System Programming

1. Introduction

Seung-Ho Lim

Dept. of Computer & Electronic Systems Eng.

Objective of SP (1)

- Understand the role of OS and the system calls of OS
  - Especially, focus on Linux and Unix
- Learn how to use the system calls for application programming
  - file I/O
  - process/thread management
  - memory management
  - IPC(interprocess communication)
  - synchronization
  - time
  - network
  - ...

Objective of SP (2)

- Practice and Experience the application development in Unix/Linux
  - Command-line interpreter (called shell)
  - Editor (vi, vim, emacs, ...)
  - GNU tools
    - compiler (gcc, g++, ...)
    - debugger (gdb)

System Programming

- Lecture Materials
  - Lecture Slides
  - Programming Exercises(HW)

- Reference Textbook
  - “리눅스 시스템 프로그래밍”, O’Reilly, 개정 2판
  - “리눅스 시스템 프로그래밍”, 김 정 국 지음, 외대 출판사, 2014.

- Ref
  - http://lxr.free-electrons.com/: Linux source navigation
  - Any books or documents on Pthread Programming
  - Linux Internals, M. Bar, Mc Graw Hill

강의 내용

주차	내용	주차	내용
1	Introduction	9	Memory
2	File IO – Library Call	10	Semaphore
3	File IO – System Call	11	Timer
4	Concurrent Process	12	IPC
5	Process Control	13	Network
6	Threads	14	Time Management
7	Lock	15	Linux Utilities
8	Mid-Term Exam	16	Final Exam

System Programming

- 평가
  - 중간고사 37%
  - 기말고사 38%
  - 과제물 15%, 기타(발표 등) 5%
  - 출석 5%

- 과제물
  - 각 단원마다 프로그래밍 숙제
  - 제출 방법
    - System Programming Server에 접속
    - HWxx directory 생성 (ex. mkdir HW01)
    - HWxx 디렉토리 아래에 작성한 소스코드, 보고서 hwp 등 제출

- 숙제 조교
  - 숙제 내용 문의
  - SystemProgramming1(화3, 목12)
    - 이재빈 (gaebin1212@gmail.com)
  - SystemProgramming2(화7, 목56)
    - 신동렬(fufehd12@naver.com)

Operating System

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - make the computer system convenient to use
  - use the computer hardware in an efficient manner

- Fundamental OS Concepts
  - Multi-user environment
  - Process and Scheduling
  - User space and Kernel space
  - Basic and Advanced I/O

Unix history (1)

- Originally developed in 1969 at Bell Labs by Ken Thompson and Dennis Ritchie.
- 1973, Rewritten in C. This made it portable and changed the history of OS
- 1974: Thompson, Joy, Haley and students at Berkeley develop the **B**erkeley **S**oftware **D**istribution (BSD) of UNIX
- two main directions emerge: BSD and what was to become "System V"
- Linux is a Unix-like OS

For more info :

http://www.unix.org/what_is_unix/history_timeline.html

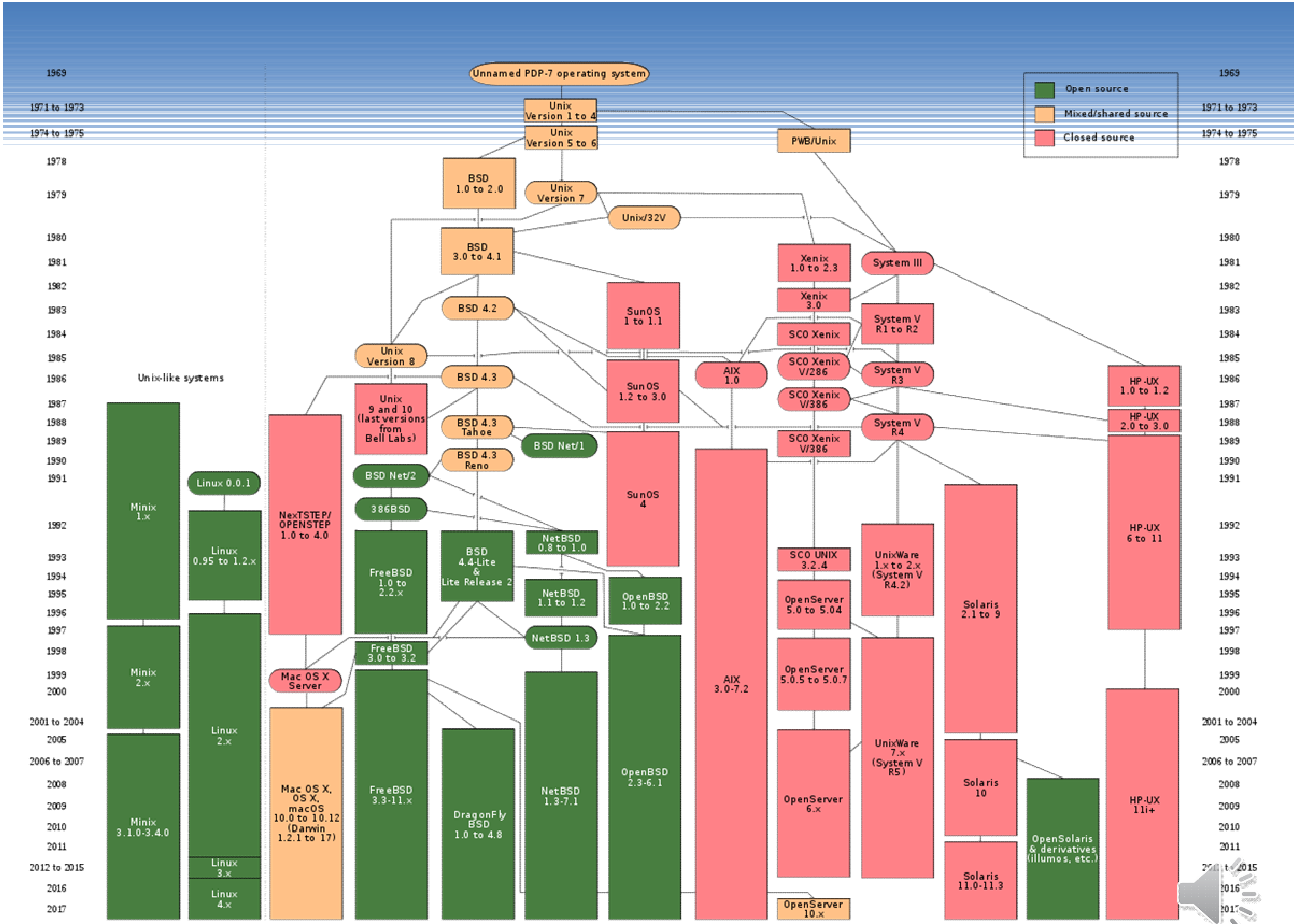
Ken Thompson and Dennis Ritchie at PDP-11 in 1971 (Photo: Courtesy of Bell Labs)

Unix history (2)

- 1984 4.2BSD released (TCP/IP)
- 1986 4.3BSD released (NFS)
- 1991 Linus Torvalds starts working on the Linux kernel
- 1993 Settlement of USL vs. BSDi; NetBSD, then FreeBSD are created
- 1994 Single UNIX Specification introduced
- 1995 4.4BSD-Lite Release 2 (last CSRG release); OpenBSD forked off NetBSD
- 2000 Darwin created (derived from NeXT, FreeBSD, NetBSD)
- 2003 Xen; SELinux
- 2005 Hadoop; DTrace; ZFS; Solaris Containers
- 2006 AWS ("Cloud Computing" comes full circle)
- 2007 iOS; KVM appears in Linux
- 2008 Android; Solaris open sourced as OpenSolaris

Unix Philosophy (1)

- Small is beautiful.
- Make each program do one thing well.
- Build a prototype as soon as possible.
- Choose portability over efficiency.
- Store data in flat text files.
- Use shell scripts to increase leverage and portability.
- Avoid captive user interfaces.
- Make every program a filter.


1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017
Open source
Mixed/shared source
Closed source
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
SunOS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
BSD 4.3
AIX 1.0
SCO Xenix V/286
System V R3
HP-UX 1.0 to 1.2
Unix-like systems
Unix 9 and 10 (last versions from Bell Labs)
SunOS 1.2 to 3.0
SCO Xenix V/386
HP-UX 2.0 to 3.0
BSD 4.3 Tahoe
System V R4
BSD Net/1
SCO Xenix V/386
BSD 4.3 Reno
BSD Net/2
Linux 0.0.1
Minix 1.x
NexTSTEP/OPENSTEP 1.0 to 4.0
386BSD
SunOS 4
NetBSD 0.8 to 1.0
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4-Lite & Lite Release 2
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
HP-UX 6 to 11
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
NetBSD 1.3
FreeBSD 3.0 to 3.2
Minix 2.x
Mac OS X Server
OpenServer 5.0.5 to 5.0.7
AIX 3.0-7.2
Linux 2.x
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
FreeBSD 3.3-11.x
DragonFly BSD 1.0 to 4.8
NetBSD 1.3-7.1
OpenBSD 2.3-6.1
OpenServer 6.x
UnixWare 7.x (System V R5)
Solaris 10
HP-UX 11i+
Minix 3.1.0-3.4.0
OpenSolaris & derivatives (illumos, etc.)
Linux 3.x
Solaris 11.0-11.3
Linux 4.x
OpenServer 10.x

Linux history (1)

- GNU Project starts in 1983 as an alternative to proprietary UNIX
  - **G**NU's **N**ot **U**nix
  - at 1985, Richard Stallman announced 'GNU Manifesto'. Evolution of open source S/W started.
- "Andrew S. Tanenbaum" announced the Minix OS (OSS) that is a variant of UNIX at 1987.
- "Linus Torvalds" announced the first Linux OS at 1991.
  - ***Linux 2.4.x → Linux 2.6.x → Linux 3.0.x → Linux 4.x.x***

Linux history (2)

- Linux first version : 1991. 11 (Linus Torvalds)
- POSIX 1003.1 standard compliance,
- Large areas of functions of System V and BSD 4.3 UNIX
- On GNU Public License : GPL & LGPL (Light GPL)
  - Pure applications on Linux: no need to open the source
  - If you modified an existing OSS, you must open the source
- Supports most of CPU chips, devices
- From ver. 2.0, supports multiprocessor systems
- Linux is widely used for servers and embedded systems
- From Linux 2.6.x : preemptible kernel
  - Enhanced for real-time systems
- Standard : LSB5.0 (Linux Standard Base: Free Standard Group)

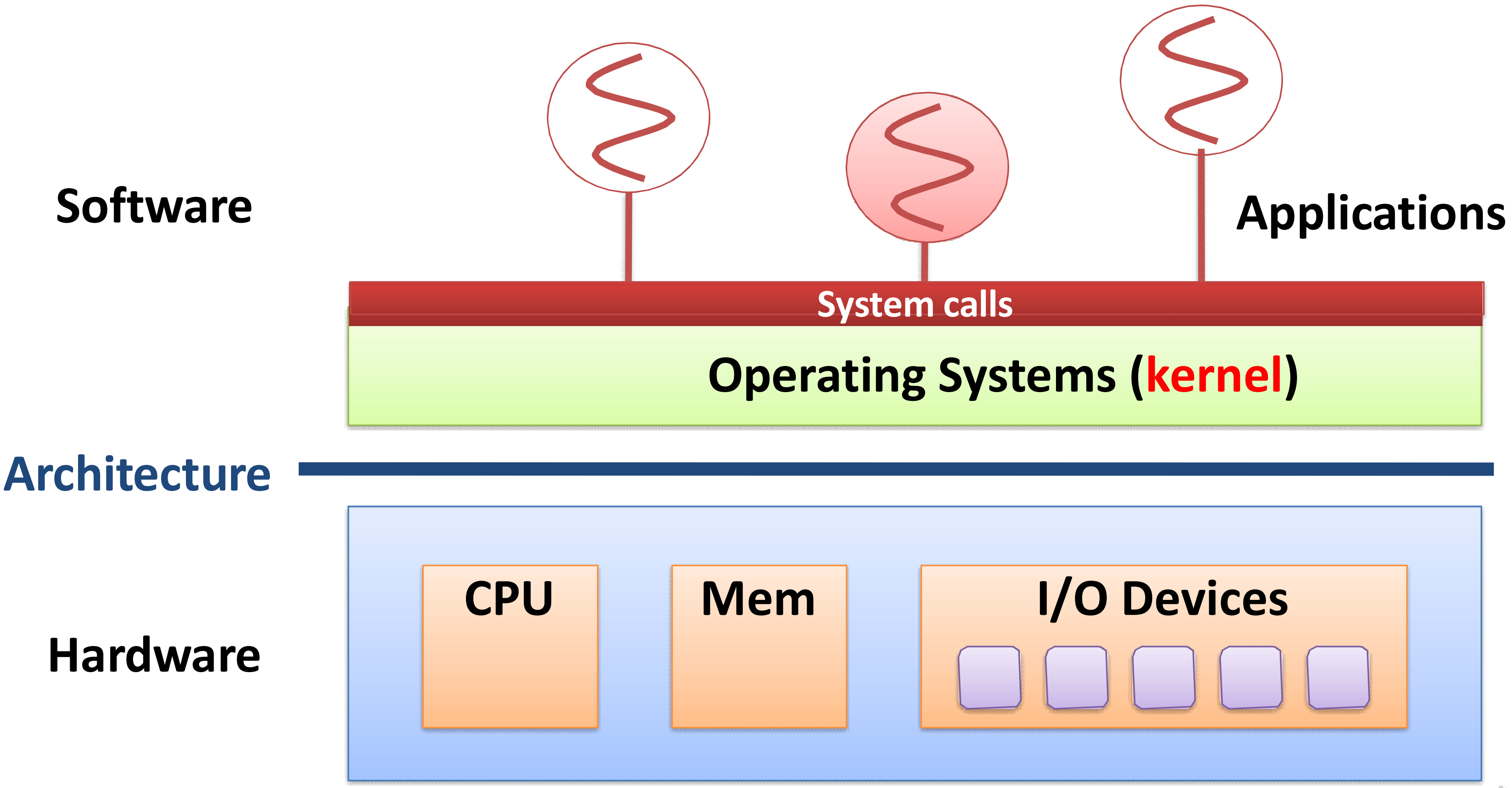
Advantages & Weakness

- Advantages
  - Open source : developer versions, stabilized version, GNU spirit(copy, modification, distribution are possible.).
  - Open developer site & User group(LUG): exchange information .
  - Royalty free.
- Weakness
  - Too fast version upgrade (many versions), many venders → follows the UNIX's way.
  - Less official programs: office, game, desktop environment.
  - Device drivers
    - Many developers develops ***non-mature*** device drivers and kernel components. (3rd party OSS).
    - Developing a device driver does not make money well. (Open source).
    - ***Hidden patents. & License problems***

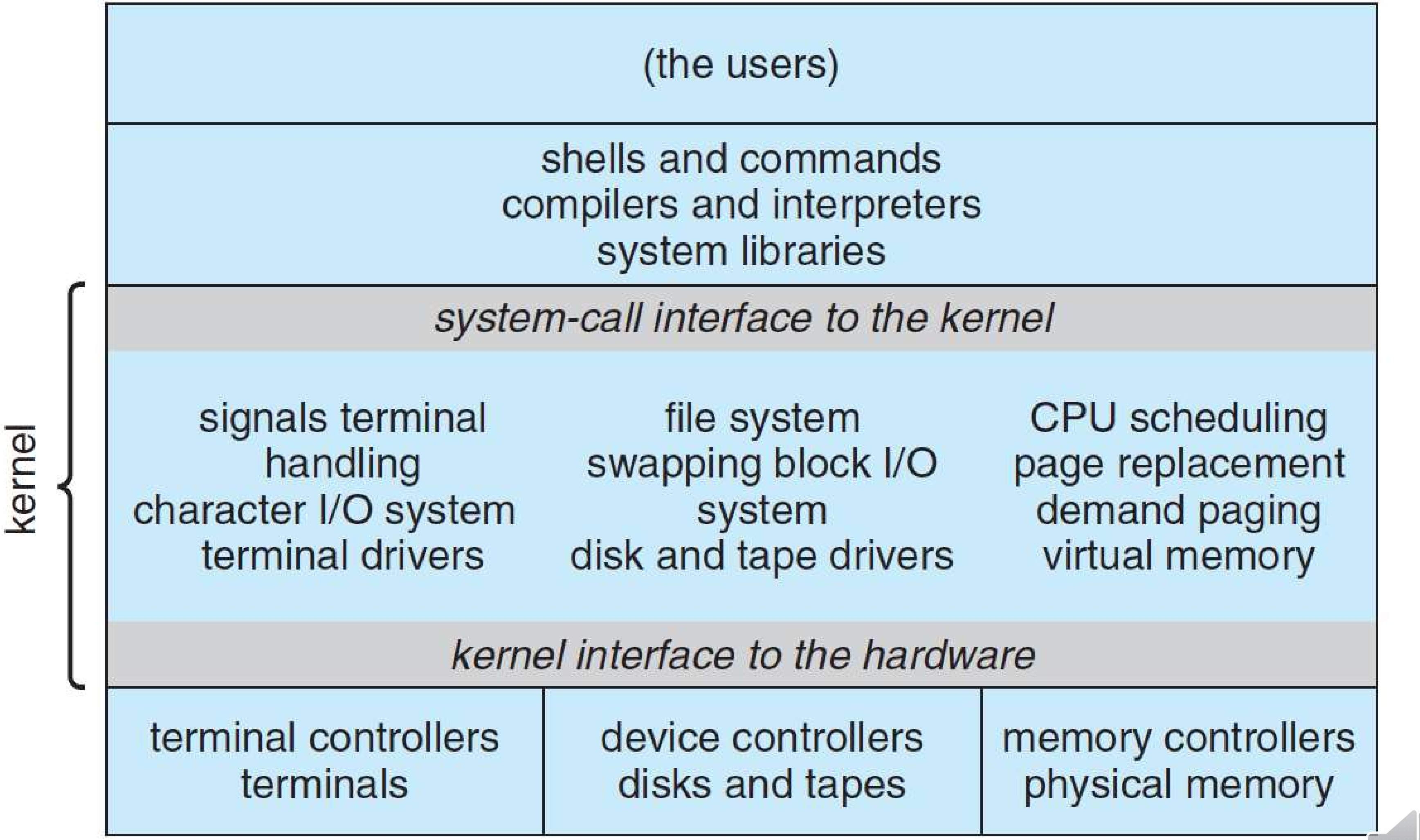
Distribution, Development, Standardization

- Linux distributors
  - Debian (Raspbian, Ubuntu, etc) : non-commercial
  - Fedora (Red Hat, etc) : commercial
  - Various other distributions (OpenSUSE, Android, etc)
- Linux archive: http://www.kernel.org
- Source navigation
  - ***http://lxr.free-electrons.com/***
- Open Projects
  - ***www.sourceforge.net,***
  - ***www.linux-foundation.org***
  - ***GNU, GNOME*** (Desktop GUI interface)
  - ***Fedora***: ***Redhat*** is the main sponsor, community supported open project
- Famous projects : ***Apache, Jakarta***, etc (Web Server & Java Environment)

Computer System Overview



Layered Linux Structure (1)



(the users)

shells and commands
compilers and interpreters
system libraries

system-call interface to the kernel

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

kernel interface to the hardware

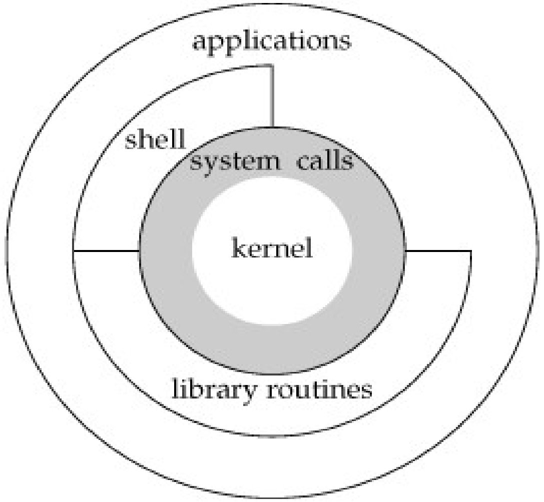
terminal controllers
terminals

device controllers
disks and tapes

memory controllers
physical memory

Layered Linux Structure (2)

- Hardware
  - CPU, Memory, Disk, Peripherals
- Kernel
  - Process management
  - File management
  - Memory management
  - Device management



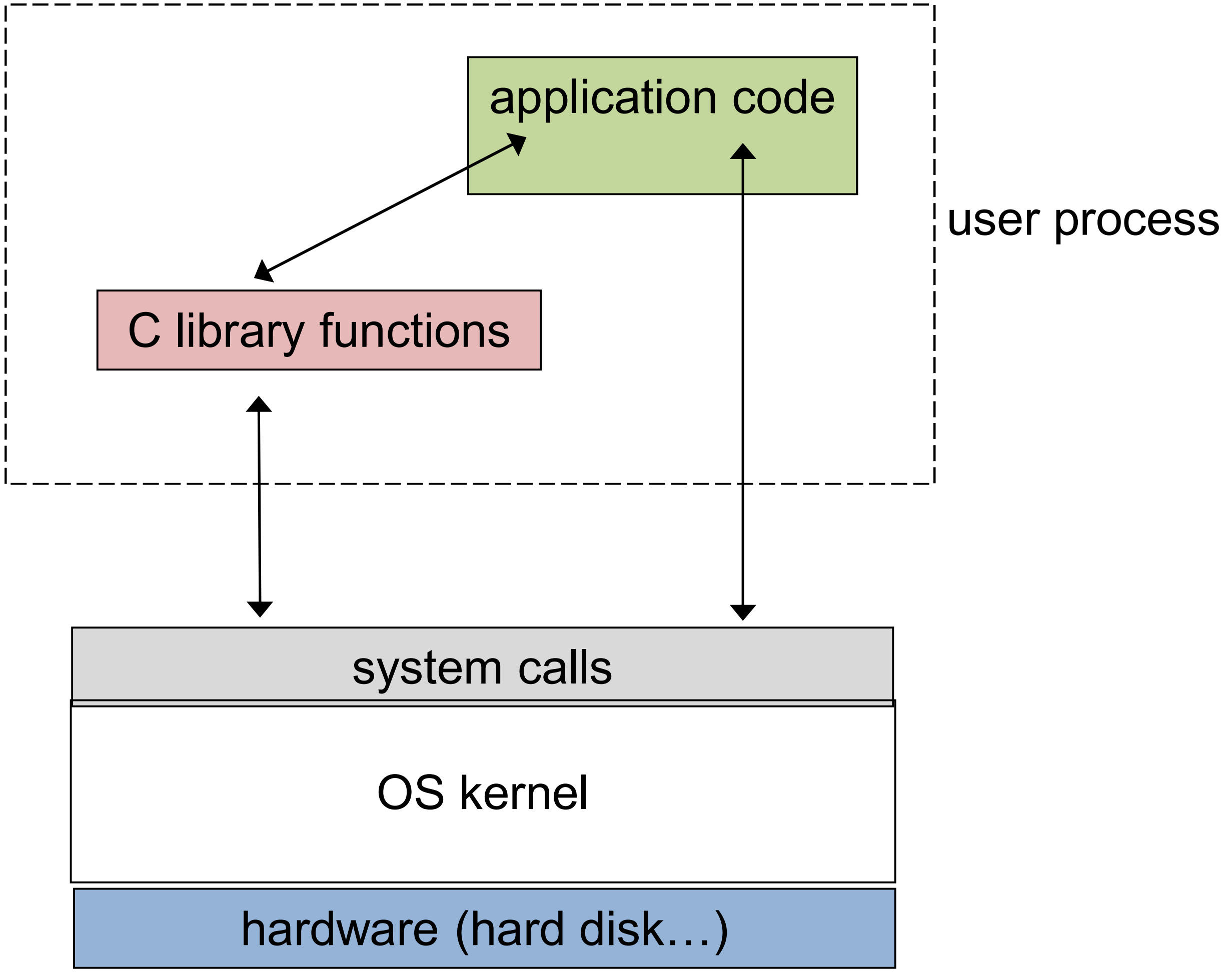
- System call
  - the programmer's functional interface to the Linux kernel
- Commands, Utilities, Application programs
  - request kernel services using library routines or system calls

System Calls vs. Library Calls (1)

- System Calls
  - they are entry points into kernel code where their functions are implemented.
  - documented in section 2 of the linux manual (e.g. `write`(2) or `man` 2 `write`)

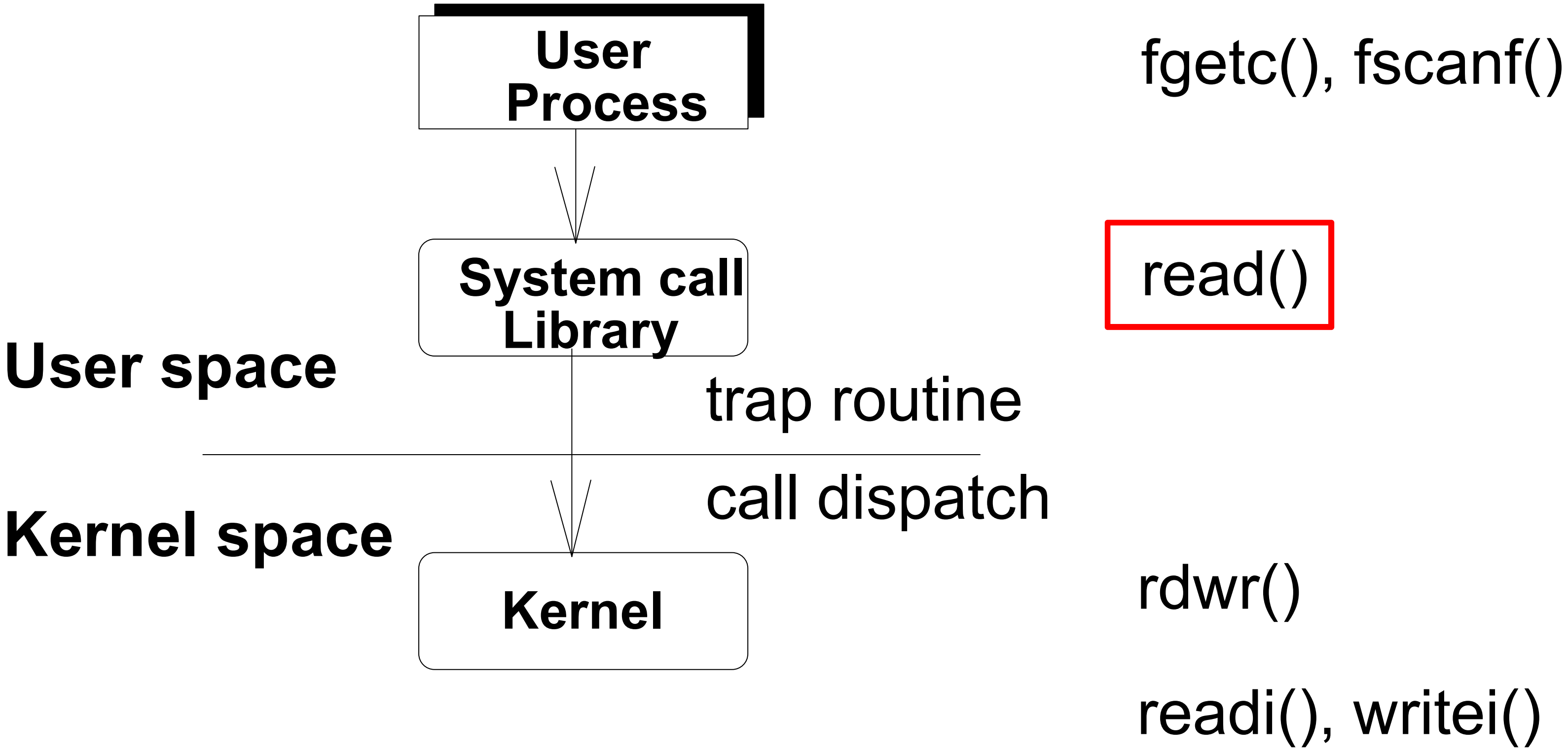
- Library Calls
  - they are transfers to user code which performs the desired functions.
  - documented in section 3 of the linux manual (e.g. `printf`(3)).
  - also called *API*(application programming interfece)

System Calls vs. Library Calls (2)



System Calls vs. Library Calls (3)

- Example: read() system call



User Process

fgetc(), fscanf()

System call Library

read()

User space

trap routine

Kernel space

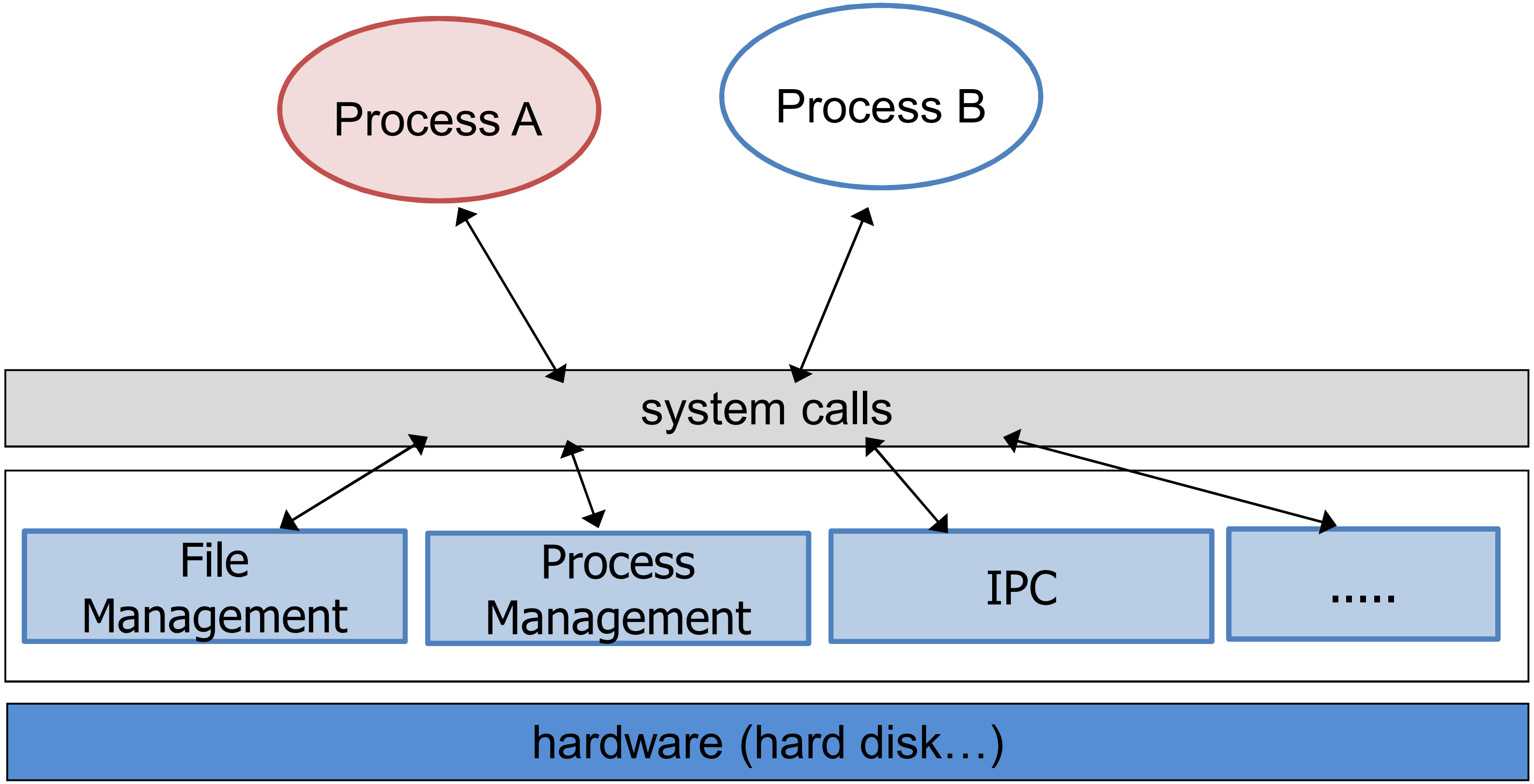
call dispatch

Kernel

rdwr()

readi(), writei()

System Calls by Processes



Linux System Calls Overview (1)

- File descriptor I/O
  - ***open(); close(); creat(), read(); write();***
  - ***seeek();*** *// random access*
  - ***fcntl();*** *// for file/record locking*
- Process control
  - ***fork(); exec(); wait(); exit();***
- Thread programming
  - ***pthread_...();***
- IPC
  - *Pipe:* ***pipe(); read(); write(); close();***
  - *Message queue: …*
- Signal handling
  - ***signal(); kill();*** *// making signal handlers;*
  - ***alarm(); pause(); sigpause(); sigblock(); sigsuspend();***
  - ***itimer (interval timer)*** *// timer creation & handling*

Linux System Calls Overview (2)

- Memory management
  - ***malloc(); free(); memcpy(); bzero();***
  - *Memory mapped files:* ***mmap(); munmap();***
- *Synchronization*
  - *File lock/unlock with* ***fcntl()***
  - *Semaphores (POSIX, SysV)*
- *Time management*
  - *Epoch time, calendar time managements*
- *Network socket API (TCP, UDP)*
  - ***socket(); close();***
  - ***bind(); listen();***
  - ***accept(); connect();***
  - ***send() recv();*** *// TCP*
  - ***sendto(); recvfrom();*** *// UCP*

Summary