System Services

Another aspect of a modern system is it's collection of system services also known as a system utilities that providing a convenient environment for program development and execution. They can be devided into these categories:

File managment. The programs manipulate files and directories.

Status information. These programs format and print the output to the terminal or other output devices or files or display. Some systems also support a registry, which is used to store and retrieve configuration information.

File modification. Text editors or special commands may be avaible to create and modify the content of files stored on disk.

Programming-language support. Compilers, assemblers, debuggers are often provided with the OS.

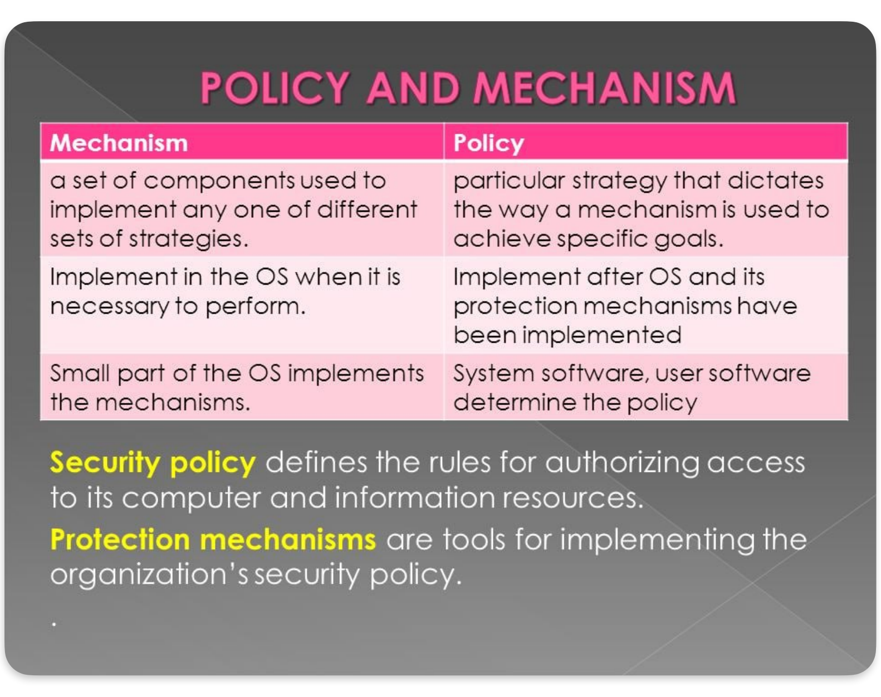
Program loading and execution. The system may provide absolute loaders relocatable loaders, linkage editors and overlay loaders to load compiled program into memory and execute it.

Communications. These programs provide the mechanism for creating virtual connections among processes, users and computer systems.

Background services. Constantly running system-program processes are known as services, subsystems or daemons. - certain system-program processes that system is launching at boot time.

Mechanism and policies

One important principle is the separation of policy from mechanism. Mechanism determine **how** to do something; policies determine **what** will be done. The separation is important for flexibility. Policies are likely to change across places or over time. In the worst case, each change in policy would require a change in the underlying mechanism. A general mechanism flexible enough to work across a range of policies is preferable.



POLICY AND MECHANISM

Mechanism	Policy
a set of components used to implement any one of different sets of strategies.	particular strategy that dictates the way a mechanism is used to achieve specific goals.
Implement in the OS when it is necessary to perform.	Implement after OS and its protection mechanisms have been implemented
Small part of the OS implements the mechanisms.	System software, user software determine the policy

Security policy defines the rules for authorizing access to its computer and information resources.

Protection mechanisms are tools for implementing the organization's security policy.

.

Operating-System Structure

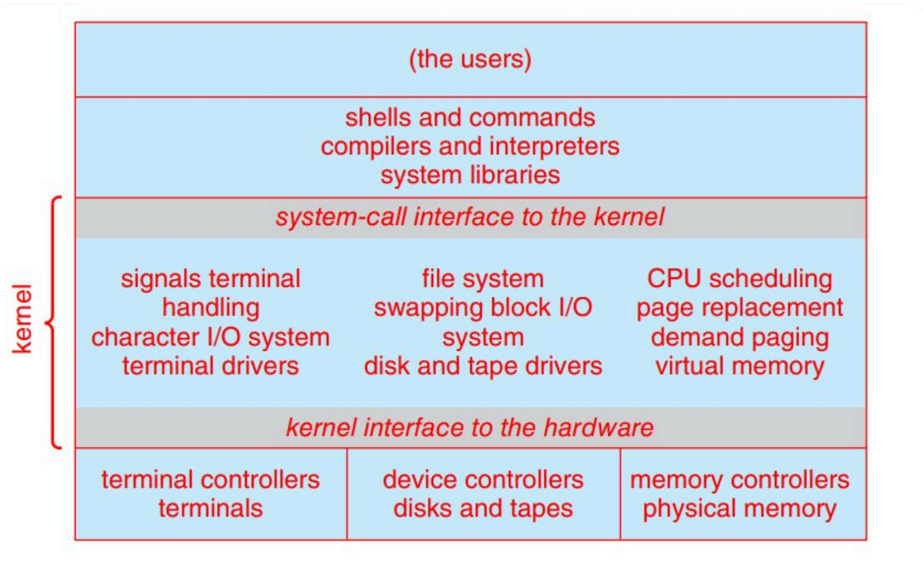

(the users)
shells and commands compilers and interpreters system libraries
system-call interface to the kernel
kernel
signals terminal handling character I/O system terminal drivers
file system swapping block I/O system disk and tape drivers
CPU scheduling page replacement demand paging virtual memory
kernel interface to the hardware
terminal controllers terminals
device controllers disks and tapes
memory controllers physical memory

Figure 2.12 Traditional UNIX system structure.

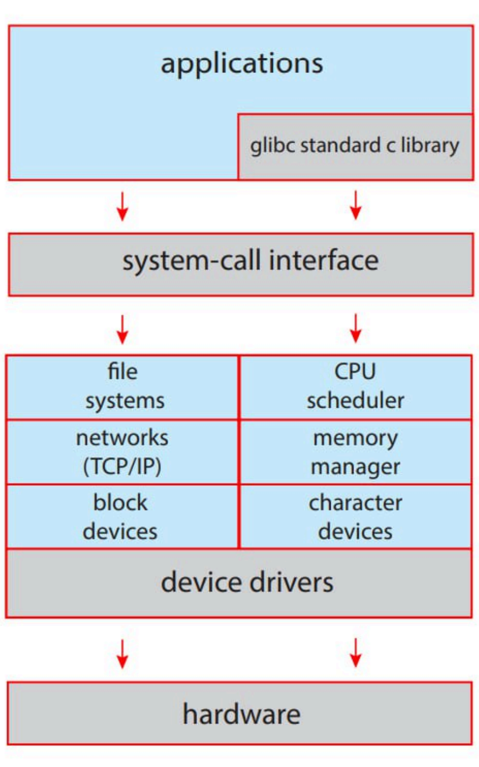


Figure 2.13 Linux system structure.

Monolitic structure - place all of the functionality of the kernel into a single, static binary file that runs in a single address space. Despite the apparent simplicity of monolithic kernels, they are difficult to implement and extend. Monolithic kernels do have a distinct perfomance however: there is very little overhead in the system-call interface, and communication within the kernel is fast.

Layered approach. The monolithic approach is often known as tightly coupled. Alternatively, we could design a loosely coupled system that is divided into separate smaller components that have specific and limited functionality. All these components together comprise the kernel. The advantage is that changes in one component affect only that component and no others.

A system can be made modular in many ways. One method is the layered approach. Each layer is implemented only with operations provided by lower-level layers.

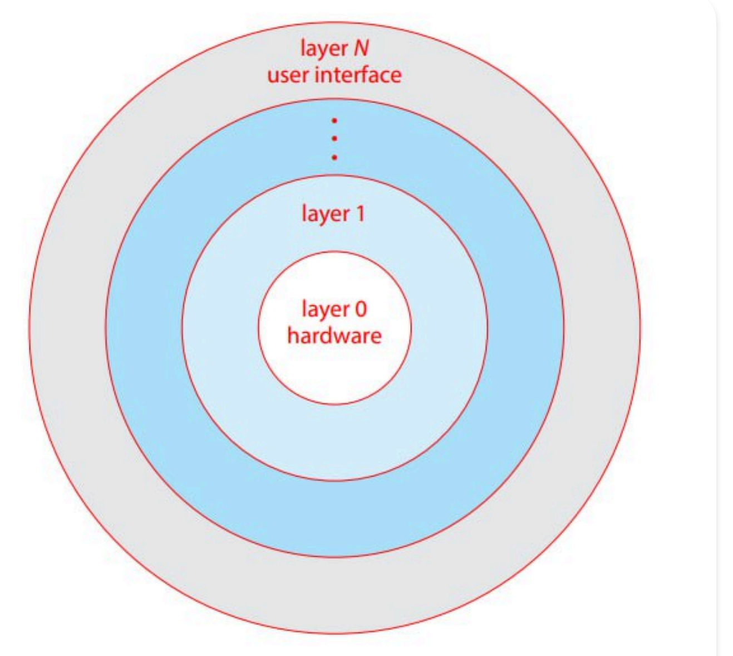


Figure 2.14 A layered operating system.

Micro kernels. This method structures the OS by removing all nonessential components from the kernel and implementing them as user-level programs that reside in separate address space. The main function of the micro-kernel is to provide communication between the client program and the various services that are also running in user space. Communication is provided through message passing. The benefit of the microkernel approach is that it makes extending the OS easier. The microkernel also provides more security and reliability, since most services are running as user processes. But the perfomance of micro-kernels can suffer due to increased system-function overhead. When two user-level services must communicate, messages must be copied between the services, which reside in separate address spaces.

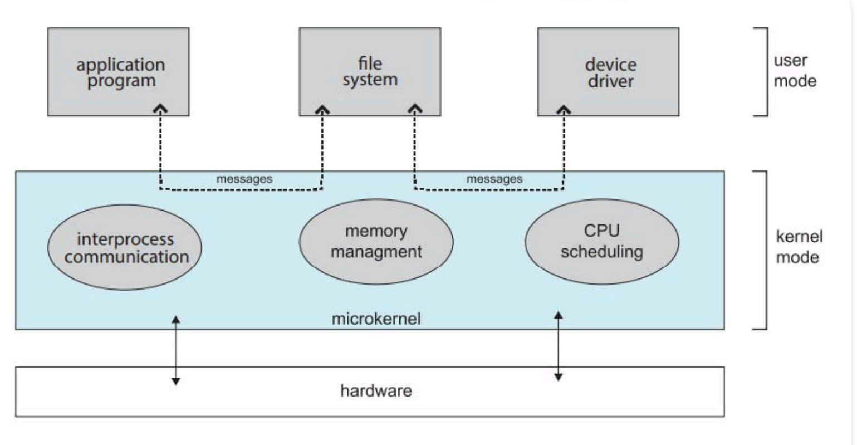
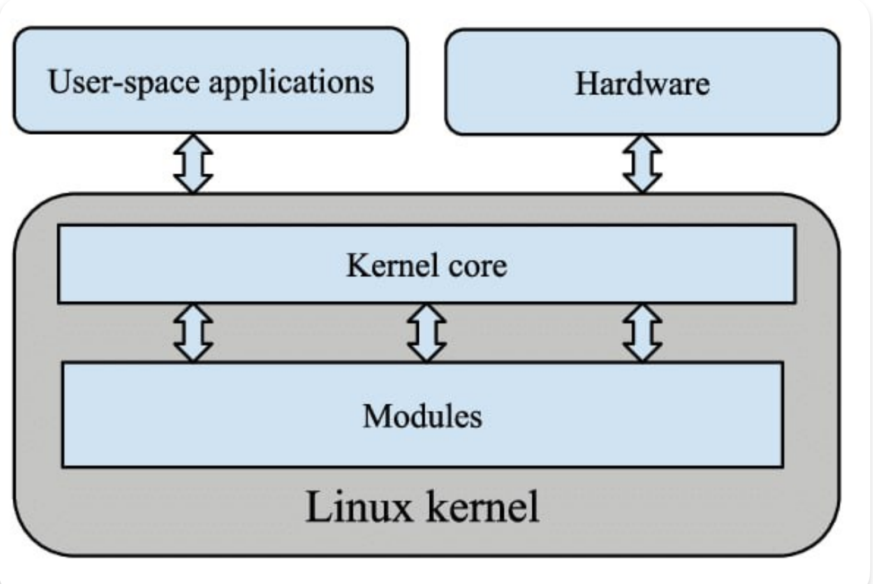


Figure 2.15 Architecture of a typical microkernel.

Modules. The best current methodology for OS design involves using loadable kernel modules LKMs. Here, the kernel has a set of core components and can link in additional services via modules, either at boot time or during runtime. The idea of the design is for the kernel to provide core services, while other services are implemented dynamically, as the kernel running. This approach doesn't require to recompile kernel after new service was added.

In practice most of OS combine different structures, resulting in hybrid systems that address perfomance, security, and usability issues.



- An operating system provides an environment for the execution of programs by providing services to users and programs.
- The three primary approaches for interacting with an operating system are (1) command interpreters, (2) graphical user interfaces, and (3) touch-screen interfaces.
- System calls provide an interface to the services made available by an operating system. Programmers use a system call's application programming interface (API) for accessing system-call services.
- System calls can be divided into six major categories: (1) process control, (2) file management, (3) device management, (4) information maintenance, (5) communications, and (6) protection.
- The standard C library provides the system-call interface for UNIX and Linux systems.
- Operating systems also include a collection of system programs that provide utilities to users.
- A linker combines several relocatable object modules into a single binary executable file. A loader loads the executable file into memory, where it becomes eligible to run on an available CPU.
- There are several reasons why applications are operating-system specific. These include different binary formats for program executables, different instruction sets for different CPUs, and system calls that vary from one operating system to another.
- An operating system is designed with specific goals in mind. These goals ultimately determine the operating system's policies. An operating system implements these policies through specific mechanisms.
- A monolithic operating system has no structure; all functionality is provided in a single, static binary file that runs in a single address space. Although such systems are difficult to modify, their primary benefit is efficiency.
- A layered operating system is divided into a number of discrete layers, where the bottom layer is the hardware interface and the highest layer is the user interface. Although layered software systems have had some success, this approach is generally not ideal for designing operating systems due to performance problems.
- The microkernel approach for designing operating systems uses a minimal kernel; most services run as user-level applications. Communication takes place via message passing.
- A modular approach for designing operating systems provides operating-system services through modules that can be loaded and removed during run time. Many contemporary operating systems are constructed as hybrid systems using a combination of a monolithic kernel and modules.
- A boot loader loads an operating system into memory, performs initialization, and begins system execution.
- The performance of an operating system can be monitored using either counters or tracing. Counters are a collection of system-wide or per-process statistics, while tracing follows the execution of a program through the operating system.