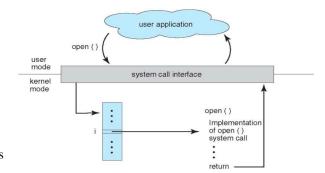
Ch.2 – OS Structures

- <u>User Interface (UI)</u> Can be <u>Command-Line (CLI)</u> or <u>Graphics User Interface (GUI)</u> or <u>Batch</u>
 - These allow for the user to interact with the system services via system calls (typically written in C/C++)
- Other system services that a helpful to the <u>user</u> include: program execution, I/O operations, file-system manipulation, communications, and error detection
- Services that exist to ensure efficient OS operation are: resource allocation, accounting, protection and security
- Most system calls are accessed by Application Program Interface (API) such as Win32, POSIX, Java
- Usually there is a number associated with each system call
 - System call interface maintains a table indexed according to these numbers
- Parameters may need to be passed to the OS during a system call, may be done by:
 - Passing in <u>registers</u>, address of parameter stored in a <u>block</u>, <u>pushed</u> onto the stack by the program and <u>popped</u> off by the OS
 - Block and stack methods do not limit the number or length of parameters being passed
- <u>Process control</u> system calls include: end, abort, load, execute, create/terminate process, wait, allocate/free memory
- <u>File management</u> system calls include: create/delete file, open/close file, read, write, get/set attributes
- <u>Device management</u> system calls: request/release device, read, write, logically attach/detach devices
- <u>Information maintenance</u> system calls: get/set time, get/set system data, get/set process/file/device attributes
- <u>Communications</u> system calls: create/delete communication connection, send/receive, transfer status information



• OS Layered approach:

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers
- <u>Virtual machine</u>: uses layered approach, treats hardware and the OS kernel as though they were all hardware.
 - Host creates the illusion that a process has its own processor and own virtual memory
 - Each guest provided with a 'virtual' copy of the underlying computer
- Application failures can generate <u>core dump</u> file capturing memory of the process
- Operating system failure can generate <u>crash dump</u> file containing kernel memory