PKU--LL104-抽象层次

阶段一：抽象与抽象层次

- Abstraction: https://en.wikipedia.org/wiki/Abstraction_(computer_science)
    - In computer science, abstraction is a technique for managing complexity of computer systems. It works by establishing a level of complexity on which a person interacts with the system, suppressing the more complex details below the current level.
    - Abstraction can apply to control or to data:
        - Control abstraction is the abstraction of actions while data abstraction is that of data structures.
        - Control abstraction involves the use of subprograms and related concepts control flows
        - Data abstraction allows handling data bits in meaningful ways. For example, it is the basic motivation behind datatype.
        - One can view the notion of an object as a way to combine abstractions of data and code.
    - Abstraction layer: https://en.wikipedia.org/wiki/Abstraction_layer
        - In computing, an abstraction layer or abstraction level is a way of hiding the implementation details of a particular set of functionality, allowing the separation of concerns to facilitate interoperability and platform independence.
        - Software models that use layers of abstraction include the OSI 7-layer model for computer network protocols, the OpenGL graphics drawing library, and the byte stream input/output (I/O) model originated by Unix and adopted by MS-DOS, Linux, and most other modern operating systems.
    - **All problems in computer science can be solved by another level of indirection; "...except for the problem of too many layers of indirection."**
        - 前半句：A famous aphorism（格言，警句） of David Wheeler
        - 后半句：Kevlin Henney's corollary（推论，必然的结果）
        - 思考：优缺点
        - Abstraction inversion: https://en.wikipedia.org/wiki/Abstraction_inversion
            - In computer programming, abstraction inversion is an anti-pattern arising when users of a construct need functions implemented within it but not exposed by its interface. The result is that the users re-implement the required functions in terms of the interface, which in its turn uses the internal implementation of the same functions.
            - 抽象倒置（Abstraction inversion）：不把用户需要的功能直接提供出来，导致他们要用更上层的函数来重复实现
        - Leaky abstraction: https://en.wikipedia.org/wiki/Leaky_abstraction
            - In software development, a leaky abstraction is an implemented abstraction where details and limitations of the implementation leak through
- HAL (Hardware abstraction layer): https://en.wikipedia.org/wiki/Hardware_abstraction
    - a layer of software that hides hardware differences from higher level programs
    - They often allow programmers to write device-independent, high performance applications by providing standard Operating System (OS) calls to hardware.
    - Device independence: https://en.wikipedia.org/wiki/Device_independence
        - Device independence is the process of making a software application be able to function on a wide variety of devices regardless of the local hardware on which the software is used.
- Platform independence: https://en.wikipedia.org/wiki/Cross-platform
    - In computing, cross-platform, or multi-platform, is an attribute conferred to computer software or computing methods and concepts that are implemented and inter-operate on multiple computer platforms. The software and methods are also said to be platform independent.
    - Cross-platform software may be divided into two types;
        - one requires individual building or compilation for each platform that it supports, and the other one can be directly run on any platform without special preparation, e.g., software written in an interpreted language or pre-compiled portable bytecode for which the interpreters or run-time packages are common or standard components of all platforms.
- Virtual Machine: https://en.wikipedia.org/wiki/Virtual_machine
    - A virtual machine (VM) is a software implementation of a machine (for example, a computer) that executes programs like a physical machine.
    - Virtual machines are separated into two major classes, based on their use and degree of correspondence to any real machine:
        - A system virtual machine provides a complete system platform which supports the execution of a complete operating system (OS).
        - A process virtual machine (also, language virtual machine) is designed to run a single program, which means that it supports a single process.
- API and ABI

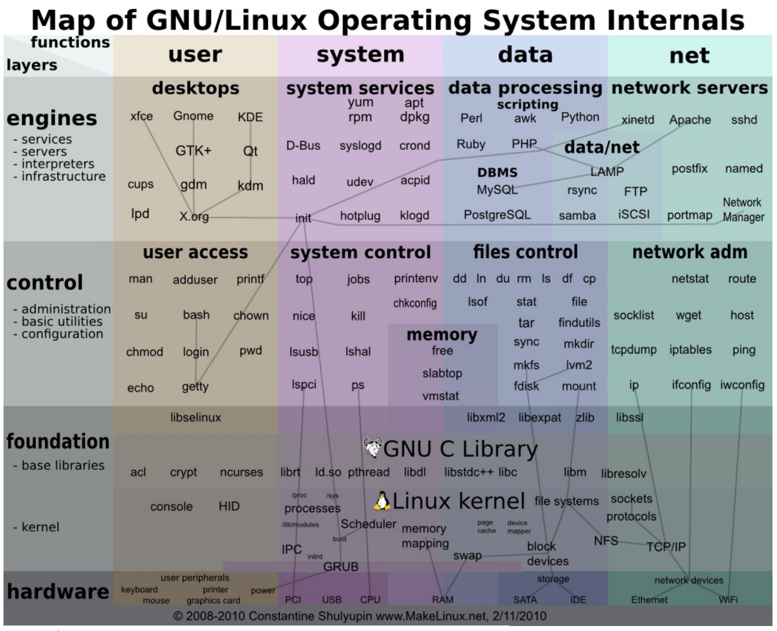
    - API: https://en.wikipedia.org/wiki/Application_programming_interface
        - In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types.
        - APIs often come in the form of a library that includes specifications for routines, data structures, object classes, and variables.
        - An API specification can take many forms, including an International Standard, such as POSIX, vendor documentation, such as the Microsoft Windows API, or the libraries of a programming language, e.g., the Standard Template Library in C++ or the Java APIs.
        - POSIX: https://en.wikipedia.org/wiki/POSIX
            - POSIX (/ p z ks/ POZ-iks), an acronym for Portable Operating System Interface, is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems.
            - POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.
    - ABI: https://en.wikipedia.org/wiki/Application_binary_interface
        - In computer software, an application binary interface (ABI) is the interface between two program modules, one of which is often a library or operating system, at the level of machine code.
        - An ABI determines such details as how functions are called and in which binary format information should be passed from one program component to the next, or to the operating system in the case of a system call.
    - An API differs from an application binary interface (ABI) in that an API is source code-based while an ABI is a binary interface. For instance POSIX is an API, while the Linux Standard Base provides an ABI.
- Linux kernel interfaces
    - https://en.wikipedia.org/wiki/Linux_kernel_interfaces
    - The Linux kernel provides several interfaces to user-space applications that are used for different purposes and that have different properties by design.
    - There are two types of application programming interface (API) in the Linux kernel that are not to be confused: the "kernel–user space" API and the "kernel internal" API.
    - Linux API: (kernel-user space API)
        - which allows programs in user space to access system resources and services of the Linux kernel
        - The Linux API, by choice, has been kept stable over the decades and never breaks; this stability guarantees the portability of source code.
        - the combination of the Linux kernel System Call Interface and glibc is what builds the Linux API
    - In-kernel APIs
        - There are a lot of kernel-internal APIs for all the subsystems to interface with one another.
        - These are being kept fairly stable, but there is no guarantee for stability.
        - In case new research or insights make a change seem favorable, an API is changed, all necessary rewrite and testing have to be done by the author.
        - The Linux kernel is a monolithic kernel, hence device drivers are kernel components. To ease the burden of companies maintaining their (proprietary) device drivers out-of-tree, stable APIs for the device drivers have been repeatedly requested. The Linux kernel developers have repeatedly denied guaranteeing stable in-kernel APIs for device drivers.

自学：

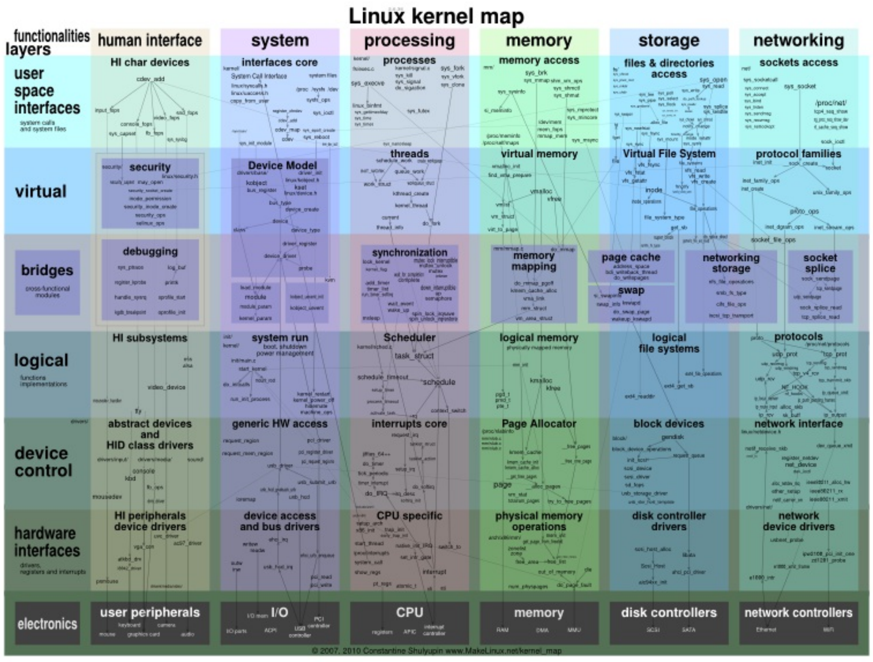
- Abstraction: https://en.wikipedia.org/wiki/Abstraction_(computer_science)
- Abstraction layer: https://en.wikipedia.org/wiki/Abstraction_layer
- Abstraction inversion: https://en.wikipedia.org/wiki/Abstraction_inversion
- Leaky abstraction: https://en.wikipedia.org/wiki/Leaky_abstraction
- Hardware abstraction: https://en.wikipedia.org/wiki/Hardware_abstraction
- Device independence: https://en.wikipedia.org/wiki/Device_independence
- Cross-platfrom: https://en.wikipedia.org/wiki/Cross-platform
- Virtual Machine: https://en.wikipedia.org/wiki/Virtual_machine
- API: https://en.wikipedia.org/wiki/Application_programming_interface
- POSIX: https://en.wikipedia.org/wiki/POSIX
- ABI: https://en.wikipedia.org/wiki/Application_binary_interface
- Linux kernel interfaces: https://en.wikipedia.org/wiki/Linux_kernel_interfaces

See Also:

- ABI Compliance Checker: http://en.wikipedia.org/wiki/ABI_Compliance_Checker
- Documentation/stable_api_nonsense.txt
- Documentation/ABI/ directory



上图来源：http://www.makelinux.net/system/new



上图来源：http://www.makelinux.net/kernel_map/

阶段二：系统调用

- 问题：如何确定一个系统服务的调用接口？
  - 在设计系统调用接口时应该遵循接口设计的三个指导原则，即简单性、完整性和高效性
    - 简单性 ：一个简单的接口更易于理解，实现起来也更易于控制，可以减少错误的发生。
    - 完整性 ：接口可以完成用户所需要的每一个任务；操作系统应该刚好做到需要它做的事情，而没有多余的功能
    - 高效性：如果一个系统调用不能被高效地实现，那么可能并不需要这个系统调用
  - 需要考虑的其它因素
    - 最少的机制
    - 不要隐藏力量
    - 面向连接和无连接
    - 可见度
- Privilege: https://en.wikipedia.org/wiki/Privilege_(computing)
  - In computing, privilege is defined as the delegation of authority over a computer system. A privilege is a permission to perform an action. Examples of various privileges include the ability to create a file in a directory, or to read or delete a file, access a device, or have read or write permission to a socket for communicating over the Internet.
  - Users who have been delegated extra levels of control are called privileged. Users who lack most privileges are defined as unprivileged, regular, or normal users.
- Principle of least privilege: https://en.wikipedia.org/wiki/Principle_of_least_privilege
  - In information security, computer science, and other fields, the principle of least privilege (also known as the principle of minimal privilege or the principle of least authority) requires that in a particular abstraction layer of a

computing environment, every module (such as a process, a user or a program depending on the subject) must be able to access only the information and resources that are necessary for its legitimate purpose.
    - The principle of least privilege is widely recognized as an important design consideration in enhancing the protection of data and functionality from faults (fault tolerance) and malicious behavior (computer security).
- System call: https://en.wikipedia.org/wiki/System_call
    - In computing, a system call is how a program requests a service from an operating system's kernel.
    - This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling.
    - System calls provide an essential interface between a process and the operating system.
    - Generally, systems provide a library or API that sits between normal programs and the operating system. On Unix-like systems, that API is usually part of an implementation of the C library (libc), such as glibc, that provides wrapper functions for the system calls, often named the same as the system calls they invoke.
    - vDSO : https://en.wikipedia.org/wiki/VDSO
        - vDSOs (virtual dynamically linked shared objects) are a way to export kernel space routines to user space applications, using standard mechanisms for linking and loading i.e. standard Executable and Linkable Format (ELF) format.
        - It helps to reduce the calling overhead on simple kernel routines, and also can work as a way to select the best system call method on some architectures.
- CPU modes: https://en.wikipedia.org/wiki/CPU_modes
    - CPU modes (also called processor modes, CPU states, CPU privilege levels and other names) are operating modes for the central processing unit of some computer architectures that place restrictions on the type and scope of operations that can be performed by certain processes being run by the CPU.
    - This design allows the operating system to run with more privileges than application software. Ideally, only highly trusted kernel code is allowed to execute in the unrestricted mode; everything else (including non-supervisory portions of the operating system) runs in a restricted mode and must use a system call to request the kernel perform on its behalf.
    - Mode types:
        - Kernel mode (master mode, supervisor mode, privileged mode, supervisor state, etc.)
        - User mode (slave mode, problem state, etc.)
            - Some CPU architectures support multiple user modes, often with a hierarchy of privileges. These architectures are often said to have ring-based security, wherein the hierarchy of privileges resembles a set of concentric rings, with the kernel mode in the center.
    - Protection ring: https://en.wikipedia.org/wiki/Protection_ring
        - In computer science, hierarchical protection domains, often called protection rings, are mechanisms to protect data and functionality from faults (by improving fault tolerance) and malicious behaviour (by providing computer security).
        - This approach is diametrically opposite to that of capability-based security.
    - Capability-based security: https://en.wikipedia.org/wiki/Capability-based_security
        - Capability-based security is a concept in the design of secure computing systems, one of the existing security models.
        - A capability (known in some systems as a key) is a communicable, unforgeable token of authority. It refers to a value that references an object along with an associated set of access rights. A user program on a capability-based operating system must use a capability to access an object.
        - Capability-based security is to be contrasted with an approach that uses hierarchical protection domains.
- strace: https://en.wikipedia.org/wiki/Strace

    - strace is a diagnostic, debugging and instructional userspace utility for Linux. It is used to monitor interactions between processes and the Linux kernel, which include system calls, signal deliveries, and changes of process state. The operation of strace is made possible by the kernel feature known as ptrace.
    - strace经常被用来作为追查程序问题的工具，但它的功能远非如此。它可以解析和记录进程的系统调用行为，这使得strace成为了一个非常有用的诊断、调查和纠错工具。举例来说，用strace可以追查到一个程序在启动之初所需加载的配置文件信息。当然，strace也有它自身的缺陷，那就是strace会严重拖慢调查对象（某个进程）的性能和运行速度。

自学：

- Privilege: https://en.wikipedia.org/wiki/Privilege_(computing)
- Principle of least privilege: https://en.wikipedia.org/wiki/Principle_of_least_privilege
- System call: https://en.wikipedia.org/wiki/System_call
- vDSO : https://en.wikipedia.org/wiki/VDSO
- CPU modes: https://en.wikipedia.org/wiki/CPU_modes
- Protection ring: https://en.wikipedia.org/wiki/Protection_ring
- Capability-based security: https://en.wikipedia.org/wiki/Capability-based_security
- strace: https://en.wikipedia.org/wiki/Strace

See also：

- On vsyscalls and the vDSO
    - http://lwn.net/Articles/446528/

阶段三：兼容性

- IBM PC compatible: https://en.wikipedia.org/wiki/IBM_PC_compatible
    - IBM PC compatible computers are those similar to the original IBM PC, XT, and AT and able to run the same software as those. Such computers used to be referred to as PC clones, or IBM clones.
- Compatibility layer：https://en.wikipedia.org/wiki/Compatibility_layer
    - In software engineering, a compatibility layer is an interface that allows binaries for a legacy or foreign system to run on a host system. This translates system calls for the foreign system into native system calls for the host system. With some libraries for the foreign system, this will often be sufficient to run foreign binaries on the host system. A hardware compatibility layer consists of tools that allow hardware emulation.
- Computer compatibility：https://en.wikipedia.org/wiki/Computer_compatibility
    - A family of computer models is said to be compatible if certain software that runs on one of the models can also be run on all other models of the family. The computer models may differ in performance, reliability or some other characteristic. These differences may affect the outcome of the running of the software.
        - Software compatibility and Hardware compatibility
    - Backward compatibility and forward-compatible
        - In telecommunications and computing, a product or technology is backward compatible (BC) or downward compatible if it can work with input generated by an older product or technology such as a legacy system.
        - Forward compatibility is the ability of a design to gracefully accept input intended for later versions of itself.
    - Source code compatibility and Binary code compatibility
        - Source code compatibility (source compatible) means that a program can run on computers (or operating systems), independently of binary code compatibility and that the source code is needed for portability.
        - Binary code compatibility (binary compatible or object code compatible) is a property of computer systems, that means they can run the same executable code, typically machine code for a general purpose computer CPU. Source code compatibility, on the other hand, means that recompilation is necessary.
    - Bug compatibility
        - Computer hardware or software is said to be bug compatible if it exactly replicates even an undesirable feature of a previous version. The phrase is found in the Jargon File.
- Legacy system: https://en.wikipedia.org/wiki/Legacy_system
    - In computing, a legacy system is an old method, technology, computer system, or application program, "of, relating to, or being a previous or outdated computer system." Often a pejorative term, referencing a system as "legacy" often implies that the system is out of date or in need of replacement.
        - pejorative 英 [p'd rtv]   美 [p'd rtv]   adj. 轻蔑的；[临床] 恶化的，变坏的 n. 轻蔑语
    - Legacy code:
        - Legacy code is source code that relates to a no-longer supported or manufactured operating system or other computer technology. The term can also mean code inserted into modern software for the purpose of maintaining an older or previously supported feature — for example supporting a serial interface even though many modern systems do not have a serial port.
        - "Legacy code" often differs from its suggested alternative by actually working and scaling.（by Bjarne Stroustrup，a computer scientist and creator of the C++ programming language.）
            - 对待遗留代码一定要慎之又慎："遗留代码"通常在实际工作和可伸缩性上与被建议的替代代码并不相同
- Software portability：https://en.wikipedia.org/wiki/Software_portability
    - Portability in high-level computer programming is the usability of the same software in different environments. The prerequirement for portability is the generalized abstraction between the application logic and system interfaces. When software with the same functionality is produced for several computing platforms, portability is the key issue for development cost reduction.
    - Porting: https://en.wikipedia.org/wiki/Porting
        - In software engineering, porting is the process of adapting software so that an executable program can be created for a computing environment that is different from the one for which it was originally designed (e.g. different CPU, operating system, or third party library). The term is also used when software/hardware is changed to make them usable in different environments.
        - Software is portable when the cost of porting it to a new platform is less than the cost of writing it from scratch. The lower the cost of porting software, relative to its implementation cost, the more portable it is said to be.

自学：

- IBM PC compatible: https://en.wikipedia.org/wiki/IBM_PC_compatible
- Compatibility layer：https://en.wikipedia.org/wiki/Compatibility_layer
- Computer compatibility：https://en.wikipedia.org/wiki/Computer_compatibility
- Backward compatibility: https://en.wikipedia.org/wiki/Backward_compatibility
- Forward compatibility: https://en.wikipedia.org/wiki/Forward_compatibility
- Source code compatibility: https://en.wikipedia.org/wiki/Source_code_compatibility
- Binary code compatibility: https://en.wikipedia.org/wiki/Binary_code_compatibility
- Bug compatibility：https://en.wikipedia.org/wiki/Bug_compatibility
- Legacy system: https://en.wikipedia.org/wiki/Legacy_system
- Legacy code: https://en.wikipedia.org/wiki/Legacy_code

- Software portability：https://en.wikipedia.org/wiki/Software_portability
- Porting: https://en.wikipedia.org/wiki/Porting

阶段四：内核中的头文件

- 内核头文件的位置
    - include dir
        - 主要目录：linux、uapi、asm-generic
        - 自动生成的：config、generated
        - 其它目录
    - arch/*/include dir
        - 每个体系都会有的：asm和uapi/asm
        - 部分体系特有的：arch-*，cpu-*，mach-*
        - 自动生成的：generated
    - 驱动自己需要的头文件，通常和驱动的源代码放在一起
    - 注意区分：<header.h>和"header.h"的区别
    - 注意如何保护头文件不被重复引用：
        - #ifndef __FILENAME_H__
        - #define __FILENAME_H__
        - ...
        - #endif
    - 注意调用的层次：
        - 先linux，再asm
        - 不要直接使用asm-generic
        - 注意避免嵌套调用
- The UAPI header file split: http://lwn.net/Articles/507794/
    - What David wants to do is to split out the user-space API content of the kernel header files in the include and arch/xxxxxx/include directories, placing that content into corresponding headers created in new uapi/ subdirectories that reside under each of the original directories.
    - As well as being a step toward solving his original problem and performing a number of other useful code cleanups, David notes that disintegrating the header files has many other benefits.
        - It simplifies and reduces the size of the kernel-only headers.
        - More importantly, splitting out the user-space APIs into separate headers has the desirable consequence that it "simplifies the complex interdependencies between headers that are [currently] partly exported to userspace".
        - By placing all of the user-space API-related definitions into files dedicated solely to that task, it becomes easier to track changes to the APIs that the kernel presents to user space.
    - User-space API: (kernel-user interface)
        - make headers_install
        - 查看：usr/include
        - 对比：/usr/include
            - diff -urN include/uapi/linux usr/include/linux
            - diff -urN usr/include/linux /usr/include/linux（只缺少了命令文件）
        - asm目录：被移动到了x86_64-linux-gnu目录下
            - diff -urN arch/x86/include/uapi/asm usr/include/asm
            - diff -urN usr/include/asm /usr/include/x86_64-linux-gnu/asm
- asm-generic
    - The directory include/asm-generic contains the header files that may be shared between individual architectures.
    - The recommended approach how to use a generic header file is to list the file in the Kbuild file.
    - See Documentation/kbuild/makefiles.txt：6.10 and 7.4
- Warning in LFS 8.3:
    - The headers in the system's include directory (/usr/include) should ***always*** be the ones against which Glibc was compiled, that is, the sanitised headers installed in Section 6.7, “Linux-4.1.2 API Headers”. Therefore, they should ***never*** be replaced by either the raw kernel headers or any other kernel sanitized headers.
    - See Also:
        - http://www.linuxfromscratch.org/lfs/view/development/chapter06/linux-headers.html
        - http://www.linuxfromscratch.org/lfs/view/development/chapter08/kernel.html

---