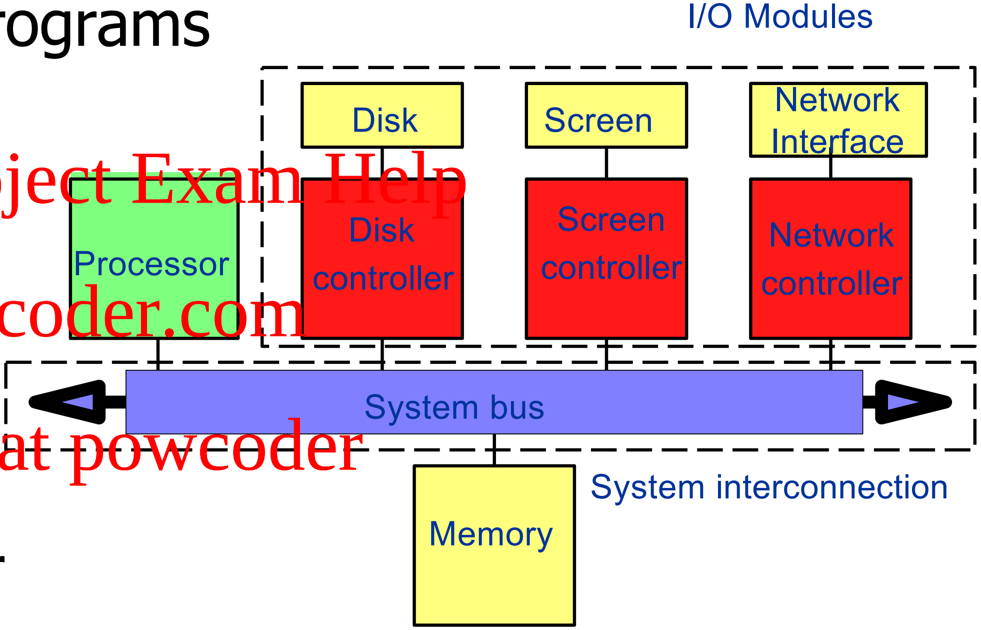
- Stores data and programs

I/O modules

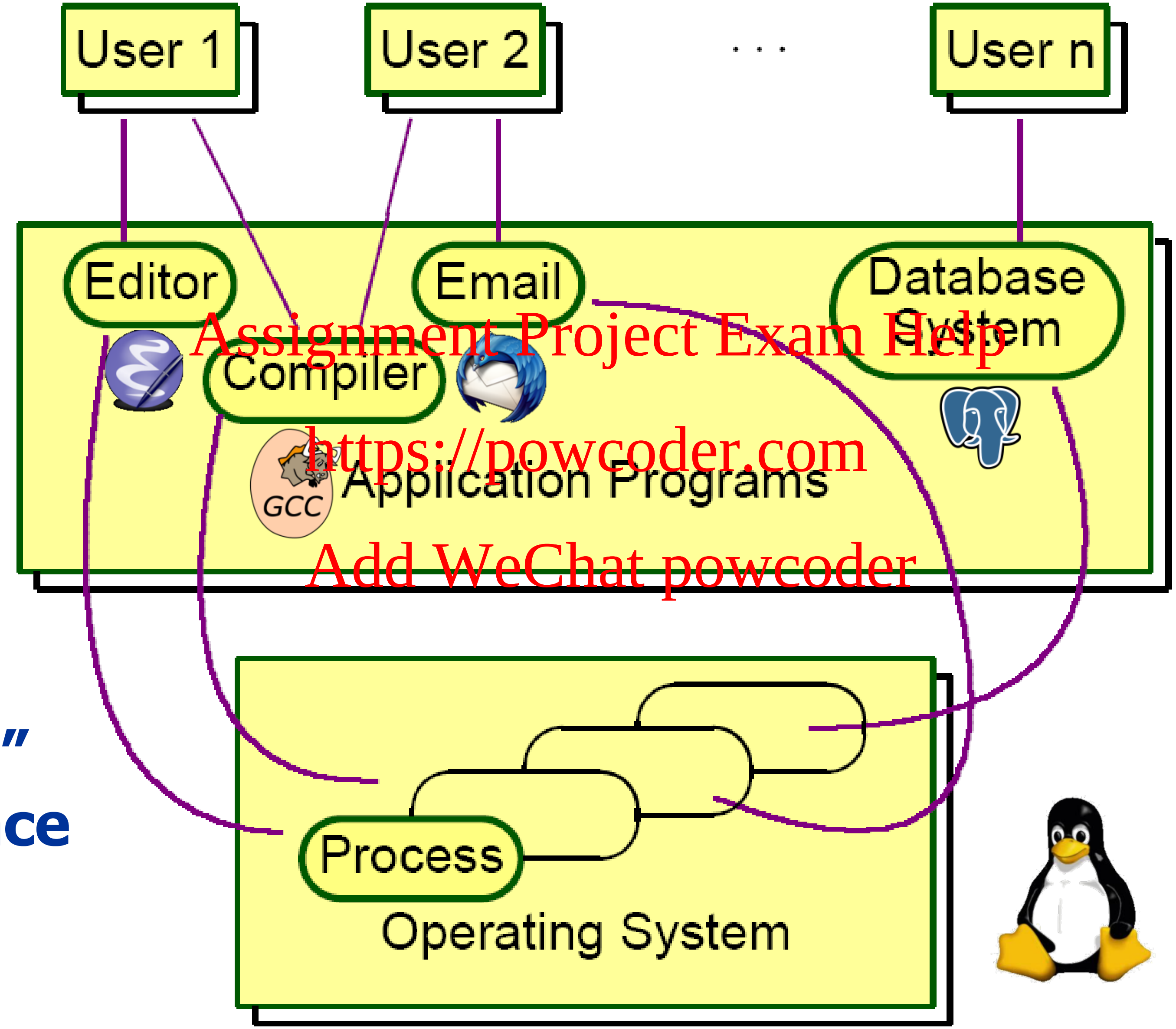
- Read and write from
  I/O devices
- Intelligence in I/O controller

System interconnection

- Connects different hardware components via bus
- Provides communication between hardware components

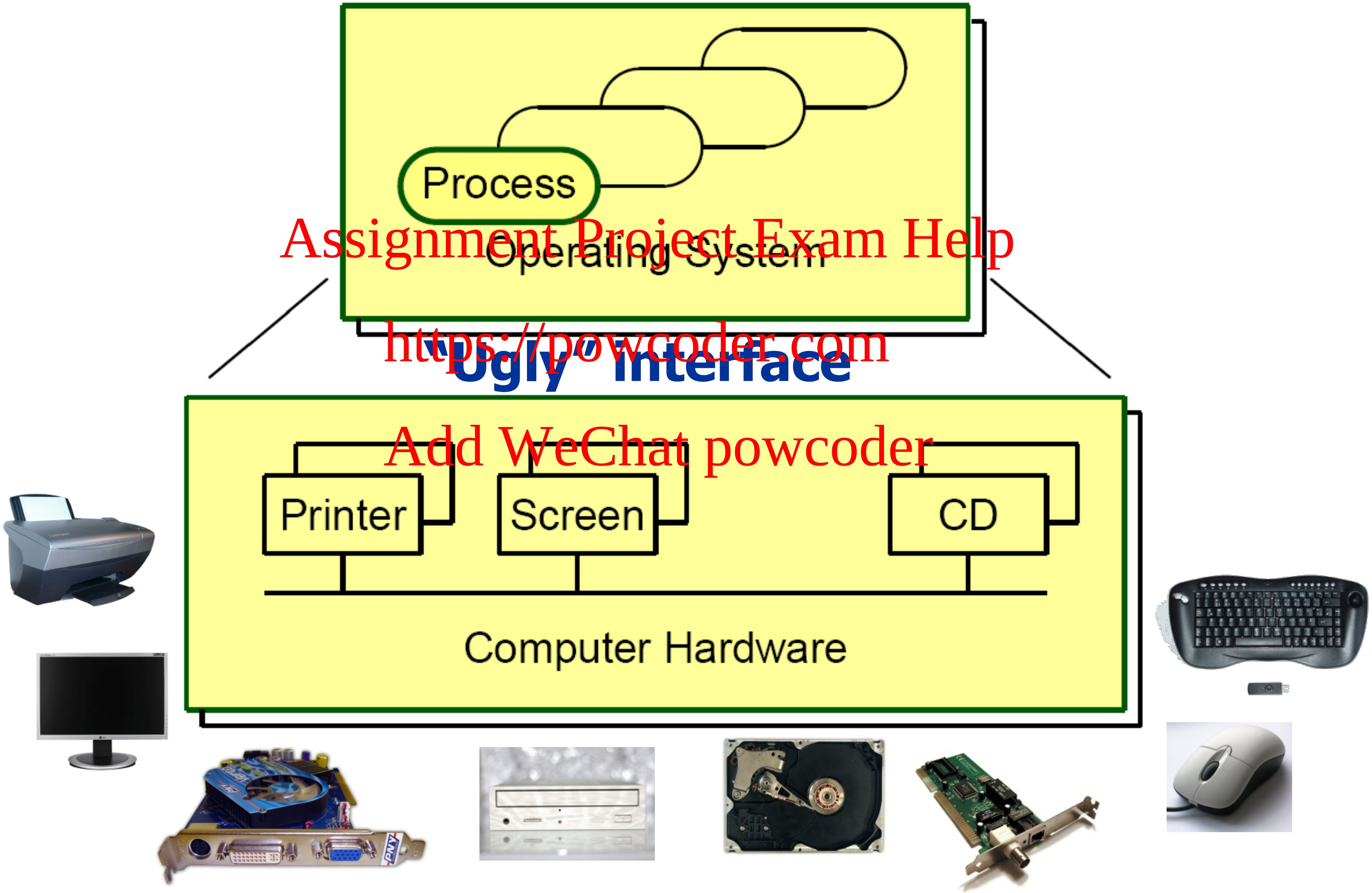


Operating Systems – Top Level View



"Clean" interface

Operating Systems – Bottom Level View



1. Resource Management

Making efficient use of (limited) available resources

- Optimise utilisation of processor, memory, disks, network etc....

Sharing resources among multiple users

- Schedule access, fair allocation
- Prevent interference

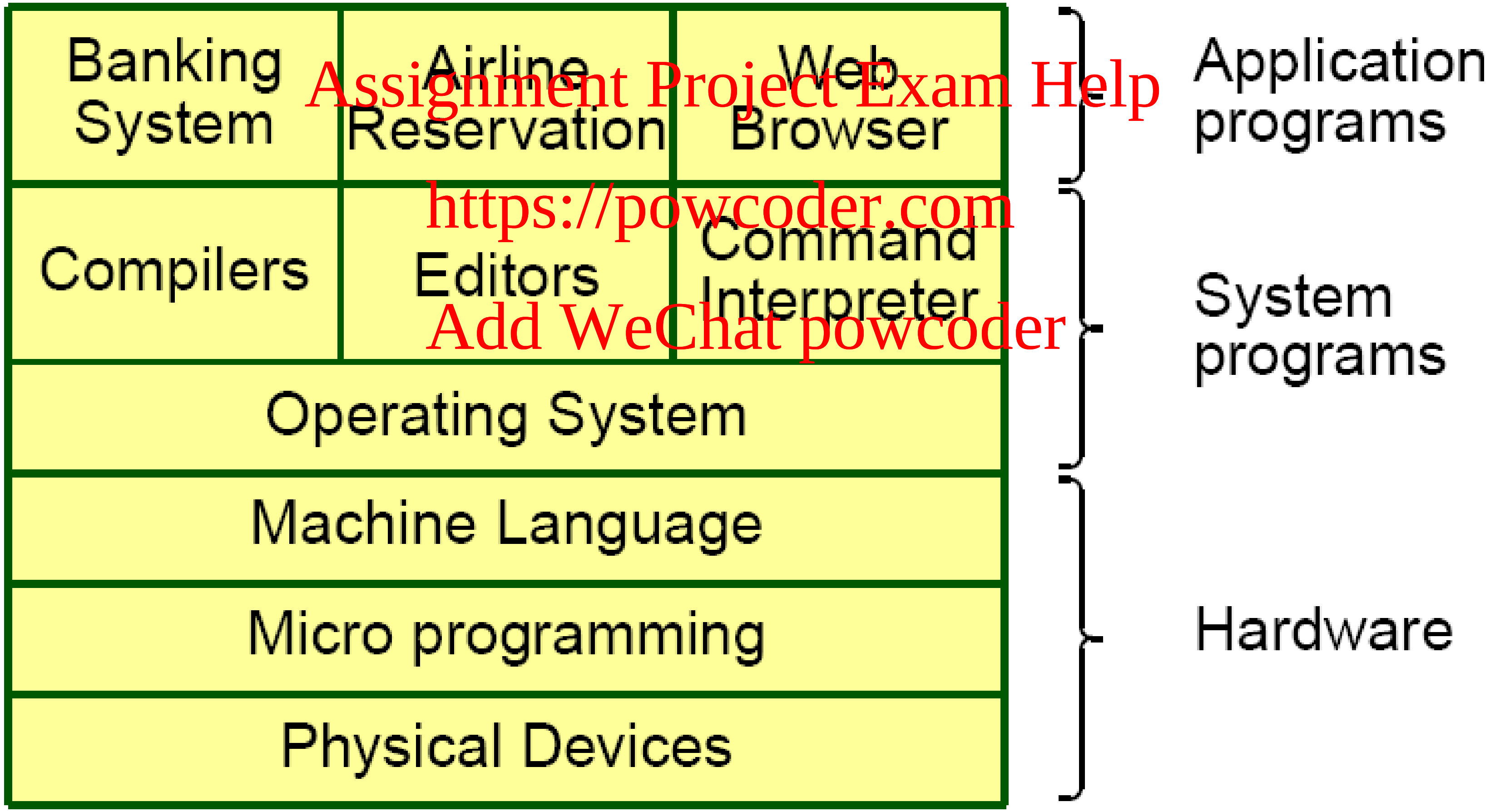
Resources

- Processors
  - Divide number and/or time
- Memory
  - RAM, cache, disks, ...
- Input/Output devices
  - Screens, printers, network interface, ...
- Internal devices
  - Clocks, timers, accelerometers ...
- Long-term storage (files)
  - Disks, storage cards, DVD, tapes, ...
- Software
  - Browsers, editors, e-mail clients, databases, …….

2. Providing Clean Interfaces

OS converts raw hardware into usable computer system

- Hides complexity of lower levels from higher levels



Banking System

Airline Reservation

Web Browser

Compilers

Editors

Command Interpreter

Operating System

Machine Language

Micro programming

Physical Devices

Application programs

System programs

Hardware

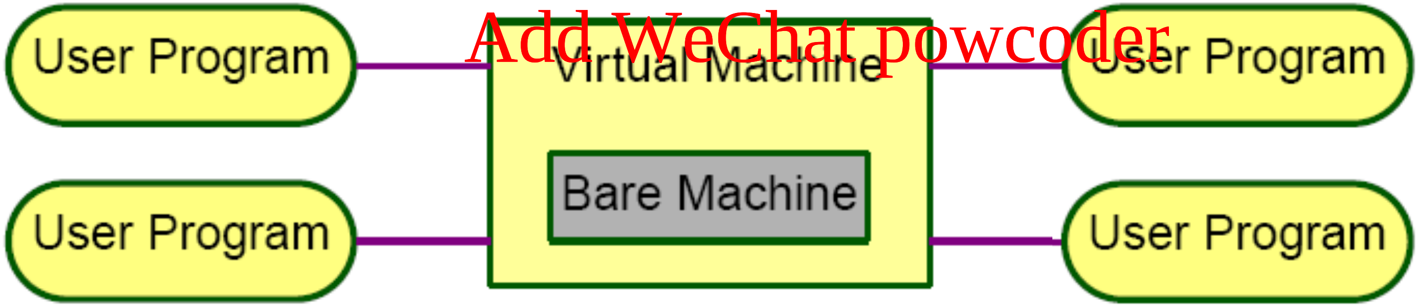
Virtual Machine Abstraction

Details of hardware kept hidden from programs

Only OS can allow access to hardware resources

User request should be abstract

- e.g. no need to know how files stored on disk



Virtual Machine Facilities

Simplified I/O: Device independence; open a file on disk, CD, screen is one operation.

Virtual Memory: Larger than real or partitioned.

Filing System: Long term storage, on disk or tape, accessed by symbolic names.

Program Interaction and communication: Pipes, semaphores, locks, monitors.

Network communication: Message passing

Protection: Prevent programs accessing resources not allocated to them.

Program Control: User interaction with programs, command language, shells.

Accounting & Management Information: Usage of processors, memory, file storage etc.

OS Characteristics: Sharing

Sharing of data, programs and hardware

- Time multiplexing and space multiplexing

Resource allocation

- Efficient and fair use of memory, CPU time, disk space, ...
- Simultaneous access to resources
  - Processor, Disks, RAM, code, network, ...

- Mutual exclusion
  - Protect multiple programs from uncontrolled access to shared resources.
  - Prevent multiple writes to same data structure or file.
- Protection against corruption
  - Accidental or malicious

OS Characteristics: Concurrency I

Several simultaneous parallel activities

- Overlapped I/O & computation
- Multiple users and programs run in parallel

Switch activities at arbitrary times

- Guarantee fairness and prompt response
- Differential responsiveness e.g. interactive vs. batch

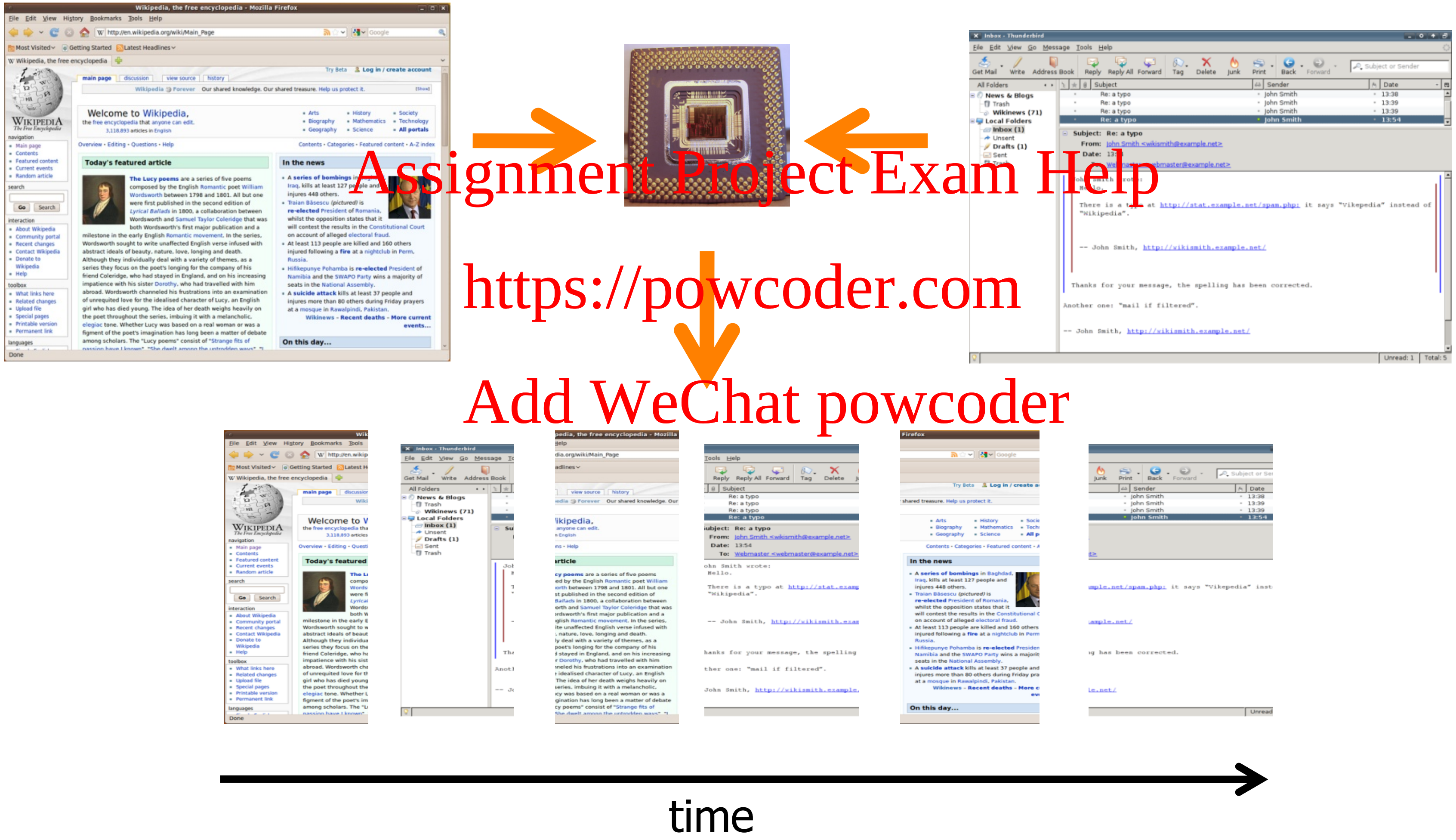
Safe concurrency

- Synchronisation of actions
  - Avoids long waiting cycles; gives accurate error handling
- Protection from interference
  - Each process has its own space

OS Characteristics: Concurrency II

Time-slicing

- Switch application running on physical CPU every 50ms



What other switching mechanisms are feasible?

OS Characteristics: Non-determinism

Non-determinism

- Results from events occurring in unpredictable order
  - e.g. timer interrupts, user input, program error, network packet loss, disk errors, . . .
- Makes programming OS hard!

OS Characteristics: Storing Data

Long term storage: File systems for disks, DVDs, memory cards …..

- Easy access to files through user-defined names
  - Directory structure, links, shared disks
- Access controls
  - Read, write, delete, execute or copy permissions
- Protection against failure (backups)
  - Daily/weekly/monthly, partial/complete
- Storage management for easy expansion
  - Add disks without need for re-compilation of OS

Mentimeter: www.menti.com OS Function Q 40 52 35

Non-determinism

Operating System Zoo

Desktop/Laptop (e.g. Windows, Mac OS X, Linux)

- Typically 2-8 cores + high resolution screen

Server OS (e.g. Linux, Windows Server 20XX, Solaris, FreeBSD,)

- Share hardware/software resources e.g. internet servers
- Typically many multicore processors + large disks

Smartphones (e.g. iOS, Android)

- Simpler CPUs, starting to be sophisticated

Real-time OS

- Guaranteed time constraints

Embedded OS (e.g. QNX, VXWorks)

- Transport, communications, banking, homes etc.
- Only trusted software

Smartcard OS

- Usually single function
- Many have JVM
- OS is primitive

Sensor Network OS (e.g. TinyOS)

- Resource/energy conscious

Resource Management Question

What are the most important resources that must be managed by the OS for the following computers?

Supercomputer

Workstations connected to servers via a network

Smartphone

OS Structure

Monolithic OS kernels (e.g. Linux, BSD, Solaris, ...)

- Single black box

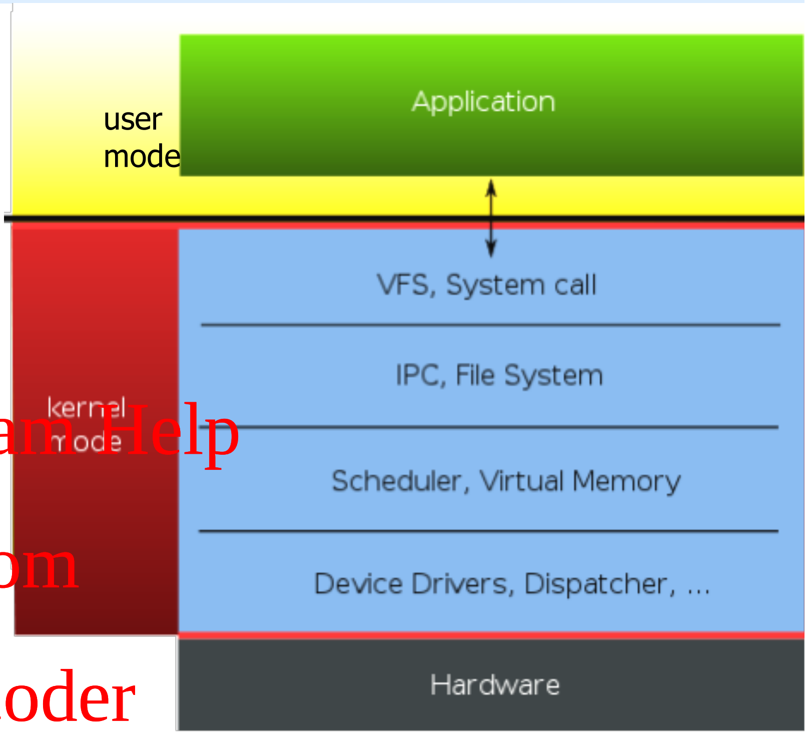
Microkernels (e.g. Symbian, L4, Mach, ...)

- Little as possible in kernel (fewer bugs)

Hybrid kernels (e.g. Windows NT, Mac OS X, ...)

- Take a guess... ☺

Monolithic Kernels



Kernel is single executable with own address space

- Structure implied through pushing parameters to stack and trap (systems calls)
- Most popular kernel style

Advantages

- Efficient calls within kernel
- Easier to write kernel components due to shared memory

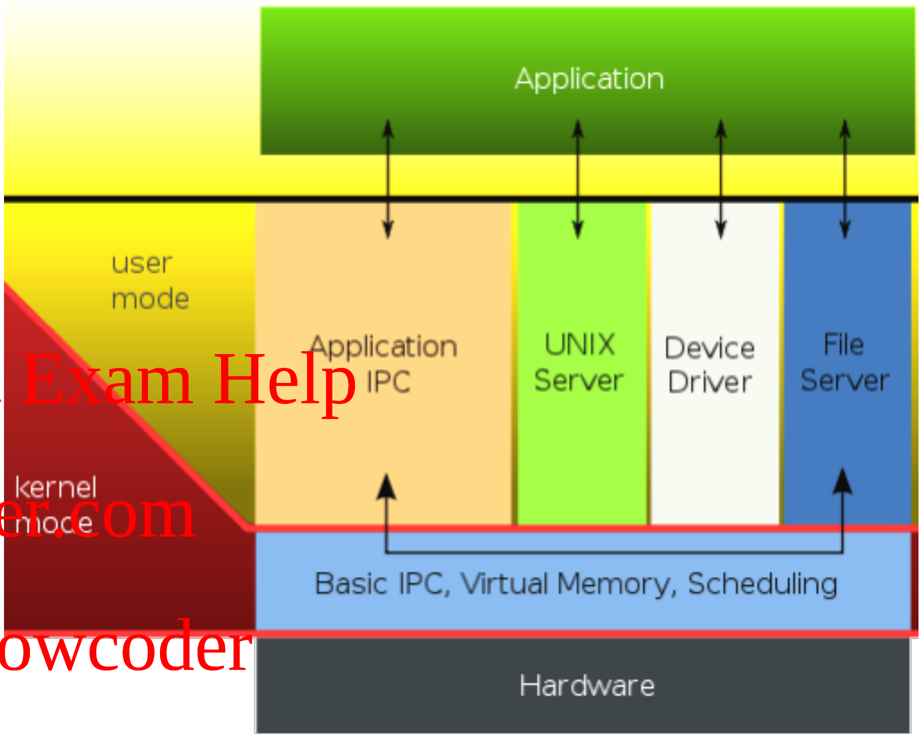
Disadvantages

- Complex design with lots of interactions
- No protection between kernel components

Microkernels

Minimal “kernel” with functionality in user-level servers

- Kernel does IPC (message-passing) between servers
- Servers for device I/O, file access, process scheduling, …



Advantages

- Kernel itself not complex ➔ less error-prone
- Servers have clean interfaces
- Servers can crash and restart without bringing kernel down

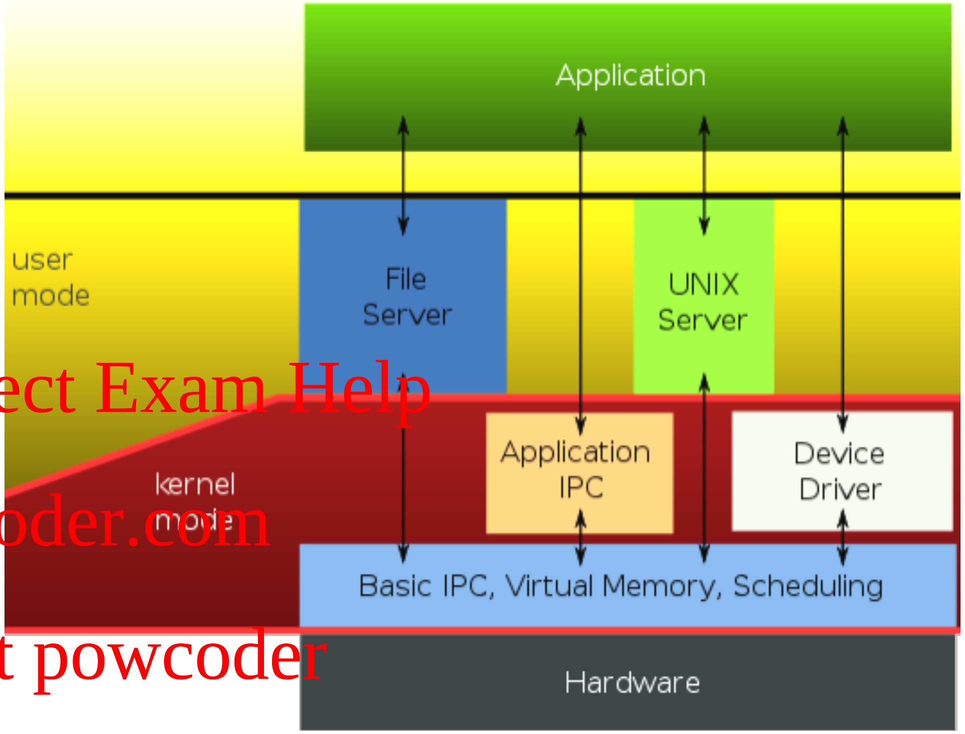
Disadvantages

- High overhead of IPC within kernel

Hybrid Kernels

Combines features of both monolithic and microkernels

- Often a design philosophy



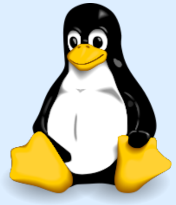
Advantages

- More structured design

Disadvantages

- Performance penalty for user-level servers

menti.com Kernel Q 40 52 35



Introduction to Linux

Linux History and Motivation

Variant of Unix like FreeBSD, System V, Solaris etc.

- Ken Thomson left Multics (Bell Labs)
  – *Uniplexed* information and computing service
- Dennis Ritchie got interested

Late 80’s: 4.3 BSD and System V r3 dominant

- Systems call libraries reconciliation POSIX

1987 Tanenbaum released MINIX microkernel

- Tractable by single person (student)

Linus Torvalds, frustrated, built fully-featured yet monolithic version ➔ Linux

- Major goal was interactivity, multiple processes and users
- Code contributed by world-wide community

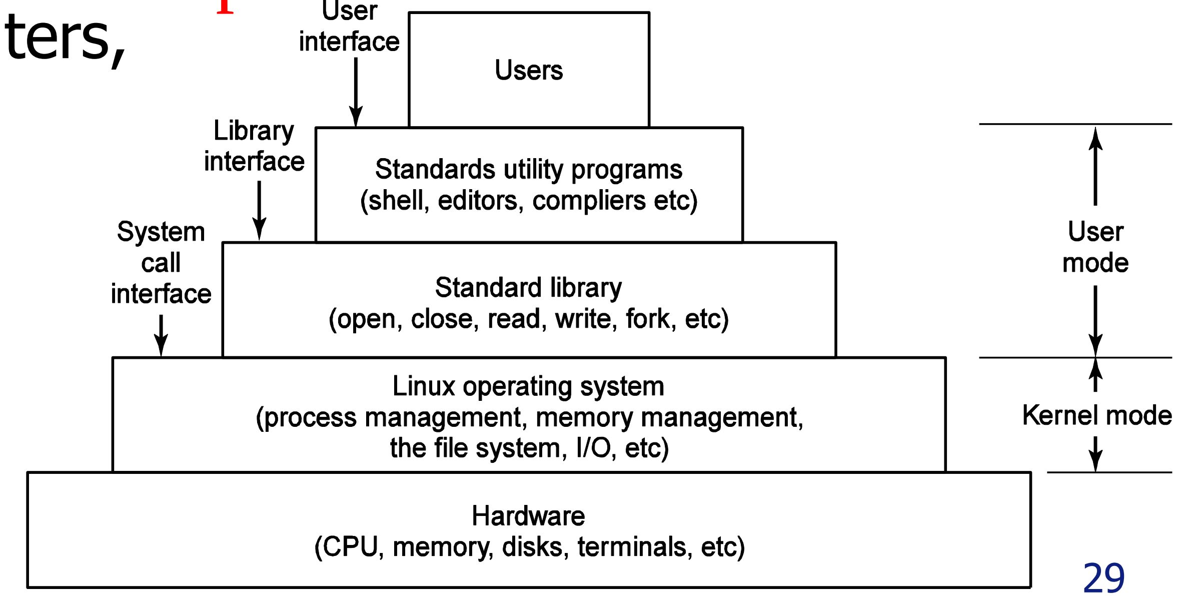
Structure and Interfaces

System calls

- Implemented by putting arguments in registers (or stack)
- Issue trap to switch from user to kernel

Rich set of programs (through GNU project)

- e.g. shells (bash, ksh, ...), compilers, editors, ...
- Desktop environments: GNOME, KDE, ...
- Utility programs: file, filters, editors, compilers, text processing, sys admin, etc



User interface

Users

Library interface

Standards utility programs
(shell, editors, compliers etc)

System call interface

Standard library
(open, close, read, write, fork, etc)

Linux operating system
(process management, memory management, the file system, I/O, etc)

Hardware
(CPU, memory, disks, terminals, etc)

User mode

Kernel mode

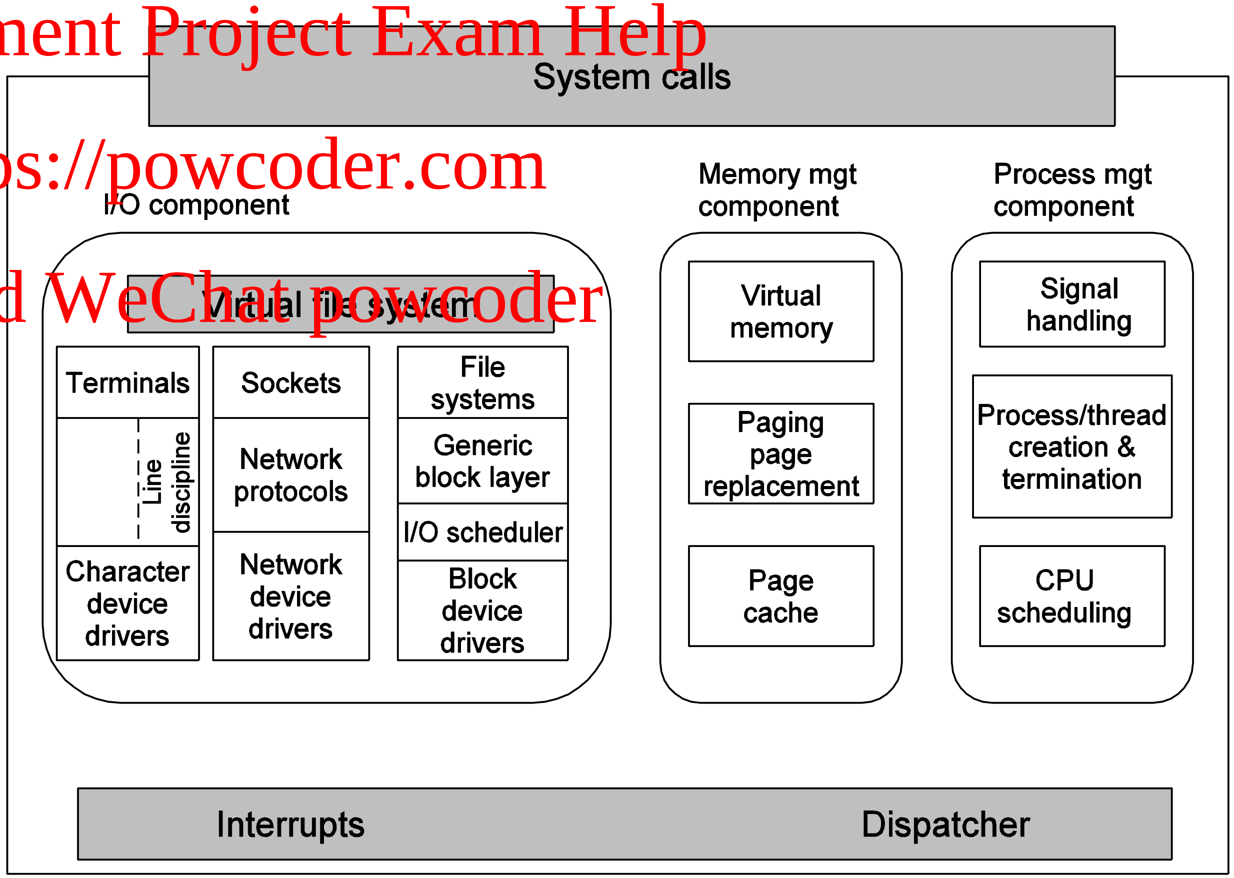
Kernel Structure

Interrupt handlers primary means to interact with devices

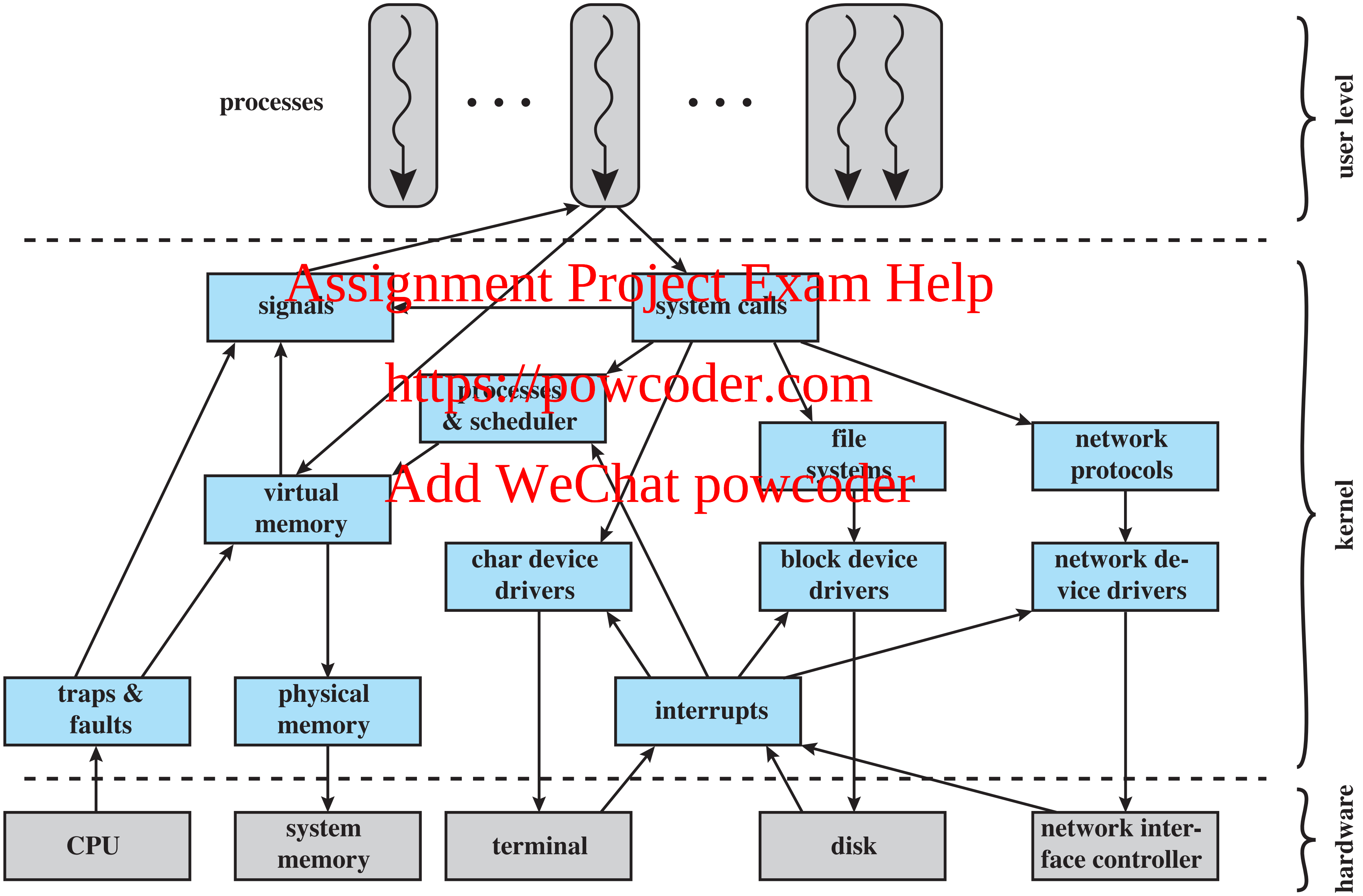
- Kicks off dispatching
  - Stop process, save state and start driver and return
- Dispatcher written in assembler

IO scheduler orders disk operation

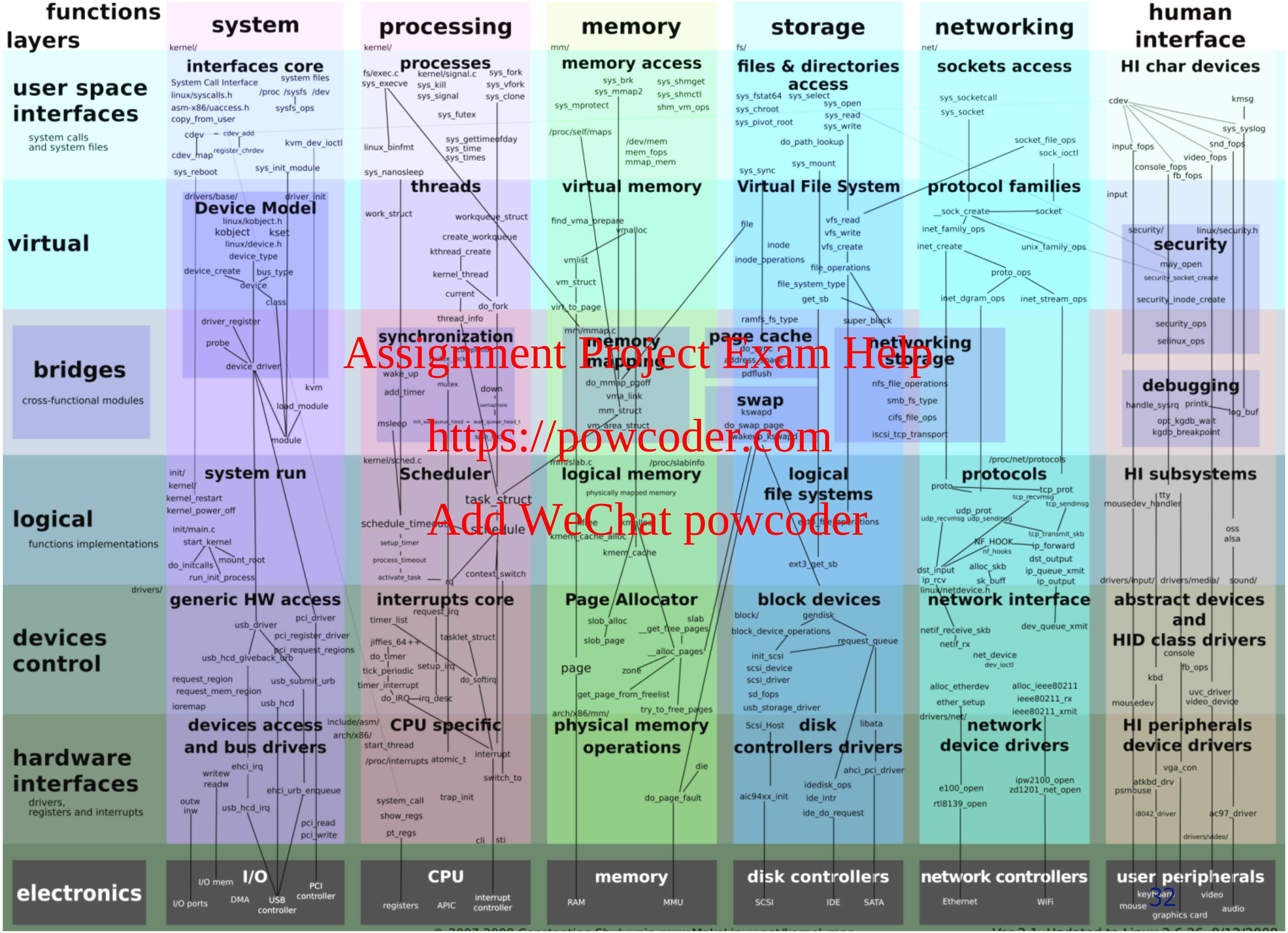
Monolithic:
Static in-kernel components
and dynamically
loadable modules
with shared internal data structures



Linux Kernel Components



Linux kernel map



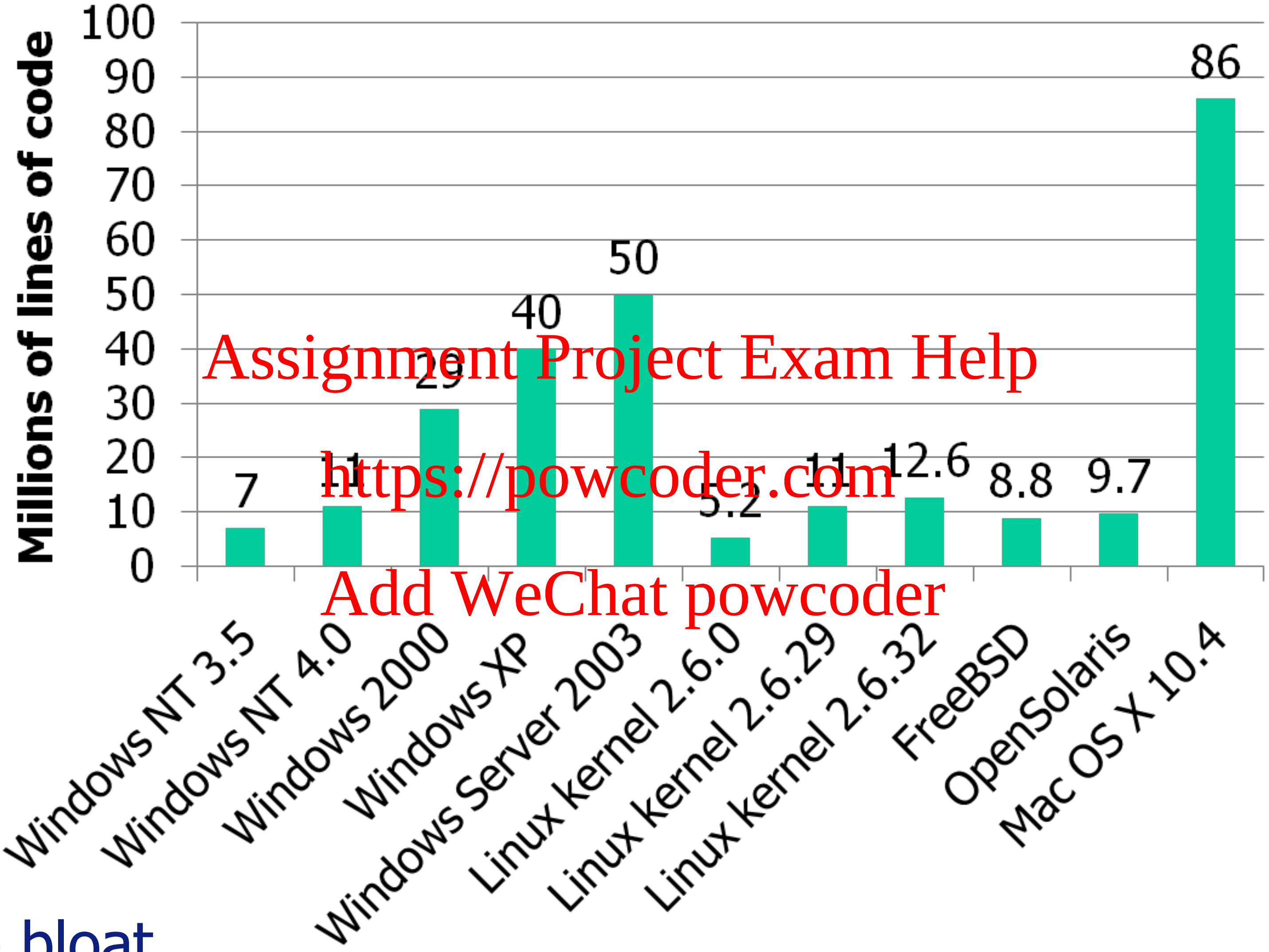
© 2007-2008 Constantine Shulyupin www.MakeLinux.net/kernel_map Ver 2.1, Updated to Linux 2.6.26, 8/12/2008

Kernel Questions

1. Why is the separation into a user mode and a kernel mode considered good operating system design?

2. Give an example in which the execution of a user processes switches from user mode to kernel mode, and then back to user mode again.

Evolution of OS Code Sizes



source: Wikipedia 2010

Code bloat

- Is lines of code useful comparison for complexity?
  - e.g. Linux scheduler (50K LoC); Vista scheduler (75K LoC)

Summary

OS Functions

- Simplify programming: device abstraction; virtual machine; memory management, file systems.
- Support concurrency, resource sharing & synchronisation

Kernel Structure

- Monolithic, Micro & Hybrid.

Operating System complexity

Portable Operating System Questions

1. Explain why it is infeasible to build an operating system that is portable from one system architecture to another without any modification.

2. Describe two general parts that you can find in an operating system that has been designed to be highly portable.