國立臺灣師範大學
NATIONAL TAIWAN NORMAL UNIVERSITY

Advanced Operating Systems
1.Introduction.7

Chun-Han Lin (林均翰)

CSIE, NTNU



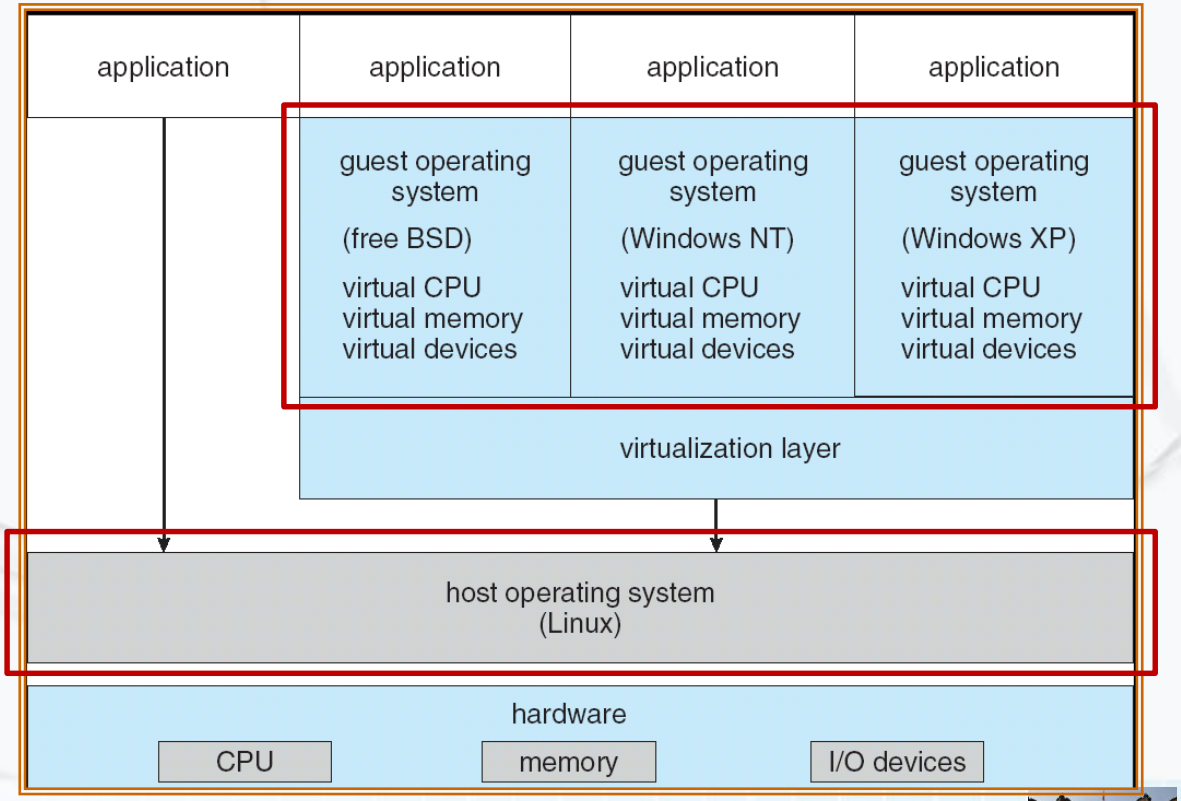
Key Points 1.7

- **Virtual machines: Layers of OS**
- **How to work on an OS easily?**
- **Computing facilities**



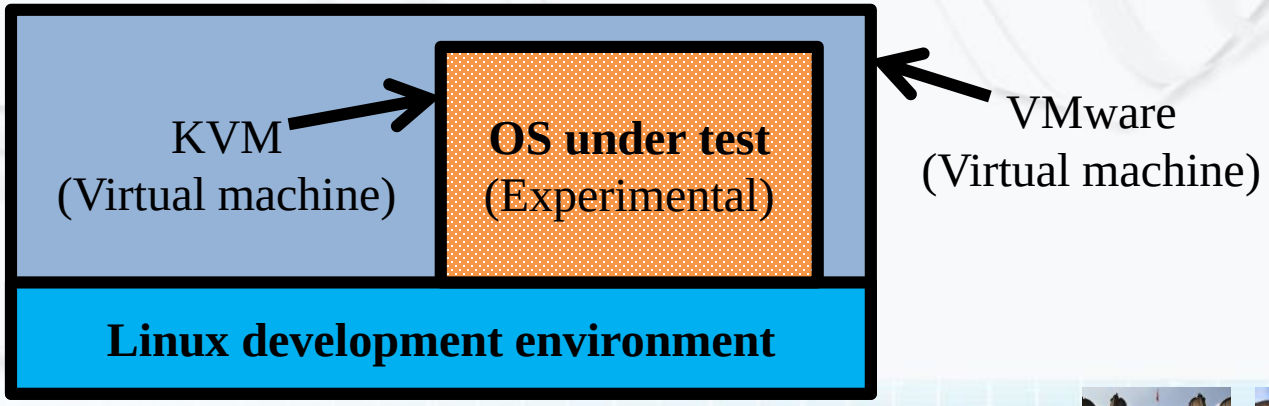
Virtual Machines: Layers of OS

- **Useful for OS development**
  - **When an OS crashes, restricted to one VM**
  - **Can aid testing programs on other OS**



How to work on an OS easily?

- Traditional approach
  - Sit at a serial console
  - Upload a new OS image somehow
  - Reboot and possibly crash (“Panic”)
  - Debug with very limited tools
- How we will do it in this class?
  - Virtual machines!
  - Or nested virtual machines



Computing Facilities

- This class is best taken with a late-model laptop
  - Look for an x86 with Vt-x extensions
  - Work in a virtual machine
  - Should be able to use Windows, Mac-OS, or Linux.



Review 1.7

- **Virtual machines: Layers of OS**
- **How to work on an OS easily?**
- **Computing facilities**



國立臺灣師範大學
NATIONAL TAIWAN NORMAL UNIVERSITY

Advanced Operating Systems
1.Introduction.8

Chun-Han Lin (林均翰)
CSIE, NTNU



Key Points 1.8

- **Why are we working on Linux?**
- **What does an OS do?**
- **What is an OS … Really?**



Why are we working on Linux? (1/2)

- **Penetration into many markets**
  - **Embedded space: Phones, routers, sensors, …**
  - **Desktops: Gnome, KDE, other X environments, …**
  - **Servers: High-end cloud services, web services, file services, …**
- **Open-source!**
  - **Can learn by "reading the source!"**



Why are we working on Linux? (2/2)

- **Extreme team-collaborative environment**
  - **Native use of development tools like “git”, “gdb”, lots of testing and development tools**
  - **Linux started the “Bizzare” method of development**
- **Negatives?**
  - **Code is not always well-designed.**
    - **No central authority enforcing “quality”**
  - **Occasionally versions of different components may “out-of-sync”**



What does an OS do? (1/2)

- Silerschatz and Gavin
  - An OS is similar to a government
  - Beg the question: Does a government do anything useful by itself?
- Coordinator and traffic cop
  - Manage all resources
  - Settle conflicting requests for resources
  - Prevent errors and improper use of the computer



What does an OS do? (2/2)

- **Facilitator**
  - **Provide facilities that everyone needs**
    - **Standard libraries, windowing systems**
  - **Make application programming easier, faster, less error-prone**
- **Some features reflect both tasks.**
  - **E.g., a file system is needed by everyone (facilitator), but file system must be protected (traffic cop)**



What is an OS … Really?

- Components
  - Memory management
  - I/O management
  - CPU scheduling
  - Communications? (Does Email belong in OS?)
  - Multitasking/multiprogramming?
- What about?
  - File system?
  - Multimedia support?
  - User interface?
  - Internet browser? ☺
- Is this only interesting to academics?



Review 1.8

- **Why are we working on Linux?**
- **What does an OS do?**
- **What is an OS … Really?**



國立臺灣師範大學
NATIONAL TAIWAN NORMAL UNIVERSITY

Advanced Operating Systems
1.Introduction.9

Chun-Han Lin (林均翰)
CSIE, NTNU



Key Points 1.9

- **OS definitions**
- **Example: Protecting processes from each other**



OS Definitions

- No a universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation.
  - But this varies wildly.
- "The one program running at all times on the computer" is the kernel.
  - Everything else is either a system program (ships with the OS) or an application program.
- Studying OS is really about the hardware/software interface (API).
  - Thus, we will hope to give you enough knowledge to …
    - Understand the interface
    - Modify the interface
    - Change the support underneath the interface



Example: Protecting Processes from Each Other

- **Problem: Run multiple applications in such a way that they are protected from one another**
- **Goals**
  - **Keep user programs from crashing OS**
  - **Keep user programs from crashing each other**
  - **[Keep parts of OS from crashing other parts?]**
- **(Some of the required) Mechanisms**
  - **Address translation**
  - **Dual mode operation**
- **Simple policy**
  - **Programs are not allowed to read/write memory of other programs or of OS.**



Review 1.9

- **OS definitions**
- **Example: Protecting processes from each other**



國立臺灣師範大學
NATIONAL TAIWAN NORMAL UNIVERSITY

Advanced Operating Systems
1.Introduction.10

Chun-Han Lin (林均翰)
CSIE, NTNU



Key Points 1.10

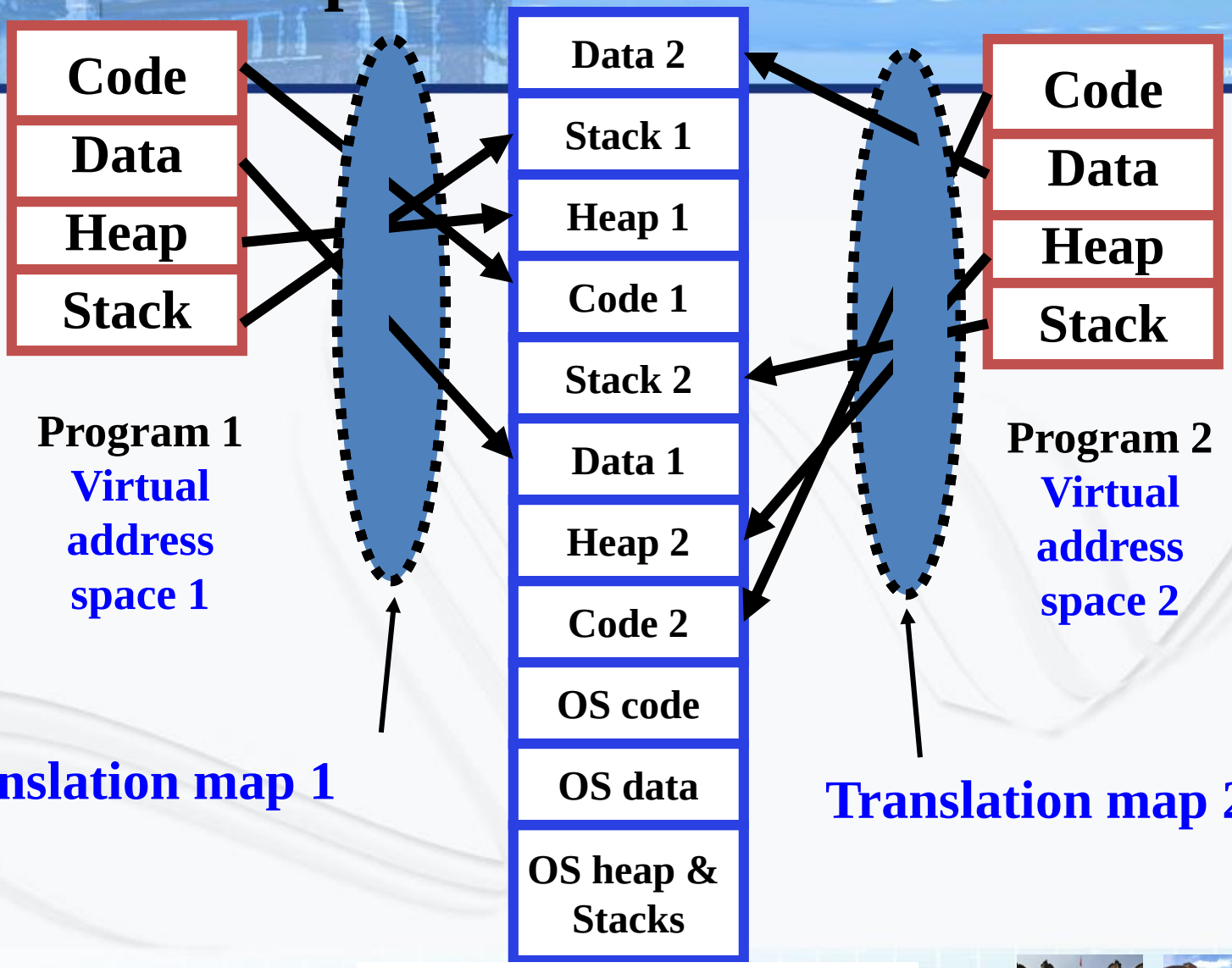
- **Address translation**
- **Dual mode operation**



Address Translation

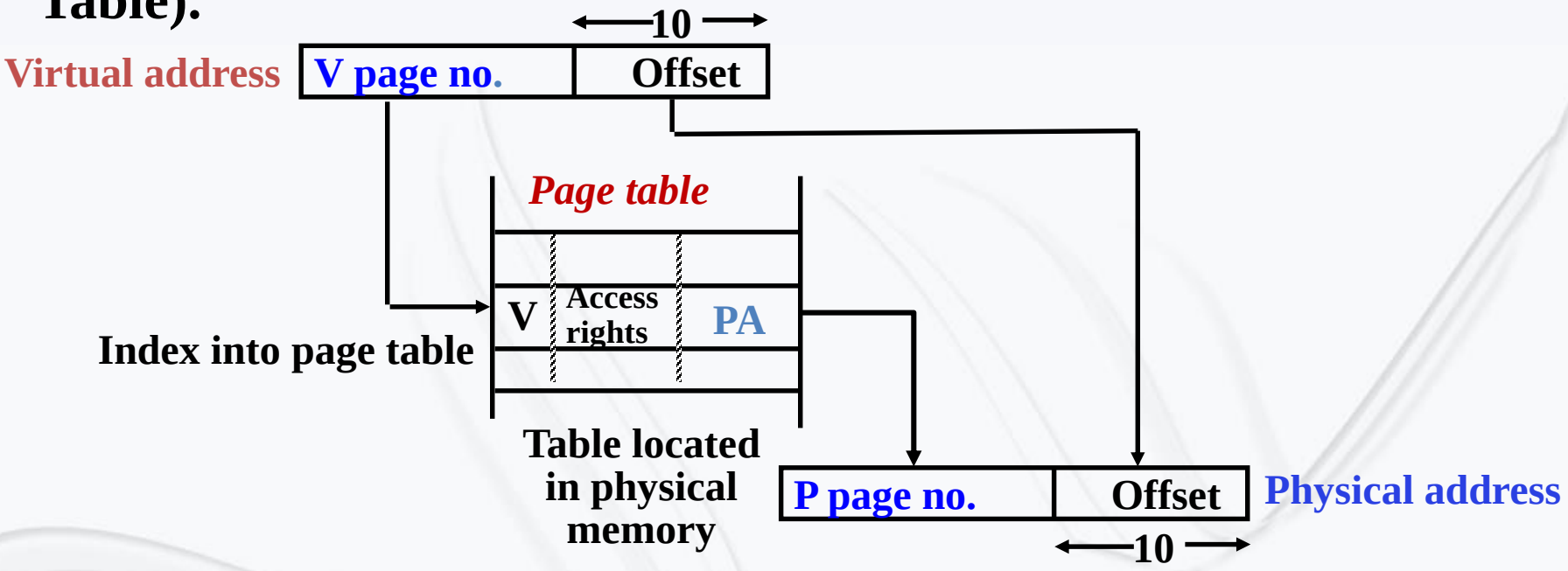
- **Address space**
  - A group of memory addresses usable by something
  - Each program (process) and kernel has potentially different address spaces.
- **Address translation**
  - Translate from virtual addresses (emitted by CPU) into physical addresses (of memory)
  - Mapping often performed in hardware by Memory Management Unit (MMU)

Example of Address Translation



Address Translation Details

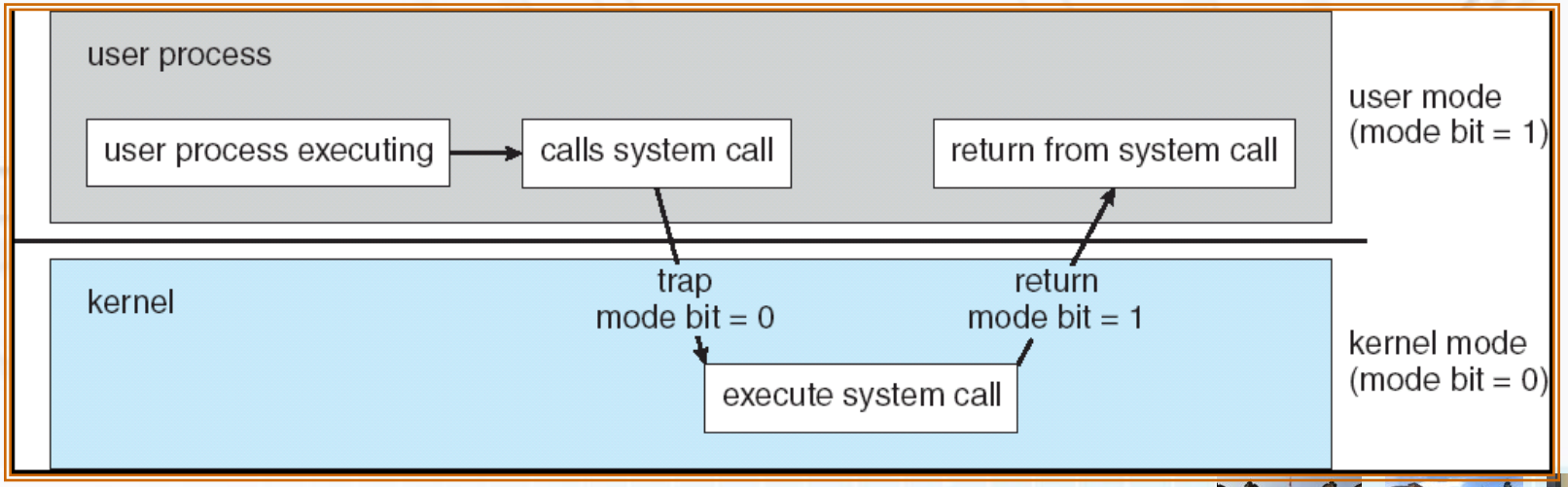
- For now, assume translation happens with a table (called a Page Table).



- Translation helps protection.
  - Control translations, control access
  - Should users be able to change page table?

Dual Mode Operation

- **Hardware provides at least two modes.**
  - **Kernel mode (or "supervisor" or "protected")**
  - **User mode: Normal programs are executed.**
- **Some instructions/operations are prohibited in user mode.**
  - **E.g., cannot modify page tables in user mode**
    - **Attempt to modify ⇒ Exception is generated**
- **Transitions from user mode to kernel mode**
  - **System calls, interrupts, and other exceptions**



Review 1.10

- **Address translation**
- **Dual mode operation**



國立臺灣師範大學
NATIONAL TAIWAN NORMAL UNIVERSITY

Advanced Operating Systems
1.Introduction.11

Chun-Han Lin (林均翰)
CSIE, NTNU

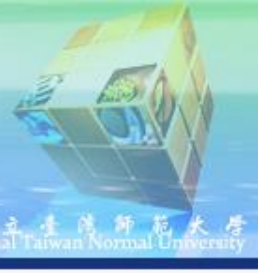


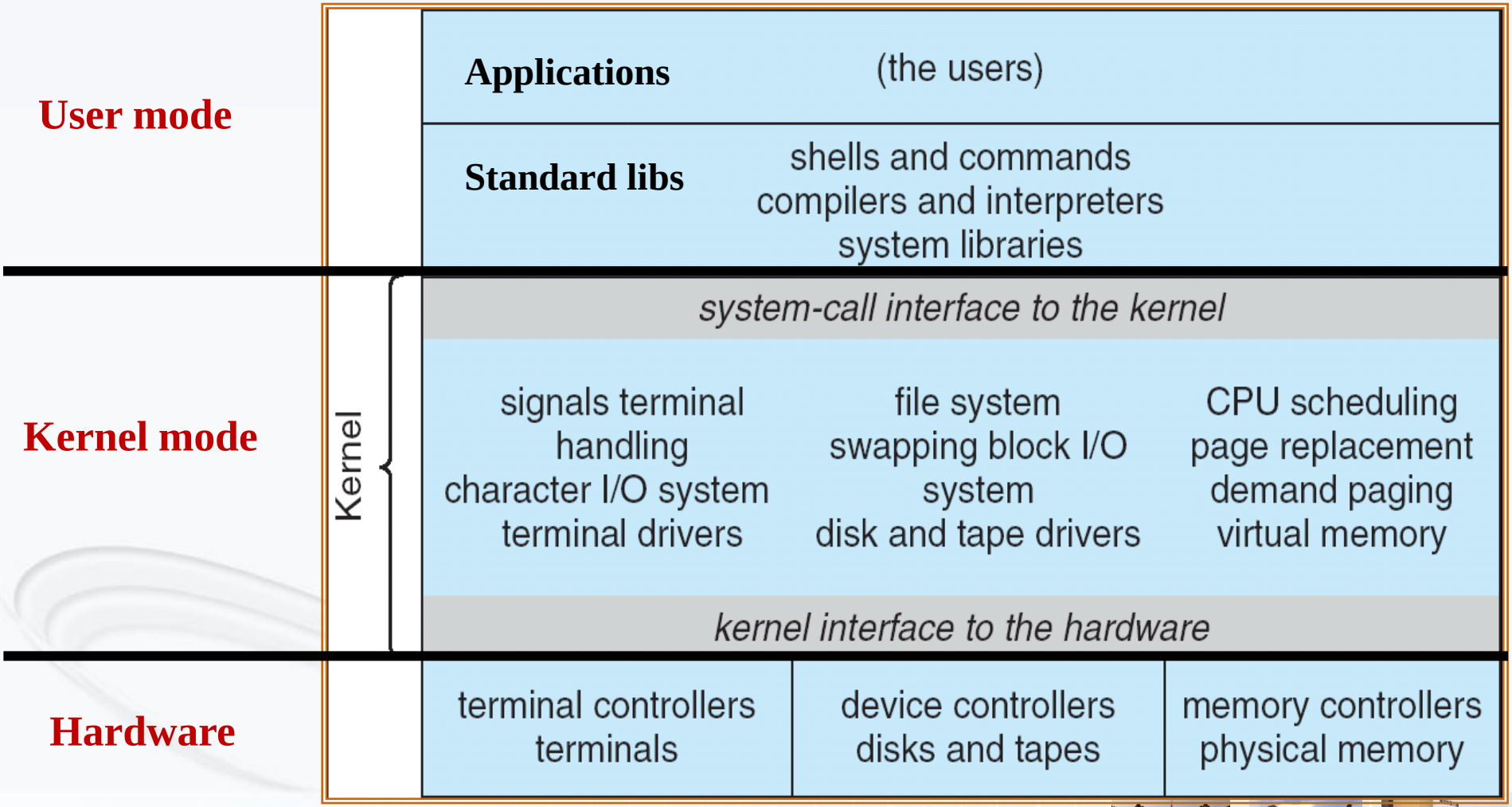
Key Points 1.11

- UNIX system structure
- Linux structure
- Map of Linux components
- History of Unix

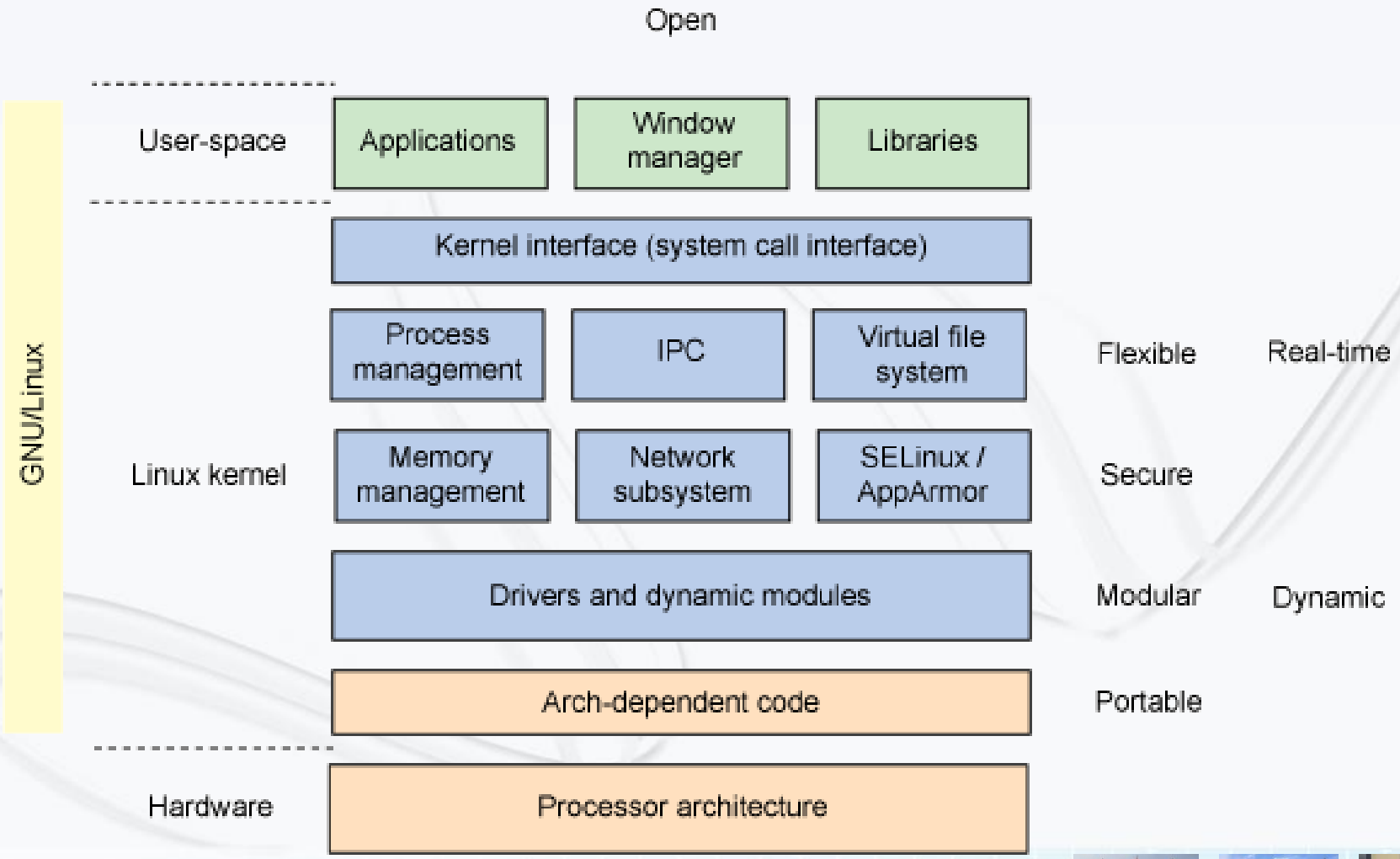


UNIX System Structure

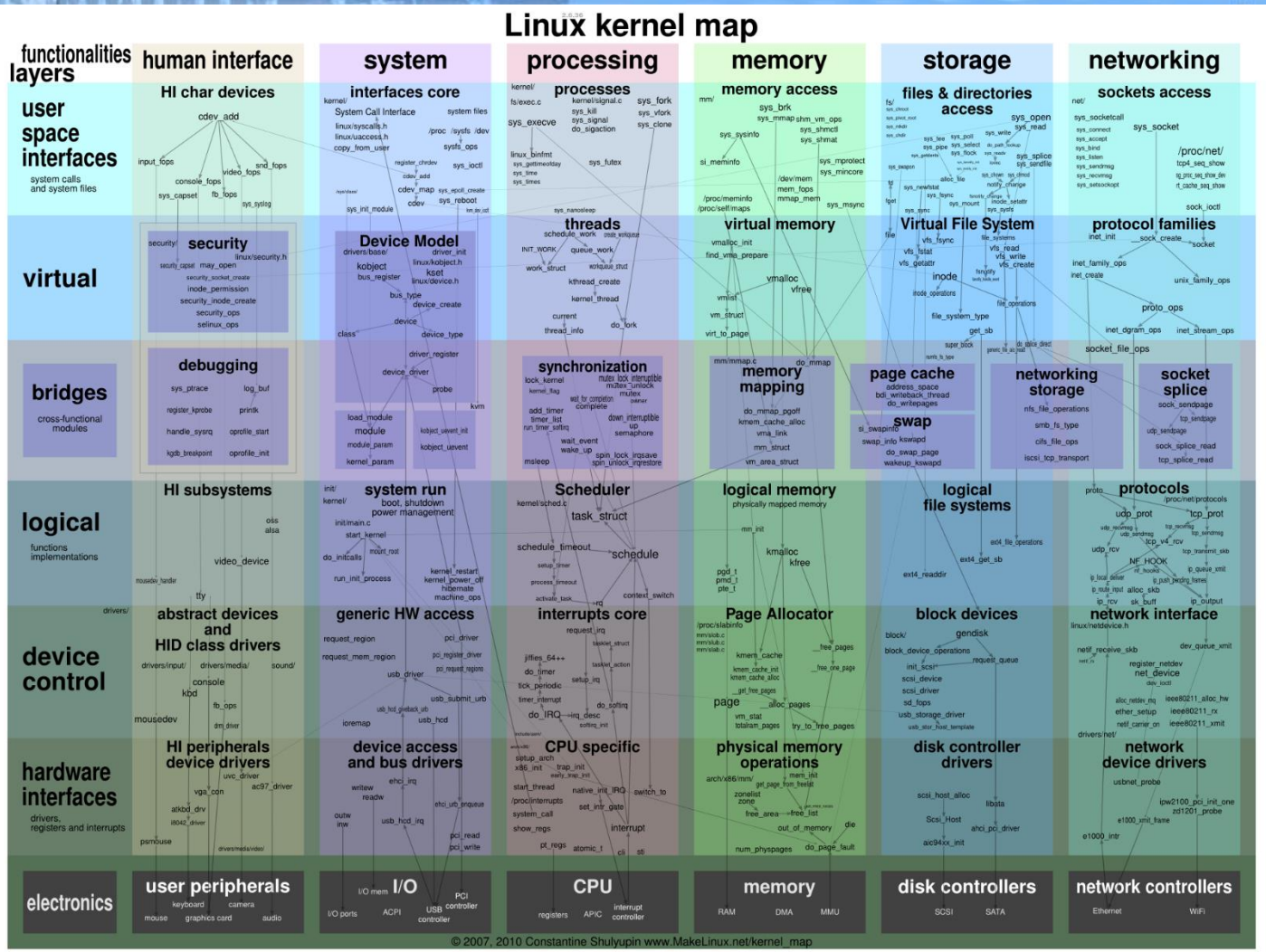




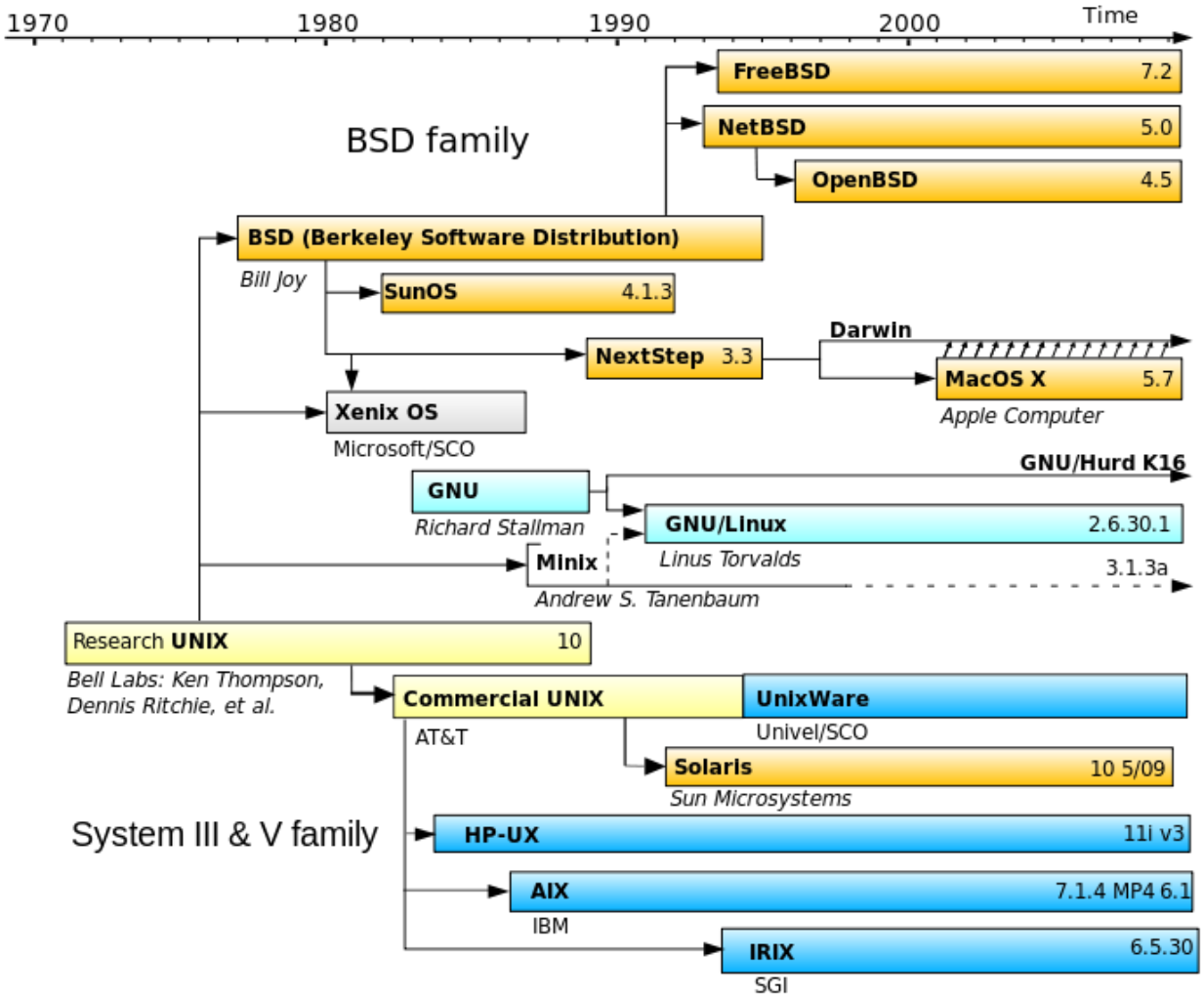
Linux Structure (More as term progresses!)



Map of Linux Components (http://www.makelinux.net/kernel_map/)



History of Unix



Review 1.11

- UNIX system structure
- Linux structure
- Map of Linux components
- History of Unix



國立臺灣師範大學
NATIONAL TAIWAN NORMAL UNIVERSITY

Advanced Operating Systems
1.Introduction.12

Chun-Han Lin (林均翰)
CSIE, NTNU

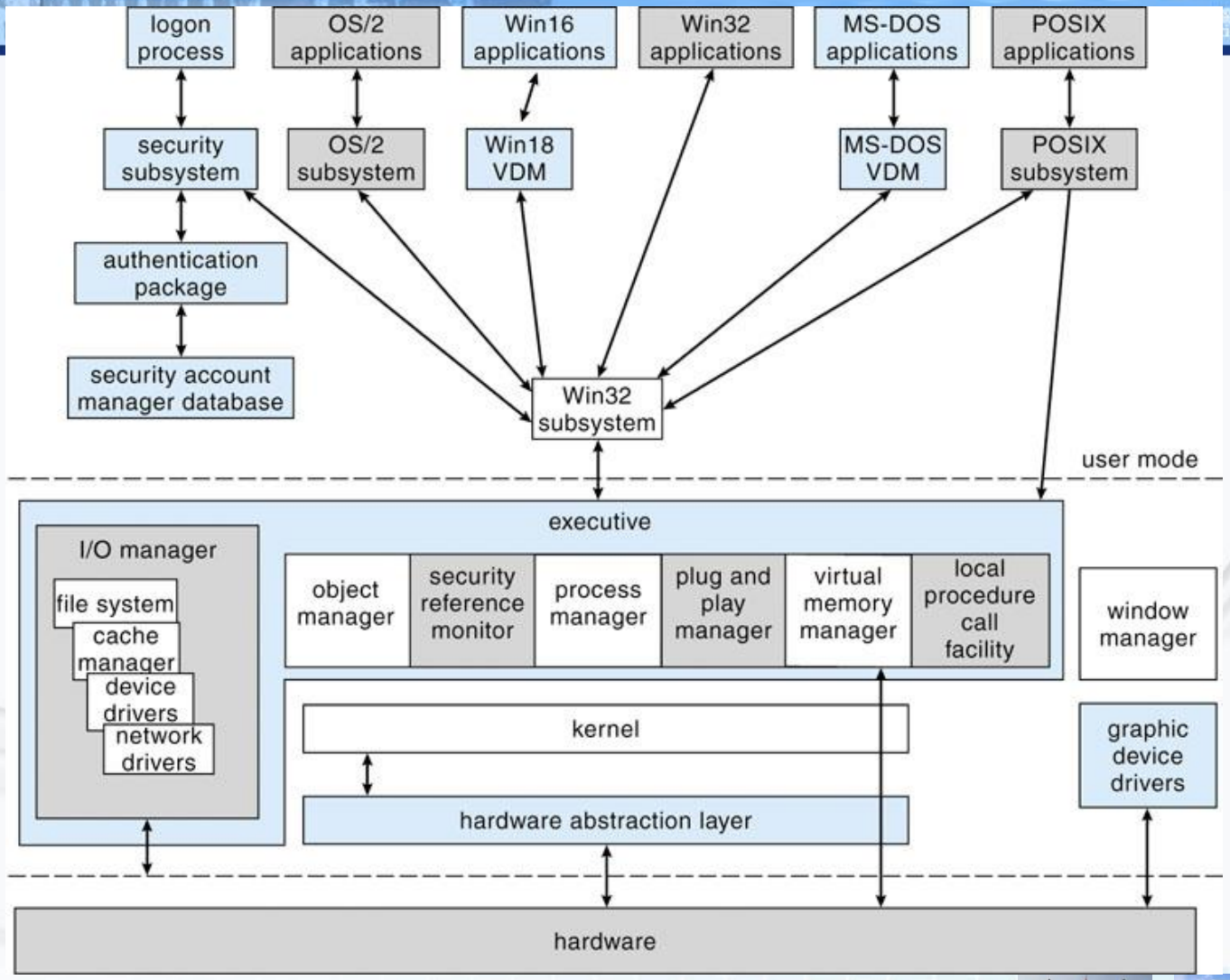


Key Points 1.12

- **Microsoft Windows structure**
- **Major Windows components**



Microsoft Windows Structure



Major Windows Components (1/2)

- Hardware abstraction layer hides hardware chipset differences from upper levels of OS.

- Kernel layer
  - Thread scheduling
  - Low-level processor synchronization
  - Interrupt/Exception handling
  - Switching between user/kernel mode



Major Windows Components (2/2)

- Executive
  - Set of services that all environmental subsystems need
    - Object manager
    - Virtual memory manager
    - Process manager
    - Advanced local procedure call facility
    - I/O manager
    - Cache manager
    - Security reference monitor
    - Plug-and-play and power managers
    - Registry
    - Booting
- Programmer interface: Win32 API



Review 1.12

- **Microsoft Windows structure**
- **Major Windows components**



國立臺灣師範大學
NATIONAL TAIWAN NORMAL UNIVERSITY

Advanced Operating Systems
1.Introduction.13

Chun-Han Lin (林均翰)
CSIE, NTNU

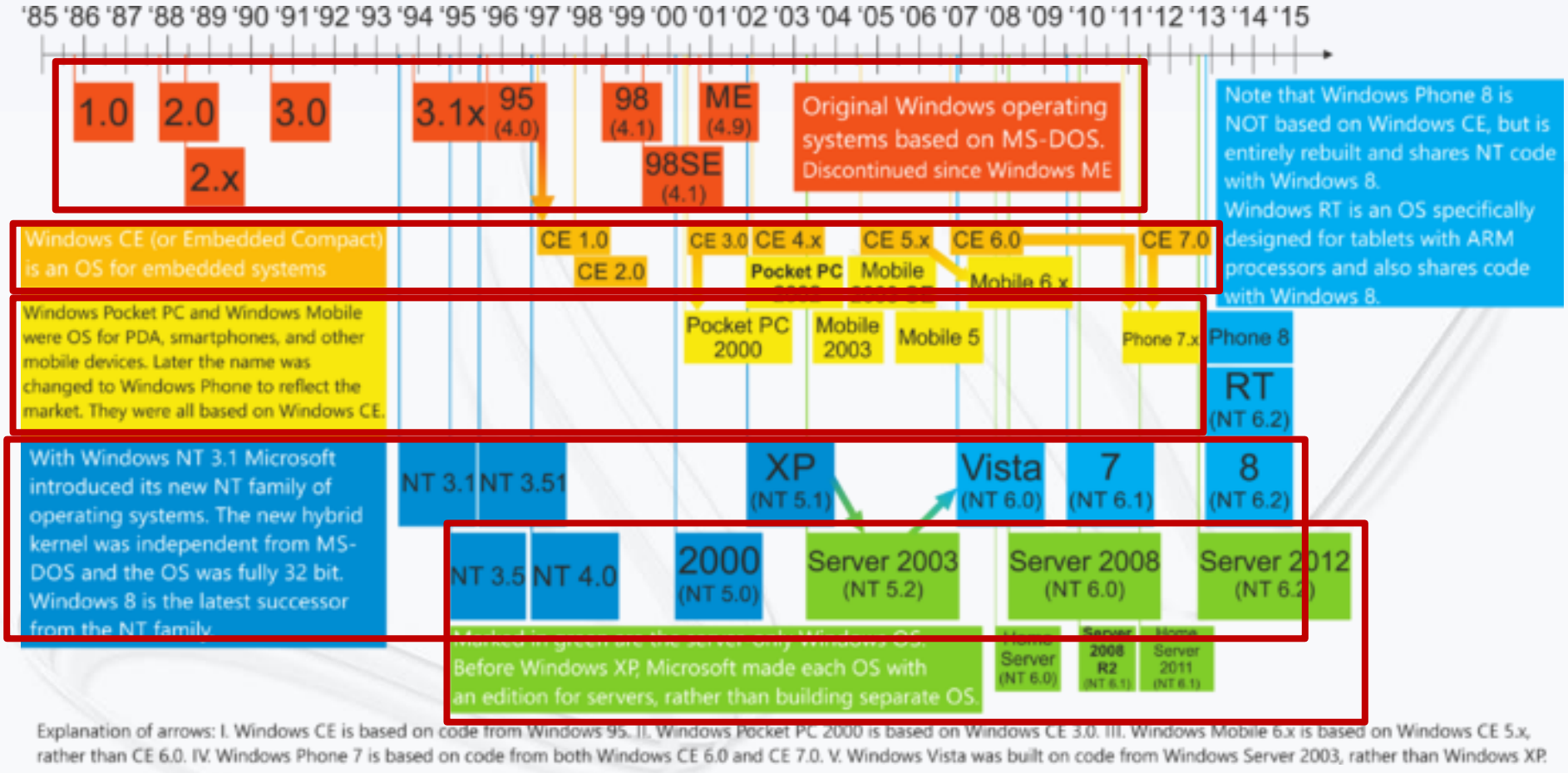


Key Points 1.13

- **History of Windows**



History of Windows



Explanation of arrows: I. Windows CE is based on code from Windows 95. II. Windows Pocket PC 2000 is based on Windows CE 3.0. III. Windows Mobile 6.x is based on Windows CE 5.x, rather than CE 6.0. IV. Windows Phone 7 is based on code from both Windows CE 6.0 and CE 7.0. V. Windows Vista was built on code from Windows Server 2003, rather than Windows XP.

Recall: increasing software complexity

Review 1.13

- **History of Windows**



國立臺灣師範大學
NATIONAL TAIWAN NORMAL UNIVERSITY

Advanced Operating Systems
1.Introduction.14

Chun-Han Lin (林均翰)
CSIE, NTNU



Key Points 1.14

- **OS principles**
- **Why study OS?**
- **Conclusion**



OS Principles (1/2)

- OS as an illusionist
  - Make hardware limitations go away
  - Provide illusion of dedicated machine with infinite memory and infinite processors
- OS as a government
  - Protect users from each other
  - Allocate resources efficiently and fairly



OS Principles (2/2)

- OS as a complex system
  - Constant tension between simplicity and functionality or performance
- OS as a history teacher
  - Learn from the past
  - Adapt as hardware tradeoffs change



Why study OS? (1/2)

- **Learn how to build complex systems**
  - **How can you manage complexity for future projects?**
- **Engineering issues**
  - **Why is a web so slow sometimes? Can you fix it?**
  - **What features should be in the next Mars Rover?**
  - **How do large distributed systems work? Kazaa, …**



Why study OS? (2/2)

- **Buying and using a personal computer**
  - **Why do different PC with same CPU behave differently?**
  - **How to choose a processor (Opteron, Itanium, Celeron, Pentium, Hexium)? Ok, made last one up.**
  - **Should you get Windows XP, 2000, Linux, Mac OS, …?**
  - **Why does Microsoft have such a bad name?**
- **Business issues**
  - **Should your division buy thin-clients vs PC?**
- **Security, viruses, and worms**
  - **What exposure do you have to worry about?**



“In conclusion…” (1/2)

- OS provide a virtual machine abstraction to handle diverse hardware.
- OS coordinate resources and protect users from each other.
- OS simplify application development by providing standard services.
- OS can provide an array of fault containment, fault tolerance, and fault recovery.



“In conclusion…” (2/2)

- AOS combines things from many other areas of computer science.
  - Programming languages, data structures, hardware, and algorithms
- AOS also introduces you to real code development.
  - GDB, source control, real implementation in Linux code base!



Conclusion

- **Studying OS is really about the hardware/software interface (API).**

- **Thus, we will hope to give you enough knowledge to**
  - **Understand this interface**
  - **Modify this interface**
  - **Change the support underneath the interface**



Review 1.14

- OS principles
- Why study OS?
- Conclusion