

Overview of Unix / Linux operating systems

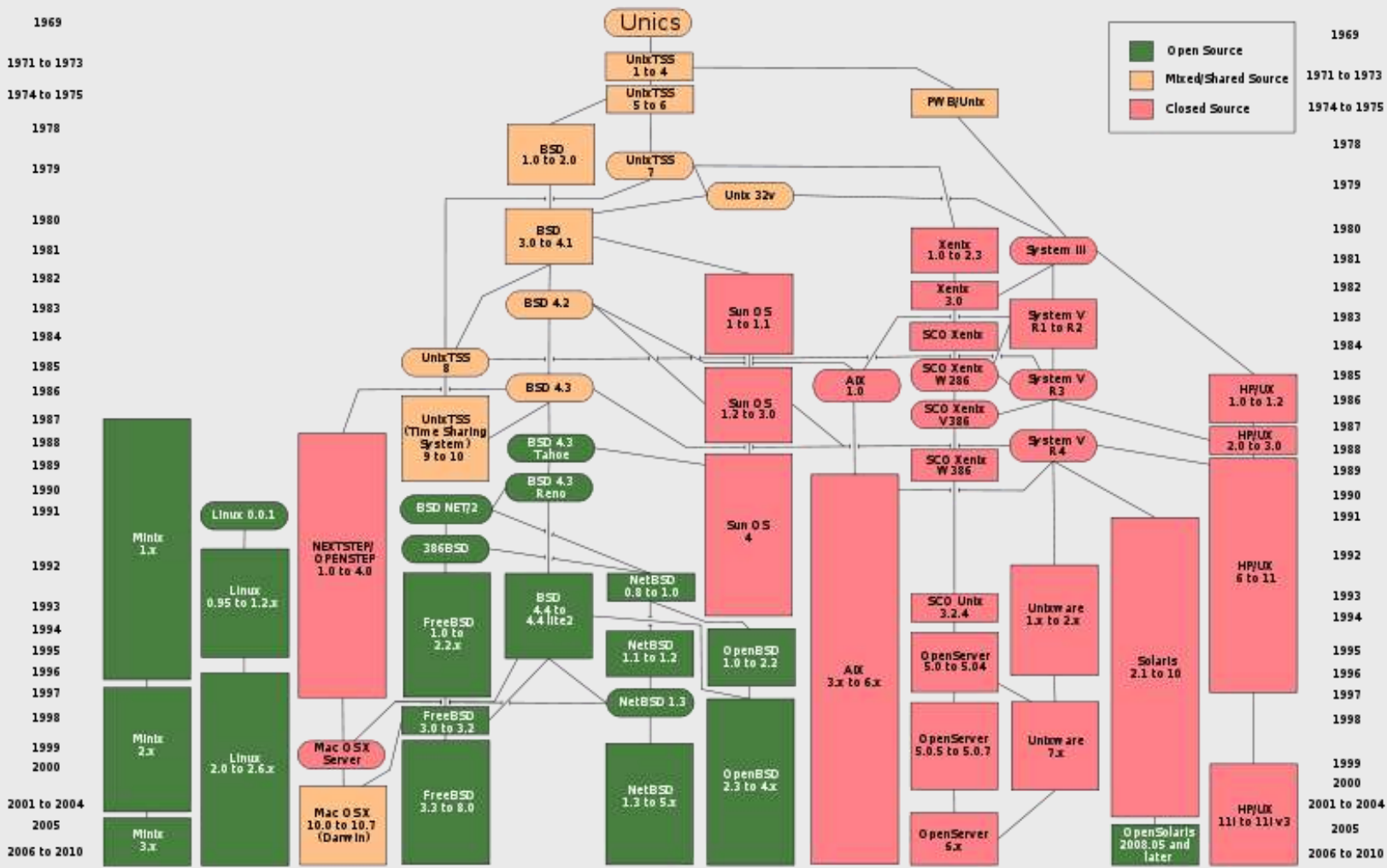
Mohammad S. Hasan

Staffordshire University, UK

Lecture Outline

- ◆ History and development of Unix / Linux
- ◆ Early versions - BSD Linux
- ◆ Design / philosophy of approach of Unix / Linux
- ◆ Overall design/structure of Unix / Linux
- ◆ Command Interpreter or Shell
- ◆ GNU General Public License (GPL)

History of different versions of Unix

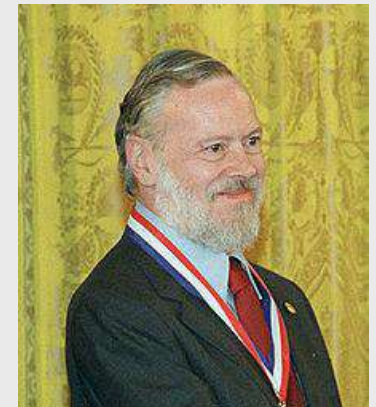


Unix Early History

- Developed by Ken Thompson and Dennis Ritchie

 - In 1969 at Bell Laboratories (AT&T)
- Originally for a small mini-computer

 - PDP 7 (only 8K of main memory)
- Small elegant operating system



Unix and C Language

➤ Early versions of Unix

- Developed in Assembly language

➤ Most of Unix version 4 (1973)

- Re-written in C Language

➤ The C language

- originally developed to provide a high level language implementation of the Unix system by Dennis Ritchie
- C Language evolved from 'B Language', developed by Ken Thompson.

```
Assembly Language  
Compare A to B  
If equal go to C  
Go to D
```

Early Advantages of Unix

- ◆ **Written in a high-level language**
- ◆ **Distributed in source form**
- ◆ **Provided powerful multi-user OS**
- ◆ **Suitable for small mini-computers**
 - **Less expensive platforms**
- ◆ **Small size, modular, clean design**

Educational involvement in development

- ◆ **Initially AT&T was not allowed to market Unix as a product**
 - **source code was made available to universities for research and education.**
- ◆ **Berkeley Software Distributions (BSD)**
 - **A group at the University of California at Berkeley played important role in the development of Unix and produced a number of influential versions**

BSD Unix

- ◆ **New features were added to new versions**
 - **demand-paged virtual memory (3 BSD)**
 - **better IPC (inter-process communication methods), TCP/IP support (4.2/4.3 BSD)**
 - **network file system, other file systems,**
 - **Portable Operating System Interface for Unix (POSIX) support (4.4 BSD)**

Commercialisation

- ◆ **Interactive Systems commercial (1977)**
- ◆ **Microsoft and SCO collaborated to port Unix to the Intel 8086 architecture**
 - **XENIX –unsuccessful as 8086 not powerful enough at the time to properly support multi-user OS**
- ◆ **1982 Bill Joy left Berkeley and founded Sun Microsystems**
 - **SunOS originally based on BSD 4.2**
 - **SunOS 5 became Solaris OS**

Commercialisation (cont.)

- ◆ In 1982 AT&T broke up and able to market Unix.
 - Released System III in 1982
 - System V the following year.
- ◆ System V UNIX
 - virtual memory (different implementation from BSD),
 - IPC (shared memory, semaphores, message queues),
 - remote file sharing, shared libraries and Streams.

Commercialisation (cont.)

- ◆ **AIX from IBM**
 - **Advanced Interactive eXecutive**
- ◆ **HP/UX from HP**
 - **Hewlett Packard Unix**
- ◆ **ULTRIX from Digital Equipment Corporation**
 - **followed by DEC OSF/1**
 - **DEC purchased by Compaq.**



Standard's Problem

◆ Problem

- **Incompatible implementations of APIs and core service components across the different Unix variants.**
- **Software written for one version of Unix would not necessarily run on other versions.**

Standard's problem and solution

◆ Solution

- Standard set of interfaces.
 - system call interface,
 - shell and system utility

◆ IEEE POSIX

- POSIX computing environments or portable OS for UNIX specifications.

◆ AT&T

- System V Interface Definition (SVID)

◆ X/Open Consortium

- X/Open Portability Guide (XPG)

Linux Begins

- ◆ A modern, free OS based on Unix standards
- ◆ History and People
 - Richard M. Stallman
 - The GNU (GNU's NOT UNIX) project
 - Linus Torvalds
 - self-contained kernel in 1991



Linux Begins (cont.)

- ◆ Designed to run efficiently and reliably on common PC hardware
 - but now also runs on a variety of other platforms.
- ◆ The core Linux operating-system kernel is entirely original
 - it can run most existing free Unix software, resulting in an entire Unix-compatible operating system free from proprietary code.

The Linux Kernel version 0.01

◆ Version 0.01 (May 1991)

- no networking,
- ran only on 80386-compatible Intel processors,
- extremely limited device driver support, and supported only the (Mini Unix) Minix file system

◆ Minix was a very simple version of a Unix, developed for teaching purposes

- Andrew S. Tanenbaum



The Linux Kernel version 1.0

◆ Linux 1.0 - March 1994

- **Unix's standard TCP/IP networking protocols**
- **BSD-compatible socket interface for networking programming**
- **Device driver support for running IP over an Ethernet**
- **Enhanced file system**
- **Support for a range of SCSI controllers for high-performance disk access**
- **Extra hardware support**

The Linux Kernel version 2.0

◆ Linux 2.0 – March 1995

- Support for multiprocessor architectures
- Improved memory-management code
- Improved TCP/IP performance
- Support for internal kernel threads
- Standardized configuration interface
- Available for Motorola 68000-series processors, Sun Sparc systems, and for PC and PowerMac systems.

Unix Design Principles

- ◆ Designed to be a time-sharing system
- ◆ Multiuser / multitasking
- ◆ Simple standard user interface (shell)
 - can be replaced
- ◆ Toolbox approach
- ◆ Combine multiple simple commands instead of using a single complex application
- ◆ Designed by programmers for programmers

Unix Design Principles (cont.)

- ◆ File system with multilevel tree-structured directories
- ◆ Files are supported by the kernel as unstructured sequences of bytes
- ◆ Supports multiple processes
 - a process can easily create new processes
- ◆ High priority given to making system interactive, and providing facilities for program development

Linux Design Principles

- ◆ Multi-user and multitasking with a full set of Unix-compatible tools.
- ◆ Adhere to traditional Unix semantics
- ◆ Fully implement the standard Unix networking model.
- ◆ Main goals
 - speed, efficiency, and standardisation.
- ◆ Linux is designed to be compliant with the relevant POSIX standards

Unix / Linux Structure

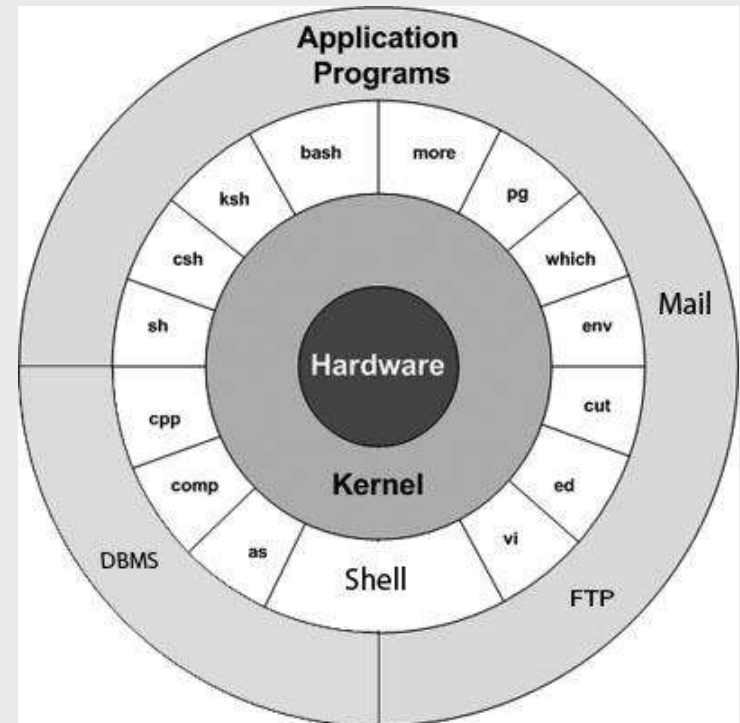
Essentially 2 layers

Above kernel interface (runs in user mode)

- Shells
- System libraries
- System programs and utilities

Below kernel interface (in supervisory mode):

- System call interface
- Kernel itself – management of processes/threads, memory, I/O, file system, device drivers



Typical Unix Structure (4.4BSD)

(the users)		
shells and commands compilers and interpreters system libraries		
<i>system-call interface to the kernel</i>		
signals terminal handling character I/O system terminal drivers	file system swapping block I/O system disk and tape drivers	CPU scheduling page replacement demand paging virtual memory
<i>kernel interface to the hardware</i>		
terminal controllers terminals	device controllers disks and tapes	memory controllers physical memory



Linux System Structure

system management programs	user processes	user utility processes	compilers
system shared libraries			
Linux kernel			
loadable kernel modules			

Unix / Linux Components

- ◆ **Unix/Linux have essentially similar set of components.**
- ◆ **The kernel**
 - **everything below the system-call interface and above the physical hardware.**
 - **file system,**
 - **CPU scheduling,**
 - **memory management,**
 - **other OS functions through system calls.**

Unix / Linux Components (Cont.)

◆ Kernel code

- executes in system mode with full access to all the physical resources.
- All kernel code and data structures are kept in the same single address space.
- This means every part of OS code/data structures is accessible from any other part of kernel

Unix / Linux Components (Cont.)

◆ The system libraries

- A standard set of functions through which applications interact with the kernel
- implement much of the operating-system functionality that does not need the full privileges of kernel code.

◆ The system programs use the kernel-supported system calls to provide useful functions

- compilation and file manipulation

Linux kernel modules

- ◆ A feature introduced by Linux
- ◆ Allows dynamic (at run-time) installation and removal of kernel components.
- ◆ A kernel module is a section of kernel code that can be compiled, loaded, and unloaded independently of the rest of the kernel.
- ◆ A kernel module may typically implement a device driver, a file system, or a networking protocol.

Linux kernel modules (cont.)

- ❖ Without kernel modules any change to kernel code would require complete re-compilation of kernel.
- ❖ The kernel module interface allows third parties to write and distribute the following items, on their own terms
 - device drivers
 - file systems etc.

Linux kernel modules (cont.)

- ◆ A Linux system to be set up with a standard, minimal kernel, without any extra device drivers.
- ◆ Linux can then allow additional modules to be loaded and integrated with the base system
 - tailored to underlying hardware and needs of system

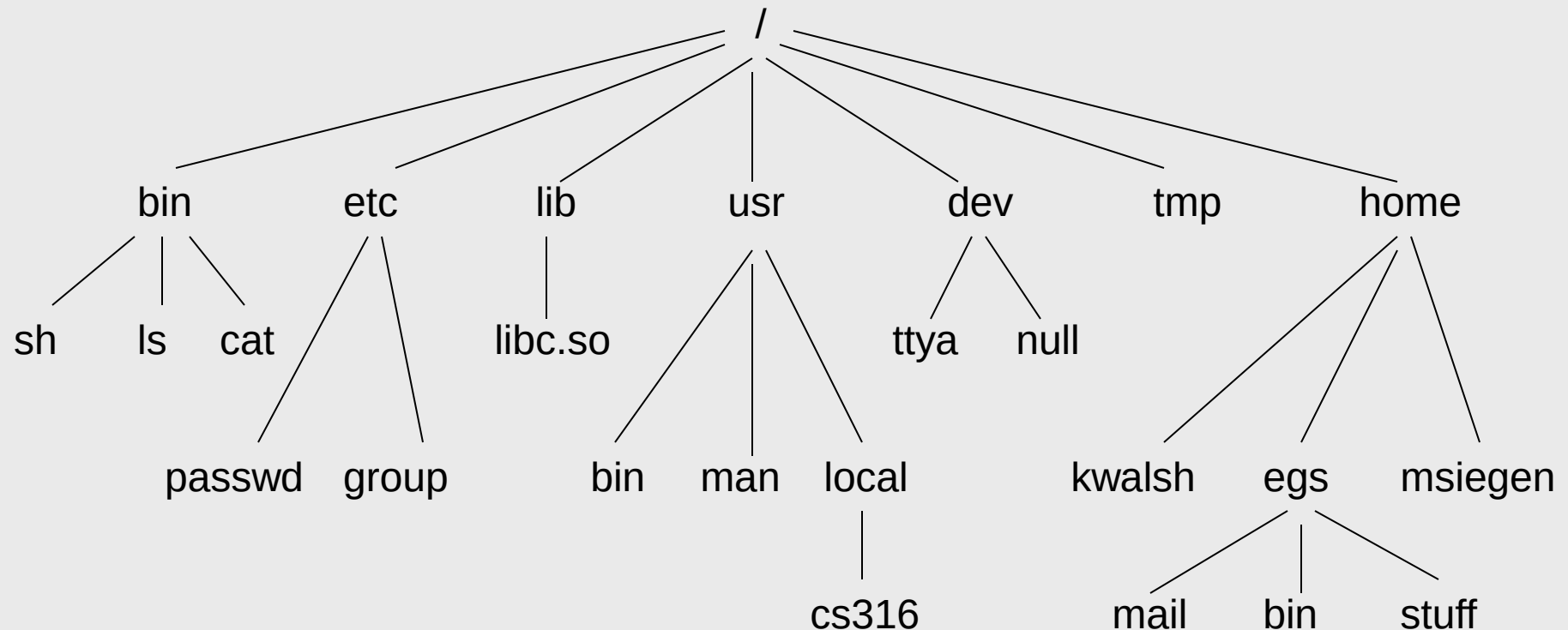
System Call Interface

- ◆ Defines the programmer interface to Unix
- ◆ The programmer and user interface define the context that the kernel must support.
- ◆ Roughly three categories of system calls in Unix.
 - File manipulation (same system calls also support device manipulation)
 - Process control
 - Information manipulation

File Manipulation

- ◆ **A file is a sequence of bytes**
 - **the kernel does not impose a structure on files.**
- ◆ **In Unix almost everything is represented by a file**
 - **whereas Windows represents everything as objects**
- ◆ **Files are organized in tree-structured directories.**
- ◆ **Directories are files**
 - **contain information on how to find other files.**

Typical Unix / Linux type directory structure



Shell and Commands



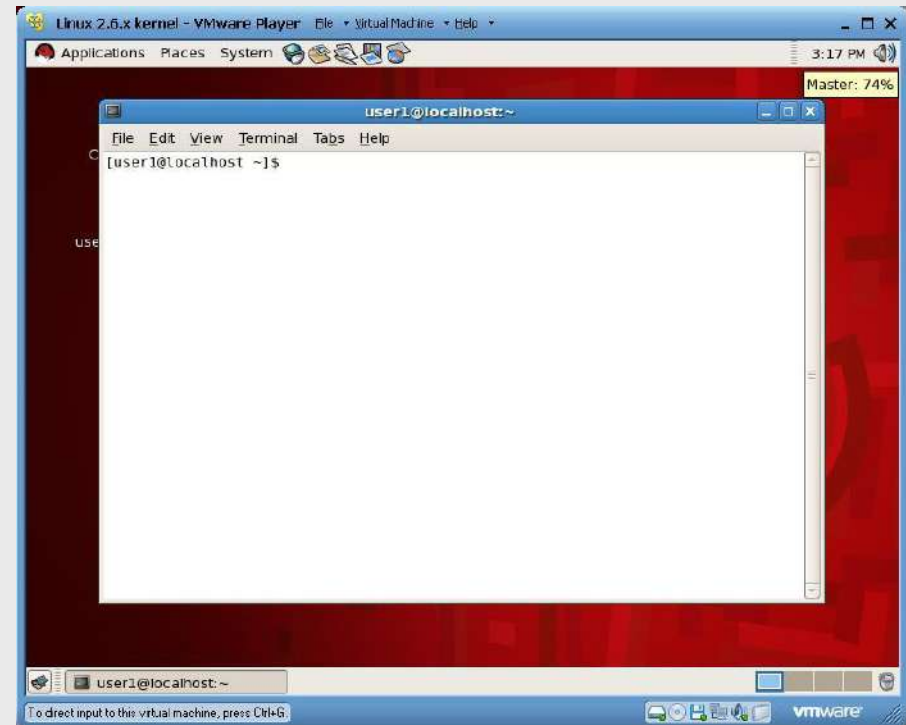
Shell

- **Command Interpreter - the user process which executes programs**
- **Called a shell, because it surrounds the kernel**
- **The shell indicates its readiness to accept another command by outputting a prompt**



Examples

- **Bourne shell (sh)**
- **Bourne again shell (bash)**
- **C Shell (csh)**



Shell and Commands (cont.)

- ◆ A typical command is an executable binary object file
- ◆ The shell looks through the specified location (search path) to find the command executable file, which is then loaded and executed
 - The directories `/bin` and `/usr/bin` are almost always in the search path
- ◆ The shell usually suspends its own execution until the command completes.
 - Parent child relationship

Shell and Commands (cont.)

- ◆ Programmers and users mainly deal with already existing systems programs
- ◆ The most common systems programs are file or directory oriented
 - `mkdir, rmdir, cd, pwd, ls, cp, mv, rm`
- ◆ X Windows system provides basis for development of GUI interfaces

Linux System Specific

◆ Linux uses many tools

- Berkeley's BSD operating system,
- MIT's X Window System,
- Free Software Foundation's GNU project.
- Networking-administration tools from BSD 4.3 code

◆ The main system libraries

- started by the GNU project, with improvements provided by the Linux community.

◆ The Linux system is maintained by a loose network of developers.

Linux Distributions

- ◆ **Standard, precompiled sets of packages, or distributions include**
 - **The Linux system kernel,**
 - **System installation and management utilities,**
 - **Ready-to-install packages of common Unix tools.**
- ◆ **Modern distributions include advanced package management and automated installation.**

Linux Distributions

◆ Many distributions now available

● Red Hat

■ Distributions from a commercial source.

● Ubuntu

● Debian

● Suse

● Many others



Linux Licensing

- ◆ **The Linux kernel is distributed under the GNU General Public License (GPL)**
 - **Anyone using Linux, or creating their own derivate of Linux, may not make the derived product proprietary**
 - **software released under the GPL may not be redistributed as a binary-only product.**

References

- ◆ **Operating System Concepts - Chapter 21 & Appendix A**