### Table: System, Distributed Systems, File Systems, Network Security

<table>
<thead>
<tr>
<th>System</th>
<th>Distributed Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>File system</td>
<td>Handling</td>
</tr>
<tr>
<td>Network security</td>
<td>Persistent file systems</td>
</tr>
<tr>
<td>File system</td>
<td>Concurrency with CPU, Interrupts</td>
</tr>
<tr>
<td>Terminal, Mouse, Printer</td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>I/O devices</td>
</tr>
<tr>
<td>Address space</td>
<td>Memory</td>
</tr>
<tr>
<td>Process</td>
<td>Synchronization</td>
</tr>
<tr>
<td>Process management, Scheduling</td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>Synchronization</td>
</tr>
<tr>
<td>Address space</td>
<td></td>
</tr>
</tbody>
</table>

### From the Architecture to the OS to the User: Architectural Resources

- Memory
- Process
- Synchronization
- File management, Secondary Storage Management
- Memory management
- Process
- Synchronization
- File systems
- I/O systems
- Distributed systems
- File system
- Network security
- Distributed systems
- File system

### From the Architecture to the User: OS Services

- Example OS Services
- User Abstraction
- Example OS Services
- User Abstraction

### OS Components

- Networking
- File system
- Process
- Memory management
- Synchronization
- I/O systems
- Distributed systems

### Today: OS Structures & Services

- Architecture & Operating Systems
OS and Processes

Processes are independent entities. Many processes can be running the same program, processes execute, and all the other information the activity needs to run. A process is a program, a process is one instance of a program in etc.) and all the other information the activity needs to run. A process includes the execution context (PC, registers, VM, resources, etc.)

Each of these activities is encapsulated in a process:
- System programs: printers, spoolers, name servers, file servers, network listeners, etc.
- Batch jobs and command scripts
- User programs

The OS manages a variety of activities:

- The OS allocates resources to processes.
- The OS manages inter-process communication and synchronization.
- The OS schedules and manages processes.
- The OS creates, deletes, suspends, and resumes processes.
removed from memory (policies).
- decide how much memory to allocate to each process, and when a process should be
- maintain the mappings from virtual to physical memory (page tables),
- deallocate memory space,
- allocate memory space for processes,

The OS must

Processes must be stored in main memory to execute.

is the direct access storage for the CPU.

Main memory

Memory & Secondary Storage Management

What can happen?

• All of the processes are trying to access the same account simultaneously.

• Balance computation, monthly interest computation and addition.

• Cooperating processes on a single account: ATM machine transaction,

Banking transactions

Synchronization Example:
maintaining mapping information, accounting, and quotas.

The file system also provides general services such as backups,

- map files onto secondary storage
- manipulate (read, write, extend, rename, copy, \texttt{mkdir}) files and directories
- create and delete files and directories

The file system provides a standard interface to

\begin{center}
\textbf{File System management}
\end{center}
Provides device driver implementations specific to individual devices.

- Provides a general device driver interface, hiding the differences among devices.
- Supports buffering and spooling of I/O.
- Supports communication with external devices: terminal, keyboard, printer, mouse.

**I/O Systems**

To execute a program, the OS must load the program from disk into memory.

**Example:** A program executable is stored in a file on disk. To execute a program, the OS must load the program from disk into memory.

- **Secondary Storage = Persistent Memory (endures system failures)**
- **Low-Level OS Routines:** Responsible for low-level disk functions, such as scheduling of disk operations, head movement, and error handling.
- **Secondary Storage:** Responsible for managing the disk space (for example).
- **I/O System:** Supports communication with external devices: terminal, keyboard, printer, mouse.
- **I/O System** supports buffering and spooling of I/O.
Linux

- Debate about what functionality goes into the kernel (above figure).

- Programs.

- The kernel is the protected part of the OS that runs in kernel mode.

---

One Basic OS Structure

---

Distributed Systems

- Independent storage devices.
  - The OS must carry out its file services across the network and manage multiple.
  - Users, servers, and storage devices are all dispersed among the various sites.

  - The OS can support a distributed file system on a distributed system.

  - That are not encountered in a centralized system.

  - The OS must provide additional mechanisms for dealing with failures and deadlock.

  - To use non-local resources in a distributed system, processes must communicate over

    memory or a clock.

  - A distributed system is a collection of processors that do not share
Hierarchical design allows communication overhead between layers, extra copying, bookkeeping.

Advantages:
- Modularity: simplicity, portability, ease of design/implementation
- Modular: less coupling, less complexity

Disadvantages:
- Communication overhead between layers

Layered OS design

Hardware
- Instruction interpreter
- CPU scheduler
- I/O channel
- Virtual memory
- Device drivers
- User programs
acceptable performance (strong reason = simulation or evaluation study).

That you need to make a particular component complicated to achieve rule simplicity except when you have a strong reason to believe a constant tradeoff between simplicity and performance. As a general

Extended?

Big Design Issue: How do we make the OS efficient, reliable, and

Summary

First Micorkernel was Hydra (CMU 70). Current systems include Chorus

(Francis) and Mach (CMU).

- mediate performance (unpredictable)
  - better reliability, easy extension, and customization

Advantages

- Implemementing as much of the OS in User-Level processes as possible.

Goal: to minimize what goes in the kernel (mechanisms, no policy),

Micorkernel Features