Introduction

Operating System Lab
Spring 2015

Roya Razmnoush
CE & IT Department, Amirkabir University of Technology

Linux and Unix history

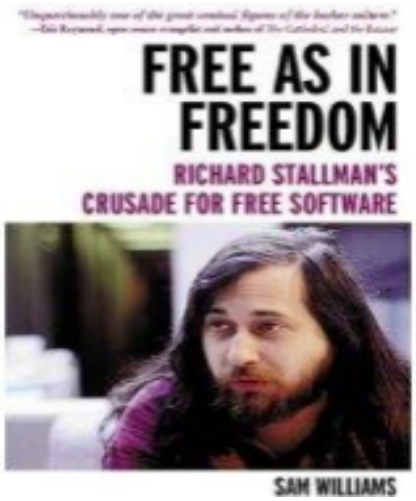
Before Linux

- In 80's, Microsoft's DOS was the dominated OS for PC
- Apple MAC was better, but expensive
- UNIX was much better, but much, much more expensive. Only for minicomputer for commercial applications
- People was looking for a UNIX based system, which is cheaper and can run on PC
- Both DOS, MAC and UNIX were proprietary, i.e., the source code of their kernel is protected
- No modification is possible without paying high license fees

Linux and Unix history

GNU project

- Established in 1984 by Richard Stallman, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs



GNU is a recursive acronym for "GNU's Not Unix"

Aim at developing a complete Unix-like operating system which is free for copying and modification

Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed

Linux and Unix history

Beginning of Linux

- A famous professor Andrew Tanenbaum developed Minix, a simplified version of UNIX that runs on PC
- Minix is for class teaching only. No intention for commercial use
- In Sept 1991, Linus Torvalds, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1

Linux and Unix history

- Soon more than a hundred people joined the Linux camp. Then thousands. Then hundreds of thousands
- It was licensed under GNU General Public License, thus ensuring that the source codes will be free for all to copy, study and to change.

Linux and Unix history

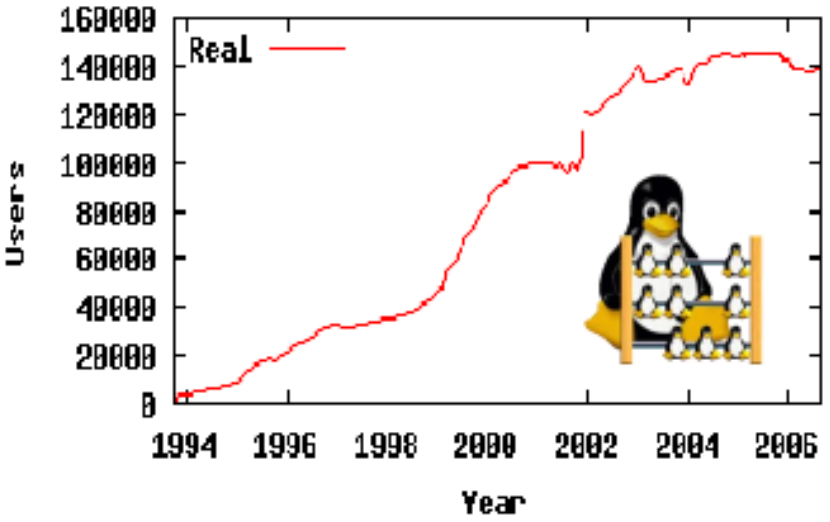
Linux Today

- Linux has been used for many computing platforms
  - PC, PDA, Supercomputer,…
- Not only character user interface but graphical user interface is available
- Commercial vendors moved in Linux itself to provide freely distributed code. They make their money by compiling up various software and gathering them in a distributable format
  - Red Hat, Slackware, etc

Linux and Unix history

Growing and growing…

In order to encourage wide dissemination of his OS, Linus made the source code open to public. At the end of 1992 there were about a hundred Linux developers. Next year there were 1000. And the numbers multiplied every year.



Linux: No of Users

Source: The Linux Counter

Recent estimates say about 29 million people use Linux worldwide. The effects of the dot-com bust, IT slowdown and global economic recession can be clearly seen.

Linux and Unix history

138712
users registered
155679
machines registered

Open Source Concept

What is Open Source Software?

- **Open Source software is software licensed under an agreement that conforms to the Open Source Definition**
  - Access to Source Code
  - Freedom to Redistribute
  - Freedom to Modify
  - Non-Discriminatory Licensing (licensee/product)
    - The license must not discriminate against any person or group of persons
  - Integrity of Authorship
    - The author's right is the right not to have the work subjected to derogatory treatment. (COPYRIGHT ACT 1968 -SECT 195AI)
    - The license may require derived works to carry a different name or version number from the original software.
  - Redistribution in accordance with the Open Source License Agreement
    - The license must explicitly permit distribution of software built from modified source code.

Open Source Concept

What is Open Source Software?

- **Any developer/licensor can draft an agreement that conforms to the OSD, though most licensors use existing agreements**
  - GNU Public License ("GPL")
  - Lesser/Library GNU Public License ("LGPL")
  - Mozilla Public License
  - Berkeley Software Distribution license ("BSD")
  - Apache Software License
  - See complete list at www.opensource.org/licenses

Open Source Concept

Open Source Licenses

- Copyleft vs. copyright
  - Copyright: prohibit others from reproducing, adapting, or distributing copies of the author's work
  - Copyleft: give every person who receives a copy of a work permission to reproduce, adapt or distribute the work as long as any resulting copies or adaptations are also bound by the same copyleft licensing scheme
- Two widely used open source licenses have “Copyleft” Provisions
  - GNU Public License (“GPL”)
  - Lesser GNU Public License or Library GNU Public License (“LGPL”)
- Most other licenses do not have Copyleft terms

Open Source Concept

Proprietary vs. Open Source Licensing Models

Proprietary Model	Open Source Model
Licensor distributes object code only; source code is kept a trade secret	Licensor distributes source code
Modifications are prohibited	Modifications are permitted
All upgrades, support and development are done by licensor	Licensee may do its own development and support or hire any third party to do it
Fees are for the software license, maintenance, and upgrades	Fees, if any, are for integration, packaging, support, and consulting
Sublicensing is prohibited, or is a very limited right	Sublicensing is permitted; licensee may have to distribute the source code to program and modifications

Open Source Concept

Key GNU Public License ("GPL") Terms

- **License Rights Granted under the GPL**
  - Licensee may run the Program
  - Licensee may copy and distribute verbatim copies of the Program's source code
  - Licensee may create "derivative works" of the Program
  - Licensee may distribute such derivative works

Open Source Concept

Key GNU Public License ("GPL") Terms

- **If a licensee of a Program distributes that Program, or any "work based on the Program," such licensee must:**
  - also distribute the source code for the Program and for the work based on the Program, and
  - cause such works to be licensed at no charge under the terms of the GPL

Open Source Concept

Key Lesser GPL("LGPL") Terms

- Very similar to the GPL Intent is to promote use of certain Libraries in conjunction with "non-free" programs
- Contains exception for linking "works that use the library" to proprietary programs, which mitigates some Copyleft concerns

Open Source Concept

Key Lesser GPL("LGPL") Terms

- **The LGPL has the same Copyleft obligations as the GPL, except:**
  - A work that uses only "numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines . . . or less)" is not subject to Copyleft obligations
  - a work that contains no derivative of any portion of the [GPL's] Library, but is designed to work with the Library by being compiled or linked with the Program, is not subject to Copyleft obligations

Open Source Concept

Linux - free software

- Free software, as defined by the FSF (Free Software Foundation), is a "matter of liberty, not price." To qualify as free software by FSF standards, you must be able to:
  - Run the program for any purpose you want to, rather than be restricted in what you can use it for.
  - View the program's source code.
  - Study the program's source code and modify it if you need to.
  - Share the program with others.
  - Improve the program and release those improvements so that others can use them.

Linux Distributions

- What required
  - OS, kernel, kernel-space
  - Applications, user-space
    - Interfaces and basic commands
    - Applications
    - Services
- Kernel by Linus and world wide developers
- Most of applications by GNU project
  - GNU : GNU is Not Unix
- Our operating system: GNU/Linux

Linux Distributions

- Red Hat & Fedora
  - Stable and commercial support
- SuSE
  - Most updated and user friendly
  - Supported by Novel
- UBUNTU
  - New fast growing user friendly Debian based
- Debian
  - Most complete distribution, the Sarge

Linux Distributions

- Bluecat
  - Linux for embedded systems
- LinuxPPC
  - Linux to run on PowerPC machines
- Astaro
  - Security applaince, Firewall, Antivirus
- Live CD
  - KNOPPIX, PHLAK, Karamad, ...

Linux Architecture

What is Kernel?

- Modules or sub-systems that provide the operating system functions.
- The Core of OS

Linux Architecture

Type of kernel

- Micro kernel (Modular kernel)
- Monolithic kernel

Linux Architecture

Micro Kernel

- It includes code only necessary to allow the system to provide major functionality.
  - IPC
  - Some memory management
  - Low level process management & scheduling
  - Low level input / output
- Such as Amoeba, Mach and …

Linux Architecture

Monolithic Kernel

- It includes all the necessary functions.
- Such as Linux and …

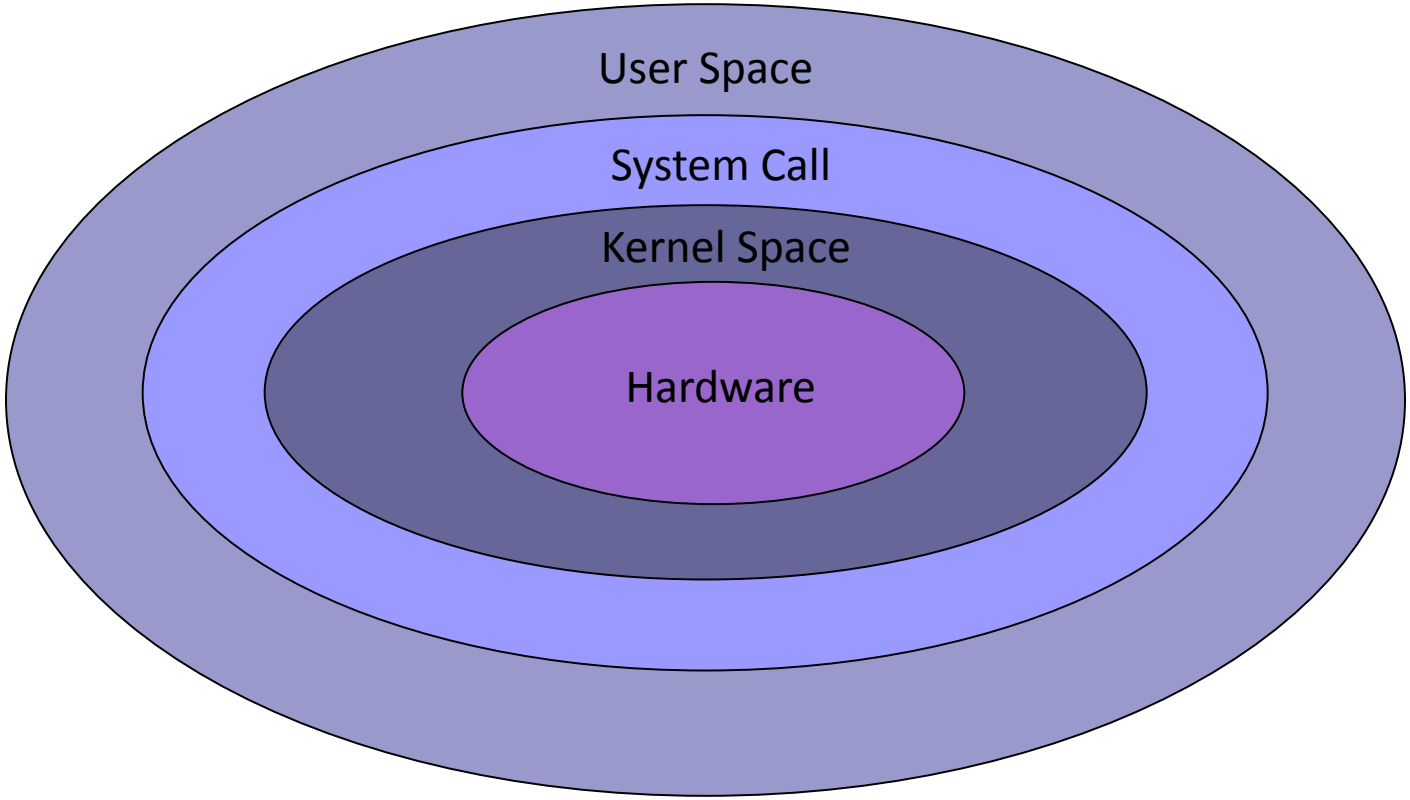
Linux Architecture

Micro vs Monolithic Kernel

- Micro
  - Flexible
  - Modular
  - Easy to implement
- Monolithic
  - Performance

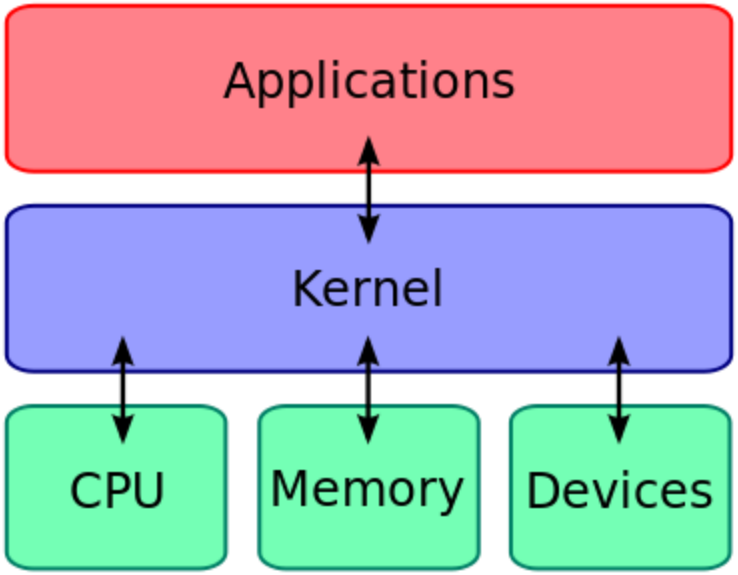
Linux Architecture

Kernel Architecture



Linux Architecture

Kernel Architecture



Linux Architecture

User Space

- The User Space is the space in memory where user processes run.
- This Space is protected.
  - The system prevents one process from interfering with another process.
  - Only Kernel processes can access a user process

Linux Architecture

Kernel Space

- The kernel Space is the space in memory where kernel processes run.
- The user has access to it only through the system call.

Linux Architecture

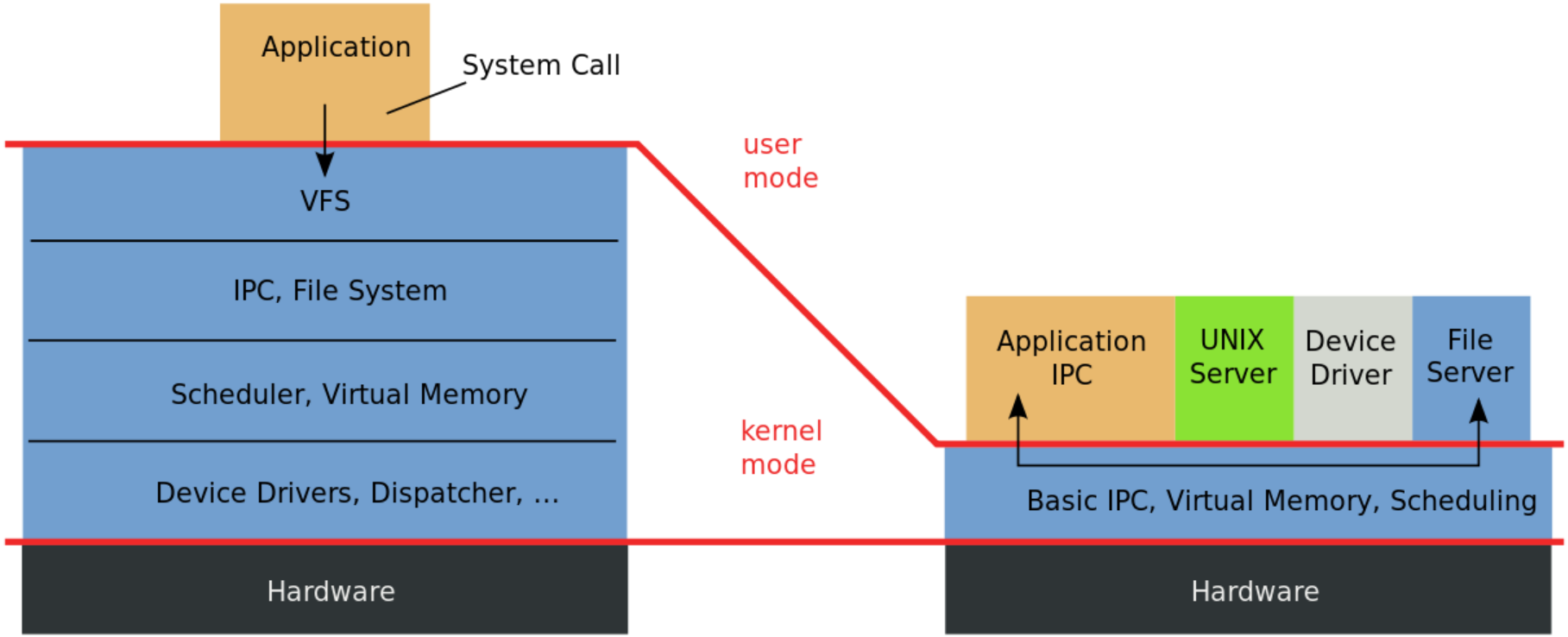
System Call

- User Space and Kernel Space are in different spaces.
- When a System Call is executed, the arguments to the call are passed from User Space to Kernel Space.
- A user process becomes a kernel process when it executes a system call.

Linux Architecture

Monolithic Kernel
based Operating System

Microkernel
based Operating System



Linux Architecture

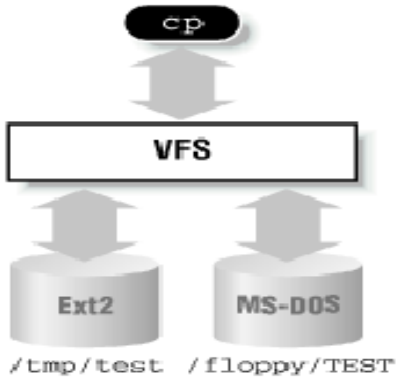
Kernel Functional Architecture

- File System
- Process Management
- Device Control
- Memory Management
- Networking

Linux Architecture

File System

- It is responsible for storing information on disk and retrieving and updating this information.
- It manages all the different file system.
- In Linux everything is file.



Linux Architecture

Process Management

- The Unix OS is a time-sharing system.
- Every process is scheduled to run for a period of time (time slice).
- Kernel creates, manages and deletes the processes

Linux Architecture

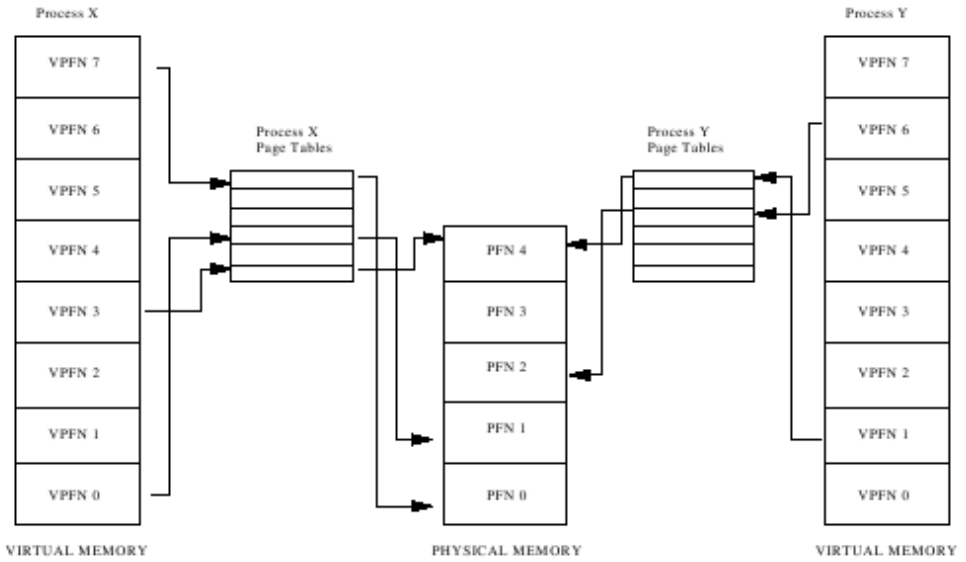
Device Control

- On of the purpose of an OS is to hide the system's hardware from user.
- Instead of putting code to manage the HW controller into every application, the code is kept in the Linux kernel.
- It abstracts the handling of devices.
  - All HW devices look like regular files.

Linux Architecture

Memory Management

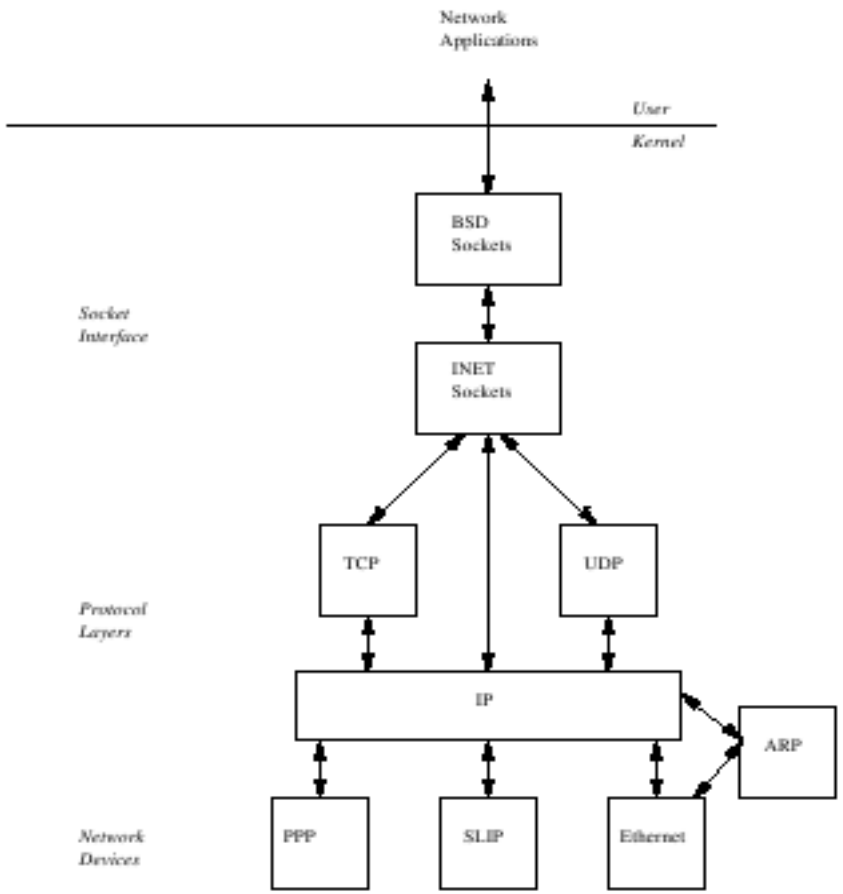
- Physical memory is limited.
- Virtual memory is developed to overcome this limitation.



Linux Architecture

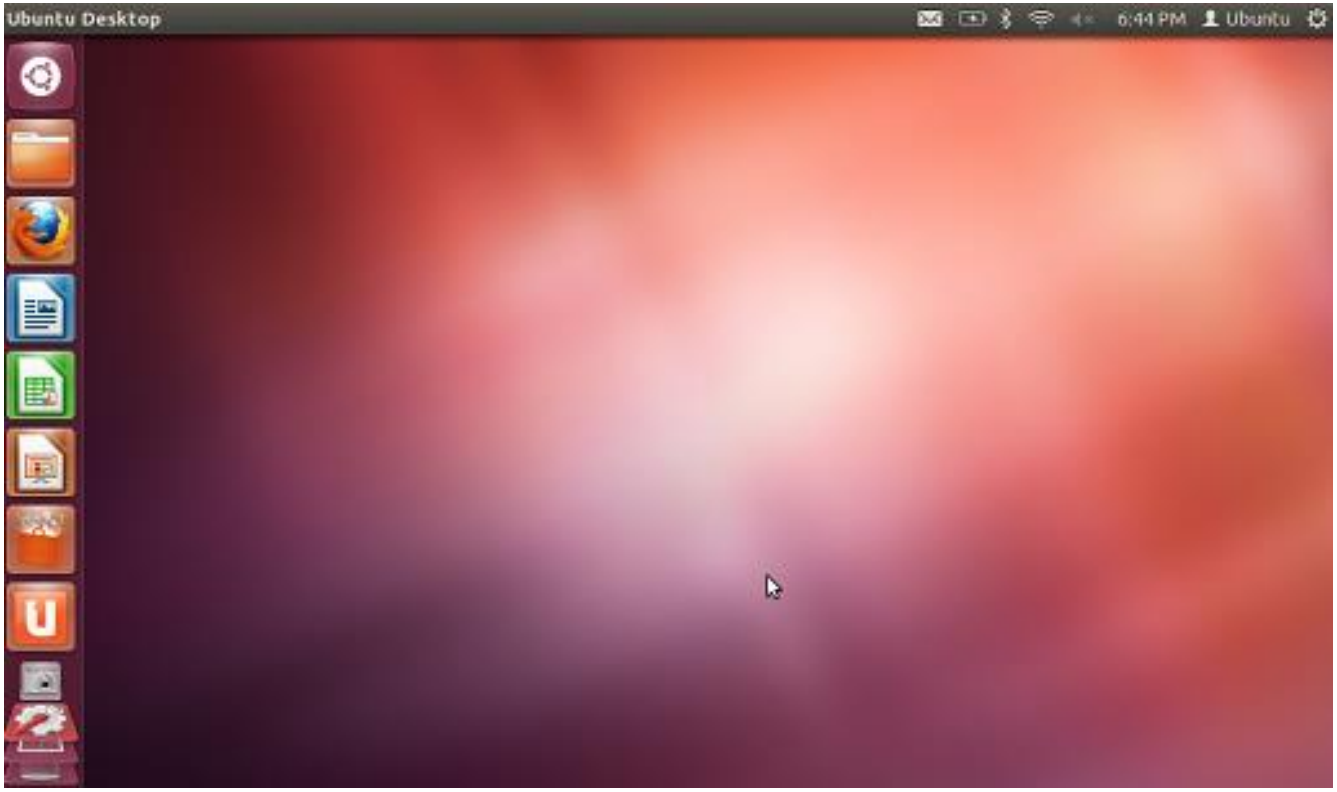
Networking

- Most network operations are not specific to a process: incoming packets are asynchronous events.
- The packets must be collected, identified, and dispatched before a process takes care of them.



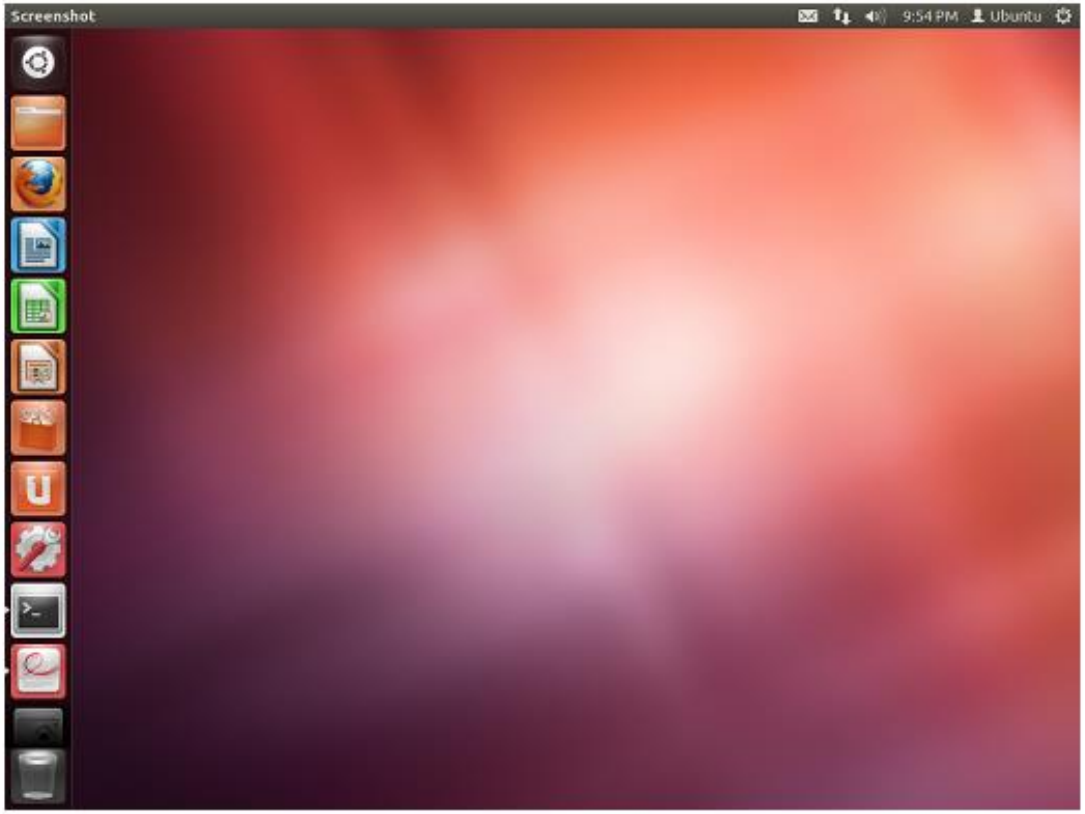
Linux Graphical Environment
Ubuntu Desktop

Ubuntu 12.04 Default Desktop



menu bar

The ubuntu 12.04 launcher on the left with a sample of applications on it



Just below the Home Folder icon, you will see the Firefox icon. Notice the triangle on the right side indicating it is the application in the foreground (on top of all other applications) and the triangle on the left side indicating there's only one window associated with Firefox at this time

The Dash

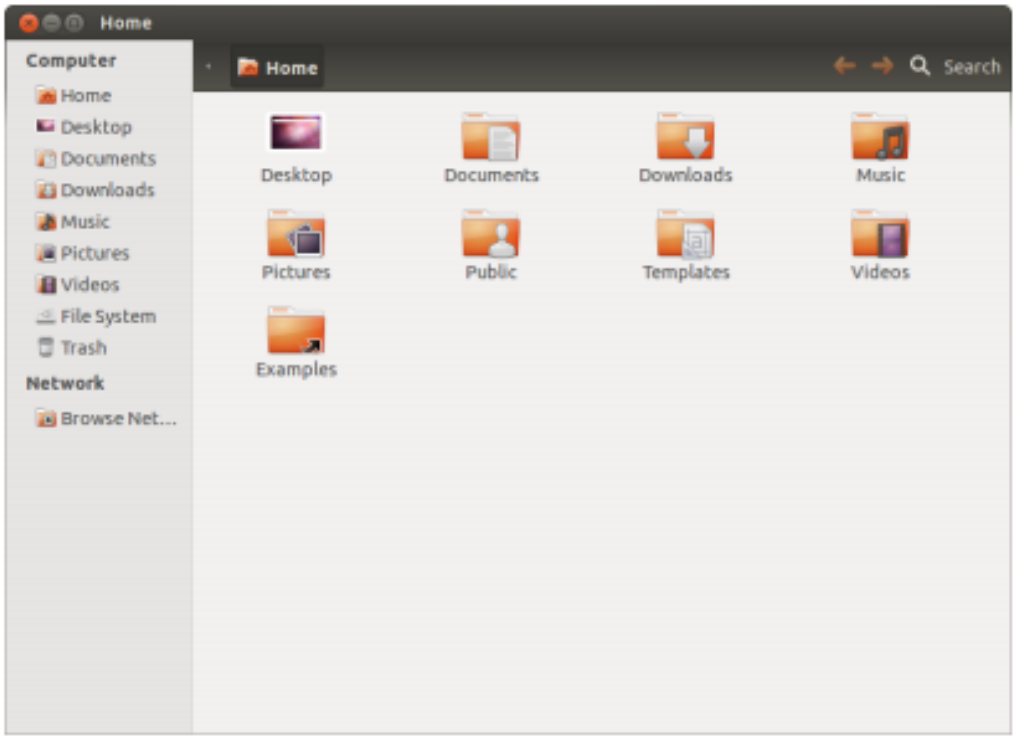

6:48 PM
Ubuntu
Search
Recent Apps
See 2 more results
Screenshot
Image Viewer
Help
Thunderbird Mail
Movie Player
Text Editor
Recent Files
02-1204-desktop.png
PNG
02-indicators.png

You can see the default results when you press Application lens, and also the criteria on the right side.

This is the top bar of a window, named titlebar. The close, minimize, and maximize buttons are on the top-left corner of window.

The workspace switcher on the Launcher

Nautilus file manager displaying your home folder.



You can change most of your system's settings here.

You can change the theme in the Look tab of the “Appearance” window

All The applications you need

- Office suits : writer,calc,math, ….
- Email applications
- Web Browsers: mozila, google chrome, ….
- PDF readers: Adobe reader, ….
- Multimedia players: VLC, Mplayer, ….
- Music players
- CD/DVD burning
- Photo management
- Graphics editors: GIMP, …

? What if you need other applications?

What is package management system?

- A **package management** system is a collection of tools to automate the process of installing, upgrading, configuring, and removing software packages from a computer.
- Packages are distributions of software and metadata such as the software's full name, description of its purpose, version number, vendor, checksum, and a list of dependencies necessary for the software to run properly.
- Upon installation, metadata is stored in a local *package database*.

Package Management System vs. Installer

Package Management System	Installer
Typically part of the operating system.	Each product comes bundled with its own installer.
Uses a single installation database.	Performs its own installation, sometimes recording information about that installation in a registry.
Can verify and manage all packages on the system.	Only works with its bundled product.
Single package management system vendor.	Multiple installer vendors.
Single package format.	Multiple installation formats.

Package Management System Functions

- **Typical functions of a package management system include:**
  - Verifying file checksums to ensure correct and complete packages.
  - Verifying digital signatures to authenticate the origin of packages.
  - Applying file archives to manage encapsulated files.
  - Upgrading software with latest versions, typically from a software repository.
  - Grouping of packages by function to help eliminate user confusion.
  - Managing dependencies to ensure a package is installed with all packages it requires.

Package Formats

- Each package manager relies on the format and metadata of the packages it can manage.
- Often a suit of tools manages the basic installation from these packages and other mangers relies on them (Provide new functionalities).
  - yum relies on rpm as a backend
  - Synaptic Package Manager uses apt library
  - apt relies on dpkg as a backend

Package Manager Past and Present

- Traditional Linux package management systems such as RPM present several problems for users.
  - Dependency problems
  - Finding packages
- Present Linux package management systems such as APT solve the problems
  - Internet-based repository

Examples of package managers

- **RPM Package Manager**
  - The file format *RPM is the baseline package format of the Linux Standard Base.*
- **Advanced Packaging Tool (APT)**
  - APT was originally designed as a front-end for dpkg to work with Debian's .deb packages but it has since been modified to also work with the RPM Package Manager system via apt-rpm.
- **Synaptic**
  - is a GTK+ graphical user interface front-end to the Advanced Packaging Tool

Working with APT package manager

- To search for a package
  - apt-cache search messenger
- Then to see information about that package:
  - apt-cache show pidgin
  - apt-cache depends pidgin
  - apt-cache showpkg pidgin
- To download source a package:
  - apt-get source pidgin
  - apt-get -b source pidgin

Working with APT package manager

- To install a package
  - apt-get install pidgin
- To reinstall a package
  - apt-get --reinstall install pidgin
- To uninstall a package
  - apt-get remove pidgin
  - apt-get --purge remove pidgin
- To update packege list
  - apt-get update

Working with APT package manager

- To upgrade your all packages in your distro
  - apt-get -u upgrade
  - Cache Limit?
    - vi */etc/apt/apt.conf.d/70debconf*
    - *APT::Cache-Limit "10000000";*
- To upgrade your distro
  - apt-get -u dist-upgrade
- To Remove unused package files
  - apt-get clean
    - /var/cache/apt/archives/
    - /var/cache/apt/archives/partial/
  - apt-get autoclean

Working with APT package manager

- apt repositories are specified in:
  - /etc/apt/sources.list
    - deb http://host/debian distribution section1 section2 section3
    - deb-src http://host/debian distribution section1 section2 section3
  - After editing this file you must run:
    - apt-get update

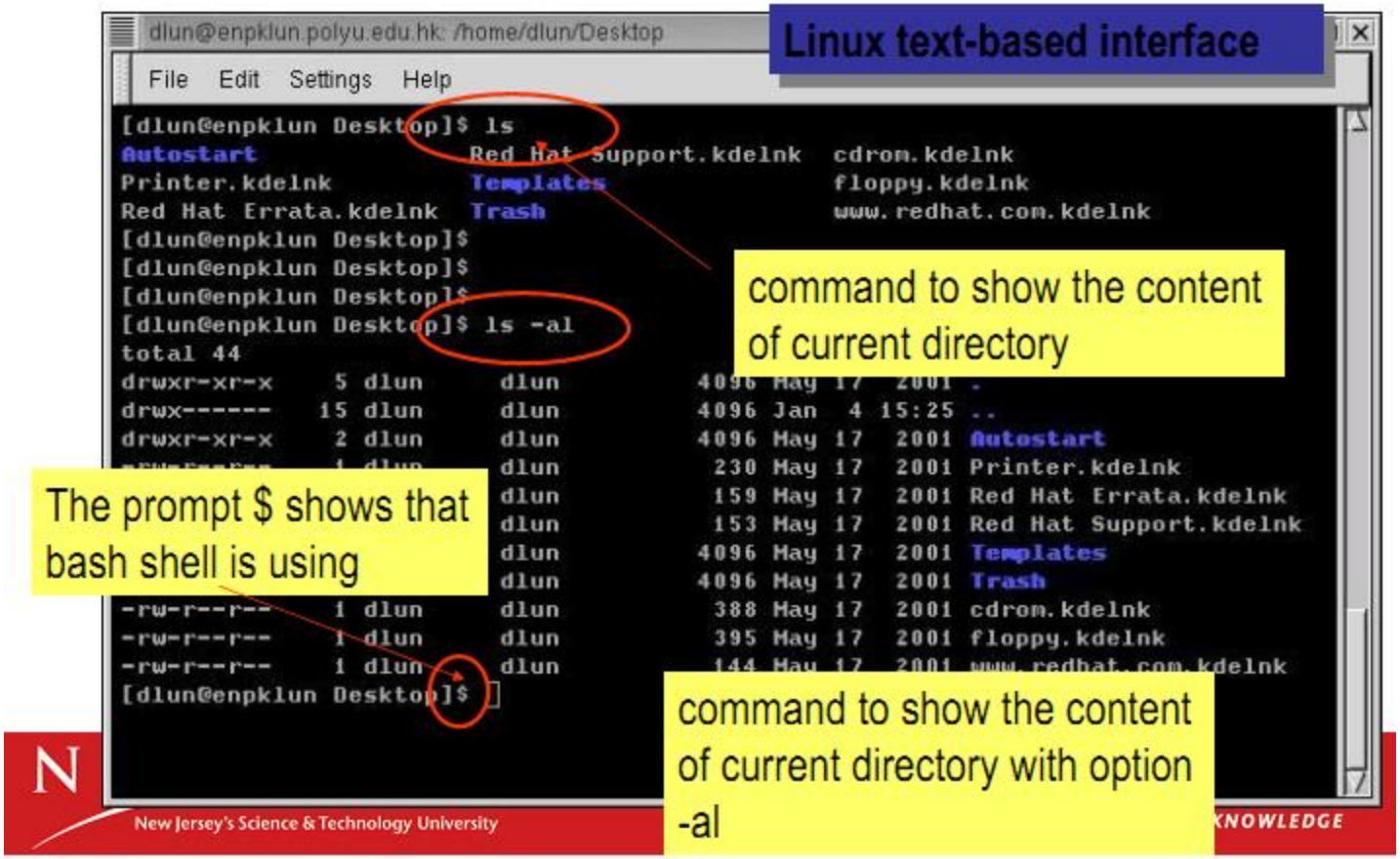
Working with APT package manager

- If an installation breaks in the middle of the process and you find that it's no longer possible to install or remove packages, try :
  - # apt-get -f install
  - # dpkg --configure -a

Ubuntu Software Center Package Management

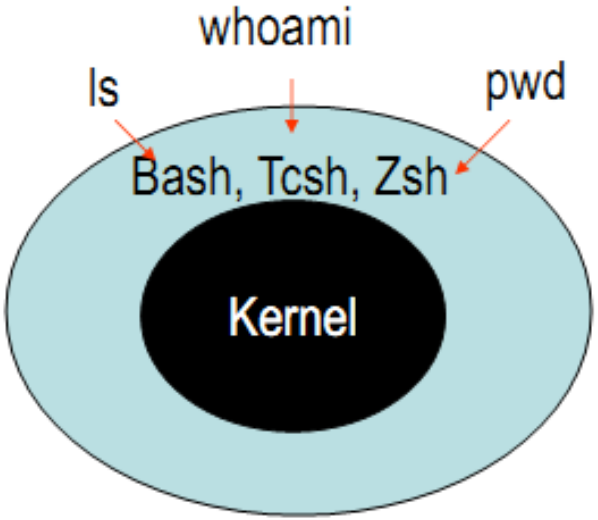
Linux Shell

All LINUX commands start with the name of the command and can be followed by options and arguments.



Linux Shell

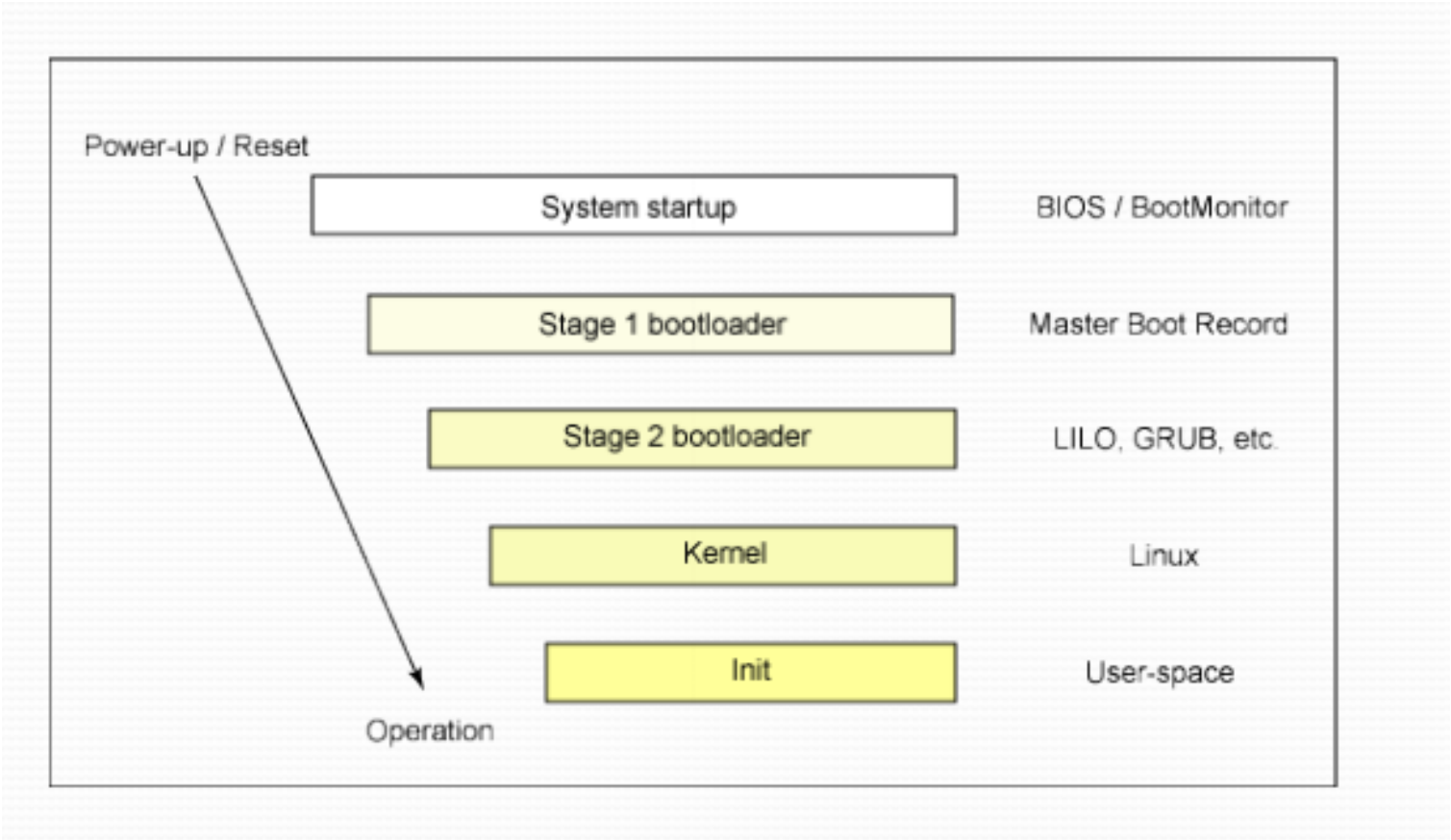
Linux Shell



- Shell interprets the command and request service from kernel
- Similar to DOS but DOS has only one set of interface while Linux can select different shell
  - Bourne Again shell (Bash), TC shell (Tcsh), Z shell (Zsh)
- Different shell has similar but different functionality
- Bash is the default for Linux
- Graphical user interface of Linux is in fact an application program work on the shell

Boot Process

Boot Process Overview



Boot Process

System Startup in PC

System Startup in PC

- booting Linux begins in the BIOS at address oxFFFFo
  - First step: Power-On Self Test (POST)
  - Second step: local device enumeration and initialization
    - Searches for devices that are both active and bootable based on the preferred order which is determined in CMOS
    - If hard disk must be boot(Suppose Linux resides in hard disk)
      - BIOS loads MBR (Master Boot Record) in RAM and yields control to it
      - MBR contains the primary boot loader
      - MBR is a 512-byte sector, located in the first sector on the disk (sector 1 of cylinder 0, head 0)

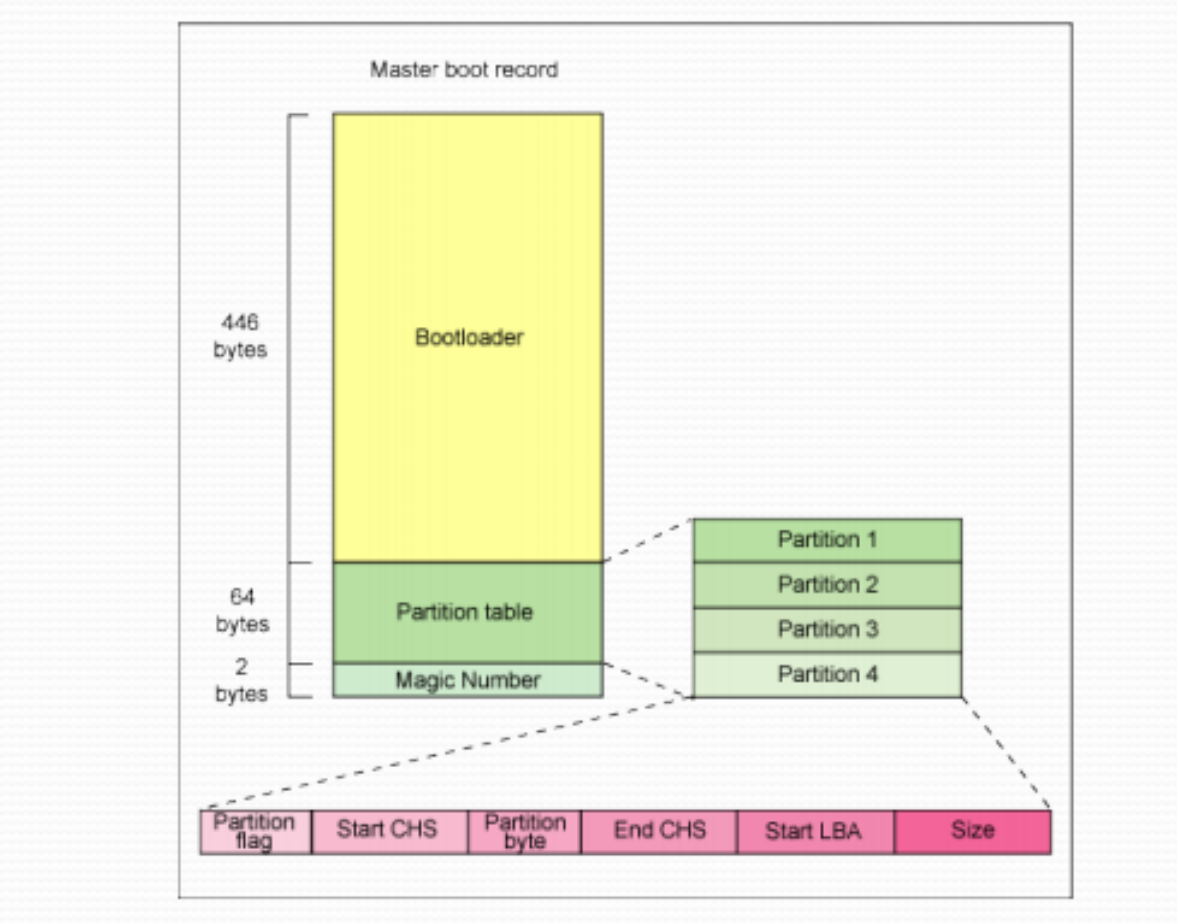
Boot Process

Stage 1 boot loader

- The primary boot loader is a 512-byte image which contains
  - Program Code (first 446 bytes)
  - Partition Table (64 bytes)
  - Magic Number (2 bytes)
  - The job of the primary boot loader is to find and load the secondary boot loader
    - First looks through the partition table for an active partition
    - When it finds an active partition, it scans the remaining partitions in the table to ensure that they're all inactive
    - When this is verified, the active partition's boot record is read from the device into RAM and executed

Boot Process

Stage 1 boot loader



Boot Process

Stage 2 boot loader (Kernel loader)

- The task at this stage is to load the Linux kernel and optional initial RAM disk.
- The first- and second-stage boot loaders combined are called Linux Loader (LILO) or GRand Unified Bootloader (GRUB) in the x86 PC environment.
- The great thing about GRUB is that it includes knowledge of Linux file systems.
  - GRUB can load a Linux kernel from an ext2 or ext3 (and also ext4) file system
  - It does this by making the two-stage boot loader into a three-stage boot loader.
  - Stage 1 (MBR) boots a stage 1.5 boot loader that understands the particular file system containing the Linux kernel image

Boot Process

Kernel loader

- Kernel image is compressed and typically this is
  - a zImage (compressed image, less than 512KB) or
  - a bzImage (big compressed image, greater than 512KB)
- At the head of this kernel image is a routine that does some minimal amount of hardware setup and then decompresses the kernel contained within the kernel image and places it into high memory.
- If an initial RAM disk image is present, this routine moves it into memory and notes it for later use.
- The routine then calls the kernel and the kernel boot begins
- During the boot of the kernel, the initial-RAM disk (initrd) that was loaded into memory by the stage 2 boot loader is copied into RAM and mounted.

Boot Process

Kernel loader

- Let see
  - ls –l /boot
    - vmliuz-2.6.28-11-generic → compressed kernel image
    - initrd.img-2.6.28-11-generic → initial RAM disk
  - ls –l /boot/grub
    - stag1 -> 512 byte -> primary boot loader
    - e2fs_stage1_5 -> ex2, ex3 or ext4 file systems
    - fat_stage1_5 -> fat file systems
    - ...
    - stage2

Boot Process

Init

- After the kernel is booted and initialized, the kernel starts the init (first user-space application).
- System V init process
  - Run levels concept
  - /etc/inittab
- Upstart init process
  - Event-driven init process
  - Ubuntu 6.10 and later
  - Fedora 9.0 and later

Boot Process

System V Init

- Runlevel
  - is a number which indicates what "mode" you want to computer to boot into
  - 0 — Halt
  - 1 — Single-user mode
  - 2 — Not used (user-definable)
  - 3 — Full multi-user mode
  - 4 — Not used (user-definable)
  - 5 — Full multi-user mode (with an X-based login screen)
  - 6 — Reboot

Boot Process

System V Init

- The default runlevel for a system to boot to and stop is configured in /etc/inittab. Like:
  - id:3:initdefault:
- The /etc/init.d directory contains the scripts executed by init at boot time and when the init state (or "runlevel") is changed
- These scripts are referenced by symbolic links in the /etc/rc*n*.d directories
- The names of the links all have the form S*mmscript* or K*mmscript* where *mm* is a two-digit number and *script* is the name of the script (this should be the same as the name of the actual script in /etc/init.d).
- When changing runlevels, init looks in the directory /etc/rc*n*.d for the scripts it should execute, where *n* is the runlevel that is being changed to, or S for the boot-up scripts.

Boot Process

System V Init

- When init changes runlevel first the targets of the links whose names start with a K are executed, each with the single argument stop, followed by the scripts prefixed with an S, each with the single argument start.
- The two-digit number *mm* is used to determine the order in which to run the scripts: low-numbered links have their scripts run first.

Boot Process

System V Init

- the chain of events for a SysV init boot is as follows:
  - The kernel looks in /sbin for init
  - init runs the /etc/rc.d/rc.sysinit script
  - rc.sysinit handles most of the boot loader's processes and then runs rc.serial (if it exists)
  - init runs all the scripts for the default runlevel
  - init runs /etc/rc.d/rc.local

Boot Process

Upstart init

- The Upstart init daemon is event-based and runs specified programs when something on the system changes
  - Instead of starting and stopping services only when the runlevel changes, Upstart can start and stop services upon receiving information that something on the system has changed.
- An *event* is a change in system state that init can be informed of.
  - the boot loader triggers the startup event
  - the system entering runlevel 2 triggers the runlevel 2 event
  - a filesystem being mounted triggers the path-mounted event
  - You can also trigger an event manually by using the initctl emit command.

Boot Process

Upstart init (jobs)

- A *job* is a series of instructions that init reads.
  - A *task* is a job that performs its work and returns to a waiting state when it is done
  - A *service* is a job that does not normally terminate by itself.
  - The /etc/event.d directory holds *job definition files* (files defining the jobs that the Upstart init daemon runs)
  - You can run and stop a job manually using the initctl start and stop commands, respectively.

Boot Process

Upstart init (task example)

- /etc/event.d/testtask

```
start on helloevent
script
        echo "hello world" > /home/test
        date >> /home/test
end script
```

- #initctl emit helloevent

Boot Process

Upstart init (Service example)

- /etc/event.d/testtask

```
start on helloevent

respawn
exec  /bin/service1  > output1.txt
```

- #initctl emit helloevent

Boot Process

References