OPERATING SYSTEM STRUCTURES

SYSTEM COMPONENTS:

PROCESS MANAGEMENT

A **process** is a **program** in execution: (A program is passive, a process active.)

A process has resources (CPU time, files) and attributes that must be managed.

Management of processes includes:

- Process Scheduling (priority, time management, . . .)
- Creation/termination
- Block/Unblock (suspension/resumption)
- Synchronization
- Communication
- Deadlock handling
- Debugging

MAIN MEMORY MANAGEMENT

- Allocation/deallocation for processes, files, I/O.
- Maintenance of several processes at a time
- Keep track of who's using what memory
- Movement of process memory to/from secondary storage.

SECONDARY STORAGE MANAGEMENT

- Disks, tapes, optical, ...
- Free space management (paging/swapping)
- Storage allocation (what data goes where on disk)
- Disk scheduling

DEVICE MANAGEMENT

- Buffer caching system
- Generic device driver code
- Drivers for each device - translate read/write requests into disk position commands.

FILE MANAGEMENT

Keep track of what's on secondary storage.

file == logical entity, disk == physical entity

- Map logical file locations onto physical disk locations.
- May involve management of a file structure (directory hierarchy)
  - Does file/directory creation/deletion
  - File manipulation (rename, move, append)
  - File backup

PROTECTION

- Of files, memory, CPU, etc.
- Means controlling of access
- Depends on the attributes of the file and user

COMMUNICATION

- Communication system between distributed processors.
- Getting information about files/processes/etc. on a remote machine.
- Can use either a message passing or a shared memory model.

SYSTEM PROGRAMS

- Command Interpreters -- Program that accepts control statements (shell, GUI interface, etc.)
- Compilers/linkers
- Communications (ftp, telnet, etc.)

HOW DO ALL THESE PIECES FIT TOGETHER?

- Who calls who? Let's draw a picture.

SYSTEM TAILORING:

- Modifying the Operating System program for a particular machine. The goal is to include all the necessary pieces, but not too many extra ones.
- Typically a System can support many possible devices, but any one installation has only a few of these possibilities.
- **Plug and play** allows for detection of devices and automatic inclusion of the code (drivers) necessary to drive these devices.
- A **sysgen** is usually a link of many OS routines/modules in order to produce an executable containing the code to run the drivers.

SYSTEM CALLS:

A System Call is the main way a user program interacts with the Operating System.

HOW A SYSTEM CALL WORKS <<< FIGURES 2.9 and 3.1 >>>

- Obtain access to system space
- Do parameter validation
- System resource collection (locks on structures)
- Ask device/system for requested item
- Suspend waiting for device
- Interrupt makes this thread ready to run
- Wrapup
- Return to user

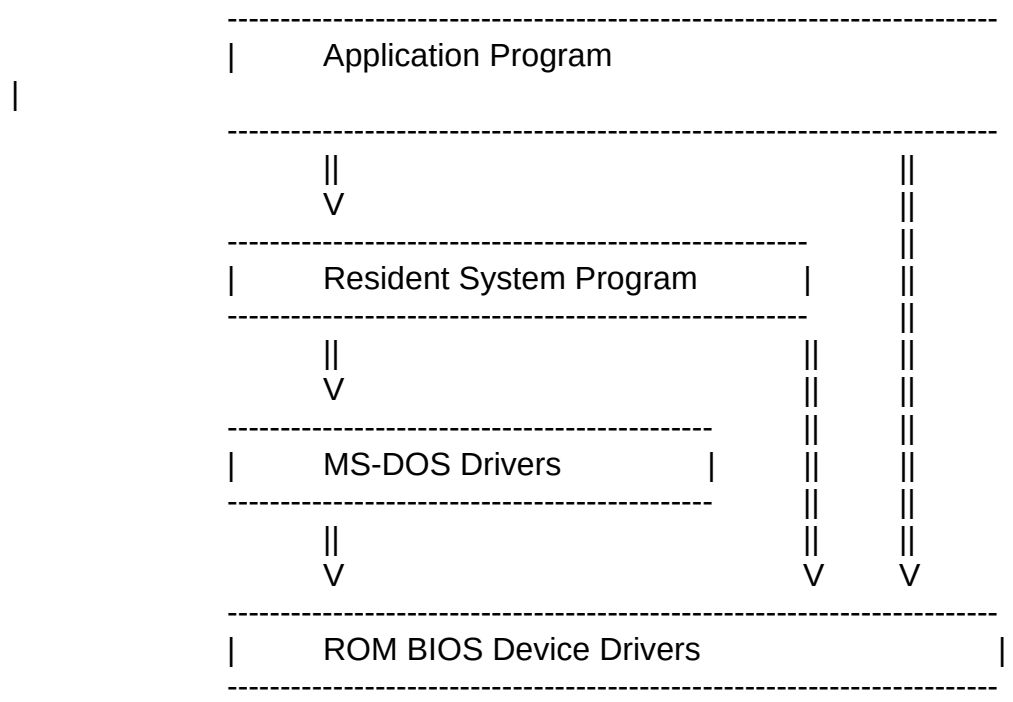
FOR EXAMPLE:

fprintf
| (Language runtimes prepare buffer)
write-string system call
| (Validate and move packet)
| (Buffer management)
write-block function call

HOW THE OPERATING SYSTEM IS STRUCTURED:

SIMPLE STRUCTURE:

Example of MS-DOS. This is **<< Figure 3.6. >>**



Note how all layers can touch the drivers!!! Bad news!!!

LAYERED STRUCTURE:

Levels of Operating systems - allows modularity and data hiding

Look at a way of doing this: **<<< FIGURE 3.7 >>> == UNIX**

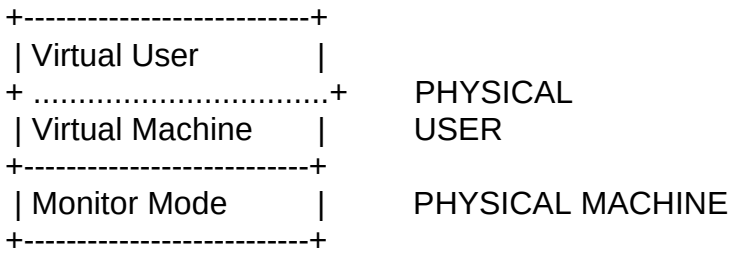
VIRTUAL MACHINE

In a Virtual Machine - each process "seems" to execute on its own processor with its own memory, devices, etc.

- IBM - VM for example. The resources of the physical machine are shared. Virtual devices are sliced out of the physical ones. Virtual disks are subsets of physical ones.

- Useful for running different OS simultaneously on the same machine.

- Protection is excellent, but no sharing possible.



Virtual privileged instructions are trapped.