Unit-II
Lecture: 2
(System Calls)

System calls are basically a way in which a program requests a service from the kernel of the OS which may include hardware-related services like accessing files form hard disk, creation and execution of new processes, etc. System calls provide the interface between a running program and the operating system.
System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do.
System calls may be available as assembly language instructions or even high-level language calls.
They resemble pre-defined function calls.

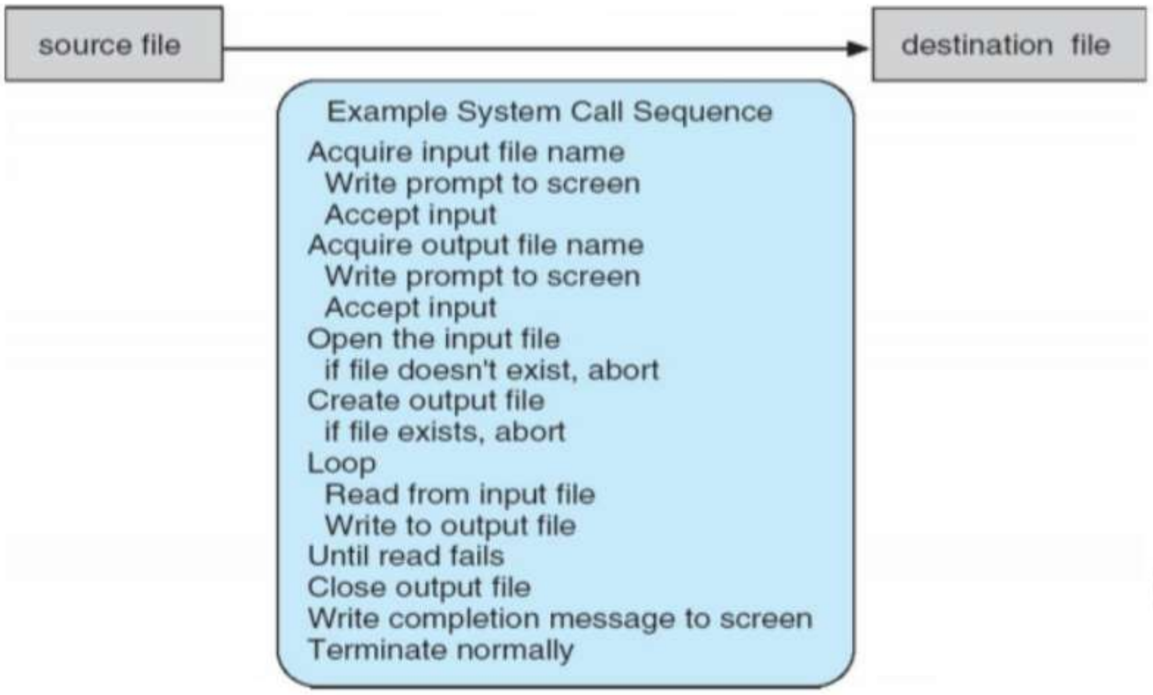


Fig: System call sequence to copy the contents of one file to another

The above figure shows the sequence of system calls that will be invoked when the contents of one file is copied to another.

- Prompting to the screen and asking the user to enter the name of a file is a system call.
- Getting the name of a source file as input is a system call.
- Again, asking the user for the name of the destination file, reading as input the destination file name, opening the source file, creating the

destination file, reading from the source file, writing to the destination file etc. are all system calls.

- So, any program written by any user will have a number of system calls in it. However, most users never see this level of details.

System Call Implementation:

- Typically, a system call number is associated with each system call. A system-call interface maintains a table indexed according to these numbers. This system call table has the address of the routine where the system call is implemented in the operating system kernel.
- The system call interface invokes the intended system call in the OS kernel and returns the status of the system call and any return values.
- The caller/user need not know anything about how the system call is implemented.
- The user just needs to obey the Application Program Interface (API) and understand what the OS will do as a result of the call.
- ***Most details of the system call are hidden from the programmer by the API.***
- The system calls are managed by a run-time support library (set of functions built into libraries included with compiler).

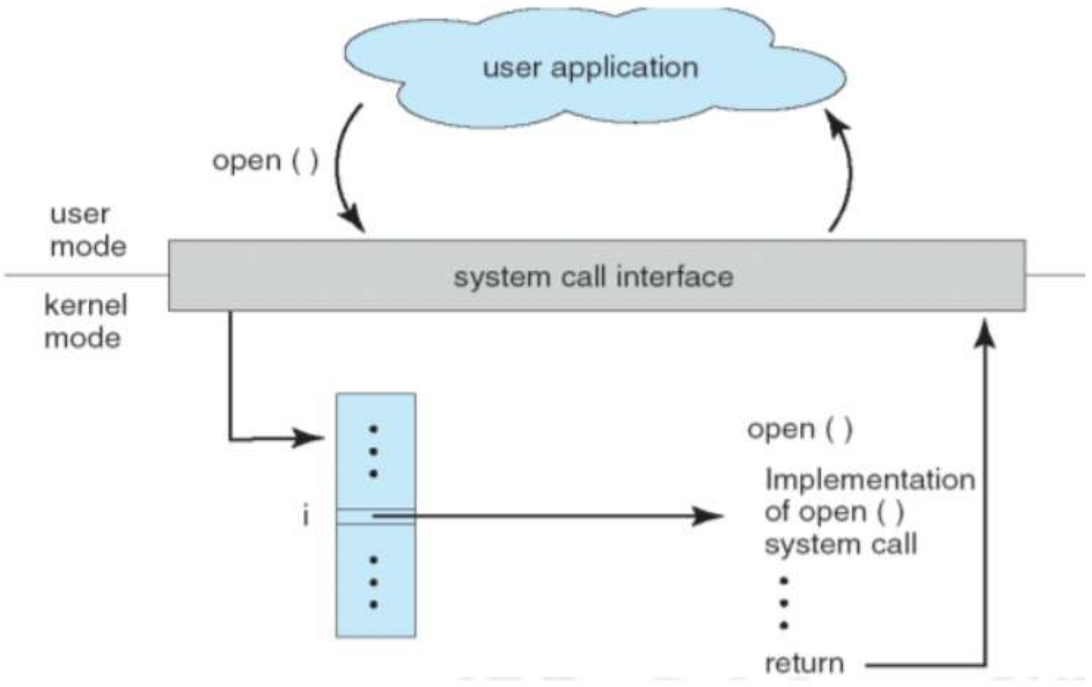
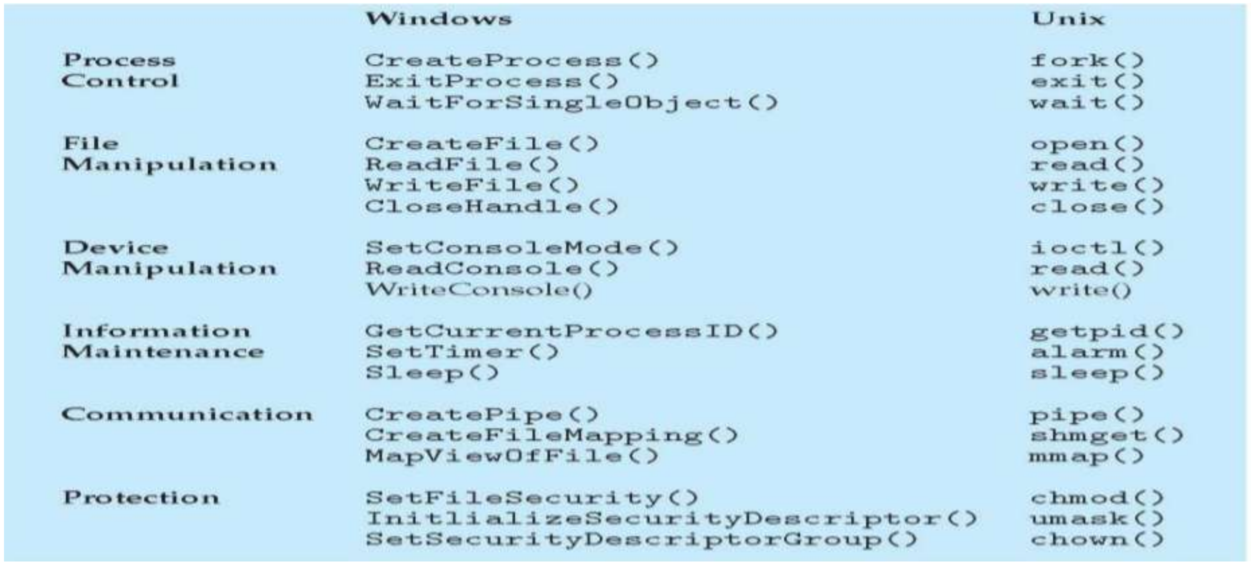


Fig: Relationship between API, system call and OS

Fig above shows a user application using the open () call to open a file.

- The open () call invokes the system call interface. There is a system call number corresponding to the open () call.
- The system call interface uses this number as an index into the system call table to find the address of the open () routine in the kernel.
- Using this address, the open () routine in the kernel is now invoked. The open routine in the kernel does whatever is necessary for opening a file and returns a file handle.
- The file handle (return value) is returned to the user.

Some examples of different types of system calls are provided by an operating system:



	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()