Course Title

Operating Systems & Systems Programming

Course No.

CSE - 3201 / CSE - 2205

Operating Systems & Systems Programming

Operating System Structure

- Monolithic Systems
- Layered Systems
- Microkernels
- Client-Server Model
- Virtual Machines
- Exokernels

System Call

When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource. This is done via something called a **system call**

System Call

In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. 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.

Operating System Structure

Monolithic Systems

In this approach the entire OS runs as a single program in kernel mode.

The OS is written as a collection of procedures, linked together into a single large executable program.

Each procedure in the system is free to call any other one, if the latter provides some useful computation that the former needs.

Operating System Structure

Monolithic Systems

Being able to call any procedure is very efficient,

But having thousands of procedures that can call each other without restriction may also lead to a system that is difficult to understand.

Also, a crash in any of these procedures will take down the entire operating system.

Operating System Structure

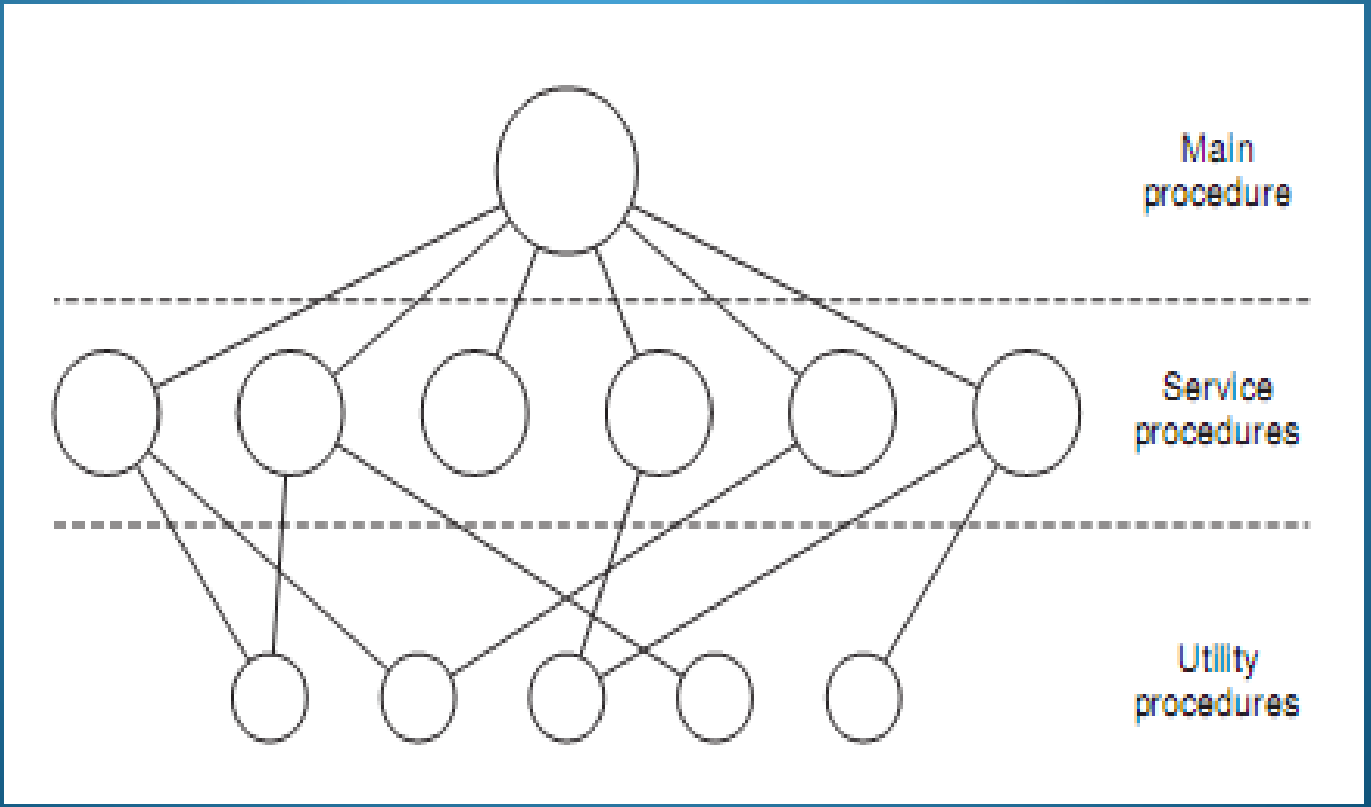
Monolithic Systems

This organization suggests a basic structure for the operating system:

- 1. A main program that invokes the requested service procedure.
- 2. A set of service procedures that carry out the system calls.
- 3. A set of utility procedures that help the service procedures.

Operating System Structure

Monolithic Systems



Operating System Structure

Monolithic Systems

In addition to the core operating system that is loaded when the computer is booted, many operating systems support loadable extensions, such as I/O device drivers and file systems. These components are loaded on demand. In UNIX they are called shared libraries. In Windows they are called DLLs (Dynamic-Link Libraries).

Operating System Structure

Layered Systems

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Operating System Structure

Layered Systems

Layer o dealt with allocation of the processor, switching between processes when interrupts occurred or timers expired

Above layer o, the system consisted of sequential processes, each of which could be programmed without having to worry about the fact that multiple processes were running on a single processor. In other words, layer o provided the basic multiprogramming of the CPU.

Operating System Structure

Layered Systems

Layer 1 did the memory management. It allocated space for processes in main memory.

Layer 2 handled communication between each process and the operator console (that is, the user).

Layer 3 took care of managing the I/O devices and buffering the information streams to and from them.

Operating System Structure

Layered Systems

Layer 4 was where the user programs were found. They did not have to worry about process, memory, console, or I/O management.

The system operator process was located in layer 5.

Operating System Structure

Microkernels

Basic idea behind microkernel design is to achieve high reliability by splitting the operating system up into small, well-defined modules. Only one of which—the microkernel—runs in kernel mode

The rest run as user processes. By running each device driver and file system as a separate user process, a bug in one of these can crash that component, but cannot crash the entire system.

Operating System Structure

Microkernels

Thus a bug in the audio driver will cause the sound to be garbled or stop, but will not crash the computer.

In contrast, in a monolithic system with all the drivers in the kernel, a buggy audio driver can easily reference an invalid memory address and bring the system to a grinding halt instantly.

Operating System Structure

Microkernels

Many microkernels have been implemented and deployed. They are dominant in real-time, industrial, avionics, and military applications that are mission critical and have very high reliability requirements. A few of the better-known microkernels include Symbian, and MINIX 3.

Operating System Structure

Microkernels

The MINIX3 microkernel manages and schedules processes, handles interprocess communication (by passing messages between processes), and offers a set of about 40 kernel calls to allow the rest of the operating system to do its work. These calls perform functions like hooking handlers to interrupts, moving data between address spaces, and installing memory maps for new processes.

Operating System Structure

Microkernels

Outside the kernel, the system is structured as three layers of processes all running in user mode.

The lowest layer contains the device drivers. Since they run in user mode, they do not have physical access to the I/O port space and cannot issue I/O commands directly.

Operating System Structure

Microkernels

Above the drivers is another user-mode layer containing the servers, which do most of the work of the operating system. One or more file servers manage the file system(s), the process server creates, destroys, and manages processes, and so on.

Operating System Structure

Microkernels

User programs obtain operating system services by sending short messages to the servers asking for the POSIX system calls. For example, a process needing to do a read sends a message to one of the file servers telling it what to read.

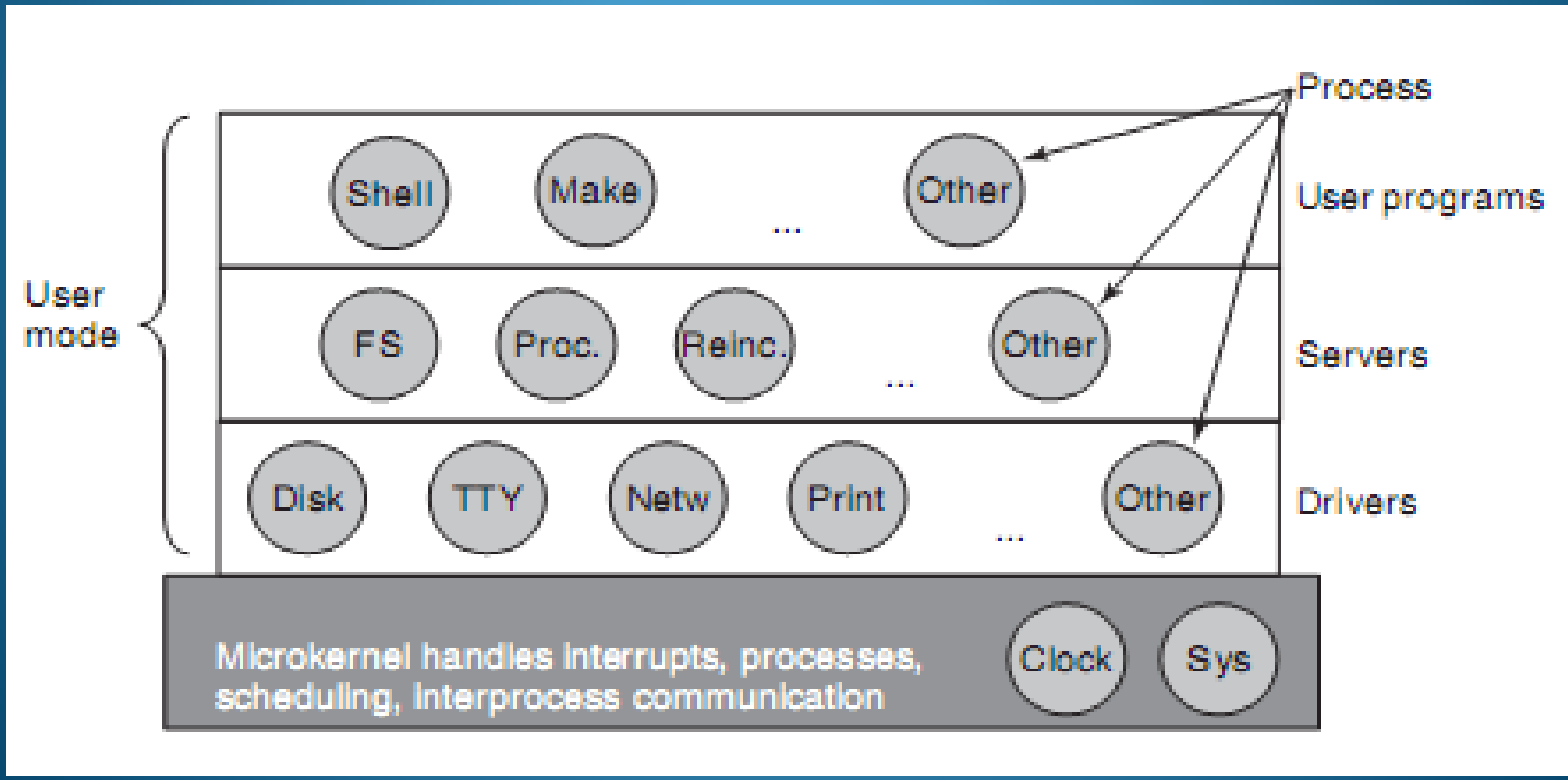
Operating System Structure

Microkernels

One interesting server is the reincarnation server, whose job is to check if the other servers and drivers are functioning correctly. In the event that a faulty one is detected, it is automatically replaced without any user intervention. In this way, the system is self healing and can achieve high reliability.

Operating System Structure

Microkernels



Operating System Structure

Client-Server Model

A slight variation of the microkernel

The idea is to distinguish classes of processes (clients) and servers

Each of servers provides some service

Each of clients use these services.

Operating System Structure

Client-Server Model

Communication between clients and servers is often by message passing.

To obtain a service, a client process constructs a message saying what it wants and sends it to the appropriate service (server).

Operating System Structure

Client-Server Model

The service (server) then does the work and sends back the answer.

If the client and server happen to run on the same machine, certain optimizations are possible, but conceptually, there is still message passing.

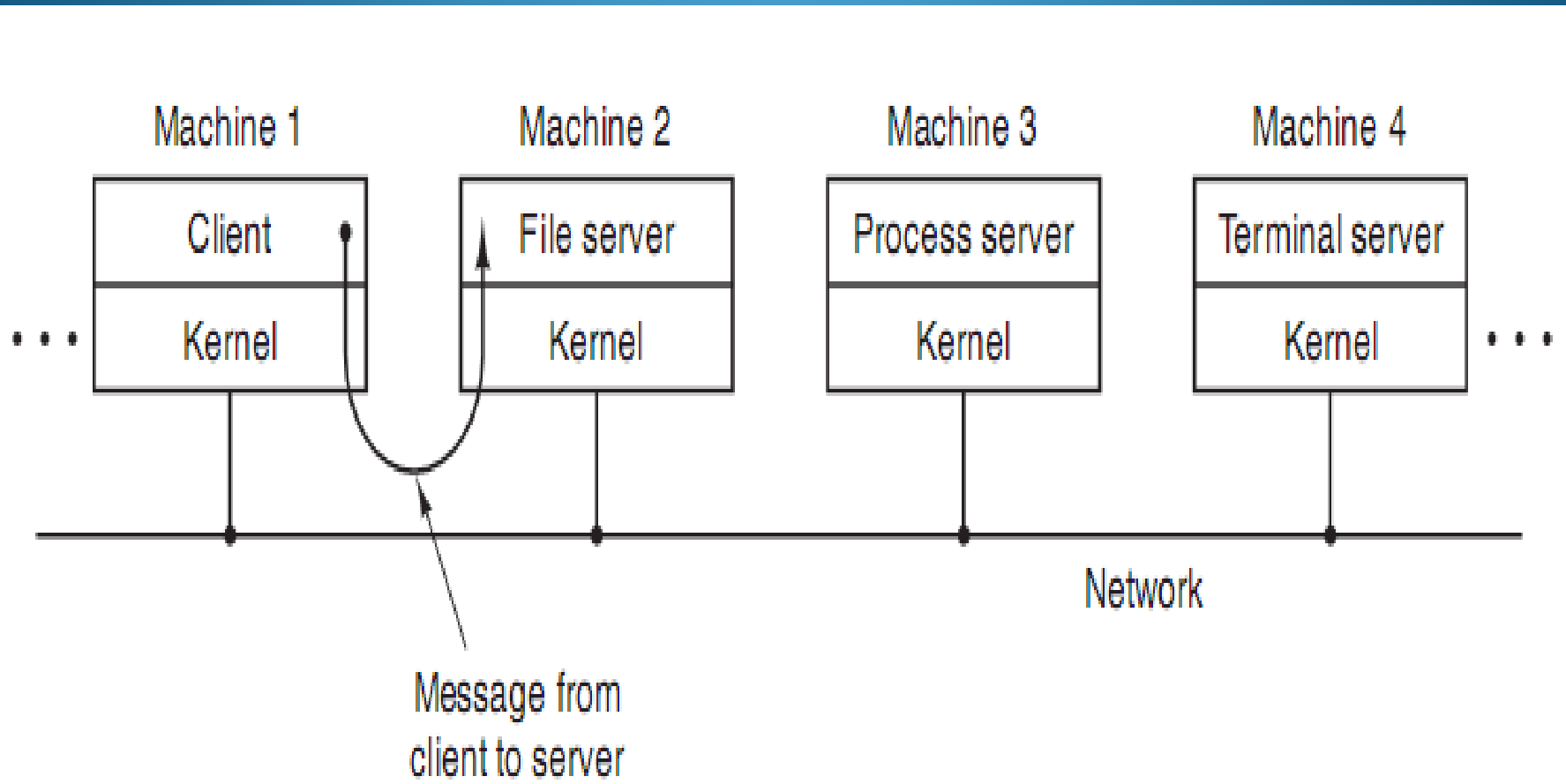
Operating System Structure

Client-Server Model

An obvious generalization of this idea is to have the clients and servers run on different computers, connected by a local or wide-area network.

Operating System Structure

Client-Server Model



Operating System Structure

Client-Server Model

As far as the client is concerned, the same thing happens in both cases: requests are sent and replies come back.

The client-server model is an abstraction that can be used for a single machine or for a network of machines.

.

Operating System Structure

Client-Server Model

Many systems involve users at their home PCs as clients and large machines elsewhere running as servers.

Much of the Web operates this way. A PC sends a request for a Web page to the server and the Web page comes back.

This is a typical use of the client-server model in a network.

Operating System Structure

Virtual Machines

VM/370 (**Virtual Machine Facility/370**) system, originally called CP/CMS (Control Program/Cambridge Monitor **System or** Control Program/Conversational Monitor System) and later renamed VM/370 (Seawright and MacKinnon, 1979), was based on an observation: a timesharing system provides (1) multiprogramming and (2) an extended machine with a more convenient interface than the bare hardware.

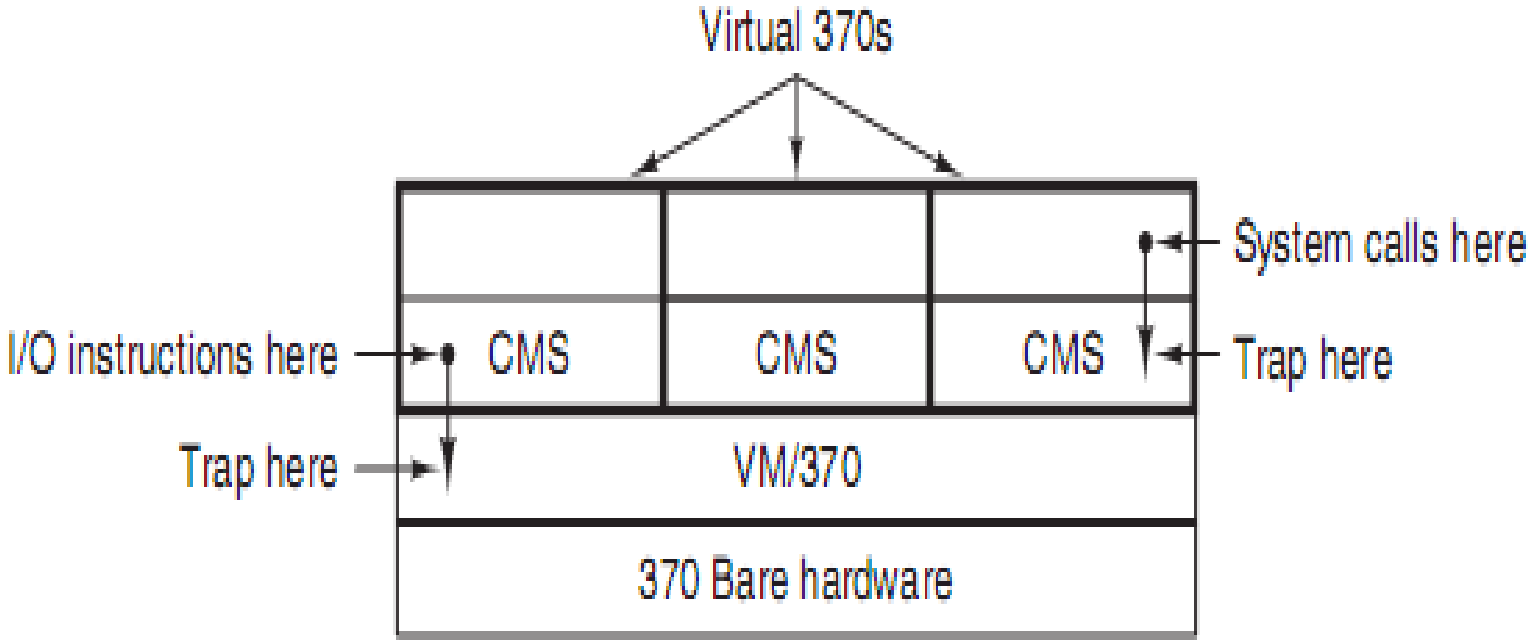
Operating System Structure

Virtual Machines

The heart of the system, known as the virtual machine monitor, runs on the bare hardware and does the multiprogramming, providing not one, but several virtual machines to the next layer up, as shown in Fig (Next Slide).

Operating System Structure

Virtual Machines



The structure of VM/370 with CMS

Operating System Structure

Virtual Machines

However, unlike all other operating systems, these virtual machines are not extended machines, with files and other nice features. Instead, they are exact copies of the bare hardware, including kernel/user mode, I/O, interrupts, and everything else the real machine has.

Operating System Structure

Virtual Machines

Because each virtual machine is identical to the true hardware, each one can run any operating system that will run directly on the bare hardware.

Operating System Structure

Virtual Machines

While IBM has had a virtual-machine product available for four decades, and a few other companies, including Oracle and Hewlett-Packard, have recently added virtual-machine support to their high-end enterprise servers, the idea of virtualization has largely been ignored in the PC world until recently.

But in the past few years, a combination of new needs, new software, and new technologies have combined to make it a hot topic.

Operating System Structure

Virtual Machines

First the needs

Many companies have traditionally run their mail servers, Web servers, FTP servers, and other servers on separate computers, sometimes with different operating systems. They see virtualization as a way to run them all on the same machine without having a crash of one server bring down the rest.

Operating System Structure

Virtual Machines

Virtualization is also popular in the Web hosting world. Without virtualization, Web hosting customers are forced to choose between shared hosting (which gives them a login account on a Web server, but no control over the server software) and dedicated hosting (which gives them their own machine, which is very flexible but not cost effective for small to medium Websites). When a Web hosting company offers virtual machines for rent, a single physical machine can run many virtual machines, each of which appears to be a complete machine.

Operating System Structure

Virtual Machines

Customers who rent a virtual machine can run whatever operating system and software they want to, but at a fraction of the cost of a dedicated server (because the same physical machine supports many virtual machines at the same time).

Operating System Structure

Virtual Machines

Another use of virtualization is for end users who want to be able to run two or more operating systems at the same time, say Windows and Linux, because some of their favorite application packages run on one and some run on the other.

Operating Systems & Systems Programming

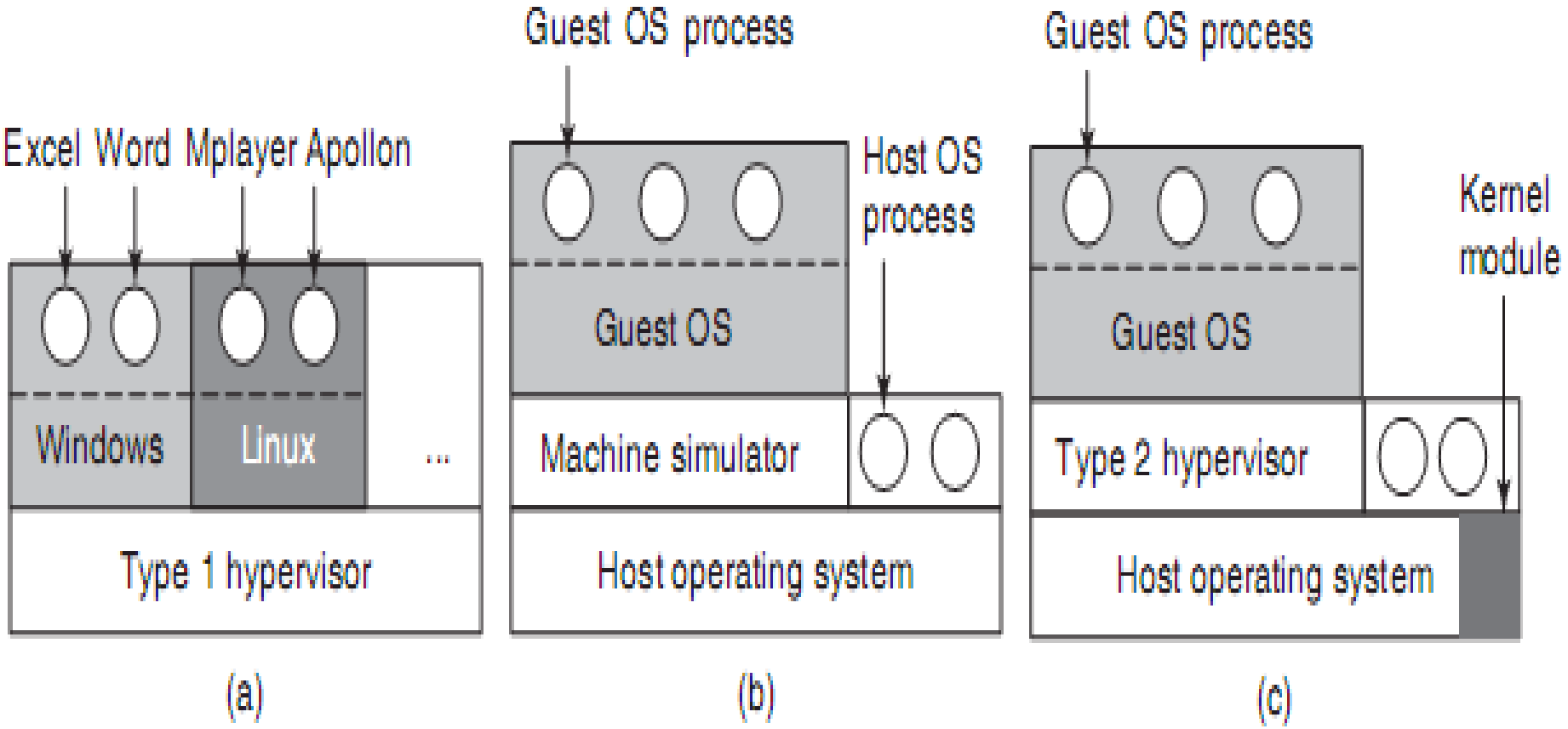


Figure . (a) A type 1 hypervisor. (b) A pure type 2 hypervisor. (c) A practical type 2 hypervisor.

Operating System Structure

Virtual Machines

The Java Virtual Machine

Another area where virtual machines are used, but in a somewhat different way, is for running Java programs. When Sun Microsystems invented the Java programming language, it also invented a virtual machine (i.e., a computer architecture) called the JVM (Java Virtual Machine).

Operating System Structure

Virtual Machines

The Java Virtual Machine

The Java compiler produces code for JVM, which then typically is executed by a software JVM interpreter. The advantage of this approach is that the JVM code can be shipped over the Internet to any computer that has a JVM interpreter and run there.

Operating System Structure

Exokernels

Rather than cloning the actual machine, as is done with virtual machines, another strategy is partitioning it, in other words, giving each user a subset of the resources. Thus one virtual machine might get disk blocks 0 to 1023, the next one might get blocks 1024 to 2047, and so on.

At the bottom layer, running in kernel mode, is a program called the exokernel (Engler et al., 1995).

Operating System Structure

Exokernels

Its job is to allocate resources to virtual machines and then check attempts to use them to make sure no machine is trying to use somebody else's resources.

Each user-level virtual machine can run its own operating system, as on VM/370 and the Pentium virtual 8086s, except that each one is restricted to using only the resources it has asked for and been allocated.

Operating System Structure

Exokernels

The advantage of the exokernel scheme is that it saves a layer of mapping. In the other designs, each virtual machine thinks it has its own disk, with blocks running from o to some maximum, so the virtual machine monitor must maintain tables to remap disk addresses (and all other resources). With the exokernel, this remapping is not needed. The exokernel need only keep track of which virtual machine has been assigned which resource

Operating System Structure
Exokernels

This method still has the advantage of separating the multiprogramming (in the exokernel) from the user operating system code (in user space), but with less overhead, since all the exokernel has to do is keep the virtual machines out of each other's hair.