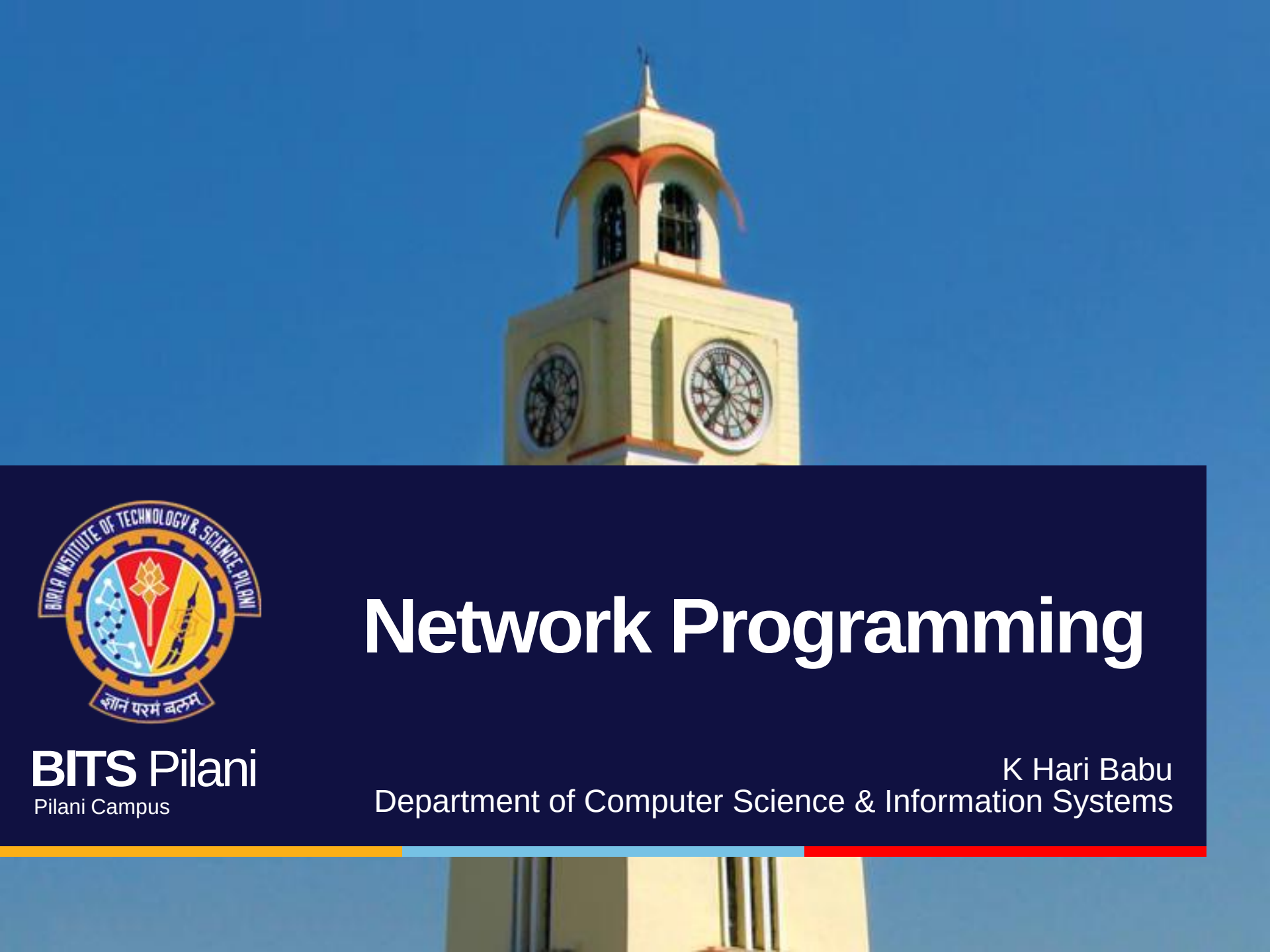


Network Programming

BITS Pilani
Pilani Campus

K Hari Babu
Department of Computer Science & Information Systems

innovate achieve lead

BITS Pilani

Pilani Campus

Course Overview

Instructor

- Dr K Hari Babu, Dept. of CS & IS
- khari@pilani.bits-pilani.ac.in
- 01596-51-5813

Objectives

innovate achieve lead

- To study the Unix/Linux API related to
  - Process management
  - File I/O
  - Memory Management
  - Network
- To develop network-aware applications.
  - Unicast/Multicast/Broadcast application development
  - Client design & implementation: Browsers etc.
  - Server design & Implementation: web servers etc
  - Server scalability

Course Topics

- System Programming
  - Process, thread and signals management
  - File System API
  - User, User Groups, Process credentials
  - Daemons
  - Inter Process Communication
- Socket Programming
  - Sockets API
  - Writing client-server applications
  - Broadcasting and multicasting
  - Raw sockets and datalink access

Course Topics

- Client & Server Design Alternatives
  - I/O Models
    - Blocking, Non-blocking, I/O Multiplexing, Signal driven, Asynchronous I/O
  - Client design Models
  - Server design Models
- Distributed Programming
  - SUN RPC
    - XDR
  - DCE RPC
  - XML RPC
  - CORBA
  - DCOM

Text Books

innovate achieve lead

- Prescribed Text Books
  - **T1:** Stevens, R.W., “Unix Network Programming, Vol-I Networking APIS : Sockets and XTI”, Prentice- Hall of India, 3rd Edition, 2004
  - **T2:** Stevens, R.W., “Unix Network Programming: Vol-II Inter Process Communications”, Prentice-Hall of India, 2nd Edition, 1999.

- Reference Books
  - **R1:** The Linux Programming Interface: Linux and UNIX System Programming Handbook by Michael Kerrisk, No Starch Press © 2010

    (http://library.books24x7.com/toc.aspx?bookid=41558)
  - **R2:** W.R. Stevens, Advanced Programming in the UNIX Environment, Addition Wisley, 1998.

Pre-requisites

innovate achieve lead

- Course will mainly focus on Linux 2.6 kernel
- Programs will be in C language.
  - We will discuss Java API for sockets which provides a simpler and higher level abstraction
  - We will also discuss Windows API for sockets. But major focus will be on Linux.
- You are expected to be familiar with C.
- Some familiarity with Networks and OS concepts will help.
  - Not a absolute requirement.

Course Delivery

innovate achieve lead

- Weekly 1 Lecture
  - Expecting participation & interactive sessions
- Labs
  - Total 8 labs on various topics
  - Helps you to make your hands dirty
  - http://lab.bits-pilani.ac.in/courses/
- Discussion Group
  - http://taxila.bits-pilani.ac.in/
  - Can post your queries
  - Will be checking frequently (twice a week) and replying you.

Evaluation Components

innovate achieve lead

- Programming Assignments (15%)
  - No groups. Individual.
  - Assignment1
    - System Programming
    - Socket Programming
  - Assignment2
    - Client & Server Design
    - Distributed Programming
  - Malpractice
    - All parties will be equally penalized. Zero marks to all parties involved.
    - No sharing of source code.
- Mid Term (35%)
- Final (50%)

History of UNIX & C

A Brief History of Time (UNIX and C)

- 1969 – First Unix Ken Thompson at AT&T Bell Labs
  - Ideas from Multics:
    - Tree structured file system
    - Program for interpreting commands (shell)
    - Files – unstructured streams of bytes
- 1970 Unix rewritten in assembly for DEC PDP-11
- C – Dennis Ritchie – a systems programming language
  - BCPL ➔ B (Thompson) ➔ C
- 1973 Kernel rewritten in C – eases porting to other machines.
- 1977 Bell Labs released Unix System III
- 1982 AT & T released Unix System V.

Berkeley Software Division (BSD)

- (1975) Thompson visiting Prof. at UC-Berkeley
- A student Bill Joy added new features
  - Vi editor
  - C shell
  - First paging virtual memory management (Unix) BSD 4.2
  - Sendmail, Pascal compiler
  - Later co-founded Sun Microsystems
- Berkely Software Distribution (BSD)
  - Released BSD 3 in 1979
  - Latest one in 1994 BSD 4.4
- Derivatives
  - Darwin, Free BSD, Net BSD, Open BSD

Unix after 1979

- BSD continued at UC-Berkeley
- Bell Labs System III ➔ Systems V
- POSIX standard (1988)
- Other Software
  - X windows
  - Free Software Foundation
  - GNU Public License
- Minix – (1988)
  - Unix like; MINI-uniX; for education; A. Tannenbaum

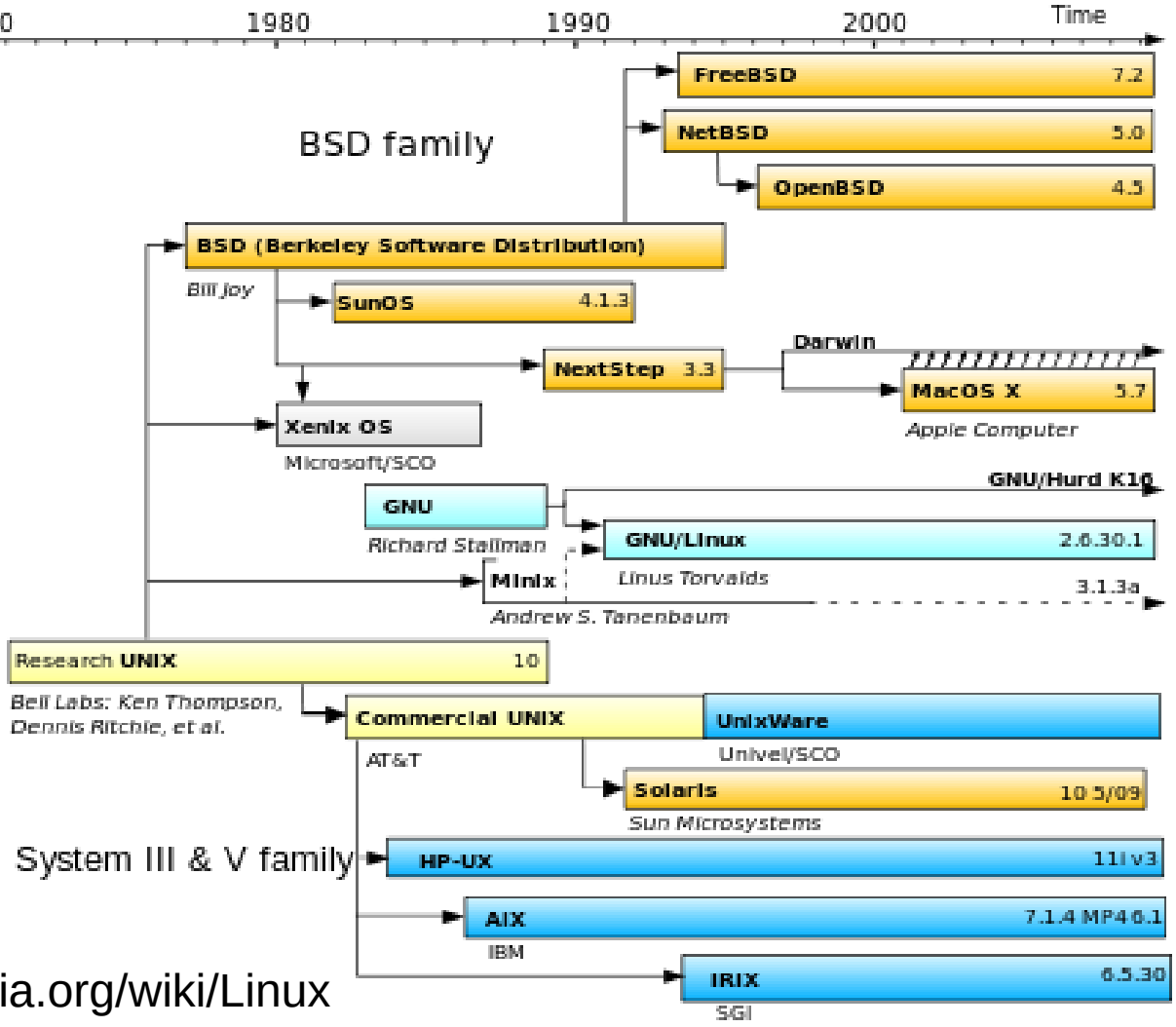
Unix Commercial Versions

- In mid 90s many companies supported high-end features and ported them to their own architecture.
  - Digital's Tru64
  - HP-UX
  - IBM AIX
  - Sequent's DYNIX
  - SGI's IRIX
  - Sun Solaris
  - SunOS

Unix Timeline

innovate achieve lead

- .



http://en.wikipedia.org/wiki/Linux

Unix Characteristics/Strengths

- Unix is Simple
  - Has only hundreds of systems calls where as other OSs have thousands.
  - Has clear design goals
- Everything is a file
  - Same system calls open(), read(), write() and close() can work with files, devices, networks sockets.
- Unix kernel and system utilities are written in C.
  - Easy portability to different architectures.
- Fast process creation time.
- Robust inter process communication (IPC).

GNU

- Richard Stallman (1983) Goal a free Unix
  - Known for Free Software movement, GNU, Emacs, gcc
  - Never really released GNU operating system

- Free Software Foundation
  - http://www.fsf.org/

http://en.wikipedia.org/wiki/GNU

Linux -

- Minix – (1988)
  - Unix like; MINI-uniX; for education; A. Tannenbaum
- (1991) Linus Torvalds
- For Intel x86 systems
- Moved to big Iron
- more than 90% of today's 500 fastest supercomputers run some variant of Linux
- Network routers
- Embedded systems
- Android

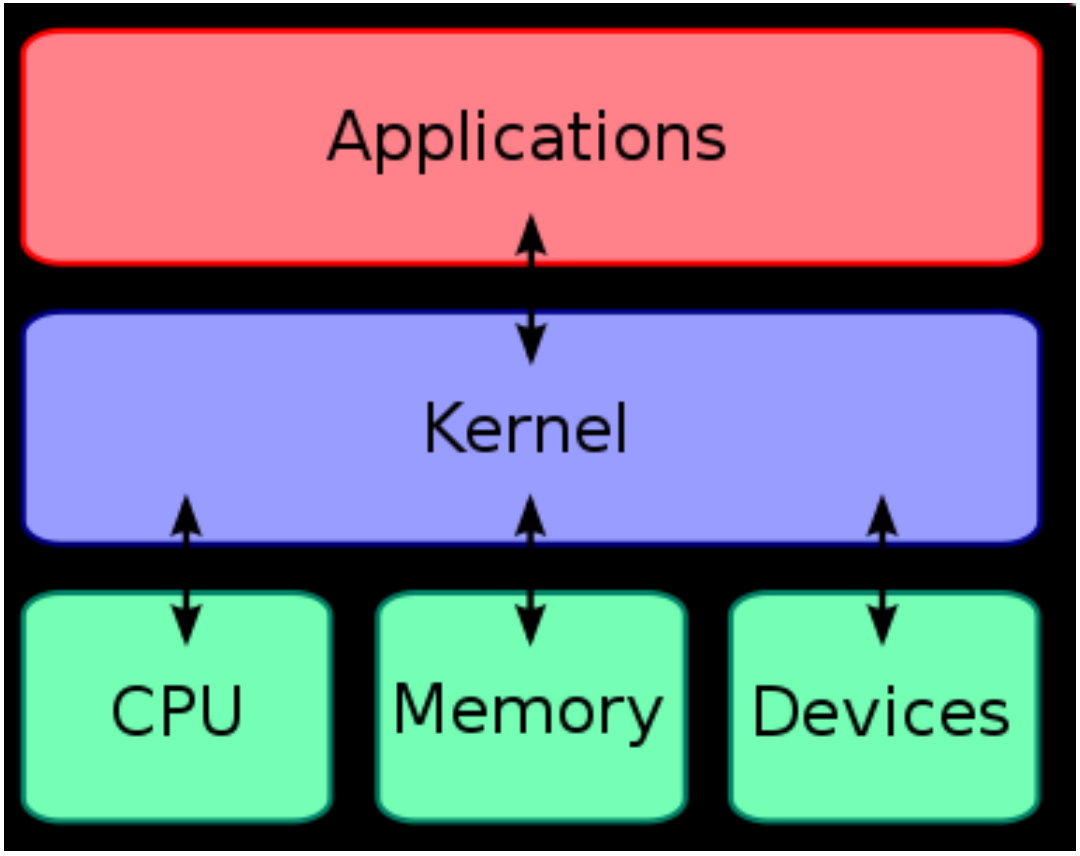
http://en.wikipedia.org/wiki/Linux

Overview of Linux OS Concepts

OS

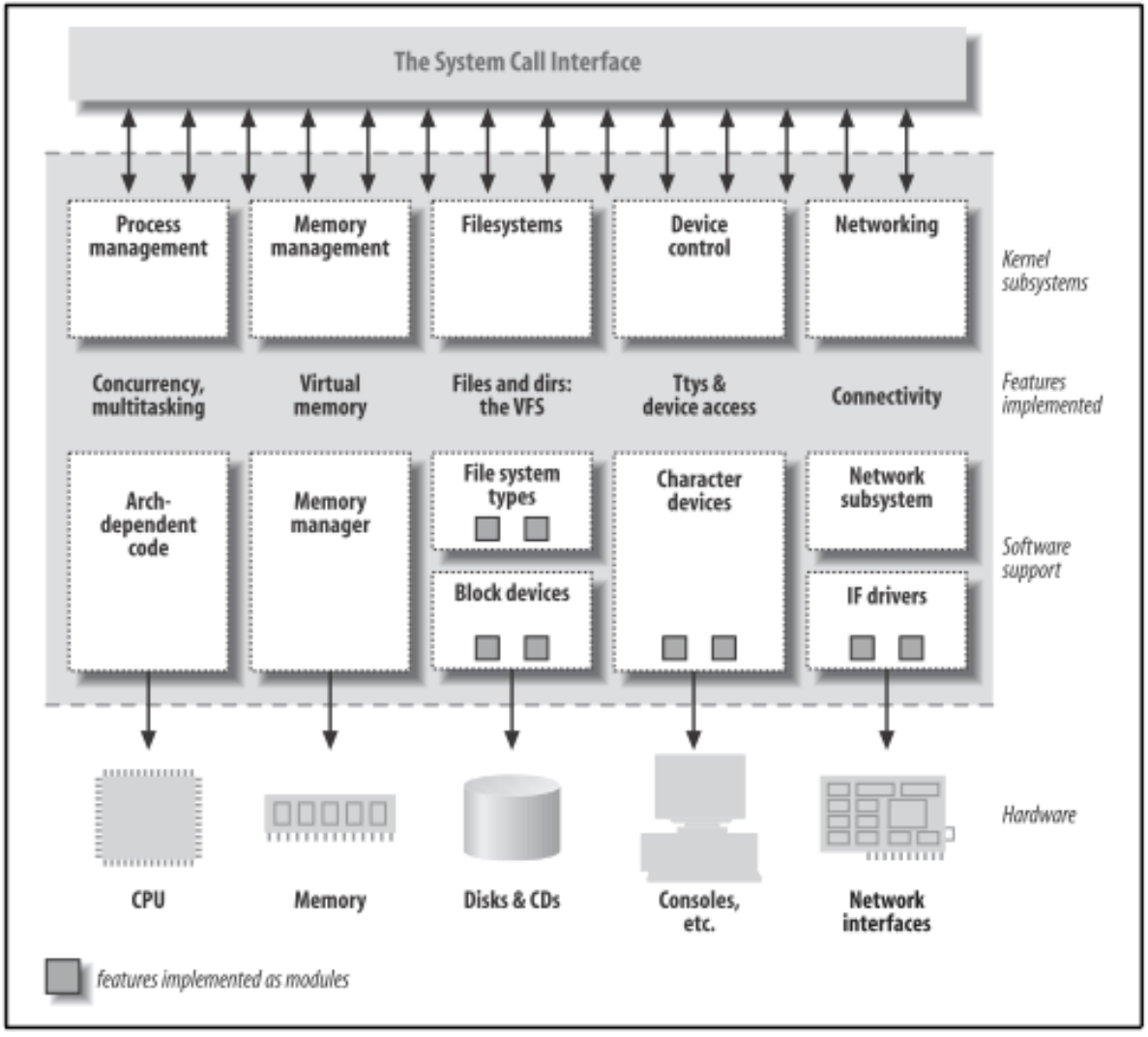
- Kernel
- C library
- Toolchain
- Basic system utilities
- Shell

About kernel



Source: Wikipedia

Kernel



Source: Linux Device Drivers

Kernel

- Central software that manages and allocates computer resources (CPU, RAM, devices etc).
- Kernel greatly simplifies writing programs and using other programs.
  - Processes
  - Virtual address space
  - Virtual CPU

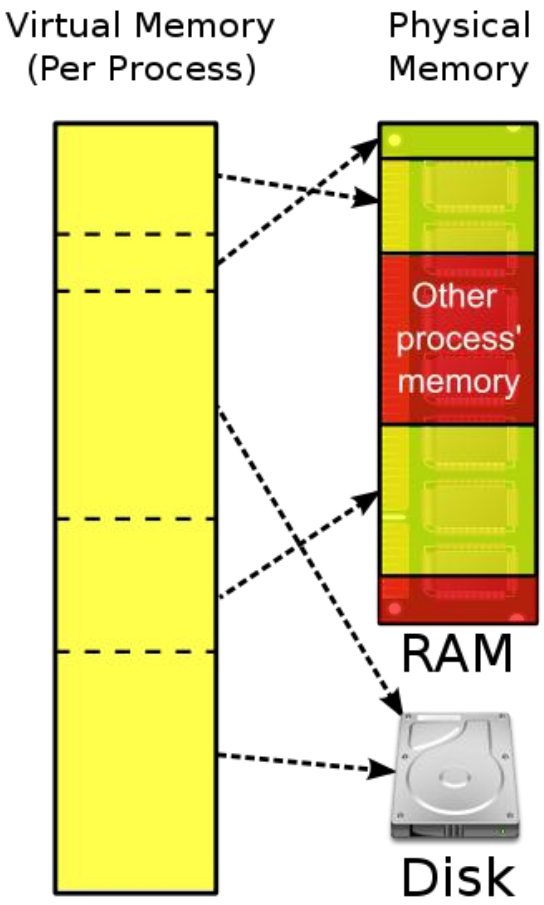
Kernel Tasks

- Process scheduling
- Memory management
- Provision of a file system
- Creation and termination of processes
- Access to devices
- Networking
- Provision of a system call application programming interface (API)

Memory Management

- Software size is enormous and RAM is a limited resource.
- Virtual Memory Management
  - Processes are isolated from one another and from the kernel.
  - Only a part of the process needs to be kept in memory. So multiple processes can be held in main memory simultaneously.
- Virtual Address Space
  - On 32 bit systems, there can be 4 GB address space.
  - Normally 2 GB is meant for Kernel and 2 GB for user process.
  - Current process has all the user address space with it.
  - Programmer flexibility.

Memory Management



Source: wikipeadia

Kernel Tasks

- Process Scheduling
  - Preemptive Multitasking
  - Multiple processes can simultaneously reside in memory
  - Preemptive means kernel decides which process and how long it gets CPU not the process itself.
- File System
  - File creation, retrieval, deletion etc.
  - More later
- Networking
  - TCP/IP stack
  - Sending/receiving messages on behalf of user processes.
  - More later

Kernel Tasks: Process Management

- Kernel can load new program into memory, provide it with the resources.
  - fork()
  - execve()
- Instance of a running program is termed as a process.
- Processes need to invoke system calls to access resources.
- Kernel executes the system calls on behalf of the process in kernel space.
- **Access to devices**:
  - Kernel provides an interface that standardizes and simplifies access to devices.
  - Keyboard, display etc.

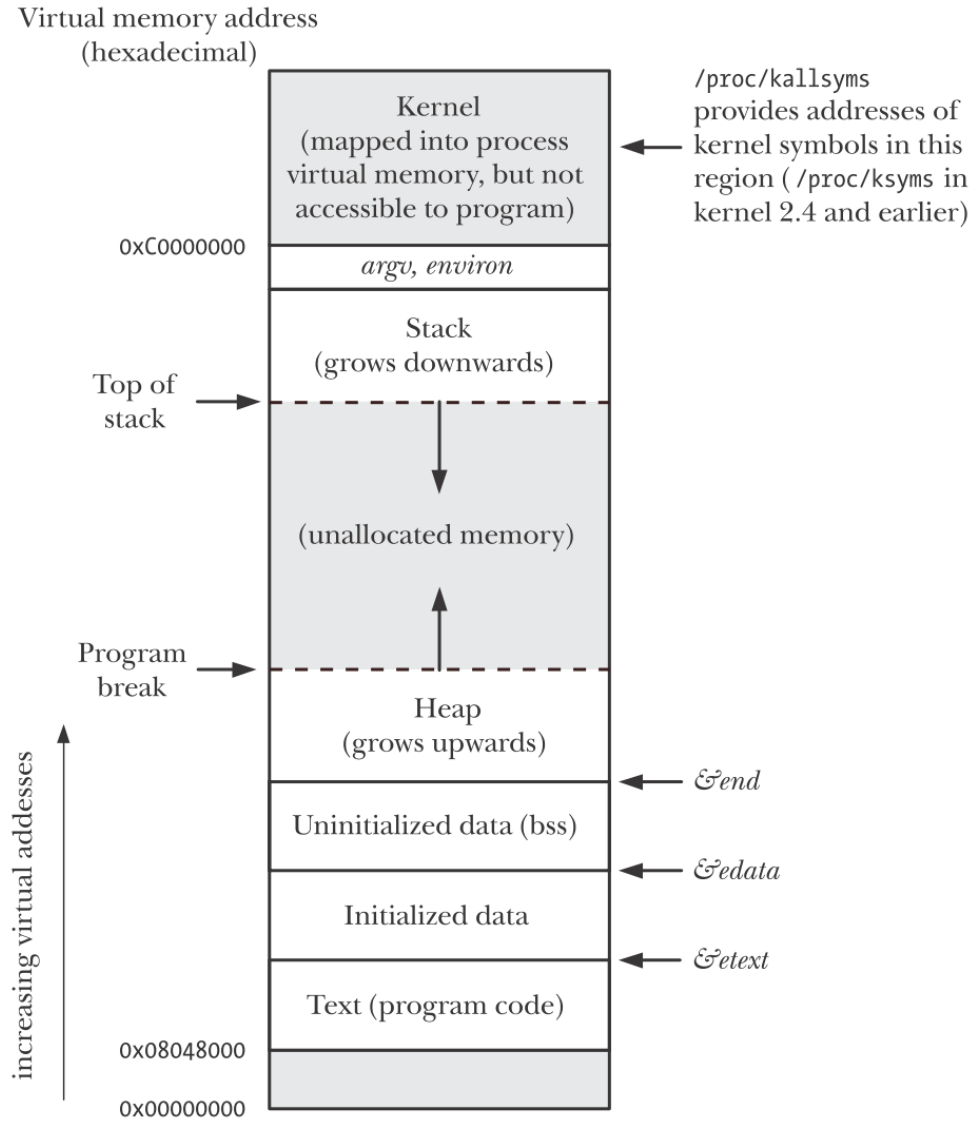
Kernel Tasks: System Calls

- Kernel entry points are known as system calls.
- System calls are software interrupts.

Kernel Mode vs User Mode

- Processor architecture allows the CPU to operate in at least two different modes:
  - User mode
  - Kernel mode
- Similarly virtual memory is marked as user space and kernel space.
  - When running user mode, the CPU can access only user space.
  - When running in kernel mode, the CPU can access both user and kernel space.
- By this arrangement, user processes can't access instructions and data structures of kernel.

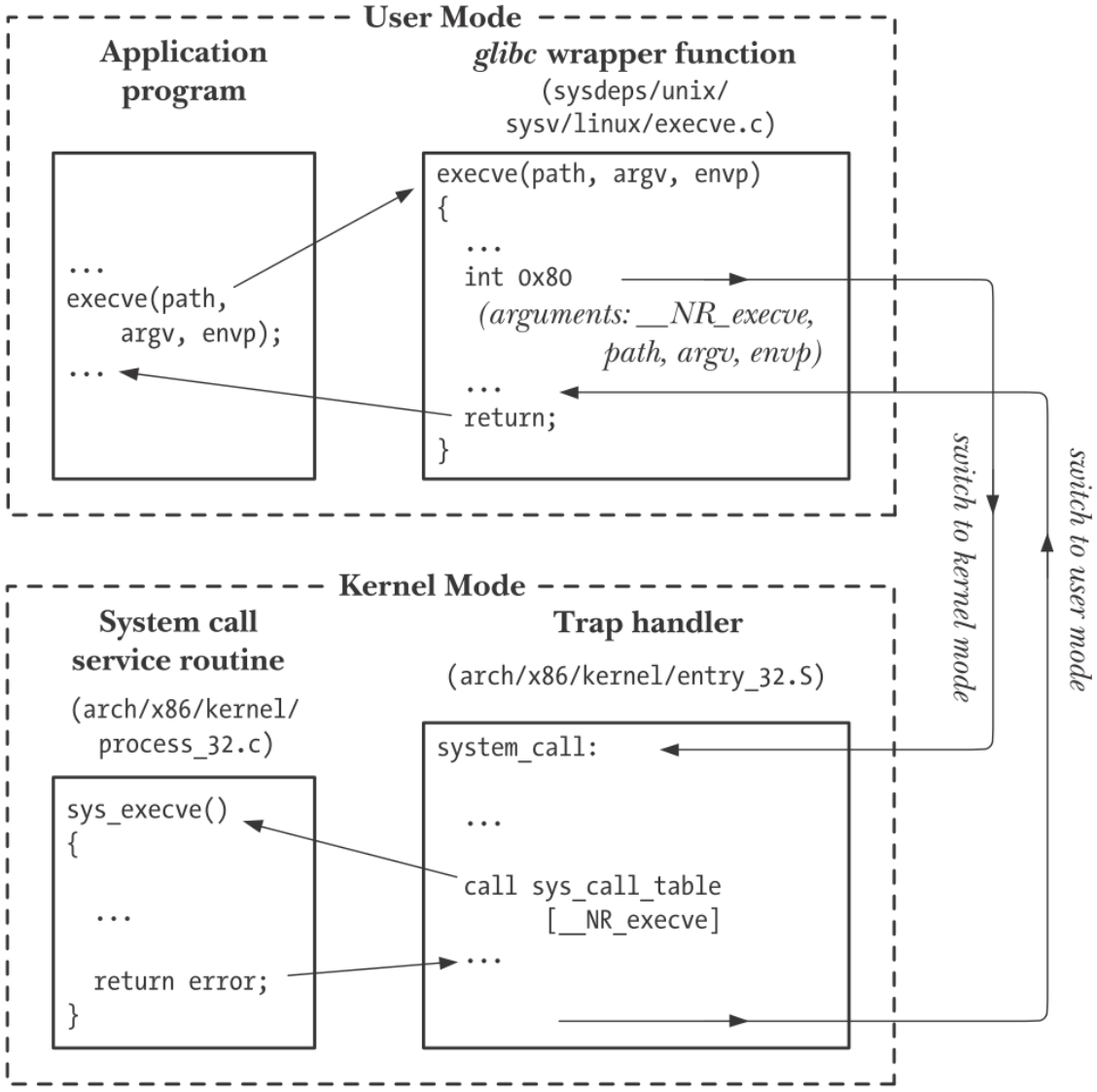
Kernel Space vs User Space



Kernel Tasks: System Calls

- A system call changes the processor from user mode to kernel mode.
- The set of sys calls are fixed. Each system call is identified by a unique number.
- System calls arguments are copied from user space to kernel space (registers).
- Wrapper function executes a trap instruction (int 0x80) causing the CPU to enter kernel mode.
- Example: execve()
  - System call number is 11.
  - Sys_call_table contains the address of the routine sys_execve().
- Not all C library calls invoke system calls. E.g.: <string.h>

Kernel Tasks: System Calls



Q&A

Next Time

innovate achieve lead

- Please read through R1: chapters 4-7

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You