Bahir Dar University Bahir Dar Institute of Technology

Faculty of Computing

Department of Software Engineering

Course Title: Operating System and System Programming

Documentation for DOS

Name: Nardos Ayalneh

ID: BDU1602231

Section: B

Submitted to: Lec. Wendmu B.

April, 2017

Contents

A Introduction to The DOS operating system

The Disk Operating System, commonly known as DOS, was one of the earliest and most influential operating systems for personal computers. Developed in the early 1980s, DOS became the backbone of IBM-compatible PCs, providing a simple yet efficient way to manage files, run software, and control hardware. Unlike today's visually driven operating systems, DOS operated entirely through text-based commands. Users had to type specific instructions to perform even basic tasks—like copying files, launching programs, or formatting disks. While this might seem cumbersome now, DOS was revolutionary at the time, offering a level of control and flexibility that helped shape modern computing.

Despite its limitations, DOS was remarkably stable and efficient, running smoothly on hardware with far less power than today's machines. Microsoft's MS-DOS and IBM's PC-DOS were the most popular versions, with MS-DOS eventually becoming the foundation for early Windows versions. Even after graphical interfaces took over, DOS remained relevant for years, especially in troubleshooting, system repairs, and running legacy software. Today, while DOS is no longer in everyday use, its influence lives on in command-line tools like the Windows Command Prompt and PowerShell. For anyone interested in computer history or low-level system operations, understanding DOS provides valuable insight into how far operating systems have come—and how much they still rely on principles established decades ago.

DOS was lightweight, fast, and ran efficiently on computers with very limited resources—some early PCs had as little as 64KB of RAM and no hard drives, relying instead on floppy disks. Microsoft's MS-DOS, the most well-known version, became the standard for IBM-compatible PCs, while other variants like DR-DOS and FreeDOS also existed. Unlike today's operating systems, DOS was single-tasking, meaning it could only run one program at a time. Yet, its simplicity made it reliable and easy to modify, which is why many classic games and business applications of the era were designed to run on it.

Today, while DOS is no longer in widespread use, its legacy persists. Modern command-line tools such as Windows Command Prompt, PowerShell, and even Linux terminals owe much of their design to DOS's straightforward, text-based approach. Additionally, retro computing enthusiasts and embedded systems developers still occasionally use DOS for its simplicity and efficiency. Understanding DOS isn't just about nostalgia—it's a lesson in computing history and a reminder of how far technology has advanced in just a few decades.

B.Objectives

Objectives of DOS (Disk Operating System)

1. Disk Management & File System Control

- DOS was primarily designed to manage storage devices, especially floppy disks and early hard drives.

- Provided commands like DIR (list files), COPY (duplicate files), DEL (delete files), and FORMAT (prepare disks) to organize data.

- Created a standardized file system (FAT12/FAT16) so programs could reliably store and retrieve information.

2. Hardware Abstraction & Device Control

- Acted as an intermediary between software and hardware components (keyboard, display, printer, etc.).

- Allowed programs to interact with hardware without needing to write custom drivers for each PC model.

- Managed memory allocation (conventional, extended) to prevent software conflicts.

3. User Command Execution

- Provided a text-based shell where users typed precise commands to perform tasks.

- Enabled scripting (via batch files with .BAT extension) to automate repetitive tasks.

- Commands were concise (e.g., CD to change directories, TYPE to display file contents) for efficiency.

4. Software Execution Platform

- Allowed applications (like WordPerfect or early games) to run in a standardized environment.

- Supported single-tasking—only one program could run at a time, ensuring stability on limited hardware.

- Used executable files (.COM, .EXE) to launch software with minimal overhead.

5. System Bootstrapping

- Loaded essential services during startup via IO.SYS, MSDOS.SYS, and COMMAND.COM.

- Provided a minimalistic environment to initialize hardware and prepare the PC for user input.

- Enabled booting from floppy disks—critical for recovery and system maintenance.

6. Resource Efficiency

- Designed to run on PCs with as little as 64KB RAM and no hard drive.

- Avoided graphical interfaces or multitasking to conserve CPU and memory.

- Optimized for speed, allowing near-instant command execution on slow processors (e.g., Intel 8088).

7. Backward Compatibility & Standardization

- Ensured software written for DOS could run across different PC clones (IBM, Compaq, etc.).

- Later versions of Windows (up to Windows ME) relied on DOS as their underlying foundation.

- Set conventions (e.g., 8.3 filenames, drive letters like C:) that persist in modern systems.

DOS achieved these objectives by prioritizing simplicity, direct hardware access, and reliability—qualities that made it the backbone of early computing and a lasting influence on operating system design.

C.REQUIREMENTS

System Requirements for MS-DOS / PC-DOS

I. Hardware Requirements

Processor (CPU):

- Minimum: Intel 8086/8088 (4.77 MHz)

- Recommended: Intel 80286 (6 MHz) or higher for later versions

Memory (RAM):

- Basic operation: 64 KB

- Optimal performance: 512 KB - 1 MB

Storage:

- Floppy disk drive (5.25" or 3.5")

  - Early versions: 160 KB - 720 KB capacity

  - Later versions: 1.2 MB - 1.44 MB

- Hard disk (optional but recommended for DOS 3.0+):

  - 10 MB - 100 MB

Display:

- Text mode: Monochrome (MDA) or CGA

- Graphics support: EGA or VGA (for later versions)

Input Devices:

- Keyboard (required)

- Mouse (optional, needed for some graphical shells)

Other Peripherals:

- Printer (parallel port)

- Serial devices (modems, mice)

II. Software Requirements

Essential System Files:

- IO.SYS (hardware interface)

- MSDOS.SYS (core OS functions)

- COMMAND.COM (command interpreter)

Included Utilities:

- File management: FORMAT, COPY, DEL

- Disk tools: CHKDSK, FDISK

- Text editor: EDIT.COM

Optional Components:

- HIMEM.SYS (extended memory manager)

- EMM386.EXE (expanded memory emulator)

- MOUSE.COM (mouse driver)

Special Notes

Memory Limitations:

- Conventional memory: 640 KB maximum

- Requires memory managers for extended/expanded memory

Compatibility:

- Single-tasking only (one program at a time)

- Can run on modern systems via emulators like DOSBox

Modern Alternatives:

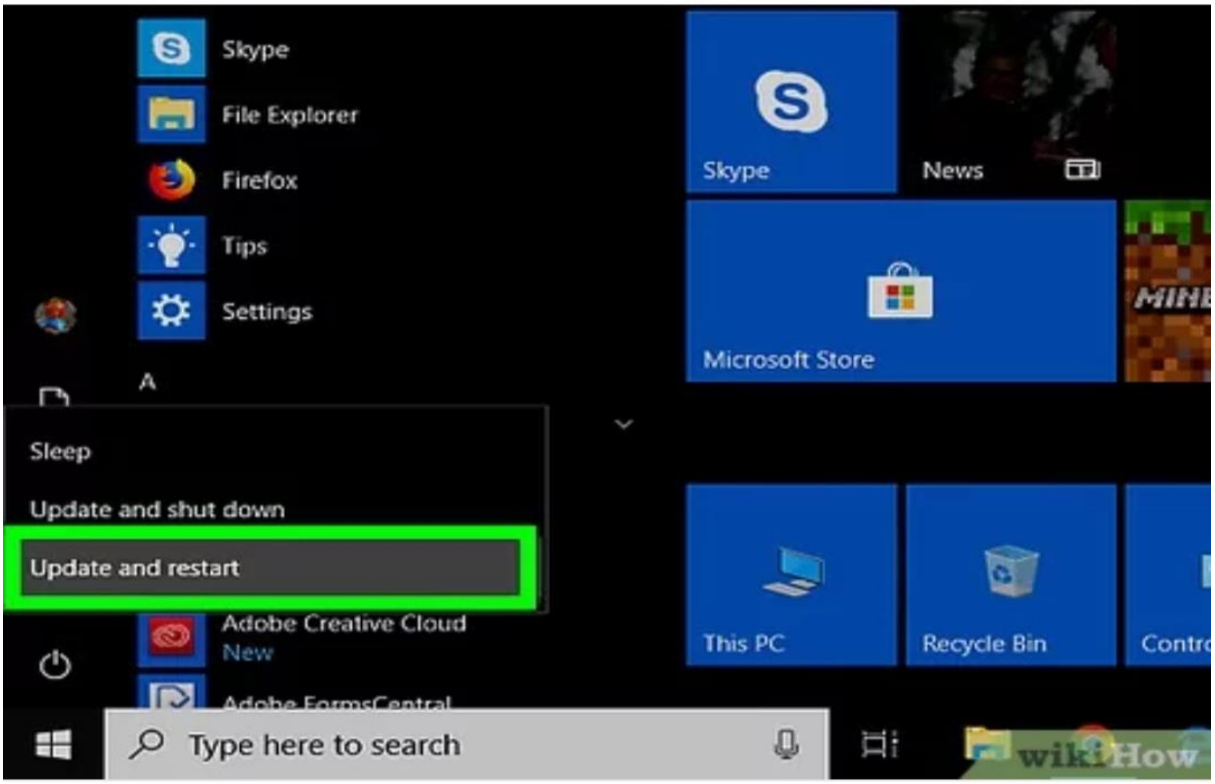
- FreeDOS (open-source compatible OS)

DOS was remarkably lightweight compared to modern operating systems, yet powerful enough to drive the PC revolution of the 1980s and early 1990s.

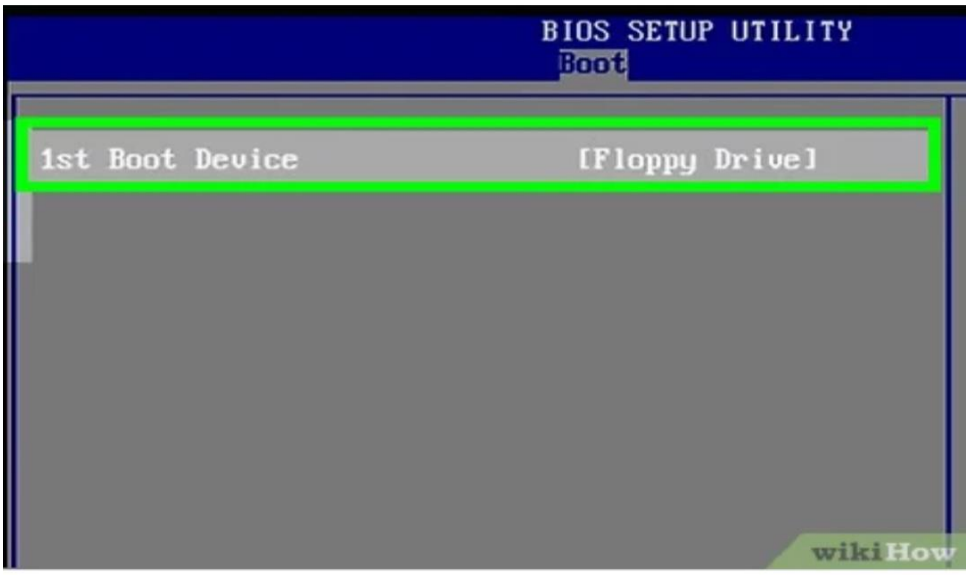
D. Installation steps

DOS is an early operating system from Microsoft that for the most part has now been replaced by Microsoft Windows. However, people still like to use DOS commands for activities like playing DOS games or using DOS programs like Robot. Since DOS is an older interface from Windows, the method to install DOS can also seem a little old-fashioned.

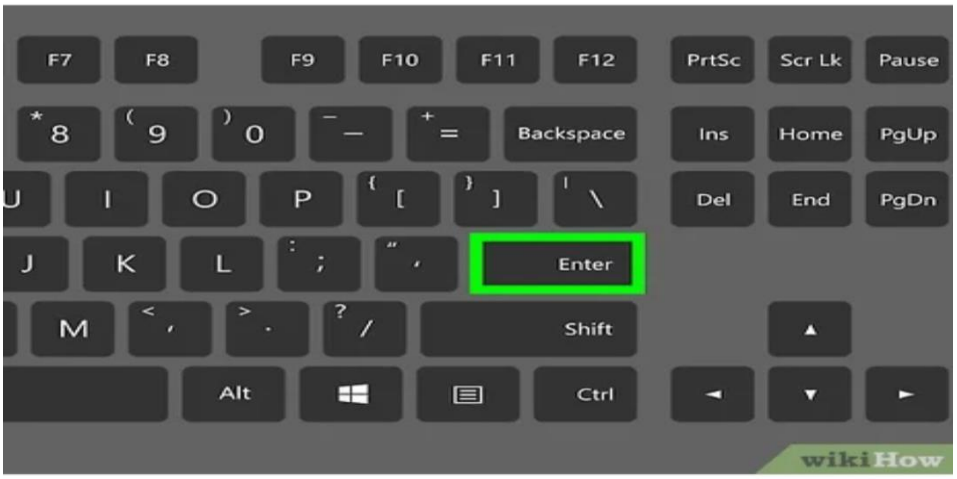
1.Purchase the DOS installation disks (they come in a set of 3 floppy disks). Usually you can purchase the set of floppy disks and an external floppy disk drive if needed for a very affordable price.



2.Insert the first installation disk in the floppy disk drive and restart your computer. The first disk is a bootable disk so you should see an option to hit a key to boot from the disk (the exact key can vary depending on which computer you have).

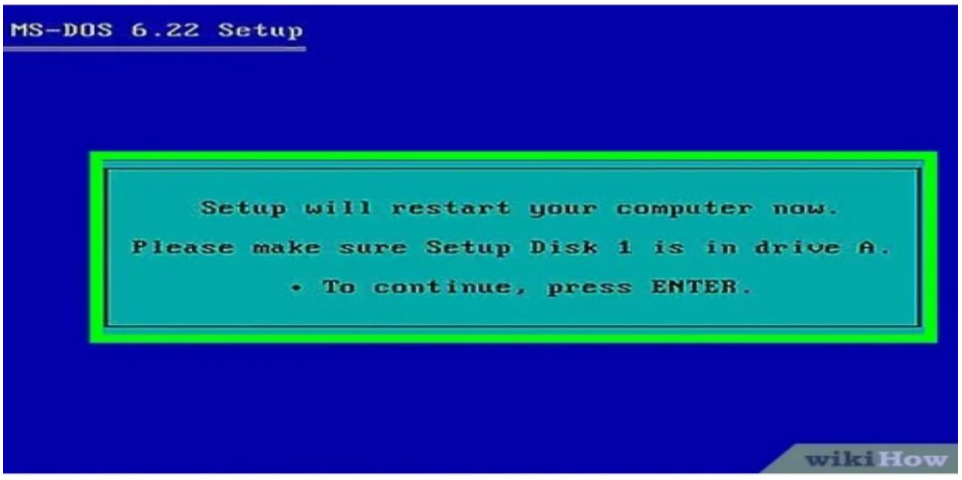


3.Hit the key to boot from the floppy drive. Your computer will abandon the usual boot process and instead you will see a blue screen with Microsoft DOS setup menu.



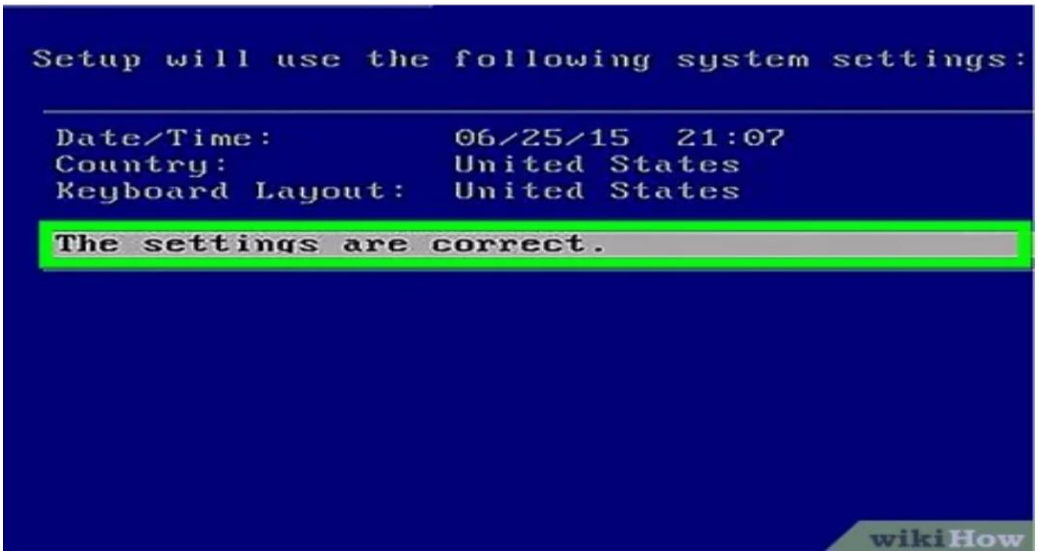
4.Press the Enter key on your keyboard to continue with the setup process.

5.Select "Configure first hard disk (recommended)" by hitting the Enter key again.

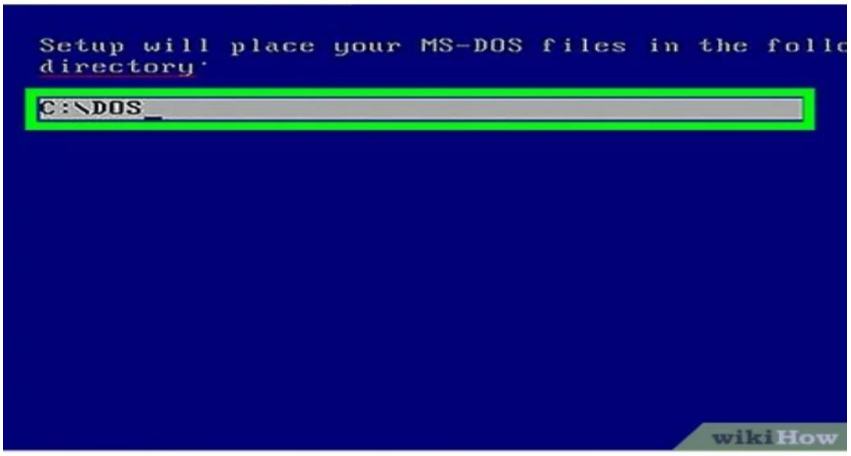


6.Read the information on the Caution window. You may want to consider the following options before continuing to install DOS.Press F3 if you want to back up your files. You will be exited from the DOS installation process.Hit the Escape key if you want to return to the previous window to review your choices.

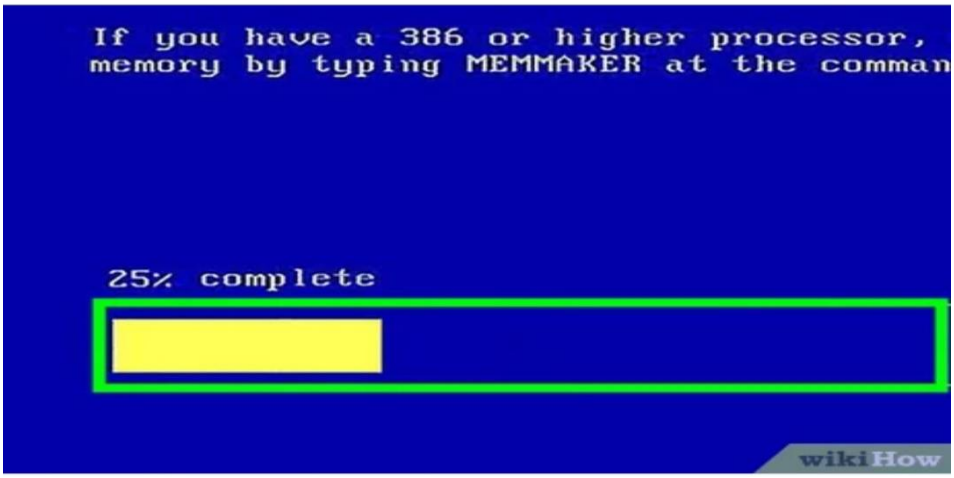
7.Wait while your system configurations are checked. After configuration check has completed your hard drive will be formatted using the FAT16 file system. Press the "Y" key on your keyboard if you want to continue installing DOS. The setup program will now continue to configure your hard drive and then restart your computer.



8.Confirm your date/time, country and keyboard layout in the next window. You can use your cursor keys to change the selections. If all settings are correct, you can highlight "The settings are correct" and press Enter.



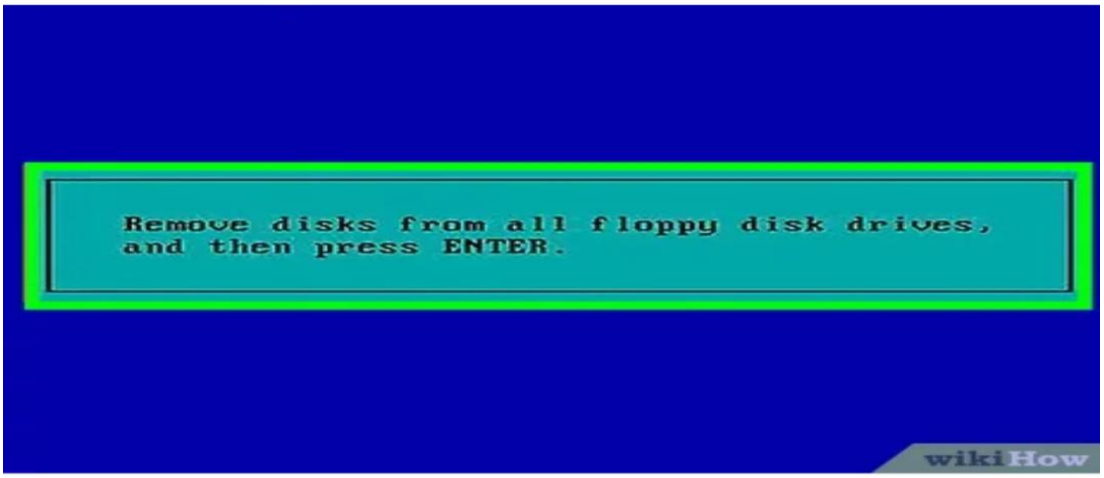
9.Install DOS in the default directory by leaving the directory location unchanged in the next window and just hitting Enter.



10.Let your computer copy the files from the first installation disk. A status bar will appear so you can see the progress.

11.Replace disk 1 with disk 2 when prompted and hit Enter to continue. You will see the status bar appear again.

12.Take out disk 2 and insert disk 3 when the next prompt window appears and press Enter. This will allow files to be copied from the third installation disk until all files have been copied.



13.Remove the final disk from your floppy drive when you see the message appear that says "Remove disks from all floppy disk drives." Press Enter after your floppy drive has been emptied.

14.Press Enter to restart your computer when you get to the window titled "MS-DOS Setup Complete.

Warnings

Data can be deleted while installing DOS if you format your hard drive. Create hard drive partitions or backup your important files to avoid data loss.

Issues Faced & Troubleshooting

Installing DOS (Disk Operating System)—such as MS-DOS, PC-DOS, or FreeDOS—on modern or older hardware can present several challenges. Here are the common problems faced during DOS installation:

1. Hardware Compatibility Issues

    - Modern PCs Lack Legacy Support – Newer motherboards may not have IDE controllers or BIOS (instead of UEFI), making it hard to boot from floppy disks or old HDDs.
    - USB Boot Problems – Many DOS versions don't natively support USB booting, requiring workarounds like PLoP Boot Manager or virtual machines.
    - Large Hard Drive Limitations – Older DOS versions (pre-DOS 7.1) struggle with disks over 8GB due to FAT16 limitations.

2. Storage Media Challenges

    - Floppy Disk Failures – If installing from floppies, disk corruption or bad sectors can cause errors.

    - CD/DVD Boot Issues – Some DOS versions don't support booting from optical drives without additional drivers.

    - SSD/Flash Drive Compatibility – DOS may not recognize NVMe SSDs or USB 3.0 drives without special drivers.

3. Partitioning & File System Problems

    - FAT32 vs. FAT16 – Older DOS versions (pre-Win95) only support FAT12/FAT16, limiting partition sizes.

    - MBR vs. GPT – DOS doesn't support GPT partitions, so disks must be MBR-formatted.

    - Unsupported NTFS/ExFAT – DOS cannot read NTFS or ExFAT partitions natively.

4. Driver & Peripheral Issues

    - No USB Mouse/Keyboard Support – DOS often requires PS/2 peripherals unless USB drivers (like DOSUSB) are loaded.

    - Lack of GPU Drivers – Modern graphics cards may not work, forcing VGA/SVGA mode.

- Sound & Network Problems – Many sound cards and network adapters need DOS-specific drivers.

5. Installation & Boot Errors

- "Invalid system disk" – BIOS not detecting the boot device correctly.

- "Bad or missing command interpreter" – Corrupt COMMAND.COM file.

- "HIMEM.SYS not loaded" – Memory management issues in CONFIG.SYS.

- "Not enough conventional memory" – Requires tweaking AUTOEXEC.BAT and CONFIG.SYS.

6. Virtual Machine (VM) Issues

- Slow Performance – Some VMs emulate old hardware poorly.

- Mouse/Keyboard Not Working – May need DOSBox or VirtualBox with legacy settings.

- Shared Folder Problems – DOS cannot access host files without additional tools.

Solutions & Workarounds

✓ Use FreeDOS (modern, better hardware support).

✓ Install in a VM (VirtualBox, DOSBox, VMware with legacy BIOS).

✓ Use a bootable USB with Rufus (FAT32, MBR).

✓ Load additional drivers (USB, HDD, network).

✓ Modify CONFIG.SYS & AUTOEXEC.BAT for memory optimization.

Filesystem support

DOS (including MS-DOS, PC-DOS, and FreeDOS) has limited native filesystem support due to its age. Here's a breakdown of which filesystems work and why:

Supported Filesystems in DOS

1. FAT12 (Default for floppies)

- Used for floppy disks (up to 16MB).

- Supported by all DOS versions.

2. FAT16 (Default for hard drives in early DOS)

- Max partition size: 2GB (with 32KB clusters) or 4GB (with 64KB clusters).

- Used in MS-DOS 4.0 to 6.22.

3. FAT32 (Introduced in Windows 95 OSR2 & later DOS versions)

- Supported by MS-DOS 7.1+ (Win95/98 DOS mode) and FreeDOS.

- Max partition size: 2TB (but DOS tools may struggle beyond 32GB).

- **Why**? FAT32 was introduced to overcome FAT16's limitations.

❌ Unsupported Filesystems in DOS

4. NTFS

- No native support (NTFS was introduced with Windows NT).

- Workaround: Use NTFS4DOS (3rd-party driver) for read-only access.

5. exFAT

- Not supported (introduced in 2006 for flash drives).

- Workaround: None (DOS lacks modern filesystem drivers).

6. ext2/ext3/ext4 (Linux)

- Not supported natively.

- Workaround: Use ext2fsd (3rd-party tool, but unreliable in DOS).

7. Btrfs/ZFS (Advanced Linux/Unix filesystems)

- No support (too modern and complex for DOS).

8. HFS+ (Mac OS Extended)

- Not supported (Mac filesystem).

9. APFS (Apple File System)

- No support (too new, designed for macOS).

Why DOS Only Supports FAT?

- Designed in the 1980s when FAT was the standard.

- No journaling or advanced features (DOS predates modern filesystems).

- No built-in drivers for newer filesystems (unlike Linux/Windows).

Best Choice for DOS?

✓ Use FAT32 if possible (larger partition support).

✓ FAT16 for maximum compatibility with older DOS versions.

✓ Avoid NTFS/ext4/exFAT unless using 3rd-party tools.

Advantages and Disadvantages of DOS (MS-DOS, PC-DOS, FreeDOS)

DOS (Disk Operating System) was the dominant OS for IBM-compatible PCs in the 1980s and early 1990s. While largely obsolete today, it still has niche uses. Below are its key advantages and disadvantages.

Advantages of DOS

1. Lightweight & Fast

- Extremely small memory footprint (can run on as little as 512KB RAM).

- Boots almost instantly compared to modern OSes.

2. Direct Hardware Access

- No abstraction layers (unlike Windows/Linux), allowing full control over hardware.

- Ideal for embedded systems, retro gaming, and low-level programming.

3. Simple & Stable

- No complex multitasking or background processes.

- Rarely crashes if hardware is compatible.

4. No Licensing Costs (FreeDOS)

- FreeDOS is open-source and free, unlike older proprietary versions (MS-DOS).

5. Legacy Software & Game Support

- Required to run old DOS applications & games that don't work on modern Windows.

- Used in industrial machines, CNC controllers, and retro computing.

6. Easy to Modify & Script

- Batch files (.BAT) allow simple automation.

- CONFIG.SYS & AUTOEXEC.BAT let users tweak system settings easily.

Disadvantages of DOS

1. No Multitasking

- Can only run one program at a time (no true multitasking).

- No protected memory (a crashing program can freeze the whole system).

2. No Native GUI (Graphical User Interface)

- Primarily command-line driven (users must type commands).

- Early GUIs (like DOS Shell) were very limited.

3. Limited Hardware Support

- No USB, Wi-Fi, or modern GPU drivers (requires third-party tools).

- No plug-and-play—users must manually configure IRQs and DMA.

4. File System Limitations

- Max 2GB partitions (FAT16) or 2TB (FAT32, but DOS tools struggle >32GB).

- No support for NTFS, exFAT, ext4, etc.

5. Security Risks

- No user permissions (any program can modify system files).

- No built-in antivirus or firewall (vulnerable to boot sector viruses).

6. Obsolete for Modern Computing

- Cannot run modern software (web browsers, office suites, etc.).

- No networking (unless using third-party TCP/IP stacks).

When Should You Use DOS Today?

✓ Retro gaming & emulation (DOSBox is a popular alternative).

✓ Legacy industrial machines (CNC, medical devices).

✓ Embedded systems & bootable rescue tools.

✓ Learning old-school computing & programming.

When Should You Avoid DOS?

- For daily computing (no modern software support).
- If you need security, multitasking, or networking.
- For large storage (beyond FAT32 limits).

Conclusion

DOS is fast, simple, and great for legacy use, but outdated for modern tasks. Alternatives like FreeDOS, DOSBox, or lightweight Linux distros (for CLI lovers) are better for most users today.

The Future Outlook and Recommendations

The Future Outlook and Recommendations for DOS (MS-DOS, PC-DOS, FreeDOS): A Comprehensive Analysis

DOS (Disk Operating System) was the dominant operating system for IBM-compatible PCs from the 1980s to the mid-1990s. While largely obsolete in mainstream computing, DOS still has niche applications in retro computing, embedded systems, and legacy industrial machines. This report explores:

- The current state of DOS in 2024.
- Future outlook for DOS-based systems.
- Recommendations for users, developers, and organizations still relying on DOS.

Current State of DOS in 2024

A. Active DOS Variants

1. FreeDOS (Open-source, actively maintained)

- The most modern DOS-compatible OS, supporting FAT32, large disks, and USB.
- Used in retro gaming, embedded systems, and legacy business applications.

2. MS-DOS / PC-DOS (Discontinued but preserved)

- Microsoft and IBM open-sourced MS-DOS 4.0 (2024), revealing historical insights.
- No official updates since the 1990s, but still used in some industrial machines.

DOSBox & Emulation

- DOSBox-X and 86Box allow running DOS on modern hardware (UEFI, x64).

B. Where DOS Is Still Used

- Retro Gaming (Classic DOS games like Doom, Prince of Persia).
- Industrial Machines (CNC, medical devices, factory automation).
- Legacy Software (Old accounting, inventory, and DOS-based databases).
- Education & Computer History (Teaching command-line basics).

3. Future Outlook for DOS

A. Long-Term Decline in Mainstream Use

- No modern software support (No web browsers, Office suites, or security updates).

- Hardware incompatibility (UEFI, NVMe SSDs, USB 3.0 not natively supported).

- Security risks (No built-in firewall, antivirus, or memory protection).

B. Niche Survival in Specific Areas

1. Retro Computing & Preservation

- DOS will remain popular among vintage computing enthusiasts.

- Projects like FreeDOS and DOSBox ensure backward compatibility.

2. Embedded & Industrial Systems

- Some factories still use DOS for CNC machines, lab equipment, and POS systems.

- Recommendation: Migrate to FreeDOS or lightweight Linux where possible.

3. Education & Historical Research

- Universities and museums use DOS to teach early computing concepts.

4. Virtualization & Emulation

- DOSBox-X and 86Box will continue improving DOS emulation on modern PCs.

C. Potential Future Developments

- FreeDOS 2.0? (Hypothetical future update with USB 3.0, UEFI boot, and better networking).

- Improved Emulation (Better GPU, sound, and input device support in DOSBox).

- Cloud-Based DOS? (Some services offer DOS in a browser for retro gaming).

4. Challenges Facing DOS in the Future

A. Hardware Obsolescence

- Floppy disks, IDE drives, and PS/2 ports are disappearing.

- Modern PCs lack legacy BIOS (UEFI is standard, making native DOS installs difficult).

B. Software & Driver Limitations

- No modern filesystem support (NTFS, exFAT, ext4).

- No USB 3.0, Wi-Fi, or GPU acceleration without third-party hacks.

C. Security Vulnerabilities

- No memory protection (Malware can corrupt the entire system).

- No encryption or secure boot (Data theft risk in industrial applications).

D. Declining Developer Interest

- Fewer programmers maintain DOS software.

- Most new development focuses on FreeDOS and emulators.

5. Recommendations for DOS Users in 2024 & Beyond

A. For Retro Gamers & Hobbyists

✓ Use DOSBox-X or 86Box for best compatibility.

✓ Try FreeDOS for modern hardware support (USB, large HDDs).

✓ Preserve old hardware (CRT monitors, Sound Blaster cards).

B. For Industrial & Legacy Systems

✓ Migrate to FreeDOS if still dependent on DOS software.

✓ Consider virtualization (Run DOS in a VM for security).

✓ Plan for eventual migration to Linux/Windows Embedded.

C. For Developers & Enthusiasts

✓ Contribute to FreeDOS (Improve USB, networking, and UEFI support).

✓ Develop DOS-compatible emulators (Better GPU and input device handling).

✓ Archive old DOS software (Prevent loss of historical programs).

D. For Educators & Historians

✓ Use DOS in computer history courses (Teach CLI basics).

✓ Document DOS's role in computing evolution.

6. Conclusion: Will DOS Survive?

- No as a mainstream OS (replaced by Windows, Linux, macOS).
- Yes in niche areas (retro gaming, embedded systems, education).
- Emulation & FreeDOS will keep DOS alive for decades.

Final Recommendation

If you rely on DOS today:

- Use FreeDOS for better hardware support.
- Switch to emulation (DOSBox) for security.
- Plan a long-term migration if used in business/industrial settings.

DOS will never regain its 1980s dominance, but it will remain a fascinating relic of computing history.

Virtualization

What is Virtualization?

Virtualization is a transformative computing technology that creates abstract, software-based versions of physical hardware, operating systems, storage, and networks, allowing multiple virtual instances to run simultaneously on a single physical machine through a hypervisor (Virtual Machine Monitor). This abstraction layer enables efficient resource utilization by partitioning CPU, memory, and storage into isolated virtual machines (VMs), each running its own guest OS as if it were on dedicated hardware, while maintaining security and flexibility. Unlike emulation, which simulates hardware at a performance cost, virtualization leverages hardware acceleration (Intel VT-x, AMD-V) for near-native speed, making it foundational for cloud computing (AWS, Azure), server consolidation, containers (Docker), and disaster recovery. By decoupling software from hardware, virtualization revolutionizes IT infrastructure, enabling scalable, cost-effective, and portable computing environments.

Virtualization is a fundamental computing paradigm that enables the creation of abstract, software-based representations of physical computing resources. At its core, virtualization decouples software from hardware, allowing multiple virtual instances to run on a single physical machine while maintaining isolation, security, and efficiency.

1. Core Definition

Virtualization is the process of creating virtual (rather than physical) versions of:

- Hardware (CPUs, memory, storage, networks)
- Operating Systems (running multiple OS instances on one machine)
- Applications (via containers or sandboxing)

These virtual instances operate as if they were running on dedicated physical hardware, even though they share underlying resources.

2. The Abstraction Layer

Virtualization introduces an abstraction layer between hardware and software. This layer is typically managed by a hypervisor (Virtual Machine Monitor, VMM), which:

- Partitions physical resources (CPU, RAM, disk, network)
- Allocates these resources to virtual machines (VMs)
- Isolates VMs from each other (for security and stability

Example:

A single server with:

- 16 CPU cores
- 64GB RAM

- 1TB SSD

Can be virtualized into four VMs, each with:

- 4 virtual CPUs
- 16GB RAM
- 250GB virtual disk

Each VM runs independently, unaware that it shares hardware with others.

3. Types of Virtualizations

Virtualization can be applied at different levels of the computing stack:

A. Hardware Virtualization (Full Virtualization)

- Creates complete virtual machines (VMs) that emulate physical hardware.
- Each VM runs its own guest OS (e.g., Windows on a Linux host).
- Requires CPU virtualization extensions (Intel VT-x, AMD-V).
- Used in data centers, cloud computing (AWS, Azure).

B. Paravirtualization

- Guest OS is modified to work efficiently with the hypervisor.
- Uses hypercalls (like system calls but for the hypervisor).
- Example: Xen (used in Amazon EC2).

C. OS-Level Virtualization (Containers)

- Lightweight alternative to full VMs.
- Shares the host OS kernel but isolates processes.
- Examples: Docker, Kubernetes, LXC.
- Faster startup, lower overhead than VMs.

D. Storage Virtualization

- Combines multiple physical storage devices into a single virtual storage pool.
- Used in SAN (Storage Area Networks), RAID, and cloud storage.

E. Network Virtualization

- Creates virtual networks independent of physical hardware.

- Used in SDN (Software-Defined Networking), VPNs, VLANs.

F. GPU Virtualization

- Allows multiple VMs to share a physical GPU.

- Used in AI/ML workloads, cloud gaming (NVIDIA vGPU).

4. How Virtualization Differs from Emulation

Feature	Virtualization	Emulation
Purpose	Run multiple OS instances efficiently	Simulate different hardware
Performance	Near-native (uses hardware acceleration)	Slower (software-based)
Use case	Cloud computing, servers	Running old console games (e.g., NES emulator)
examples	VMware, Hyper-V	QEMU (without KVM), DOSBox

5. Key Concepts in Virtualization

A. Hypervisor (VMM)

- The software/firmware that creates and manages VMs.

- Type 1 (Bare-Metal): Runs directly on hardware (e.g., VMware ESXi, Microsoft Hyper-V).

- Type 2 (Hosted): Runs on an OS (e.g., VirtualBox, VMware Workstation).

B. Virtual Machine (VM)

- A software-based computer with its own OS, CPU, memory, and storage.

- Can be snapshotted, cloned, and migrated between hosts.

C. Guest OS vs. Host OS

- Host OS: The primary OS running on physical hardware (for Type 2 hypervisors).

- Guest OS: The OS running inside a VM (e.g., Ubuntu on a Windows host).

D. Virtual Hardware

- Virtual CPUs (vCPUs), virtual disks (VHD, QCOW2), virtual NICs.

- Managed by the hypervisor.

6. Why Virtualization is Revolutionary

1. Breaks the "One Machine, One OS" Limitation

   - Before virtualization, each server ran a single OS. Now, one machine can run dozens of VMs.

2. Enables Cloud Computing

   - AWS, Azure, and Google Cloud rely on virtualization to offer scalable VMs.

3. Improves Disaster Recovery

   - VMs can be backed up and restored in seconds.

4. Reduces Hardware Costs

   - Companies no longer need a physical server for every workload.

5. Supports DevOps & CI/CD

   - Developers test applications in isolated VMs/containers.

7. Real-World Examples

- ✓ Cloud Computing: AWS EC2 uses Xen/KVM virtualization.
- ✓ Enterprise IT: Companies run VMware vSphere for server consolidation.
- ✓ Containers: Docker packages apps in lightweight virtualized environments.
- ✓ Gaming: NVIDIA GeForce Now streams games from virtualized GPUs.

8. Conclusion

Virtualization is the foundation of modern IT infrastructure, enabling efficiency, scalability, and flexibility. By abstracting hardware, it powers cloud computing, cybersecurity, DevOps, and AI workloads.

Next Steps:

- ✓ Explore how hypervisors work (CPU scheduling, memory ballooning).
- ✓ Compare virtualization vs. containers (Docker vs. VMware).
- ✓ Learn about nested virtualization (running VMs inside VMs).

System call(connect())

A system call is a fundamental mechanism that allows a user-level process to request a service from the operating system (OS) kernel. It acts as an interface between user applications and the OS, enabling programs to perform privileged operations such as file access, process management, and hardware interactions, which would otherwise be restricted for security and stability reasons.

Implementing a system call like connect() involves several layers of the operating system, including user-space interfaces, kernel-level socket handling, and network stack management. Below is a step-by-step explanation of how connect() works and how you might implement it in a Unix-like OS (e.g., Linux).

1. System Call Definition (User Space)

The connect() system call is used to establish a connection to a remote socket. Its signature is:

c

int connect(int sockfd, const struct sockaddr addr, socklen_t addrlen);

- sockfd: Socket file descriptor (created by socket()).

- addr: Pointer to a sockaddr structure (contains IP/port of the remote host).

- addrlen: Size of the sockaddr structure.

User-Space Example

c

include <sys/socket.h>

include <netinet/in.h>

include <stdio.h>

int main() {

    int sockfd = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in server_addr = {

        .sin_family = AF_INET,

        .sin_port = htons(8080),

```
    .sin_addr.s_addr = inet_addr("192.168.1.1")
  };

  if (connect(sockfd, (struct sockaddr )&server_addr, sizeof(server_addr)) < 0) {
    perror("Connection failed");
    return -1;
  }
  printf("Connected successfully!\n");
  return 0;
}
```

2. Kernel-Space Implementation

When connect() is called, the following happens in the kernel:

Step 1: System Call Entry (Switching to Kernel Mode)

- The user program executes connect(), which triggers a software interrupt (int 0x80 on older x86, syscall on modern x86-64).
- The CPU switches to kernel mode and jumps to the system call handler.

Step 2: System Call Dispatching

- The kernel checks the system call number (e.g., __NR_connect in Linux) in the system call table.
- The corresponding kernel function (sys_connect()) is called.

Step 3: Socket Lookup & Validation

- The kernel retrieves the socket object from the file descriptor (sockfd).
- Checks:
  - Is sockfd valid?
  - Is the socket type supported (e.g., SOCK_STREAM for TCP)?
  - Is the address family correct (AF_INET for IPv4)?

Step 4: Network Stack Interaction

- The kernel initiates a 3-way TCP handshake (if using TCP):

  1. SYN → Client sends a synchronization packet.

  2. SYN-ACK → Server acknowledges.

  3. ACK → Client confirms.

- For UDP, it simply binds the socket to the remote address.

Step 5: Error Handling & Return

- If successful, the socket is marked as connected.

- If failed (e.g., timeout, unreachable host), an error code (ECONNREFUSED, ETIMEDOUT) is returned.

3. Linux Kernel Implementation (Simplified)

Here's a pseudo-code version of sys_connect() in the Linux kernel:

c

```
SYSCALL_DEFINE3(connect, int, sockfd, struct sockaddr __user , uservaddr, int, addrlen) {
   struct socket sock;
   struct sockaddr_storage address;
   int err;
   // 1. Get socket from file descriptor
   sock = sockfd_lookup_light(sockfd, &err);
   if (!sock)
      return err;
   // 2. Copy user-space address to kernel
   if (copy_from_user(&address, uservaddr, addrlen)) {
      err = -EFAULT;
      goto out;
```

```
    }

    // 3. Security checks (e.g., SELinux)
    err = security_socket_connect(sock, (struct sockaddr )&address, addrlen);
    if (err)
        goto out;

    // 4. Protocol-specific connect (e.g., TCP/IP)
    err = sock->ops->connect(sock, (struct sockaddr )&address, addrlen, sock->file->f_flags);

out:
    sockfd_put(sock); // Release socket reference
    return err;
}
```

Key Kernel Structures Involved

Structure	Purpose
struct socket	Represents a network socket.
struct sock	Contains protocol-specific data (e.g., TCP state).
struct proto_ops	Hooks for protocol operations (connect(), bind(), etc.).

4. OS-Level Networking Management

The OS manages networking through:

1. Socket Layer

- Abstracts different protocols (TCP, UDP, RAW).

- Manages file descriptors and socket states.

2. Protocol Stack (TCP/IP)

- Handles packet routing, congestion control, retransmissions.

3. Device Drivers (NIC)

- Sends/receives packets via network interfaces.

4. Firewall & Security (Netfilter, eBPF)

- Filters connections (e.g., iptables rules).

5. Possible Errors & Edge Cases

- ECONNREFUSED → Remote host rejected the connection.

- ETIMEDOUT → No response from the server.

- EINPROGRESS (for non-blocking sockets) → Connection in progress.

- EACCES → Firewall blocked the connection.

6. Extending connect() for IPC (Inter-Process Communication)

For local communication (Unix domain sockets), connect() works similarly but skips the network stack:

c

```
struct sockaddr_un addr = {
   .sun_family = AF_UNIX,
   .sun_path = "/tmp/my_socket"
};
connect(sockfd, (struct sockaddr )&addr, sizeof(addr));
```

- The kernel routes the connection via in-memory pipes instead of TCP/IP.

Conclusion

The connect() system call is a critical part of networking and IPC, bridging user applications with the OS kernel's networking stack. Its implementation involves:

1. User → Kernel Transition (via syscall mechanism).
2. Socket & Protocol Handling (TCP/UDP).
3. Network Stack Interaction (3-way handshake for TCP).
4. Error Handling & Return.