Operating System

Lecture Two /

- Linux History
- Distributions Debian / Ubuntu
- Installation (Single/Dual mode –Virtual Box – Portable)
- Command line (Terminal)
- Chapter 1 & 2 exercise

OS Linux History

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source.**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copy left" **GNU Public License (GPL)**
- Examples include **GNU/Linux** 

and **BSD** *(Berkeley Software Distribution)* **UNIX** (including core of **Mac OS X**), and many more
- Can use VMM (Virtual Machine Management) like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)

  Use to run guest operating systems for exploration

OS Linux History

Firmware

is programming that's written to the read-only memory (ROM) of a computing device, which is added at the time of manufacturing, is used to run user programs on the device. (IBM prefers the term microcode)

OS Linux History

Traditional UNIX Systems

- Were developed at Bell Labs and became operational on a PDP-7 (Microcomputer 1965) in 1970. (Written by Assembly Language – Ken Thompson)
- Incorporated many ideas from Multics (Multiplexed Information and Computing Service) is a timesharing operating system begun in 1965 and used until 2000.
- PDP-11(1970 -1990) was a milestone because it first showed that UNIX would be an OS for all computers.

OS Linux History

Traditional UNIX Systems

- Next milestone was rewriting UNIX in the programming language C
  - demonstrated the advantages of using a high-level language for system code (ken Thompson & Dennis Ritchie C Author) 1973
- Was described in a technical journal for the first time in 1974
- First widely available version outside Bell Labs was Version 6 in 1976
- Version 7, released in 1978 is the ancestor of most modern UNIX systems
- Most important of the non-AT&T systems was UNIX BSD (Berkeley Software Distribution)

OS Unix Structure

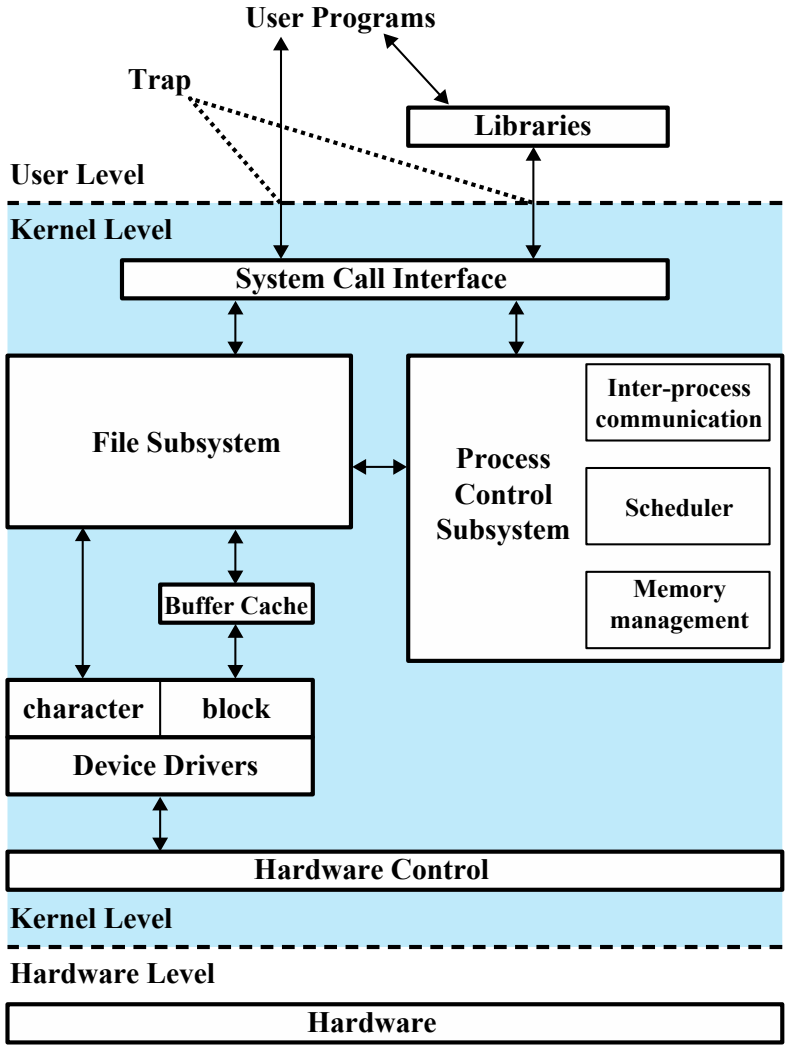


Figure 2.16 Traditional UNIX Kernel

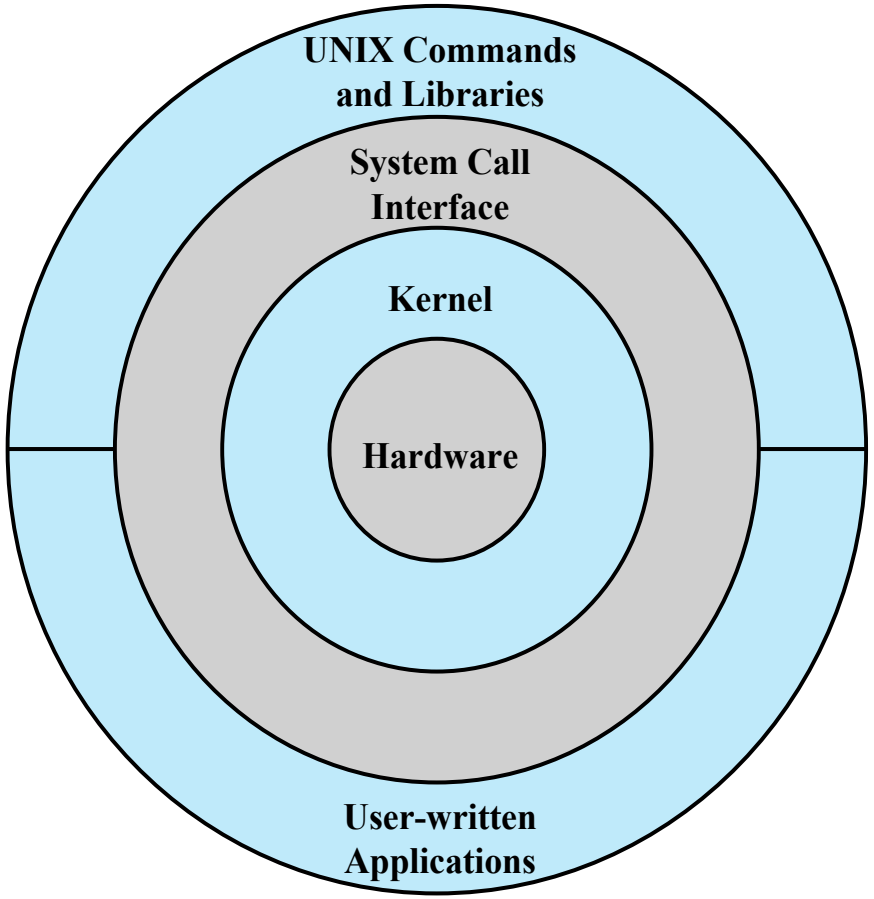


Figure 2.15 General UNIX Architecture

LINUX Overview

- Started out as a UNIX variant for the IBM PC
- Linus Torvalds, a Finnish student of computer science, wrote the initial version
- Linux was first posted on the Internet in 1991
- Today it is a full-featured UNIX system that runs on several platforms & Distributions of linux (Red Hat, Solaris, Debian, Ubuntu, Fedora, etc.)
- Is free and the source code is available
- Key to success has been the availability of free software packages
- Highly modular and easily configured

OS Modern Unix Kernel Structure

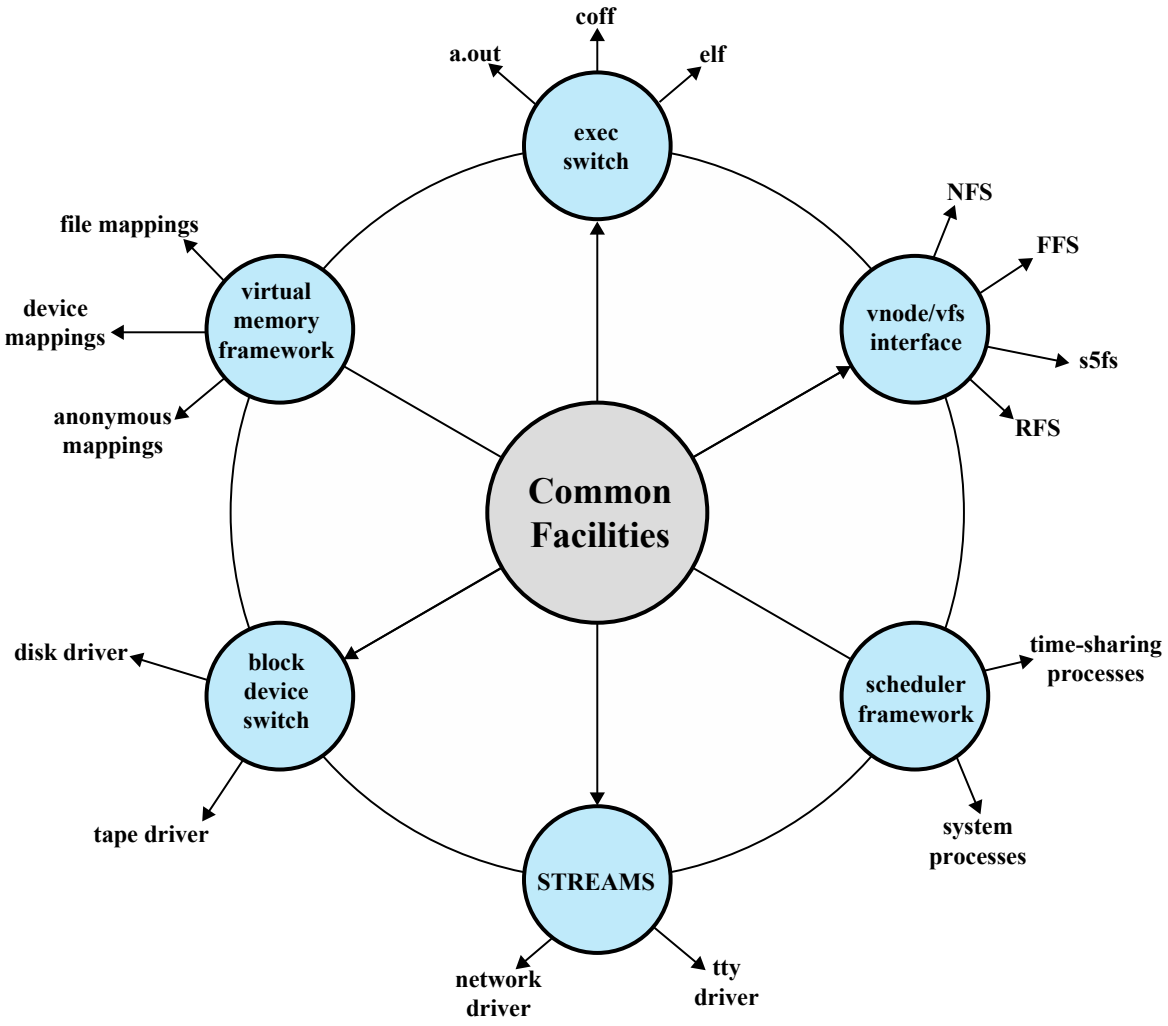


Figure 2.17 Modern UNIX Kernel [VAHA96]

OS Modes of Operation

User Mode

- user program executes in user mode
- certain areas of memory are protected from user access
- certain instructions may not be executed

Kernel Mode

- monitor executes in kernel mode
- privileged (higher priority) instructions may be executed
- protected areas of memory may be accessed

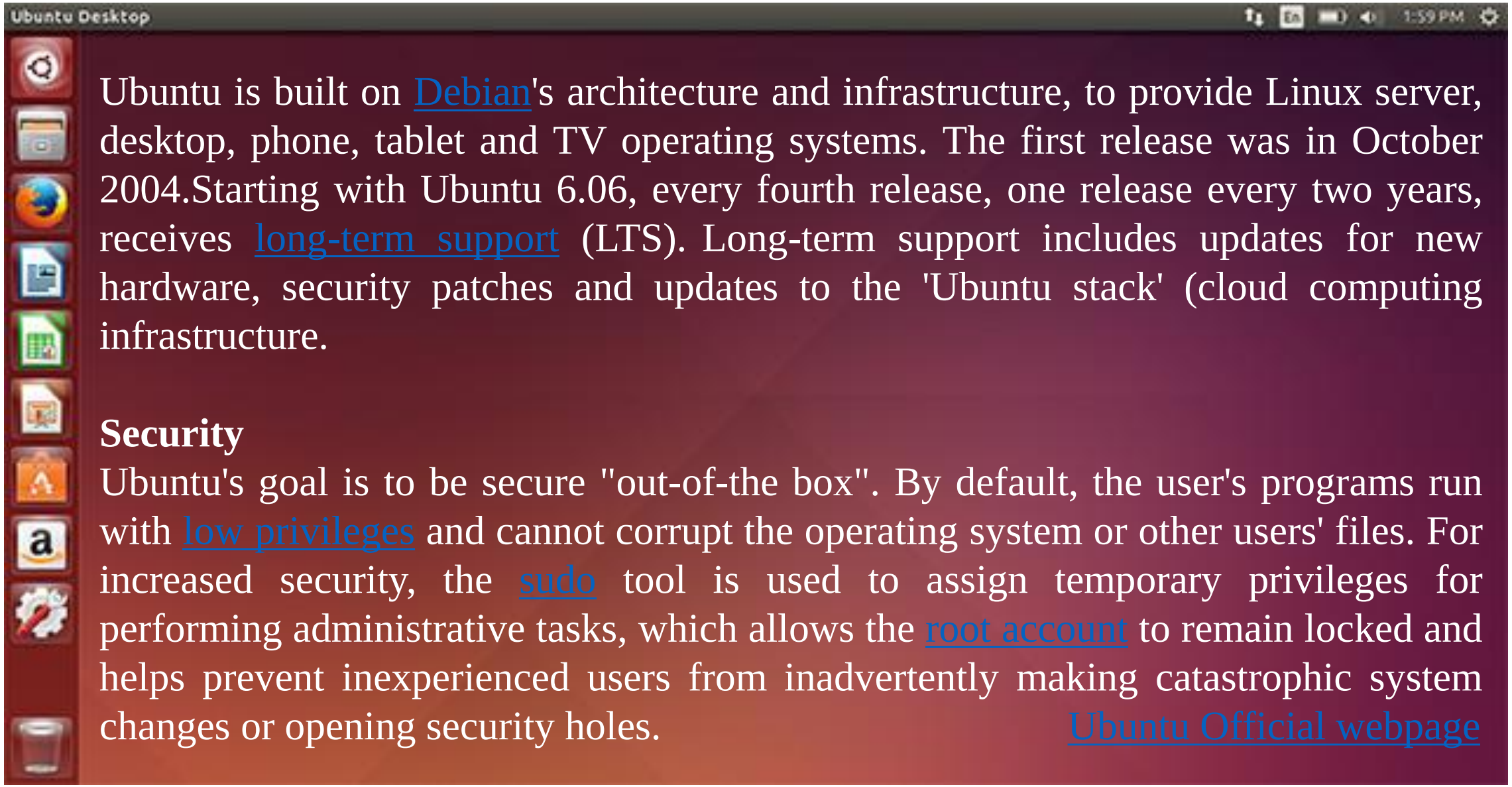
OS Linux Debian

The Debian Project was first announced in 1993 by **Ian Murdock**, Debian 0.01 was released on September 15, 1993,and the first stable release was made in 1996.

The Debian stable release branch is one of the most popular for personal computers and network servers, and has been used as a base for several other distributions.

Debian Official Webpage

OS Linux Ubuntu



Ubuntu is built on Debian's architecture and infrastructure, to provide Linux server, desktop, phone, tablet and TV operating systems. The first release was in October 2004.Starting with Ubuntu 6.06, every fourth release, one release every two years, receives long-term support (LTS). Long-term support includes updates for new hardware, security patches and updates to the 'Ubuntu stack' (cloud computing infrastructure.

Security

Ubuntu's goal is to be secure "out-of-the box". By default, the user's programs run with low privileges and cannot corrupt the operating system or other users' files. For increased security, the sudo tool is used to assign temporary privileges for performing administrative tasks, which allows the root account to remain locked and helps prevent inexperienced users from inadvertently making catastrophic system changes or opening security holes.

Ubuntu Official webpage

OS Linux Installation

Downloading & Preparing ISO Memory Stick/ DVD

- Debian
- Ubuntu LTS

- AOMEI Partition Assistant
- Bootable Memory Stick (Rufus)
- Bootable DVD (IsoCreator, Free ISO Burner)

- Virtual Box

OS Linux Installation Mode

1- Single Mode (only one OS) easy straight forward

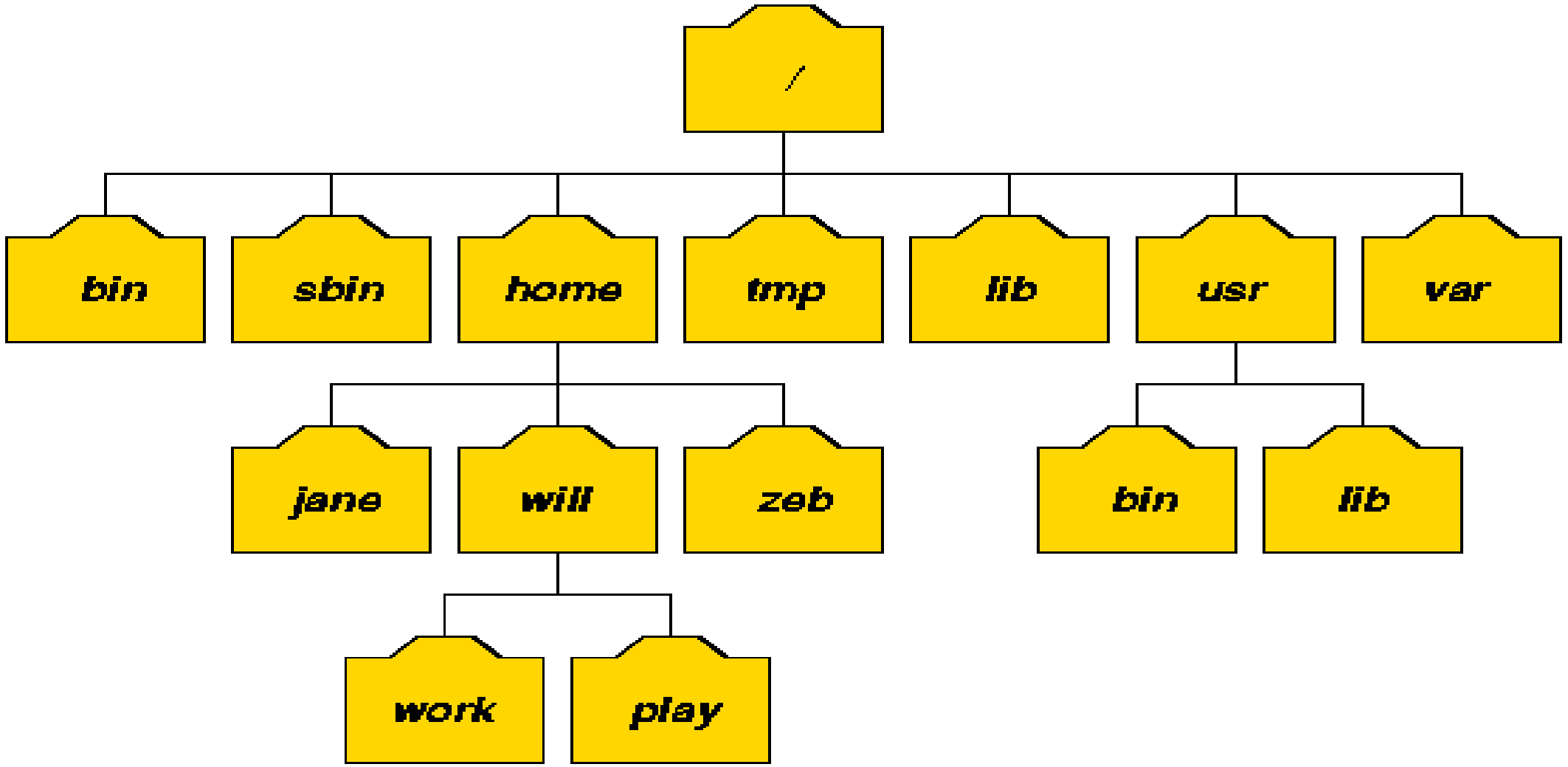
2- Parallel Dual Mode (Win / Linux) depending on system hardware VGA, RAM, CPU core, Boot BIOS (basic input/output system) / UEFI (Unified Extensible Firmware Interface) (precisely updated UEFI)

Installation Error Requires Community following for solutions.

3- Virtual Box Dual Mode window main platform

4- Portable Mode Memory Stick

OS Typical UNIX Directory Structure



OS Typical UNIX Directory Structure

Directory	Typical Contents
/	The "root" directory
/bin	Essential low-level system utilities
/usr/bin	Higher-level system utilities and application programs
/sbin	Superuser system utilities (for performing system administration tasks)
/lib	Program libraries (collections of system calls that can be included in programs by a compiler) for low-level system utilities
/usr/lib	Program libraries for higher-level user programs
/tmp	Temporary file storage space (can be used by any user)
/home or /homes	User home directories containing personal file space for each user. Each directory is named after the login of the user.
/etc	UNIX system configuration and information files
/dev	Hardware devices
/proc	A pseudo-filesystem which is used as an interface to the kernel. Includes a sub-directory for each active program (or process).

OS Sys Call

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

OS Linux Filesystem Categories

Every item stored in a UNIX filesystem belongs to one of four types:

1 - Ordinary files can contain text, data, or program information.

2 - Directories are containers or folders that hold files, and other directories.

3 - Devices To provide applications with easy access to hardware devices, UNIX allows them to be used in much the same way as ordinary files. There are two types of devices in UNIX **- block-oriented** devices which transfer data in blocks (e.g. hard disks) and **character-oriented** devices that transfer data on a byte-by-byte basis (e.g. modems and dumb terminals).

4 - Links A link is a pointer to another file. There are two types of links **- a hard link** to a file is indistinguishable from the file itself. A **soft link** (or symbolic link) provides an indirect pointer or shortcut to a file.

OS Linux Terminal Commands

Linux man

Some Examples

- **pwd (print [current] working directory)**

  $ pwd (Enter) output /usr/bin

- **ls (list directory)**

  $ ls (Enter) out put bin dev home mnt share usr var boot etc lib proc sbin tmp vol

  Options –a (all), -l (long listing) (etc, from ls man)

  $ ls –al output

OS Linux Terminal Commands

- **cd (change [current working] directory)**

  $ cd *path*

  $ cd ../.. (Enter) Output root

  $ cd .. (Enter) Output one step backward (Parent directory)

  $ cd (Enter) Output home directory

Or the path to a particular distention (the current home dir /directory name/directory name from root ../user or specific directory/…….

- **mkdir (make directory)**

  $ mkdir OS2016/17 output (OS2016/17 Subdirectory in current directory)

- **rmdir (remove directory)**

  $ rmdir OS2016/17 output (Removes Subdirectory OS2016/17 in current directory)

OS Linux Terminal Commands

- **cp (copy)**

  $ cp source-file(s) destination

To copy entire directories (including their contents), use a recursive copy:

  $ cp -rd source-directories destination-directory

- **mv (move/rename)**

  $ mv source destination

is used to rename files/directories and/or move them from one directory into another. Exactly one source and one destination must be specified.

- **rm (remove/delete)**

  $ rm target-file(s) (works as shift delete)

  -r (recursive) -f (force) -rf (forces deleting everything)

- **Sudo** Superuser

OS File and Directory Permissions

Permission	File	Directory
read	User can look at the contents of the file	User can list the files in the directory
write	User can modify the contents of the file	User can create new files and remove existing files in the directory
execute	User can use the filename as a UNIX command	User can change into the directory, but cannot list the files unless (s)he has read permission. User can read files if (s)he has read permission on them.

OS File and Directory Permissions

---	0
--x	1
-w-	2
-wx	3
r--	4
r-x	5
rw-	6
rwx	7

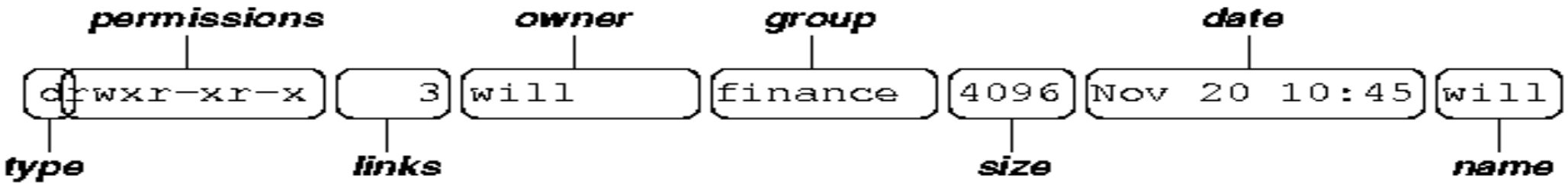
chmod (change [file or directory] mode)

$ chmod options files

For example the command:

$ chmod 600 private.txt

$ chmod ug=rw,o-rw,a-x *.txt



OS Linux Online Tutorial

Youtube

https://www.youtube.com/watch?v=9t_gJWC32zk

Linux Handout & Tutorial

http://www.guru99.com/unix-linux-tutorial.html

William Knottenbelt Imperial college London 2001

http://www.doc.ic.ac.uk/~wjk/UnixIntro/index.html

WORLD OF ASIC 2014

http://www.asic-world.com/scripting/unix3.html

OS Ch 1 Exc

1.1 What are the three main purposes of an operating system?

1.2 We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to “waste” resources? Why is such a system not really wasteful?

1.3 What is the main difficulty that a programmer must overcome in writing an operating system for a real-time environment?

1.4 Keeping in mind the various definitions of ***operating system,*** consider whether the operating system should include applications such as web browsers and mail programs. Argue both that it should and that it should not, and support your answers.

OS Ch 1Exc

1.5 How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?

1.6 Which of the following instructions should be privileged?

a. Set value of timer. / b. Read the clock. / c. Clear memory. / d. Issue a trap instruction.

e. Turn off interrupts. / f. Modify entries in device-status table.

g. Switch from user to kernel mode. / h. Access I/O device.

1.7 Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.

1.8 Some CPUs provide for more than two modes of operation. What are two possible uses of these multiple modes?

OS Ch 1 Exc

1.9 Timers could be used to compute the current time. Provide a short description of how this could be accomplished.

1.10 Give two reasons why caches are useful. What problems do they solve?
What problems do they cause? If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?

1.11 Distinguish between the client–server and peer-to-peer models of distributed systems.

OS Ch 2 Exc

2.1 What is the purpose of system calls?

2.2 What are the five major activities of an operating system with regard to process management?

2.3 What are the three major activities of an operating system with regard to memory management?

2.4 What are the three major activities of an operating system with regard to secondary-storage management?

2.5 What is the purpose of the command interpreter? Why is it usually separate from the kernel?

OS Ch 2 Exc

2.6 What system calls have to be executed by a command interpreter or shell in order to start a new process?

2.7 What is the purpose of system programs?

2.8 What is the main advantage of the layered approach to system design? What are the disadvantages of the layered approach?

2.9 List five services provided by an operating system, and explain how each creates convenience for users. In which cases would it be impossible for user-level programs to provide these services? Explain your answer.

2.10 Why do some systems store the operating system in firmware, while others store it on disk?

2.11 How could a system be designed to allow a choice of operating systems from which to boot? What would the bootstrap program need to do?

OS Next Lecture

Chapter 3

Process Concept

Process Scheduling

Operations on Processes