CS353 Linux Kernel

Chentao Wu吴晨涛
Associate Professor
Dept. of CSE, SJTU
wuct@cs.sjtu.edu.cn

Introduction of Myself

- Dual Ph.D.
  - 2012, Electrical and Computer Engineering, *Virginia Commonwealth University (VCU)*, Richmond, VA, USA
  - 2010, Computer Architecture, *Huazhong University of Science and Technology (HUST)*, Wuhan, China
- Research Interest: Data Storage Systems
  - Storage management for Big Data
  - Cloud storage, Green storage
  - Reliable storage systems (e.g., disk arrays)
  - Semantic file systems (e.g., object-based storage sys.)
  - Cache Algorithms in storage systems

Our Lab

- Leader: Prof. Minyi Guo (Dean of CSE Dept.)
  - http://epcc.sjtu.edu.cn
  - Parallel and Distributed Computing
    - Parallel and Distributed Systems/Networks
    - High Performance Computing
    - Cloud Computing
    - Big Data
  - Welcome to participate in our lab
  - 15+ master students per year, 5+ doctoral students per year

上海交通大學

Download Lectures and Upload Projects

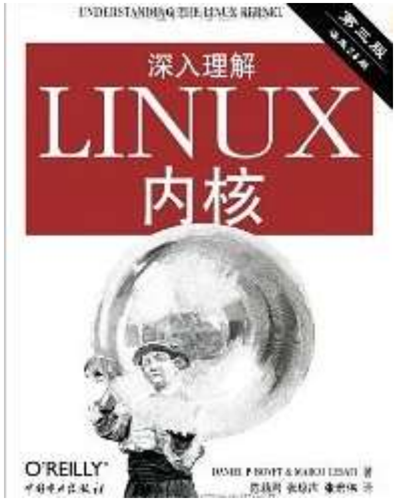
- **ftp://public.sjtu.edu.cn**
- User: wuct
- Password: wuct123456

Teaching Assistant

- Chao Tan谭超
  - Email: 345243921@qq.com
  - Mobile Phone: 15821274485
- Help me to review Projects and the Final Exam

Books

- No textbook
- References
  - Understanding the Linux Kernel
    - 3rd Edition
  - Linux Kernel Drivers
    - 3rd Edition
  - Linux Kernel Development
    - 3rd Edition



Syllabus (1)

- Requirements:
  - Computer Organization, Operating System
  - C/C++ Programming
- Goals: Successful course participants will:
  - Understand C programming in Linux Kernel (Module programming in Linux Kernel).
  - Understand the core concepts of operating systems, including processes, threads, synchronization, virtual memory policies, and file management.

Syllabus (2)

- Goals (contd.)
  - The idea of the course is to learn how computers really work in Linux Kernel, from the chip level up to the application level. When we finish, you will understand what is actually happening when a computer system is running a set of programs, and will be able to make informed choices as a developer, project manager, or system customer.

Course Meeting Times

- Lectures:
  - 2 classes per week (Friday)
  - 2 classes per dual weeks (Virtual Time)
- Questions:
  - Ask me directly between/after the classes
  - Go to my office: SEIEE 3-513
  - Send me an email: wuct@cs.sjtu.edu.cn
  - Ask teaching assistant

Final Grades

- Participation 10%
  - Randomly
- Project 20%
  - Four Projects
  - Source Code and Document
- Report (Reading Kernel Code or Update Experimental Manual) 20%
- Final Exam 50% (Close Book)

Late Policy

- Late Policy: Deadlines will be given in each assignment. These deadlines are strict.
- Typically, project will be given on Fridays, you should submit your homework by the next three weeks.

Cooperation

- This course is established with Intel-Shanghai Corporation.
  - Several Intel engineers will share their experiences on Linux Kernel Programming.

1. The Linux Kernel: Introduction

Chentao Wu
Associate Professor
Dept. of CSE, SJTU
wuct@cs.sjtu.edu.cn

上海交通大學

What’s Linux?

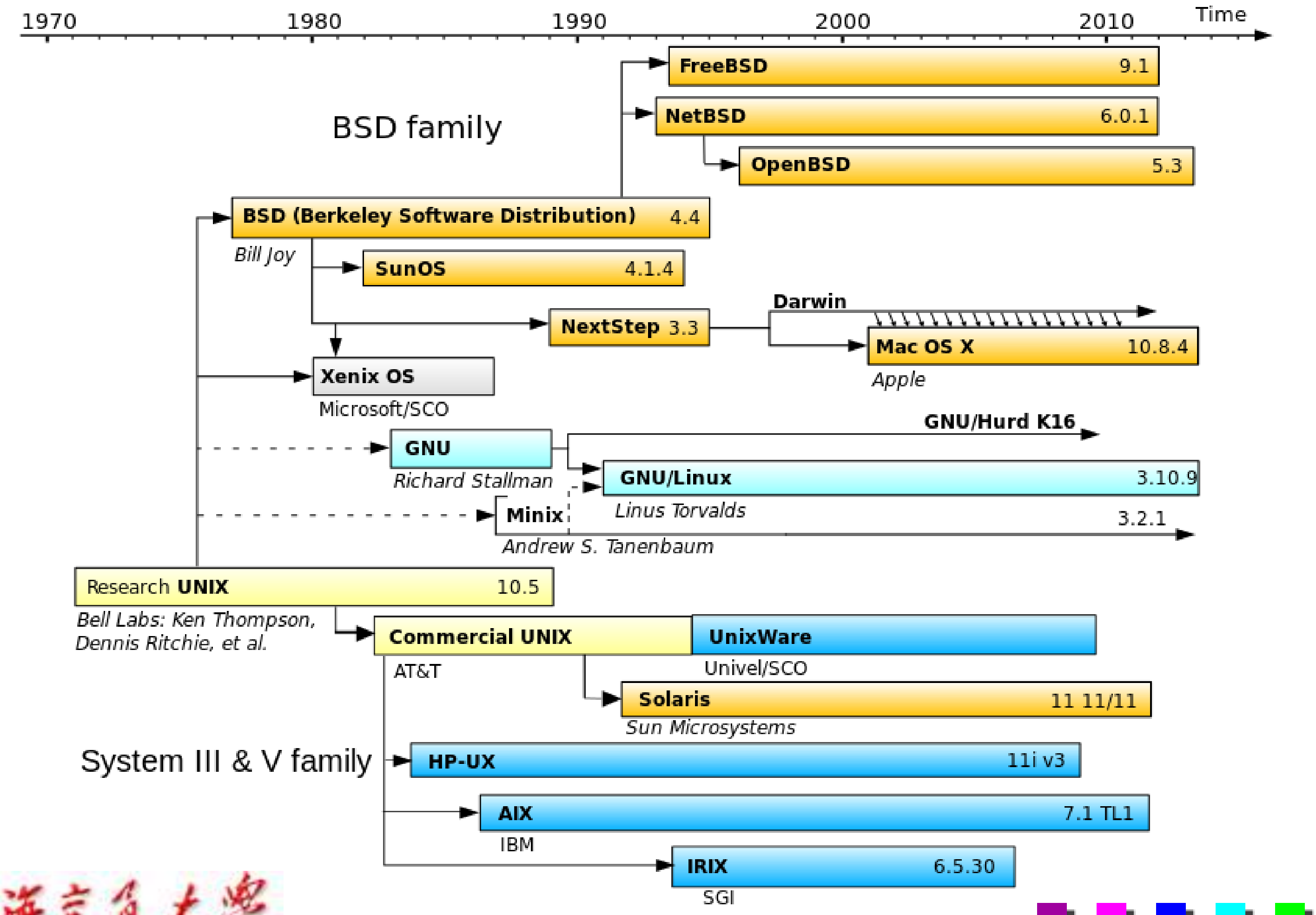
- **Linux** is a *Unix-like* and *POSIX-compliant* computer operating system assembled under the model of free and open source software development and distribution. The defining component of Linux is the **Linux kernel**, an operating system kernel first released on 5 October 1991 by **Linus Torvalds**.

  (from http://www.wikipedia.org)

History

- UNIX: 1969 Thompson & Ritchie AT&T Bell Labs.
- BSD: 1978 Berkeley Software Distribution.
- Commercial Vendors: Sun, HP, IBM, SGI, DEC.
- GNU: 1984 Richard Stallman, FSF.
- POSIX: 1986 IEEE Portable Operating System unIX.
- Minix: 1987 Andy Tannenbaum.
- SVR4: 1989 AT&T and Sun.
- Linux: 1991 Linus Torvalds Intel 386 (i386).
- Open Source: GPL.

UNIX Family

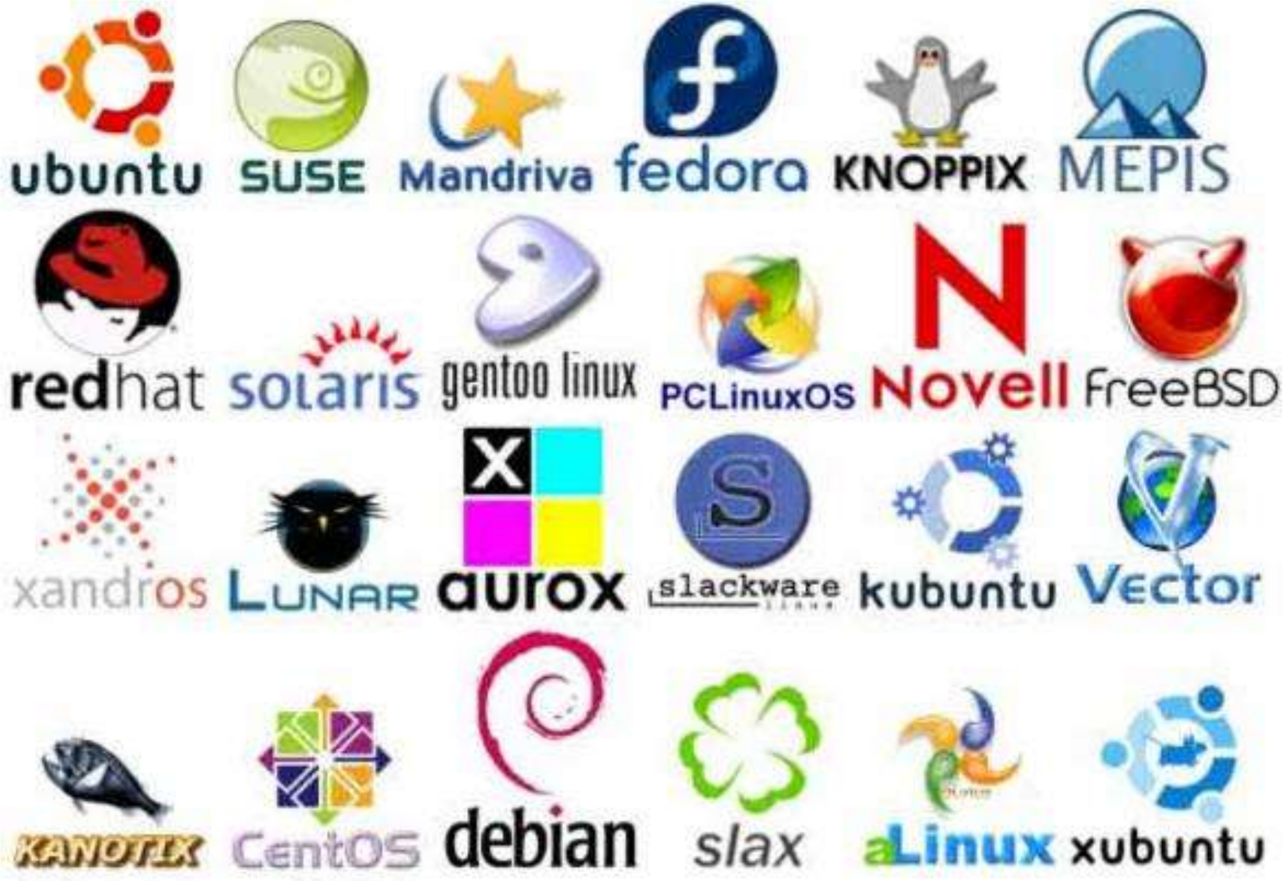


1970
1980
1990
2000
2010
Time
FreeBSD 9.1
NetBSD 6.0.1
OpenBSD 5.3
BSD family
BSD (Berkeley Software Distribution) 4.4
Bill Joy
SunOS 4.1.4
Darwin
NextStep 3.3
Mac OS X 10.8.4
Apple
Xenix OS
Microsoft/SCO
GNU/Hurd K16
GNU
Richard Stallman
GNU/Linux 3.10.9
Linus Torvalds
Minix
3.2.1
Andrew S. Tanenbaum
Research UNIX 10.5
Bell Labs: Ken Thompson, Dennis Ritchie, et al.
Commercial UNIX
AT&T
UnixWare
Univel/SCO
Solaris 11 11/11
Sun Microsystems
System III & V family
HP-UX 11i v3
AIX 7.1 TL1
IBM
IRIX 6.5.30
SGI

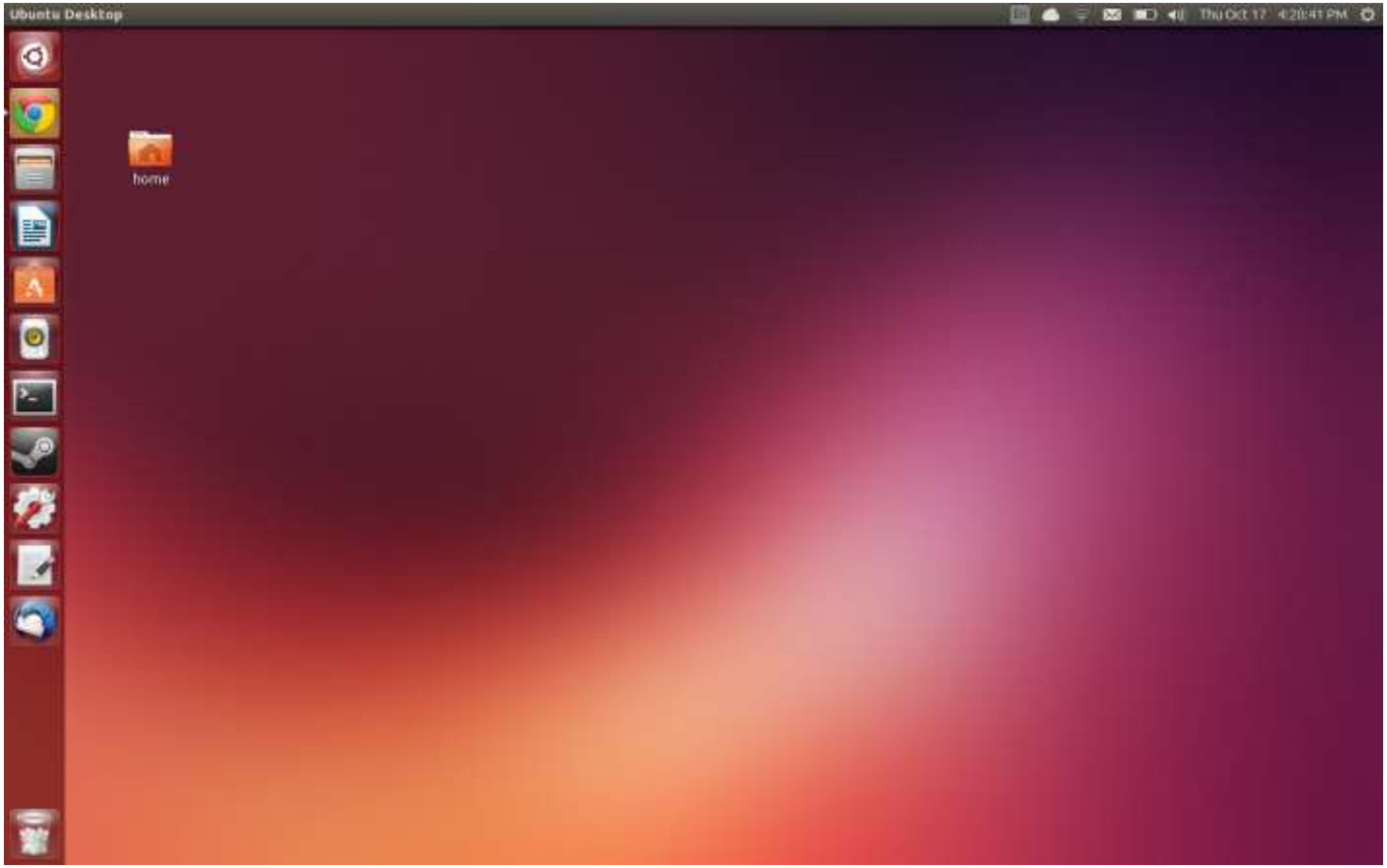
Linux Features

- UNIX-like operating system.
- Features:
  - Preemptive multitasking.
  - Virtual memory (protected memory, paging).
  - Shared libraries.
  - Demand loading, dynamic kernel modules.
  - Shared copy-on-write executables.
  - TCP/IP networking.
  - SMP support.
  - Open source.

Linux Distribution



Ubuntu



Two Modes within Linux

User mode	Application software (bash, LibreOffice, Blender, 0 A.D.)		
	Complex libraries (GLib, GTK+, Qt, SDL, EFL)		Application software
	Complex libraries (GLib, GTK+, Qt)	Simple libraries (*sin, opendbm*)	Application software
	open, exec, sbrk, socket, fopen, calloc C standard library: *glibc* (1,187,911 lines of code) / *uClibc* (342,842 lines of code)		
Kernel mode	System calls: *TRAP, CALL, BRK, INT, IOCTL* (depends on the hardware)		
	Linux kernel (16,223,920 lines of code) (device drivers, process-scheduler, networking stack, file systems) ALSA, DRI, evdev, LVM, device mapper, Linux Process Scheduler, Linux Network Scheduler,Netfilter Linux Security Modules: *SELinux*, *TOMOYO*, *AppArmor*, *Smack*		
	Hardware (CPU(s), Memory, other Microprocessors, Devices etc)		

What’s a Kernel?

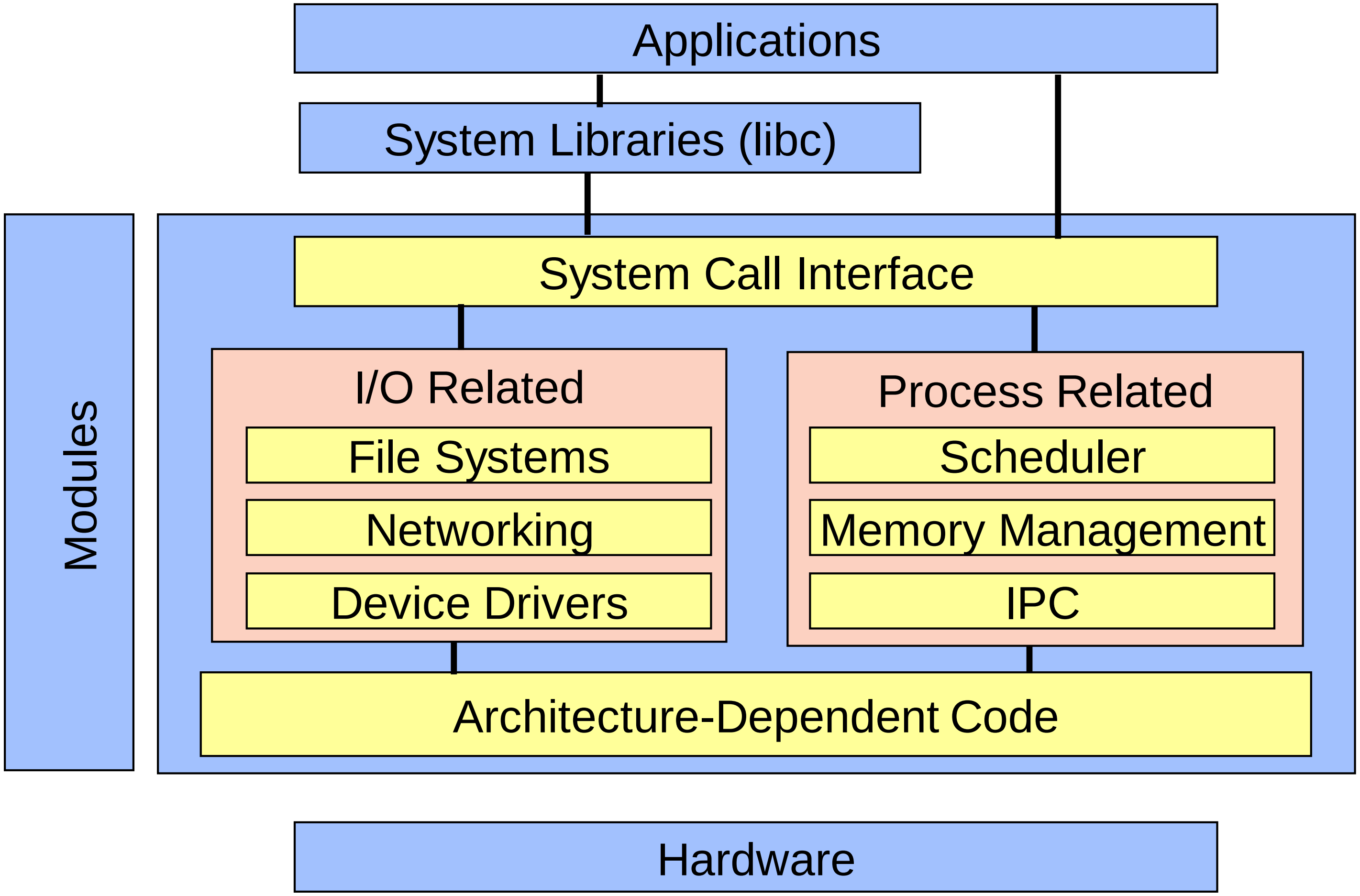
- AKA: executive, system monitor.
- Controls and mediates access to hardware.
- Implements and supports fundamental abstractions:
  - Processes, files, devices etc.
- Schedules / allocates system resources:
  - Memory, CPU, disk, descriptors, etc.
- Enforces security and protection.
- Responds to user requests for service (system calls).
- Etc…etc…

Kernel Design Goals

- Performance: efficiency, speed.
  - Utilize resources to capacity with low overhead.
- Stability: robustness, resilience.
  - Uptime, graceful degradation.
- Capability: features, flexibility, compatibility.
- Security, protection.
  - Protect users from each other & system from bad users.
- Portability.
- Extensibility.

上海交通大學

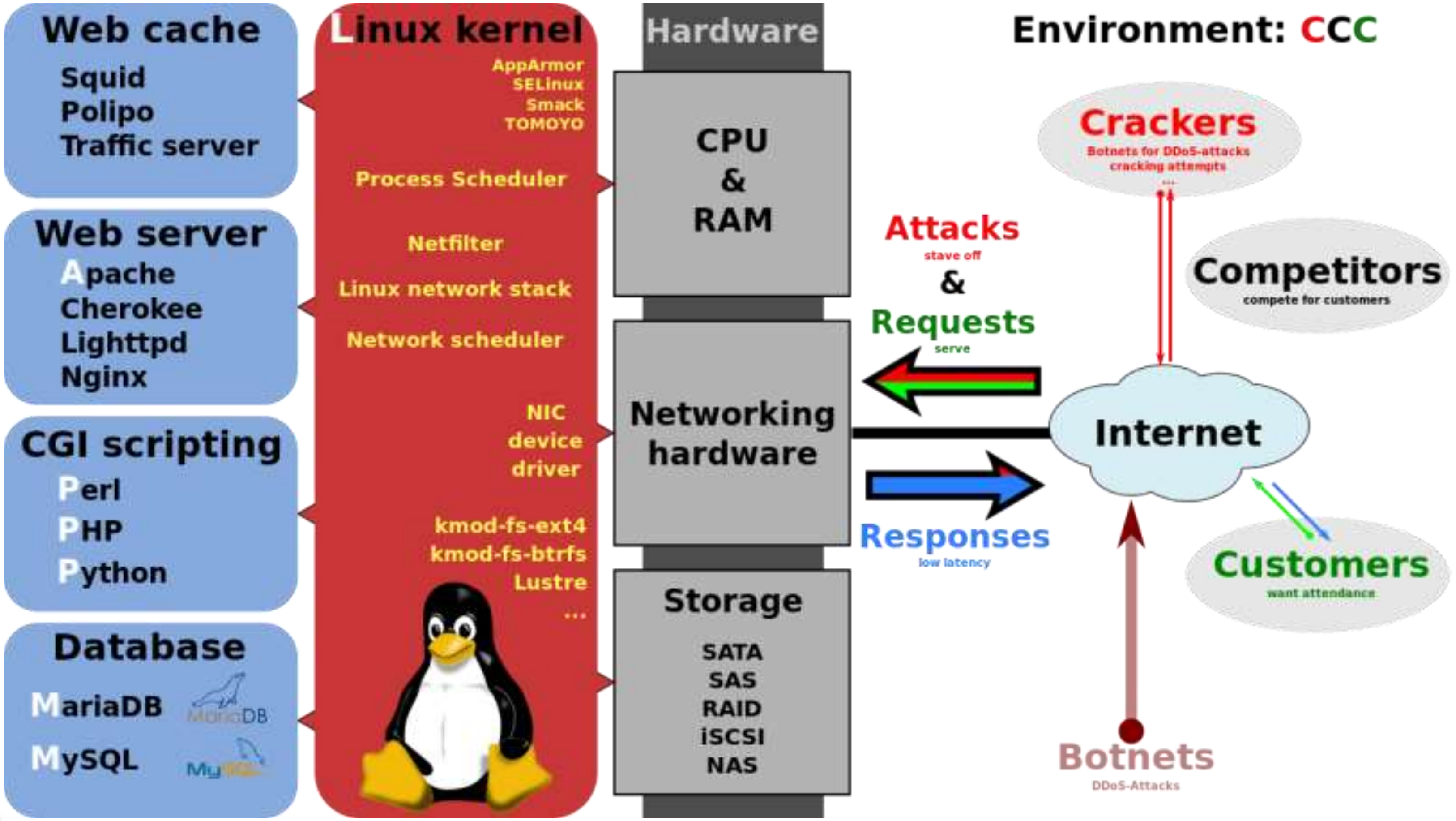
Example “Core” Kernel



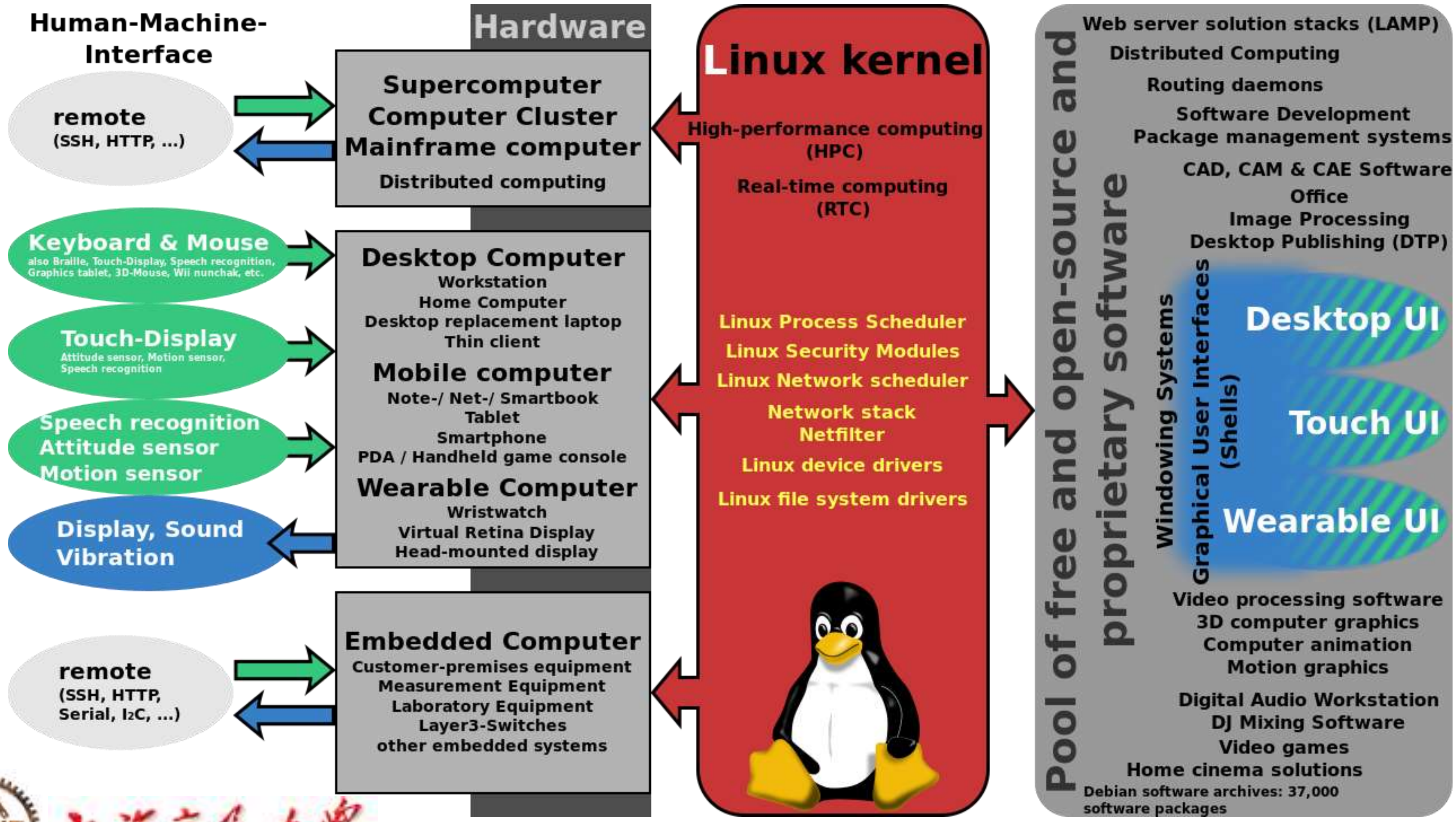
Architectural Approaches

- Monolithic.
- Layered.
- Modularized.
- Micro-kernel.
- Virtual machine.

Board View of Linux Kernel



Various Types of Hardware Are Running on Linux



Human-Machine-Interface

remote
(SSH, HTTP, ...)

Keyboard & Mouse
also Braille, Touch-Display, Speech recognition, Graphics tablet, 3D-Mouse, Wii nunchak, etc.

Touch-Display
Attitude sensor, Motion sensor, Speech recognition

Speech recognition
Attitude sensor
Motion sensor

Display, Sound
Vibration

remote
(SSH, HTTP, Serial, I$_2$C, ...)

Hardware

Supercomputer
Computer Cluster
Mainframe computer
Distributed computing

Desktop Computer
Workstation
Home Computer
Desktop replacement laptop
Thin client

Mobile computer
Note-/ Net-/ Smartbook
Tablet
Smartphone
PDA / Handheld game console

Wearable Computer
Wristwatch
Virtual Retina Display
Head-mounted display

Embedded Computer
Customer-premises equipment
Measurement Equipment
Laboratory Equipment
Layer3-Switches
other embedded systems

Linux kernel

High-performance computing (HPC)

Real-time computing (RTC)

Linux Process Scheduler
Linux Security Modules
Linux Network scheduler
Network stack
Netfilter
Linux device drivers
Linux file system drivers

Pool of free and open-source and proprietary software

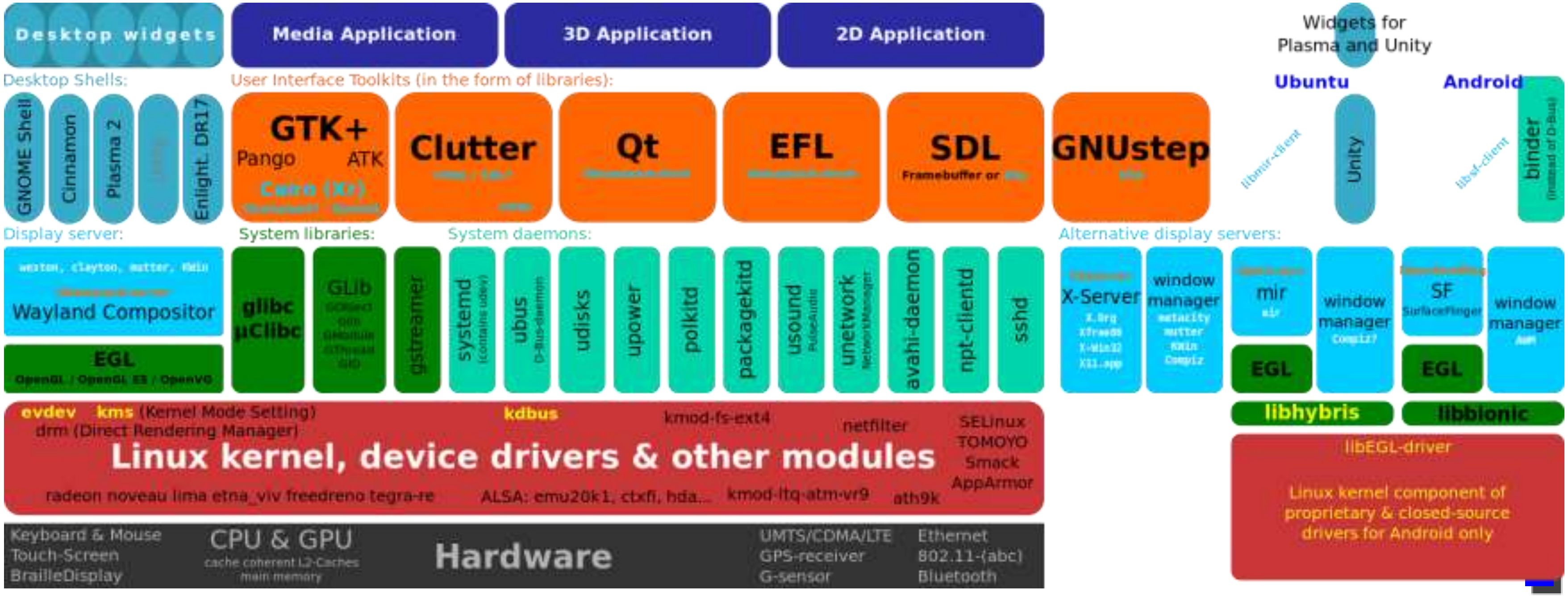
Web server solution stacks (LAMP)
Distributed Computing
Routing daemons
Software Development
Package management systems
CAD, CAM & CAE Software
Office
Image Processing
Desktop Publishing (DTP)

Windowing Systems
Graphical User Interfaces (Shells)

Desktop UI
Touch UI
Wearable UI

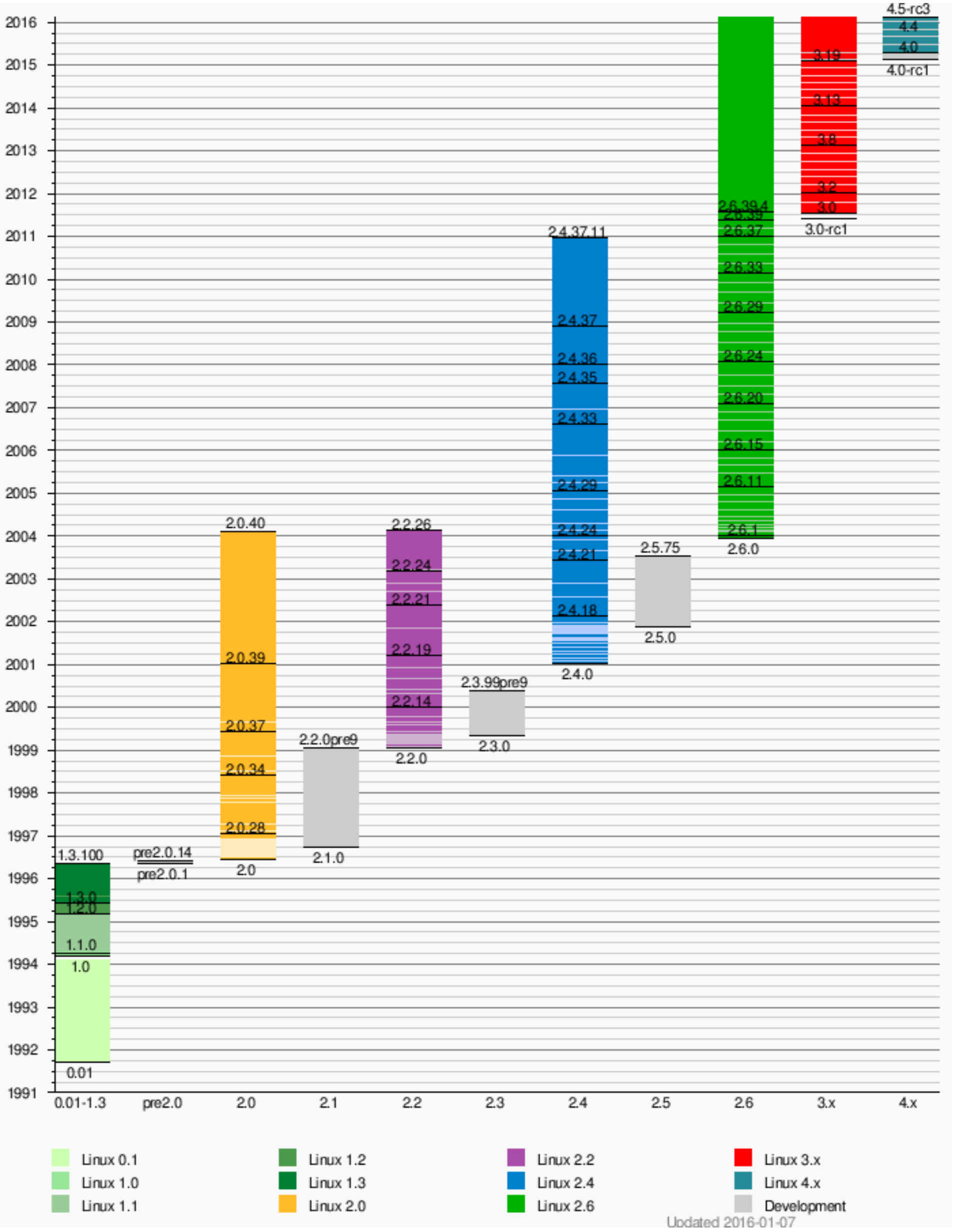
Video processing software
3D computer graphics
Computer animation
Motion graphics
Digital Audio Workstation
DJ Mixing Software
Video games
Home cinema solutions
Debian software archives: 37,000 software packages

Software Components of Linux Desktop

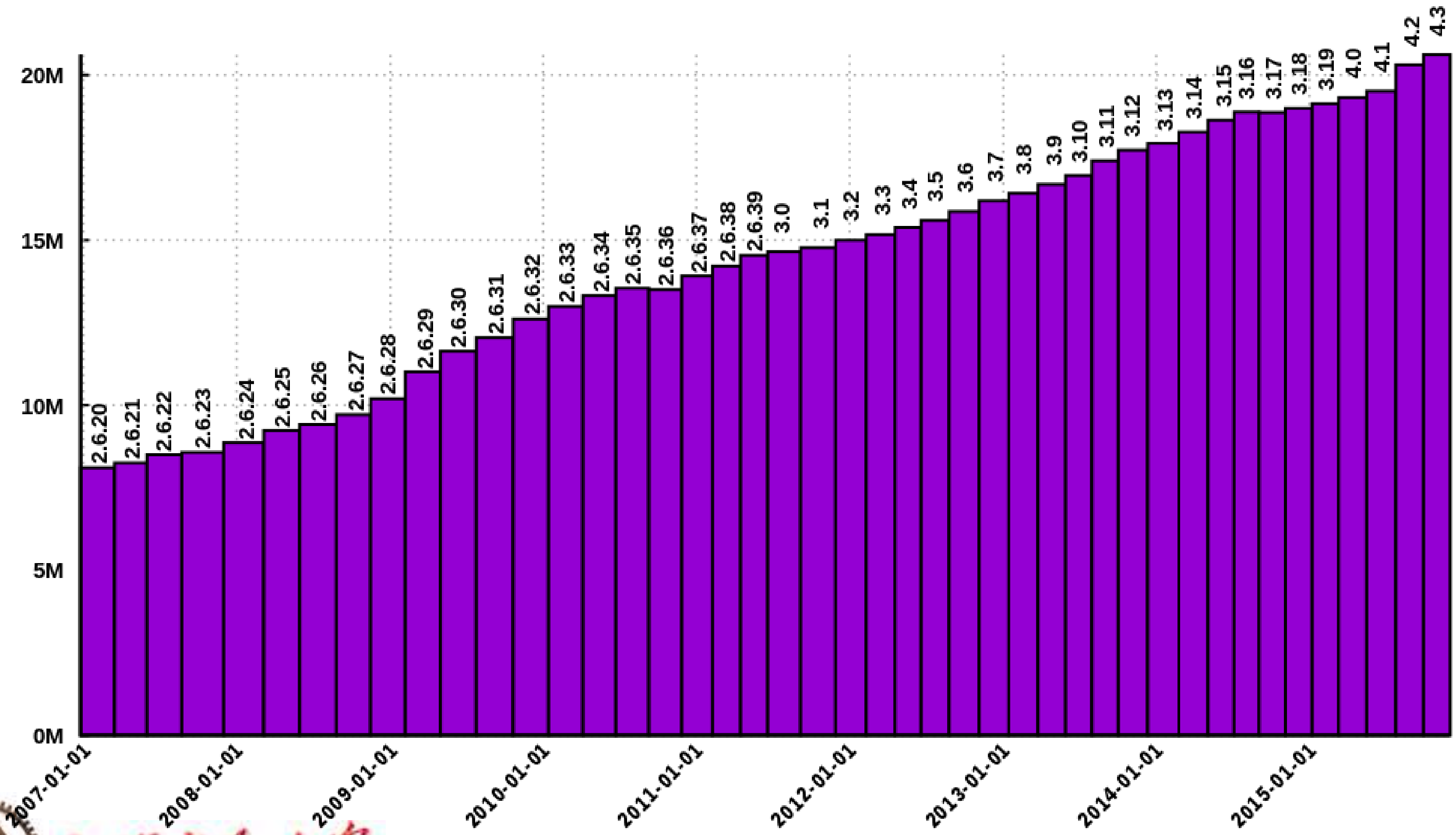


History of Linux Kernel Versions

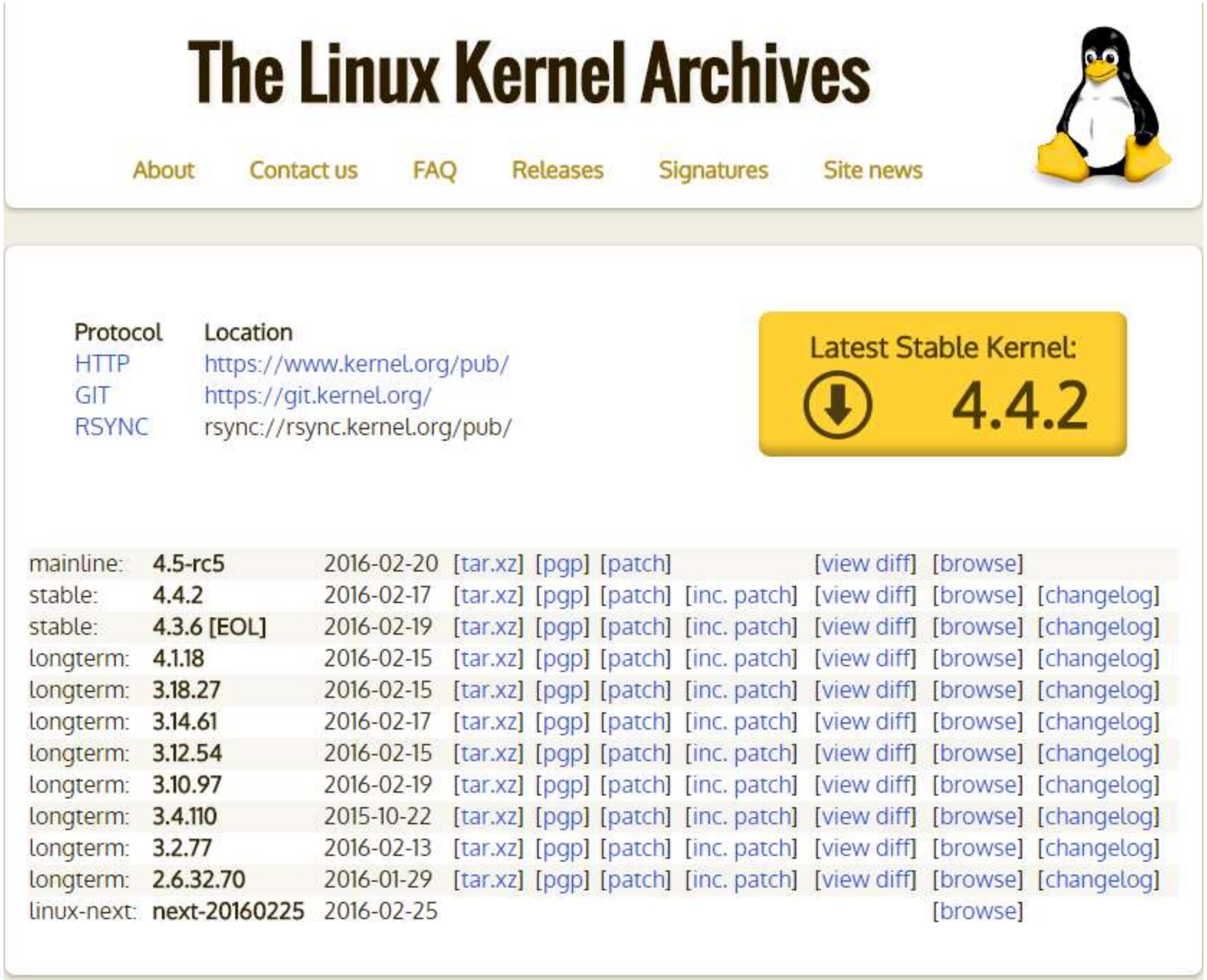
- 0.01 -- 17 Sep. 1991
- 1.0 -- 14 Mar. 1994
- 2.0 -- 9 June 1996
- 2.4 -- 4 Jan. 2001
- 2.6 -- 18 Dec. 2003
- 3.0 -- 21 July 2011
- 4.0 -- 12 Apr. 2015
- 4.5 – 21 Feb. 2016



Lines of Source Code in Linux



kernel.org



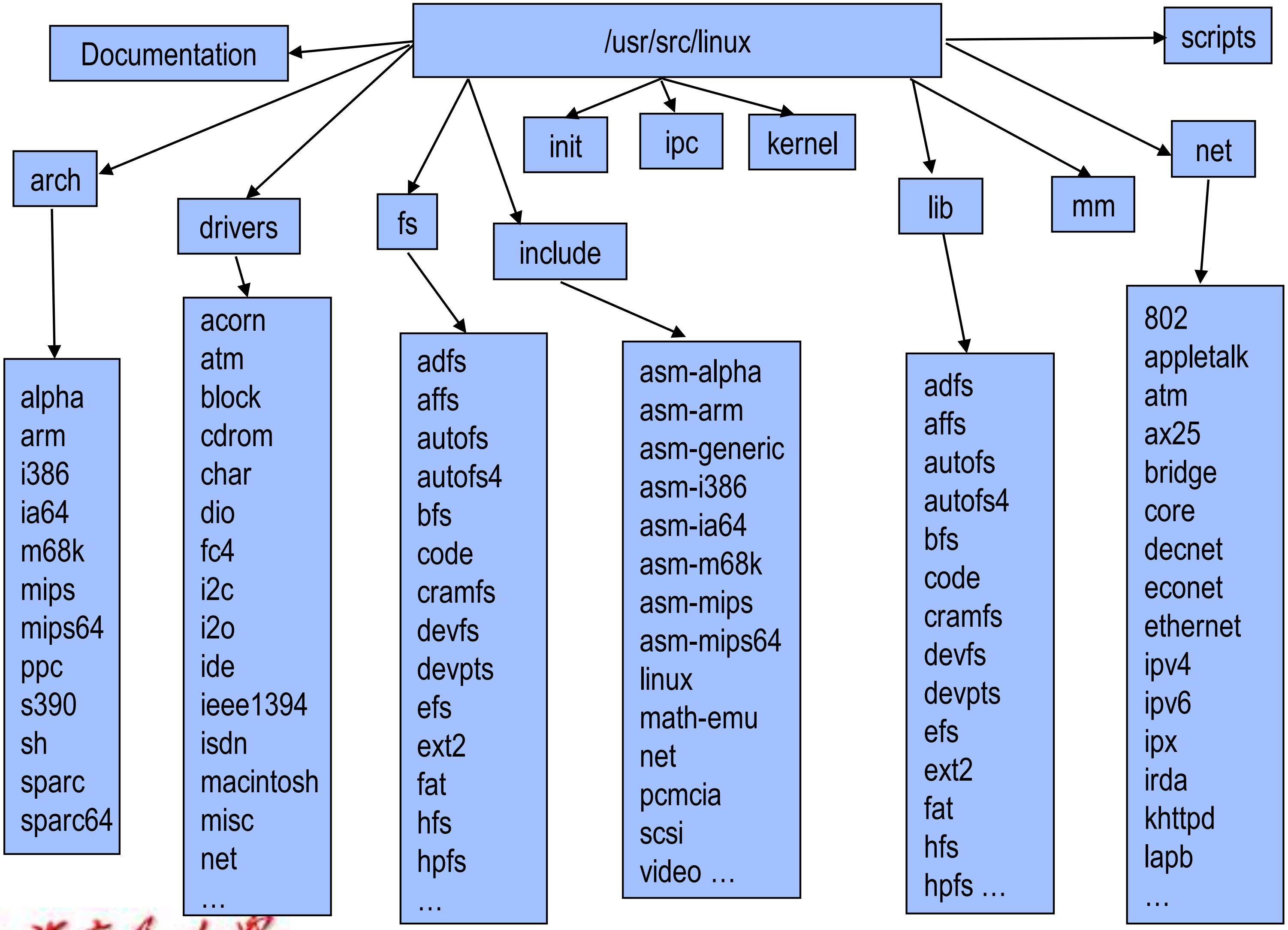
The Linux Kernel Archives

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel:
4.4.2

mainline:	4.5-rc5	2016-02-20	[tar.xz]	[pgp]	[patch]		[view diff]	[browse]	
stable:	4.4.2	2016-02-17	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
stable:	4.3.6 [EOL]	2016-02-19	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.1.18	2016-02-15	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.18.27	2016-02-15	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.14.61	2016-02-17	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.12.54	2016-02-15	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.10.97	2016-02-19	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.4.110	2015-10-22	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.2.77	2016-02-13	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	2.6.32.70	2016-01-29	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20160225	2016-02-25						[browse]	

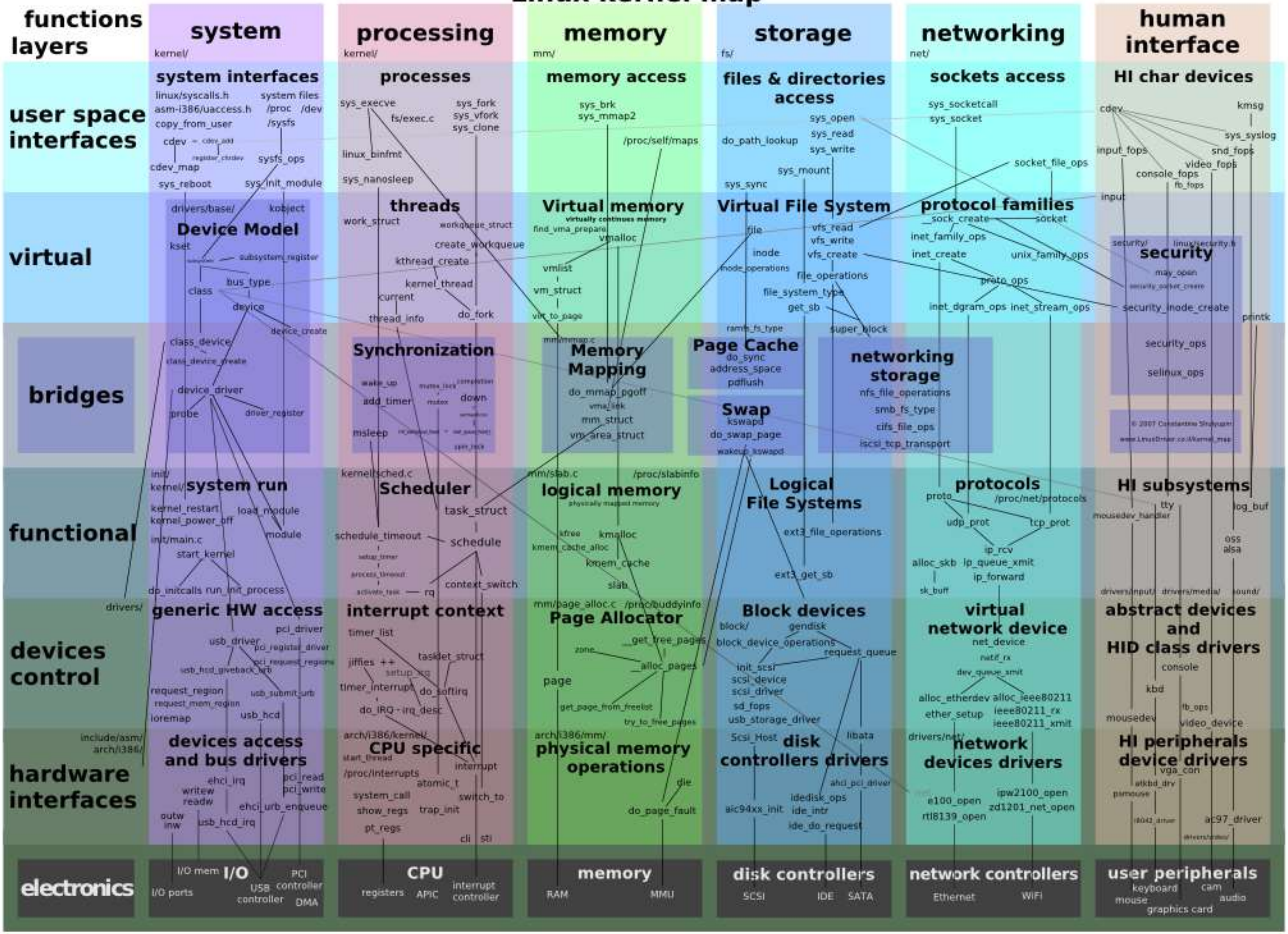
Linux Source Tree Layout (2.6 or earlier)



/usr/src/linux

- Documentation
- scripts
- init
- ipc
- kernel
- arch
  - alpha
  - arm
  - i386
  - ia64
  - m68k
  - mips
  - mips64
  - ppc
  - s390
  - sh
  - sparc
  - sparc64
- drivers
  - acorn
  - atm
  - block
  - cdrom
  - char
  - dio
  - fc4
  - i2c
  - i2o
  - ide
  - ieee1394
  - isdn
  - macintosh
  - misc
  - net
  - …
- fs
  - adfs
  - affs
  - autofs
  - autofs4
  - bfs
  - code
  - cramfs
  - devfs
  - devpts
  - efs
  - ext2
  - fat
  - hfs
  - hpfs
  - …
- include
  - asm-alpha
  - asm-arm
  - asm-generic
  - asm-i386
  - asm-ia64
  - asm-m68k
  - asm-mips
  - asm-mips64
  - linux
  - math-emu
  - net
  - pcmcia
  - scsi
  - video …
- lib
  - adfs
  - affs
  - autofs
  - autofs4
  - bfs
  - code
  - cramfs
  - devfs
  - devpts
  - efs
  - ext2
  - fat
  - hfs
  - hpfs …
- mm
- net
  - 802
  - appletalk
  - atm
  - ax25
  - bridge
  - core
  - decnet
  - econet
  - ethernet
  - ipv4
  - ipv6
  - ipx
  - irda
  - khttpd
  - lapb
  - …

Linux kernel map



Linux Source Tree (1)

- All directories are grouped under the root entry "/"
- **root** - The home directory for the root user
- **home** - Contains the user's home directories along with directories for services
  - ftp
  - HTTP
  - samba

Linux Source Tree (2)

- **bin** - Commands needed during booting up that might be needed by normal users
- **sbin** - Like bin but commands are not intended for normal users. Commands run by LINUX.
- **proc** - This filesystem is not on a disk. It is a virtual filesystem that exists in the kernels imagination which is memory
  - 1 - A directory with info about process number 1. Each process has a directory below proc.

Linux Source Tree (3)

- **usr** - Contains all commands, libraries, man pages, games and static files for normal operation.
  - **bin** - Almost all user commands. some commands are in /bin or /usr/local/bin.
  - **sbin** - System admin commands not needed on the root filesystem. e.g., most server programs.
  - **include** - Header files for the C programming language. Should be below /user/lib for consistency.
  - **lib** - Unchanging data files for programs and subsystems
  - **local** - The place for locally installed software and other files.
  - **man** - Manual pages
  - **info** - Info documents
  - **doc** - Documentation
  - **tmp**
  - **X11R6** - The X windows system files. There is a directory similar to usr below this directory.
  - **X386** - Like X11R6 but for X11 release 5

Linux Source Tree (4)

- **boot** - Files used by the bootstrap loader. Kernel images are often kept here.
- **lib** - Shared libraries needed by the programs on the root filesystem
- **modules** - Loadable kernel modules, especially those needed to boot the system after disasters.
- **dev** - Device files
- **etc** - Configuration files specific to the machine.
- **skel** - When a home directory is created it is initialized with files from this directory
- **sysconfig** - Files that configure the linux system for devices.

Linux Source Tree (5)

- **var** - Contains files that change for mail, news, printers log files, man pages, temp files
  - **file**
  - **lib** - Files that change while the system is running normally
  - **local** - Variable data for programs installed in /usr/local.
  - **lock** - Lock files. Used by a program to indicate it is using a particular device or file
  - **log** - Log files from programs such as login and syslog which logs all logins and logouts.
  - **run** - Files that contain information about the system that is valid until the system is next booted
  - **spool** - Directories for mail, printer spools, news and other spooled work.
  - **tmp** - Temporary files that are large or need to exist for longer than they should in /tmp.
  - **catman** - A cache for man pages that are formatted on demand

Linux Source Tree (6)

- **mnt** - Mount points for temporary mounts by the system administrator.
- **tmp** - Temporary files. Programs running after bootup should use /var/tmp

linux/arch

- Subdirectories for each current port.
- Each contains **kernel**, **lib**, **mm**, **boot** and other directories whose contents override code stubs in architecture independent code.
- **lib** contains highly-optimized common utility routines such as memcpy, checksums, etc.
- **arch** as of 2.4:
  - alpha, arm, i386, ia64, m68k, mips, mips64.
  - ppc, s390, sh, sparc, sparc64.

linux/drivers

- Largest amount of code in the kernel tree (~1.5M).
- device, bus, platform and general directories.
- drivers/char – n_tty.c is the default line discipline.
- drivers/block – elevator.c, genhd.c, linear.c, ll_rw_blk.c, raidN.c.
- drivers/net –specific drivers and general routines Space.c and net_init.c.
- drivers/scsi – scsi_*.c files are generic; sd.c (disk), sr.c (CD-ROM), st.c (tape), sg.c (generic).
- General:
  - cdrom, ide, isdn, parport, pcmcia, pnp, sound, telephony, video.
- Buses – fc4, i2c, nubus, pci, sbus, tc, usb.
- Platforms – acorn, macintosh, s390, sgi.

linux/fs

- Contains:
  - virtual filesystem (VFS) framework.
  - subdirectories for actual filesystems.
- vfs-related files:
  - exec.c, binfmt_*.c - files for mapping new process images.
  - devices.c, blk_dev.c – device registration, block device support.
  - super.c, filesystems.c.
  - inode.c, dcache.c, namei.c, buffer.c, file_table.c.
  - open.c, read_write.c, select.c, pipe.c, fifo.c.
  - fcntl.c, ioctl.c, locks.c, dquot.c, stat.c.

linux/include

- include/asm-*:
  - Architecture-dependent include subdirectories.
- include/linux:
  - Header info needed both by the kernel and user apps.
  - Usually linked to /usr/include/linux.
  - Kernel-only portions guarded by #ifdefs
    - #ifdef __KERNEL__
    - /* kernel stuff */
    - #endif
- Other directories:
  - math-emu, net, pcmcia, scsi, video.

linux/init

- Just two files: version.c, main.c.
- version.c – contains the version banner that prints at boot.
- main.c – architecture-independent boot code.
- start_kernel is the primary entry point.

linux/ipc

- System V IPC facilities.
- If disabled at compile-time, util.c exports stubs that simply return –ENOSYS.
- One file for each facility:
  - sem.c – semaphores.
  - shm.c – shared memory.
  - msg.c – message queues.

linux/kernel

- The core kernel code.
- sched.c – “the main kernel file”:
  - scheduler, wait queues, timers, alarms, task queues.
- Process control:
  - fork.c, exec.c, signal.c, exit.c etc…
- Kernel module support:
  - kmod.c, ksyms.c, module.c.
- Other operations:
  - time.c, resource.c, dma.c, softirq.c, itimer.c.
  - printk.c, info.c, panic.c, sysctl.c, sys.c.

linux/lib

- kernel code cannot call standard C library routines.
- Files:
  - brlock.c – “Big Reader” spinlocks.
  - cmdline.c – kernel command line parsing routines.
  - errno.c – global definition of errno.
  - inflate.c – “gunzip” part of gzip.c used during boot.
  - string.c – portable string code.
    - Usually replaced by optimized, architecture-dependent routines.
  - vsprintf.c – libc replacement.

linux/mm

- Paging and swapping:
  - swap.c, swapfile.c (paging devices), swap_state.c (cache).
  - vmscan.c – paging policies, kswapd.
  - page_io.c – low-level page transfer.
- Allocation and deallocation:
  - slab.c – slab allocator.
  - page_alloc.c – page-based allocator.
  - vmalloc.c – kernel virtual-memory allocator.
- Memory mapping:
  - memory.c – paging, fault-handling, page table code.
  - filemap.c – file mapping.
  - mmap.c, mremap.c, mlock.c, mprotect.c.

linux/scripts

- Scripts for:
  - Menu-based kernel configuration.
  - Kernel patching.
  - Generating kernel documentation.

Summary

- Linux is a modular, UNIX-like monolithic kernel.
- Kernel is the heart of the OS that executes with special hardware permission (kernel mode).
- “Core kernel” provides framework, data structures, support for drivers, modules, subsystems.
- Architecture dependent source sub-trees live in /arch.

Project 1:
Compile the Linux Kernel

Preparation

- For each student
  - Prepare one laptop for each group
  - Install Virtual Machine Software: **VMware Workstation** or **VirtualBox**
  - Install the Linux Operating System in a virtual machine: **RedHat Enterprise Linux (RHEL), Fedora**, Gentoo, **Ubuntu**, etc.
  - Linux kernel version: 4.3 or earlier

Compile The Linux Kernel (1)

- Download the latest kernel version (4.4.2, updated on 2016-02-17) from **http://www.kernel.org**
  - Download the complete source code (not a patch or change log)
- Copy to a directory
  - **4.X.X is the kernel version number**

```
# cp linux-4.X.X.tar.gz  /usr/src
# cd /usr/src
```

- Extract the kernel

```
# tar –xzvf linux-4.X.X.tar.gz  or
# tar –xjvf linux-4.X.X.tar.bz2
```

Compile The Linux Kernel (2)

- Preparation

```
# cd /usr/src/linux
# make clean
# make mrproper
```

Note: make clean and make mrproper can be passed with the first compilation.

Compile The Linux Kernel (3)

- Configure the kernel

# make menuconfig

/*creates a menu where you can browse options on what the kernel supports*/

# make xconfig

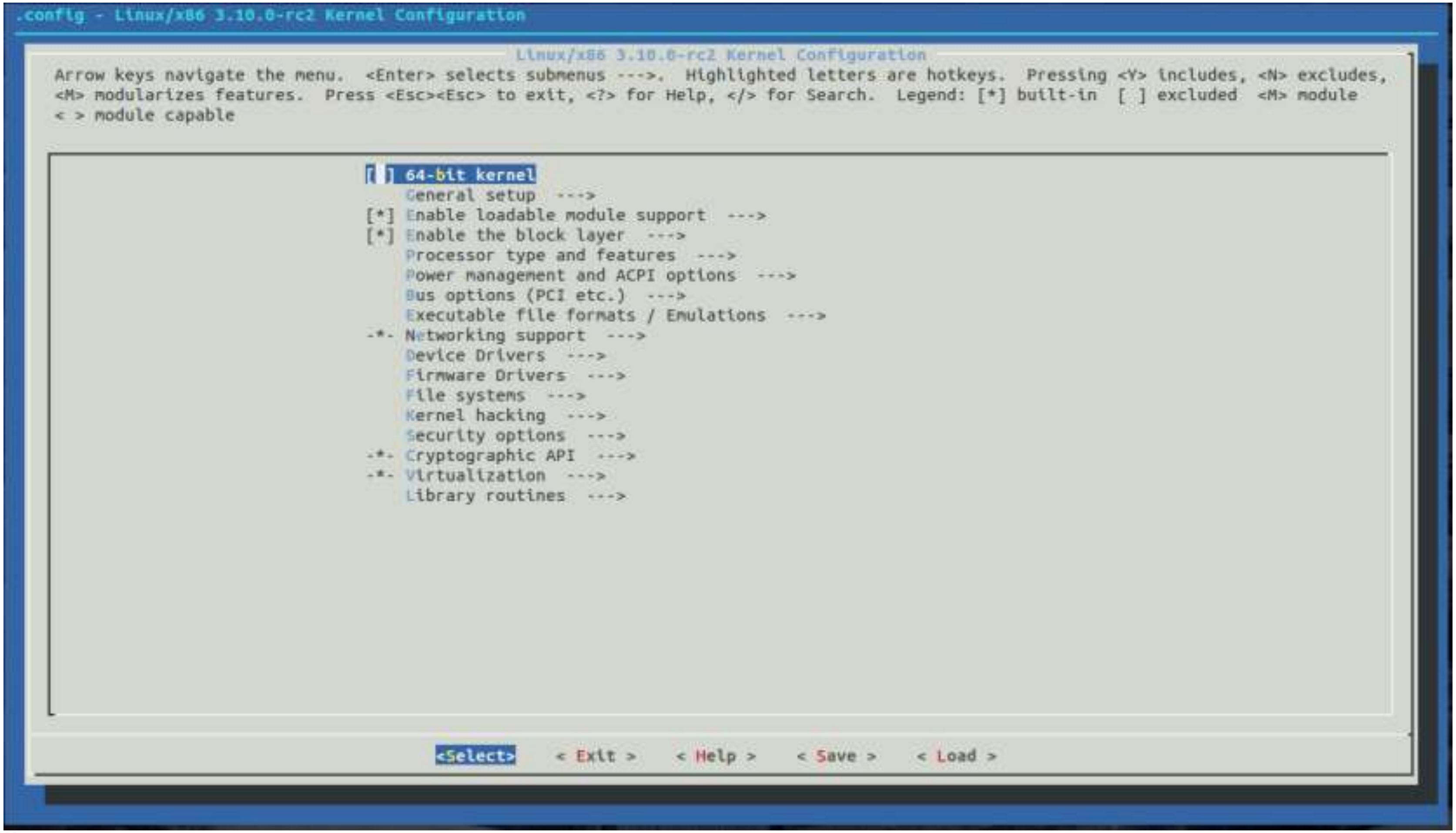
/*same as menuconfig, except that now the configuration menu is graphics based**/**

# make oldconfig

/*minor revision on previous kernel*/

# make config /*not be recommended*/

Make menuconfig



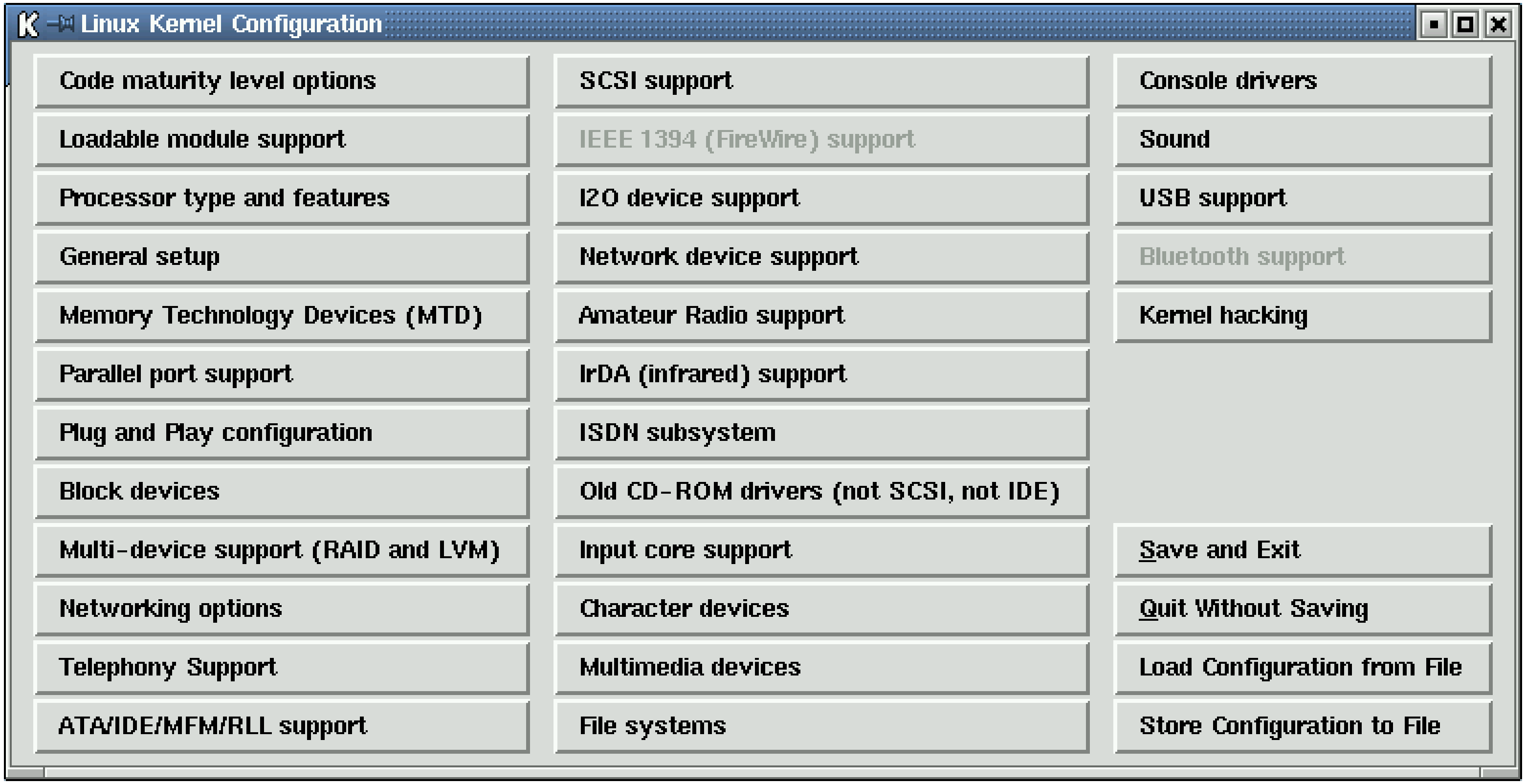
```
.config - Linux/x86 3.10.0-rc2 Kernel Configuration

                         Linux/x86 3.10.0-rc2 Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
 <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module
 < > module capable

                              [ ] 64-bit kernel
                                  General setup  --->
                              [*] Enable loadable module support  --->
                              [*] Enable the block layer  --->
                                  Processor type and features  --->
                                  Power management and ACPI options  --->
                                  Bus options (PCI etc.)  --->
                                  Executable file formats / Emulations  --->
                              -*- Networking support  --->
                                  Device Drivers  --->
                                  Firmware Drivers  --->
                                  File systems  --->
                                  Kernel hacking  --->
                                  Security options  --->
                              -*- Cryptographic API  --->
                              -*- Virtualization  --->
                                  Library routines  --->

                  <Select>    < Exit >    < Help >    < Save >    < Load >
```

Make Xconfig



Compile The Linux Kernel (4)

- Customize the Kernel
  - e.g., you may add support for NTFS file system from “File System >> DOS/FAT/NT/ >> select NTFS file system support”

Compile The Linux Kernel (5)

- Compile and Install the Kernel (~20 min)

```
# make
# make modules_install
# make install
```

- Update GRUB (or LILO)

```
    # cd /boot
    # mkinitrd –o initrd.img-<kernel version>
```

- Reboot the system