Objectives

• To describe System calls.

• To Explain the importance/significance of system calls in operating systems
System Calls

• System calls provide an interface to the services made available by an operating system. What are these services?

• These calls are generally available as functions.....Remember functions in programming?
System Calls

- The occurrence of an event is usually signaled by an interrupt from either the hardware or the software.
System Calls

- Software may trigger an interrupt by executing a special operation called a **system call** (also called a **monitor call**).
Class Exercise

• Write down an algorithm to copy the contents of a file A to file B [which you create](Assume an icon based system and that you are doing this on the desktop)
Class Exercise

1. Create file B and give it a name. This requires a system call to create file

   » What if a file with a similar name exists?

2. Locate and open file A

   ➢ What if a file does not exist?
Class Exercise

3. Copy contents of A-B. This is a loop

» What if disk gets full?

4 Close File. Terminate system call
Example of System Calls

- System call sequence to copy the contents of one file to another file

Example System Call Sequence
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
  Until read fails
Close output file
Write completion message to screen
Terminate normally
System Calls

• Most programmers never see this level of detail. They hence develop programs according to an application programming interface (API) rather than direct system call use.
System Calls

Three most common APIs are

• Win32 API for Windows,
• POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X),
• Java API for the Java virtual machine (JVM)
The API specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and return values expected.
System Calls

These functions that make up the API invoke the actual system call

Example the win32 function Create process() is used to create a new process but in actual sense, this function calls the Ntcreateprocess() system call in windows kernel.
System Calls

- Why would a programmer use APIs rather than invoke actual system calls?
System Calls

• **Portability:** An application programmer designing a program using an API can expect his/her program to compile and run on any system that supports the same API.

• Actual system calls may be more detailed and hence difficult to work with.
Handling of a user application invoking a system call

user application

open ()

user mode

system call interface

kernel mode

Implementation of open ()

system call

return
System Call Parameter Passing

- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in registers
  - In some cases, may be more parameters than registers

- What do we do?
System Call Parameter Passing

– Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register

• This approach taken by Linux
System Call Parameter Passing

– Parameters placed, or *pushed*, onto the *stack* by the program and *popped* off the stack by the operating system

– Block and stack methods do not limit the number or length of parameters being passed
Parameter Passing via Table

X: parameters for call
load address X
system call 13

X
register

use parameters from table X

user program

operating system
code for system call 13

Operating System Services
Types of System Calls that an OS provides
Can be grouped into five...
1. Process control
   – end, abort
   – load, execute
   – create process, terminate process
   – get process attributes, set process attributes
   – wait for time
   – wait event, signal event
   – allocate and free memory
Types of System Calls

• File management
  – create file, delete file
  – open, close file
  – read, write, reposition
  – get and set file attributes
Types of System Calls (Cont.)

• Device management
  – request device, release device
  – read, write, reposition
  – get device attributes, set device attributes
  – logically attach or detach devices
Types of System Calls (Cont.)

- Information maintenance
  These are calls that exist for transferring information between the user program and the OS.
  - get time or date, set time or date
  - Number of current users
  - Amount of free memory or disk space
Types of System Calls (Cont.)

• Communications

Deals with inter process communication

– create, delete communication connection
– send, receive messages
– transfer status information
– attach and detach remote devices
## Examples of Windows and Unix System Calls

<table>
<thead>
<tr>
<th>Windows</th>
<th>Unix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process Control</strong></td>
<td></td>
</tr>
<tr>
<td>CreateProcess()</td>
<td>fork()</td>
</tr>
<tr>
<td>ExitProcess()</td>
<td>exit()</td>
</tr>
<tr>
<td>WaitForSingleObject()</td>
<td>wait()</td>
</tr>
<tr>
<td><strong>File Manipulation</strong></td>
<td></td>
</tr>
<tr>
<td>CreateFile()</td>
<td>open()</td>
</tr>
<tr>
<td>ReadFile()</td>
<td>read()</td>
</tr>
<tr>
<td>WriteFile()</td>
<td>write()</td>
</tr>
<tr>
<td>CloseHandle()</td>
<td>close()</td>
</tr>
<tr>
<td><strong>Device Manipulation</strong></td>
<td></td>
</tr>
<tr>
<td>SetConsoleMode()</td>
<td>ioctl()</td>
</tr>
<tr>
<td>ReadConsole()</td>
<td>read()</td>
</tr>
<tr>
<td>WriteConsole()</td>
<td>write()</td>
</tr>
<tr>
<td><strong>Information Maintenance</strong></td>
<td></td>
</tr>
<tr>
<td>GetCurrentProcessID()</td>
<td>getpid()</td>
</tr>
<tr>
<td>SetTimer()</td>
<td>alarm()</td>
</tr>
<tr>
<td>Sleep()</td>
<td>sleep()</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td></td>
</tr>
<tr>
<td>CreatePipe()</td>
<td>pipe()</td>
</tr>
<tr>
<td>CreateFileMapping()</td>
<td>shmget()</td>
</tr>
<tr>
<td>MapViewOfFile()</td>
<td>mmap()</td>
</tr>
<tr>
<td><strong>Protection</strong></td>
<td></td>
</tr>
<tr>
<td>SetFileSecurity()</td>
<td>chmod()</td>
</tr>
<tr>
<td>InitializeSecurityDescriptor()</td>
<td>umask()</td>
</tr>
<tr>
<td>SetSecurityDescriptorGroup()</td>
<td>chown()</td>
</tr>
</tbody>
</table>