Name: Kshitij Chandrakar

Batch: 49

SAP: 500124827

Case Study On The Following Topics With Respect to the Windows and The Linux Operating Systems

1. Computer Hardware Review
2. Computer System
3. Introduction to Operating System: Definition Operating System view
4. History
5. Types of Operating Functions of Operating System
6. Services of Operating System
7. Computing Environments
8. Virtualization and Containerization
9. Operating System Structures
10. Operating System Operations
11. System boot. System Calls
12. Types of System Calls (Windows and Unix System Calls examples)
13. Open Source Operating Systems

Introduction

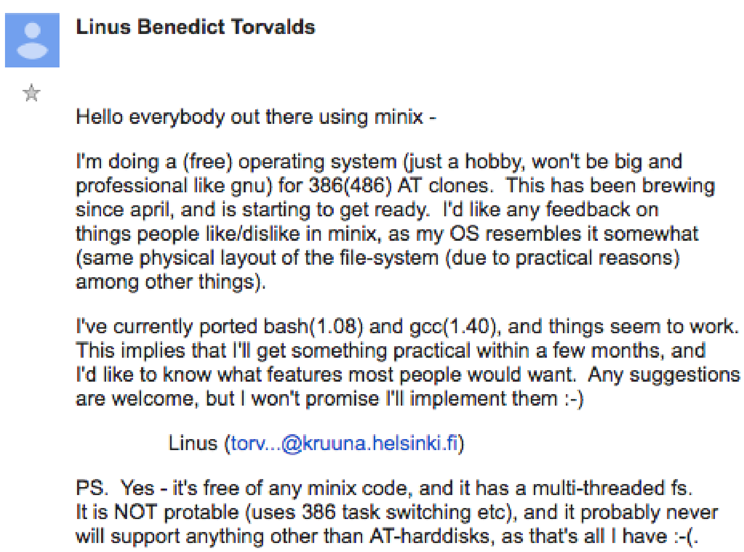
This case study will look into the comparative aspects of Windows and Linux operating systems, focusing on their core components, functionalities, and underlying principles. I will explore the historical context of these systems, their primary types, and the services they provide to users.

I will examine the concepts of virtualization and containerization, the structural components of these operating systems, and their operational mechanisms, the system boot process, system calls, and the significance of open-source operating systems.

The Windows Operating system is a Proprietary OS provided by Microsoft, with the First Version Releasing on November 20, 1985, as a response to the growing popularity of GUI based operating systems in opposition to MS-DOS. It has had multiple different versions since then, with the latest being Windows 11 at the time of writing.

Linux as an operating system is a generic term for a family of operating systems running on an Open Source UNIX like kernal called the linux kernal. It was released on September 17 1991 By Linus Torvalds [1]. Linux is typically packaged as a Linux distribution (distros), which includes the kernel and supporting system software and libraries, many of which are provided by the GNU Project.

As of writing, multiple Linux based distributions exist, some popular ones include Ubuntu, Linux Mint Red Hat, Debian etc.



Linus Benedict Torvalds

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torv...@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Figure 1: Linus Torvalds' Announcement of Linux

1. https://groups.google.com/g/comp.os.minix/c/dlNtH7RRrGA/m/SwRavCzVE7gJ

On Computer Hardware

Computer hardware refers to the physical components of a computer system. These components work together to perform various tasks, such as processing information, storing data, and communicating with other devices.

Key Hardware Components:

1. **Central Processing Unit (CPU):** Often referred to as the "brain" of the computer, the CPU is responsible for executing instructions and performing calculations.
2. **Motherboard:** Serves as the main circuit board that connects all the components of the computer system.
3. **Memory:** Stores data and instructions temporarily while the computer is running.
4. **Storage Devices:** Store data in a non volatile fashion.
5. **Input Devices:** Allow users to enter data into the computer.
6. **Output Devices:** Used to Display or present information from the computer.

7. **Power Supply Unit (PSU):** Converts AC power from the electrical outlet to DC power for the computer components.

Windows: The Windows OS is Typically associated with consumer-grade hardware, Windows often runs on a wider range of devices, including desktops, laptops, tablets, and smartphones. It has a broad compatibility with various hardware components and peripherals. However Performance in comparison with Linux is worse off.

Linux: Known for its flexibility and adaptability, Linux can run on a diverse range of hardware, from low-powered devices to high-performance servers. As it is very lightweight, it can run in older and/or weaker computers. It is often used in embedded systems, supercomputers, and cloud environments.

Computer System

In its most basic form, a computer system is a programmable electronic device that can accept input; store data; and retrieve, process and output information.

Introduction to Operating System

Definition: An operating system (OS) is a software program that manages a computer's hardware and software resources. It provides a platform for applications to run and interact with the system.

Operating System View: The OS can be viewed from two perspectives:

- **User View:** Presents a user-friendly interface for interacting with the system.
- **System View:** Manages the system's resources and handles communication between hardware and software components.

Views Of an Operating System

An operating system can be defined or observed in two ways

- User View
- System View

User View

It Focuses on how users interact with application programs. Types include single-user, multiple-user, handled user, and embedded user viewpoints.

Single User Viewpoint

- These systems are designed for a single user experience and meet the needs of a single user
- The performance is not given focus as the multiple user systems.

Multiple User Viewpoint

- These systems consists one mainframe computer and many users on their computers trying to interact with their kernels over the mainframe to each other.
- In such systems, memory allocation by the CPU must be done effectively to give a good user experience.
- The client-server architecture is another good example where many clients may interact through a remote server

Handled User Viewpoint

- These systems are lies under touchscreen era that comes with best handheld technology ever. Smartphones interact via wireless devices to perform numerous operations,
- Such operating system is a great example of creating a device focused on the user's point of view.

Embedded User Viewpoint

- Systems in which remote control used to turn on or off the tv is all part of an embedded system in which the electronic device communicates with another program where the user viewpoint is limited and allows the user to engage with the application.

System View

- A computer system comprises various sources, such as hardware and software, which must be managed effectively. The operating system manages the resources, decides between competing demands, controls the program execution, etc.

- According to this point of view, the operating system's purpose is to maximize performance. The operating system is responsible for managing hardware resources and allocating them to programs and users to ensure maximum performance.

From a system viewpoint, the hardware interacts with the operating system than with the user. The hardware and the operating system interact for a variety of reasons, including:

Resource Allocation

- The hardware contains several resources like registers, caches, RAM, ROM, CPUs, I/O interaction, etc. These are all resources that the operating system needs when an application program demands them.

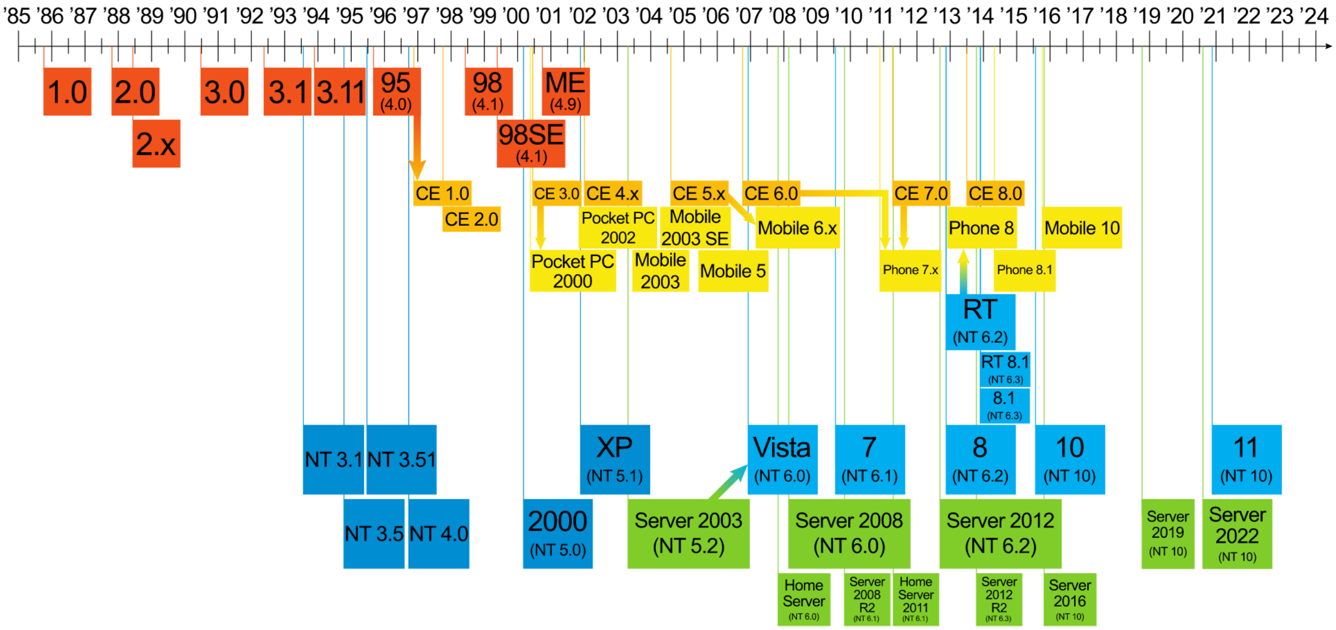
- Only the operating system can allocate resources with several tactics and strategies to maximize its processing and memory space. The operating system uses a variety of strategies to get the most out of the hardware resources, including paging, virtual memory, caching, and so on.

Control Program

- The control program controls how input and output devices (hardware) interact with the operating system.

- The user may request an action that can only be done with I/O devices; in this case, the operating system must also have proper communication, control, detect, and handle such devices.

History



A Timeline of Windows Releases

Windows: A Brief History of Windows

Microsoft Windows, one of the most widely used operating systems today, has a history dating back to the 1980s. Its roots can be traced to the MS-DOS operating system, which was a command-line interface primarily used for business applications.

In 1985, Microsoft introduced Windows 1.0, a graphical user interface (GUI) that aimed to make computing more accessible to the average user. While it was a significant step forward, the early versions of Windows were relatively limited in functionality and performance.

Over the years, Microsoft continued to refine and improve Windows, releasing major updates with new features and capabilities. Windows 95, released in 1995, was a breakthrough moment for the operating system, introducing a more user-friendly interface and improved multitasking.

Windows XP, released in 2001, became one of the most popular versions of Windows, known for its stability and ease of use. It helped to solidify Windows' position as the dominant operating system for personal computers.

Windows 7 Continued This Trend of Breakthroughs, becoming another one of the most popular systems in use.

Windows 8, Failed to compete with the giants that came before

Windows 9 was a ghost

and Windows 10, Releasing in 2015, was Another Breakthrough for Microsoft.

Linux

Originated in the early 1990s as a free and open-source operating system. It was inspired by Unix and has gained immense popularity due to its flexibility, reliability, and security. While it does have versions, as a community project, multiple different forks have arisen.
Many developers of open-source software agree that the Linux kernel was not designed but rather evolved through natural selection. Torvalds considers that although the design of Unix served as a scaffolding, "Linux grew with a lot of mutations – and because the mutations were less than random, they were faster and more directed than alpha-particles in DNA." [2]

"Linux evolved in a completely different way. From nearly the beginning, it was rather casually hacked on by huge numbers of volunteers coordinating only through the Internet. Quality was maintained not by rigid standards or autocracy but by the naively simple strategy of releasing every week and getting feedback from hundreds of users within days, creating a sort of rapid Darwinian selection on the mutations introduced by developers." [3]

It evolved as a form of rebellion against the Locking down of UNIX as a Proprietary Software. A Few Precursors existed such as FreeBSD, Minix, NetBSD, OpenBSD etc. It was created by Linus in Frustration of the Minix software restricting its free use to Educational Use.

2 https://lwn.net/2001/1206/a/no-design.php3 https://web.archive.org/web/20210812201159/https://lwn.net/2001/1206/a/no-design.php3

3 *"Anatomy of a Linux System" (PDF). O'Reilly. July 23–26, 2001. Archived (PDF) from the original on September 4, 2019. Retrieved October 10, 2018.* https://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/linux_anatomy.pdf

Types of Functions of Operating Systems

Windows: Primarily a single-user, multi-tasking operating system designed for personal computers.

Linux: Can be single-user or multi-user, and can support both single-tasking and multi-tasking environments. It is widely used in servers, workstations, and embedded systems.

Functions of Operating System

Both Windows and Linux perform essential functions, including:

Functions and Services of an Operating System

An operating system (OS) is a software program that manages a computer's hardware and software resources. It provides a platform for applications to run and interacts with the user. Here are the primary functions of an OS:

Process Management

- **Process creation and termination:** The OS creates and manages processes, which are instances of a program. It also terminates processes when they are no longer needed.
- **Process scheduling:** The OS determines the order in which processes will be executed.
- **Process synchronization:** The OS ensures that multiple processes can access shared resources without causing conflicts.
- **Process communication:** The OS provides mechanisms for processes to communicate with each other.

Memory Management

- **Memory allocation:** The OS allocates memory to processes and data structures.
- **Memory deallocation:** The OS reclaims memory that is no longer needed.
- **Memory protection:** The OS ensures that processes cannot access memory that they are not authorized to use.
- **Virtual memory:** The OS creates a virtual memory space for each process, which may be larger than the physical memory available.

I/O Management

- **Device drivers:** The OS provides device drivers that interface with hardware devices.
- **I/O scheduling:** The OS determines the order in which I/O requests will be serviced.
- **Buffering:** The OS may buffer I/O data to improve performance.

File System Management

- **File creation and deletion:** The OS creates and deletes files.

- **File access:** The OS provides mechanisms for processes to access files.
- **File organization:** The OS organizes files into directories.
- **File protection:** The OS ensures that files are protected from unauthorized access.

Secondary Storage Management

- **Disk scheduling:** The OS determines the order in which disk I/O requests will be serviced.
- **Disk formatting:** The OS formats disks to prepare them for use.
- **Disk partitioning:** The OS partitions disks into logical drives.

User Interface

- **Command-line interface (CLI):** The OS provides a text-based interface for users to interact with the system.
- **Graphical user interface (GUI):** The OS provides a visual interface with icons and windows.

Other Functions Include

- **Security:** The OS protects the system from unauthorized access.
- **Networking:** The OS manages network connections and communication.
- **Error handling:** The OS handles errors that occur during system operation.

On Services:

Both Windows and Linux provide various services, Other than the aforementioned Functions, Such as:

User Interface (UI)

- **Graphical User Interface (GUI):**
    - Provides a visual interface with icons, windows, and menus.
    - Makes interaction with the computer more intuitive.
    - Examples: Windows, macOS, Linux (with desktop environments like GNOME, KDE)
- **Command-Line Interface (CLI):**
    - A text-based interface that requires users to enter commands.
    - Offers more control and flexibility for advanced users.
    - Examples: Unix shells (bash, zsh, csh, ksh), PowerShell and Command Prompt(Windows)

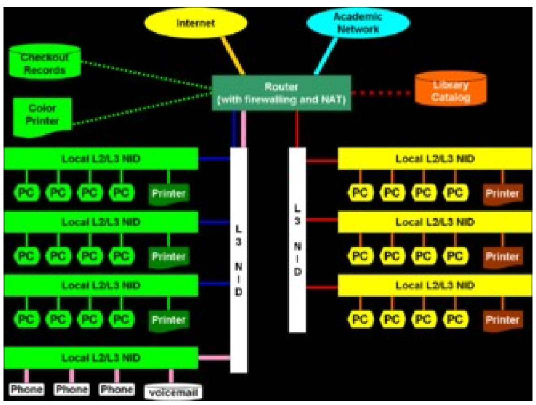
Security

- **Authentication:**
    - Verifies the identity of users or processes.
    - Methods include passwords, biometrics (fingerprints, facial recognition), tokens, or multi-factor authentication.
- **Authorization:**

    - Determines what actions a user or process is allowed to perform. Based on roles, permissions, or access control lists (ACLs).
- **Access Control:**
    - Enforces security policies to prevent unauthorized access.
    - Techniques include firewalls, intrusion detection systems (IDS), and encryption.

Networking

- **Network Protocols:**
    - Defines the rules for communication between devices on a network.
    - Examples: TCP/IP, HTTP, FTP, SMTP.
- **Network Devices:**
    - Hardware components that facilitate network communication.
    - Examples: routers, switches, modems, network interface cards (NICs).



Typical library network, in a branching tree map and controlled access to resources

Application Programming Interfaces (APIs)

- **System Calls:**
    - Functions that applications can use to request services from the operating system.
    - Examples: open(), read(), write(), create_process().
- **Libraries:**
    - Collections of related functions and data structures.
    - Examples: standard C library, Java class libraries.
- **API Documentation:**
    - Provides information about how to use APIs.
    - Includes descriptions of functions, parameters, and return values.

Computing Environments

Computing environments refer to the technology infrastructure and software platforms that are used to develop, test, deploy, and run software applications. There are several types of computing environments According to organization of different computer devices and communication processes, which each have their own Advantages and Disadvantages, Some of them are:

1. **Mainframe:** A large and powerful computer system used for critical applications and large-scale data processing.
2. **Client-Server:** A computing environment in which client devices access resources and services from a central server.
3. **Cloud Computing:** A computing environment in which resources and services are provided over the Internet and accessed through a web browser or client software.
4. **Mobile Computing:** A computing environment in which users access information and applications using handheld devices such as smartphones and tablets.
5. **Grid Computing:** A computing environment in which resources and services are shared across multiple computers to perform large-scale computations.
6. **Embedded Systems:** A computing environment in which software is integrated into devices and products, often with limited processing power and memory.

Windows: Primarily used in personal computers, workstations, and some servers.

Linux: Widely used in servers, workstations, embedded systems, and supercomputers. It is also gaining popularity in cloud computing environments. However, in recent years along with the release of the steam deck, Compatibility of Gaming is increasing on Linux.

Virtualization and Containerization

Virtualization (or Virtualisation) is a word used in computing. Virtualization means that the users (programs, or real people) only see an abstraction of a computer resource. Virtualization can be done in software, or with hardware.[4]

1. **RAID:** RAID is used to virtualize computer storage. A RAID system appears as one "disk". The fact that it is made of several disks that work together is hidden.

2. **Virtual memory:** This Makes it possible to use more memory than is physically in the computer. The computer figures out a way to write contents of certain memory blocks to disk.

3. **Storage virtualization:** This takes the ideas developed by RAID further. This is what Storage area networks commonly use. All the storage appears as a single big disk. Certain administrators can specify that this large disk is made of "data pools" (which are also virtual). The disk pools are made of single physical disks (or RAID arrays).

4. Some computers (mostly mainframes) allow to run several operating systems at the same time. Each operating system believes it is the only one running.

5. Data virtualization is used by businesses to put together data from a few sources in one place. This helps applications, reporting tools and end users to access data with no need in details about the source structure, location and original data.

---

4. Virtualization: what are these, the main types

Operating System Structures

Simple/Monolithic Structure:

A simple/monolithic structure integrates all components of an operating system into a single executable. This approach offers efficiency and ease of implementation but can be less flexible and prone to system crashes if a single component fails. Examples include MS-DOS and early versions of UNIX.

Micro-Kernel Structure:

A micro-kernel structure divides the operating system into a small kernel that handles essential functions and user-level processes for most services. This modular approach enhances flexibility, reliability, and maintainability but can introduce performance overhead due to inter-process communication. Examples include Mach, QNX, and L4.

Hybrid-Kernel Structure:

A hybrid-kernel structure combines elements of monolithic and micro-kernel structures. It integrates some services into the kernel while implementing others as user-level processes. This approach aims to balance efficiency and flexibility, but can be complex to design and implement. Examples include Windows NT and Linux.

Exo-Kernel Structure:

An exo-kernel structure has a minimal kernel that provides only the most basic services. Most system functions are implemented as micro-kernels that run on top of the exo-kernel. This highly modular approach offers flexibility and reliability but can introduce performance overhead due to multiple layers of abstraction. Examples include Singularity and EROS.

Layered Structure:

A layered structure divides the operating system into layers, with each layer providing services to the layer above it. This approach promotes modularity, understandability, and maintainability but can be less efficient due to multiple layers of abstraction. Examples include VMS and early versions of Windows.

Modular Structure:

A modular structure allows the operating system to be composed of dynamically loadable and unloadable modules. This approach offers flexibility, customization, and easier updates but can be complex to manage. Examples include Linux and Windows.

Virtual Machines:

Virtual machines create isolated environments on a single physical machine, allowing multiple operating systems to run simultaneously. This approach provides isolation, security, and facilitates testing and development but can introduce performance overhead and resource management complexity. Examples include VMware, VirtualBox, and Hyper-V.

Windows: Typically has a monolithic kernel structure, where the kernel handles all system functions.

Linux: Uses a modular kernel structure, allowing for flexibility and customization.

Operating System Operations

Both Windows and Linux perform similar operations, such as:

- **System Boot:** The process of loading the operating system into memory and starting it.
- **System Calls:** Functions that allow applications to request services from the operating system.

System Boot and System Calls

The system boot process involves:

1. **Power-On Self-Test (POST):** Checks the hardware components for errors. This shows that the hardware works.
2. **Basic Input/Output System (BIOS/UEFI):** Loads the operating system bootloader into the memory.
3. **Bootloader:** Loads the operating system kernel into memory.
4. **Kernel Initialization:** The kernel initializes system resources and starts essential services.

Types of System Calls

The architecture of most modern processors, with the exception of some embedded systems, involves a security model. For example, the rings model specifies multiple privilege levels under which software may be executed: a program is usually limited to its own address space so that it cannot access or modify other running programs or the operating system itself, and is usually prevented from directly manipulating hardware devices.

However, many applications need access to these components, so system calls are made available by the operating system to provide standard, well-defined, and safe implementations for such operations for the applications to interact with them.

They Can Be Broadly classified into 6 Categories:

Process control

For Windows The System Calls Are

System Call	Description
CreateProcess	Creates a new process and its primary thread.
CreateProcessWithLogonW	Creates a new process with specified logon credentials.
TerminateProcess	Terminates a process.
ExitProcess	Terminates the current process.
GetProcessInformation	Retrieves information about a process, such as its handle, process ID, and thread ID.
EnumProcesses	Enumerates all processes in the system.
GetProcessTimes	Retrieves information about the CPU time used by a process.
SetPriorityClass	Sets the priority class for a process.
SetThreadPriority	Sets the priority of a thread within a process.

WaitForSingleObject	Waits for a single object (e.g., process, thread, event) to become signaled.
WaitForMultipleObjects	Waits for multiple objects to become signaled.
ReadFile	Reads data from a file or device.
WriteFile	Writes data to a file or device.
VirtualAlloc	Allocates virtual memory for a process.
VirtualFree	Frees virtual memory allocated by a process.
GetEnvironmentVariable	Retrieves an environment variable.
SetEnvironmentVariable	Sets an environment variable.
OpenProcess	Opens a handle to an existing process.
DuplicateHandle	Duplicates a handle to a process.

For Linux The System Calls Are

System Call	Description
fork	Creates a new process that is a copy of the calling process.
execve	Replaces the current process with a new process specified by the pathname and arguments.
exit	Terminates the current process.
_exit	Terminates the current process without flushing buffers.
getpid	Retrieves the process ID of the current process.
getppid	Retrieves the process ID of the parent process.
getpgrp	Retrieves the process group ID of the current process.
nice	Changes the priority of the current process.
setpriority	Changes the priority of a specified process.
wait	Waits for a child process to terminate.
waitpid	Waits for a specific child process to terminate.
read	Reads data from a file or device.
write	Writes data to a file or device.
malloc	Allocates memory.
free	Frees allocated memory.
getenv	Retrieves an environment variable.
setenv	Sets an environment variable.

File management

Windows

System Call	Description
CreateFile	Creates a file or opens an existing file.
CloseHandle	Closes a file handle.

ReadFile	Reads data from a file.
WriteFile	Writes data to a file.
SetFilePointer	Sets the file pointer to a specific position.
GetFileSize	Retrieves the size of a file.
CreateDirectory	Creates a directory.
RemoveDirectory	Removes a directory.
FindFirstFile	Finds the first file or directory in a specified directory.
FindNextFile	Finds the next file or directory in a specified directory.
FindClose	Closes a file search handle.
MoveFile	Moves a file or directory.
CopyFile	Copies a file.
DeleteFile	Deletes a file.
GetFileAttributes	Retrieves the attributes of a file.
SetFileAttributes	Sets the attributes of a file.
LockFileEx	Locks a range of bytes in a file.
UnlockFileEx	Unlocks a range of bytes in a file.
FlushFileBuffers	Flushes the file buffers of a file.
GetLastError	Retrieves the last error that occurred.

Linux

System Call	Description
open	Opens a file.
close	Closes a file descriptor.
read	Reads data from a file.
write	Writes data to a file.
lseek	Sets the file pointer to a specific position.
stat	Retrieves information about a file.
mkdir	Creates a directory.
rmdir	Removes a directory.
opendir	Opens a directory.
readdir	Reads the next entry from a directory.
closedir	Closes a directory.
rename	Renames a file or directory.
unlink	Deletes a file.
chmod	Changes the permissions of a file.
chown	Changes the owner and group of a file.
access	Tests whether a process has permission to access a file.
fcntl	Performs various file control operations.
ioctl	Performs device-specific control operations.

Device management

Windows

System Call	Description
CreateFile	Creates a handle to a device.
CloseHandle	Closes a device handle.
ReadFile	Reads data from a device.
WriteFile	Writes data to a device.
DeviceIoControl	Performs device-specific control operations.
QueryDosDevice	Retrieves the device name associated with a volume.
SetVolumeInformation	Sets volume information.
GetVolumeInformation	Retrieves volume information.
GetDriveType	Retrieves the type of a drive.
GetDiskFreeSpaceEx	Retrieves information about the free space on a drive.

Linux

System Call	Description
open	Opens a device file.
close	Closes a device file descriptor.
read	Reads data from a device.
write	Writes data to a device.
ioctl	Performs device-specific control operations.
mknod	Creates a device node.
rmnod	Removes a device node.
statfs	Retrieves information about a file system.
mount	Mounts a file system.
umount	Unmounts a file system.
fstatfs	Retrieves information about a file system associated with a file descriptor.

Information maintenance

Windows

System Call	Description
GetSystemInfo	Retrieves information about the system.
GetSystemTime	Retrieves the current system time.
GetSystemTimeAsFileTime	Retrieves the current system time as a FILETIME structure.
SetSystemTime	Sets the current system time.
GetLocalTime	Retrieves the current local time.
SetLocalTime	Sets the current local time.
GetTimeZoneInformation	Retrieves information about the current time zone.
SetTimeZoneInformation	Sets the current time zone.
GetTickCount	Retrieves the number of milliseconds that have elapsed since Windows was started.

QueryPerformanceCounter	Retrieves a high-resolution performance counter value.
QueryPerformanceFrequency	Retrieves the frequency of a high-resolution performance counter.

Linux

System Call	Description
time	Measures the elapsed time for a command or program.
gettimeofday	Retrieves the current system time.
clock_gettime	Retrieves the current time with a specified clock.
clock_getres	Retrieves the resolution of a clock.
getrusage	Retrieves resource usage statistics for a process.
sysinfo	Retrieves system information.
uname	Retrieves system information, such as the operating system name, host name, and kernel version.
getuid	Retrieves the user ID of the current process.
geteuid	Retrieves the effective user ID of the current process.
getgid	Retrieves the group ID of the current process.
getegid	Retrieves the effective group ID of the current process.

Communication

Windows:

System Call	Description
CreatePipe	Creates a pipe for communication between processes.
ConnectNamedPipe	Connects a named pipe to a client process.
NamedPipeClientConnect	Connects a client process to a named pipe.
ReadFile/WriteFile	Reads and writes data to/from a pipe.
DisconnectNamedPipe	Disconnects a named pipe.
CreateMailslot	Creates a mailslot for communication.
GetMailslotInfo	Retrieves information about a mailslot.
ReadMailslot	Reads data from a mailslot.
WriteMailslot	Writes data to a mailslot.

Linux:

System Call	Description
pipe	Creates a pipe for communication between processes.
mkfifo	Creates a named pipe.
open	Opens a pipe or named pipe.
read/write	Reads and writes data to/from a pipe.
close	Closes a pipe or named pipe.
poll	Polls for events on file descriptors, including pipes.
select	Selects file descriptors for reading, writing, or exception handling.

epoll	Efficiently waits for events on file descriptors, including pipes.

Protection

Windows:

System Call	Description
OpenProcess	Opens a handle to an existing process.
DuplicateHandle	Duplicates a handle to a process.
CreateProcessWithLogonW	Creates a new process with specified logon credentials.
ImpersonateLoggedOnUser	Impersonates a logged-on user.
RevertToSelf	Reverts to the current thread's security context.
AdjustTokenPrivileges	Adjusts the privileges associated with a token.
GetSecurityInfo	Retrieves security information about an object.
SetSecurityInfo	Sets security information about an object.
CheckTokenMembership	Determines whether a token belongs to a specified group.
LookupPrivilegeValue	Retrieves the value of a privilege.
LookupAccountSid	Retrieves information about a security identifier (SID).

Linux:

System Call	Description
setuid	Sets the effective user ID of the current process.
seteuid	Sets the effective user ID of the current process.
setgid	Sets the effective group ID of the current process.
setegid	Sets the effective group ID of the current process.
setpgid	Sets the process group ID of the current process.
setpgrp	Sets the process group ID of the current process and its children.
setgroups	Sets the supplementary group IDs of the current process.
geteuid	Retrieves the effective user ID of the current process.
getegid	Retrieves the effective group ID of the current process.
getpgrp	Retrieves the process group ID of the current process.
getgroups	Retrieves the supplementary group IDs of the current process.
chroot	Changes the root directory of the current process.
setfacl	Sets access control lists (ACLs) for a file or directory.
getfacl	Retrieves access control lists (ACLs) for a file or directory.

Open-Source Operating Systems

Here are Some Examples of Various Open Source Operating Systems:

Linux Distributions

- **Ubuntu:** One of the most popular Linux distributions, known for its user-friendly interface and extensive software repositories.
- **Debian:** A stable and reliable distribution that forms the basis for many other Linux distributions.
- **Fedora:** A community-driven distribution sponsored by Red Hat, focusing on innovation and cutting-edge features.
- **CentOS:** A community-supported distribution based on Red Hat Enterprise Linux, often used for servers and enterprise environments.
- **Arch Linux:** A rolling release distribution known for its flexibility and control, but requiring more technical knowledge to install and configure.

BSD-Based Systems

- **FreeBSD:** A popular BSD-based operating system, often used for servers and embedded systems.
- **OpenBSD:** Known for its focus on security and reliability, often used in critical infrastructure.
- **NetBSD:** A highly portable operating system that can run on a wide range of hardware architectures.

Other Notable Open-Source Operating Systems

- **Android:** The mobile operating system used by most smartphones and tablets, based on a modified version of the Linux kernel.
- **Chrome OS:** A lightweight operating system designed for Chromebooks, primarily focused on web-based applications.
- **Haiku:** An open-source operating system inspired by BeOS, known for its user-friendly interface and multimedia capabilities.
- **ReactOS:** A project aimed at creating a Windows-compatible operating system.