4 USER OPERATING SYSTEM INTERFACE

User Operating System Interface - CLI

- CLI or **command interpreter** allows direct command entry
  - Sometimes implemented in kernel, sometimes by systems program
  - Sometimes multiple flavors implemented – **shells**
  - Primarily fetches a command from user and executes it
  - Sometimes commands built-in, sometimes just names of programs
    - If the latter, adding new features doesn't require shell modification

User Operating System Interface - GUI

- User-friendly **desktop** metaphor interface
  - Usually mouse, keyboard, and monitor
  - **Icons** represent files, programs, actions, etc.
  - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a **folder**)
  - Invented at Xerox PARC
- Many systems now include both CLI and GUI interfaces
  - Microsoft Windows is GUI with CLI "command" shell
  - Apple Mac OS X is "Aqua" GUI interface with UNIX kernel underneath and shells available
  - Unix and Linux have CLI with optional GUI interfaces (CDE, KDE, GNOME)

Touchscreen Interfaces

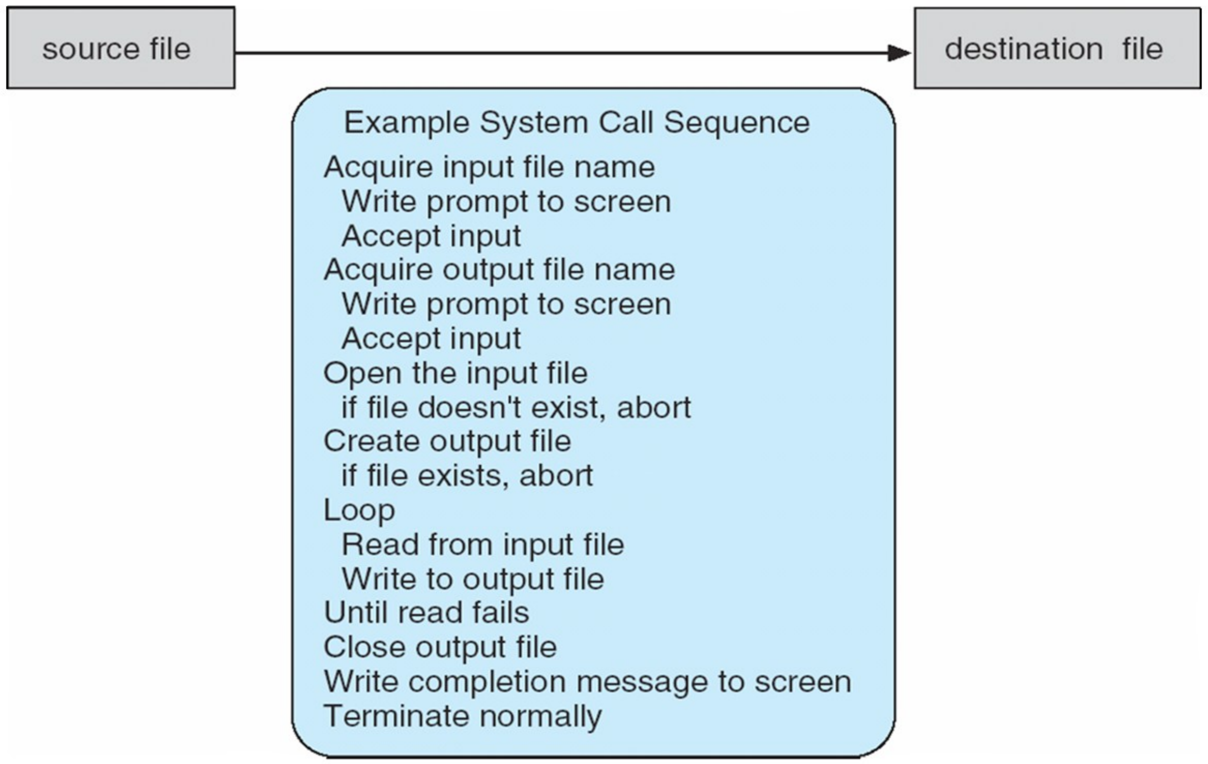
- Touchscreen devices require new interfaces
  - Mouse not possible or not desired
  - Actions and selection based on gestures
  - Virtual keyboard for text entry
- Voice commands.

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Programming Interface (API)** rather than direct system call use
- 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), and Java API for the Java virtual machine (JVM)

Example of System Calls

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



Types of System Calls

- Process control
    - create process, terminate process
    - end, abort
    - load, execute
    - get process attributes, set process attributes
    - wait for time
    - wait event, signal event
    - allocate and free memory
    - Dump memory if error
    - **Debugger** for determining **bugs, single step** execution
    - **Locks** for managing access to shared data between processes

- File management
    - create file, delete file
    - open, close file
    - read, write, reposition
    - get and set file attributes

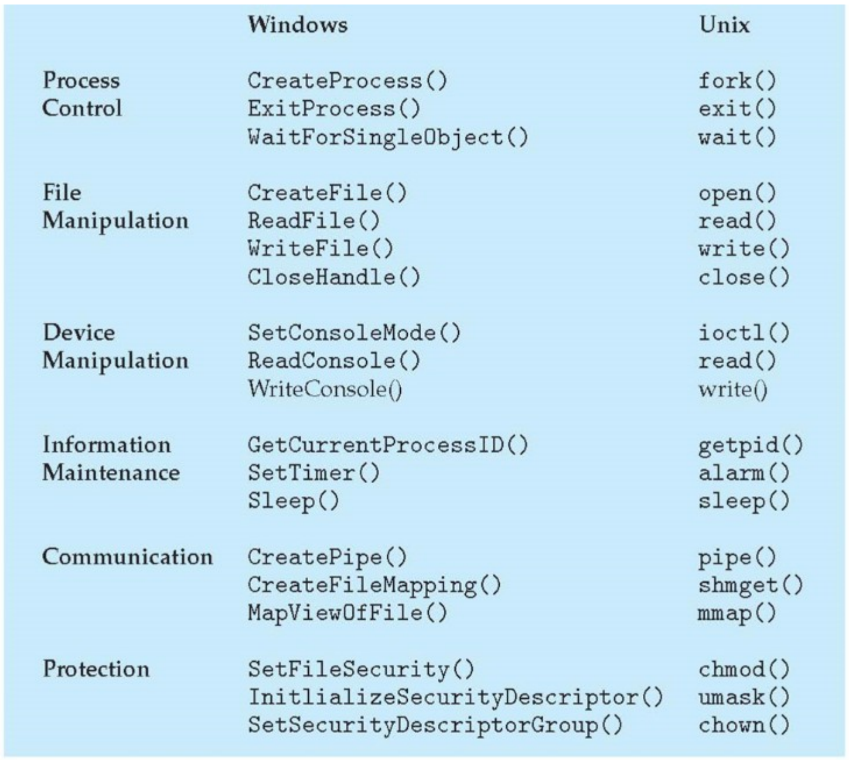
- Device management
    - request device, release device
    - read, write, reposition
    - get device attributes, set device attributes
    - logically attach or detach devices

- Information maintenance
    - get time or date, set time or date
    - get system data, set system data
    - get and set process, file, or device attributes

- Communications
    - create, delete communication connection
    - send, receive messages if **message passing model** to **host name** or **process name**
        - From **client** to **server**
    - **Shared-memory model** create and gain access to memory regions
    - transfer status information
    - attach and detach remote devices

- Protection
    - Control access to resources
    - Get and set permissions
    - Allow and deny user access

Examples of Windows and Unix System Calls



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

System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into:
    - File manipulation
    - Status information sometimes stored in a File modification
    - Programming language support
    - Program loading and execution
    - Communications
    - Background services
    - Application programs
- Most users' view of the operating system is defined by system programs, not the actual system calls
- Provide a convenient environment for program development and execution
    - Some of them are simply user interfaces to system calls; others are considerably more complex

- **File management** - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories

- **Status information**
    - Some ask the system for info - date, time, amount of available memory, disk space, number of users
    - Others provide detailed performance, logging, and debugging information
    - Typically, these programs format and print the output to the terminal or other output devices
    - Some systems implement a **registry** - used to store and retrieve configuration information

- **File modification**
    - Text editors to create and modify files
    - Special commands to search contents of files or perform transformations of the text

- **Programming-language support** - Compilers, assemblers, debuggers and interpreters sometimes provided

- **Program loading and execution**- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language

- **Communications** - Provide the mechanism for creating virtual connections among processes, users, and computer systems
    - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

- **Background Services**
    - Launch at boot time
        - Some for-system startup, then terminate
        - Some from system boot to shutdown
    - Provide facilities like disk checking, process scheduling, error logging, printing
    - Run in user context not kernel context
    - Known as **services**, **subsystems**, **daemons**

- **Application programs**
    - Don’t pertain to system
    - Run by users
    - Not typically considered part of OS
    - Launched by command line, mouse click, finger poke

Note:

Read our reference book to learn more about this lesson.

Hi I’m Flashee!

You have reached the end of the lesson. Be sure to answer the corresponding **activity of this lesson** on the activities folder of our class materials in the file server.