Portable Executable

From Wikipedia, the free encyclopedia

[Jump to navigation](https://en.wikipedia.org/wiki/Portable_Executable#mw-head)[Jump to search](https://en.wikipedia.org/wiki/Portable_Executable#searchInput)

*Not to be confused with*[*Portable application*](https://en.wikipedia.org/wiki/Portable_application)*.*

|  |  |
| --- | --- |
|  | This article **needs additional citations for**[**verification**](https://en.wikipedia.org/wiki/Wikipedia:Verifiability). Please help [improve this article](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit) by [adding citations to reliable sources](https://en.wikipedia.org/wiki/Help:Referencing_for_beginners). Unsourced material may be challenged and removed. *Find sources:* ["Portable Executable"](https://www.google.com/search?as_eq=wikipedia&q=%22Portable+Executable%22) – [news](https://www.google.com/search?tbm=nws&q=%22Portable+Executable%22+-wikipedia) **·** [newspapers](https://www.google.com/search?&q=%22Portable+Executable%22+site:news.google.com/newspapers&source=newspapers) **·** [books](https://www.google.com/search?tbs=bks:1&q=%22Portable+Executable%22+-wikipedia) **·** [scholar](https://scholar.google.com/scholar?q=%22Portable+Executable%22) **·** [JSTOR](https://www.jstor.org/action/doBasicSearch?Query=%22Portable+Executable%22&acc=on&wc=on) *(December 2010) (*[*Learn how and when to remove this template message*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)* |

|  |  |
| --- | --- |
| **Portable Executable** | |
| [**Filename extension**](https://en.wikipedia.org/wiki/Filename_extension) | .acm, .ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp |
| [**Internet media type**](https://en.wikipedia.org/wiki/Media_type) | application/vnd.microsoft.portable-executable[[1]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-1) |
| **Developed by** | Currently: [Microsoft](https://en.wikipedia.org/wiki/Microsoft) |
| **Type of format** | [Binary](https://en.wikipedia.org/wiki/Binary_file), [executable](https://en.wikipedia.org/wiki/Executable), [object](https://en.wikipedia.org/wiki/Object_code), [shared libraries](https://en.wikipedia.org/wiki/Shared_libraries) |
| **Extended from** | [DOS MZ executable](https://en.wikipedia.org/wiki/DOS_MZ_executable) [COFF](https://en.wikipedia.org/wiki/COFF) |

The **Portable Executable** (PE) format is a [file format](https://en.wikipedia.org/wiki/File_format) for [executables](https://en.wikipedia.org/wiki/Executable), [object code](https://en.wikipedia.org/wiki/Object_file), [DLLs](https://en.wikipedia.org/wiki/Dynamic-link_library) and others used in 32-bit and 64-bit versions of [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) [operating systems](https://en.wikipedia.org/wiki/Operating_system). The PE format is a data structure that encapsulates the information necessary for the Windows OS loader to manage the wrapped executable code. This includes [dynamic library references for linking](https://en.wikipedia.org/wiki/Library_(computer_science)#Dynamic_linking), [API](https://en.wikipedia.org/wiki/Application_programming_interface) export and import tables, resource management data and [thread-local storage](https://en.wikipedia.org/wiki/Thread-local_storage) (TLS) data. On [NT](https://en.wikipedia.org/wiki/Windows_NT) operating systems, the PE format is used for [EXE](https://en.wikipedia.org/wiki/EXE), [DLL](https://en.wikipedia.org/wiki/Dynamic-link_library), [SYS](https://en.wikipedia.org/wiki/.sys) ([device driver](https://en.wikipedia.org/wiki/Device_driver)), and other file types. The [Extensible Firmware Interface (EFI)](https://en.wikipedia.org/wiki/Extensible_Firmware_Interface) specification states that PE is the standard executable format in EFI environments.[[2]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-2)

On Windows NT operating systems, PE currently supports the [x86](https://en.wikipedia.org/wiki/X86), [IA-32](https://en.wikipedia.org/wiki/IA-32), [x86-64](https://en.wikipedia.org/wiki/X86-64) (AMD64/Intel 64), [IA-64](https://en.wikipedia.org/wiki/IA-64), [ARM](https://en.wikipedia.org/wiki/ARM_architecture) and [ARM64](https://en.wikipedia.org/wiki/ARM64) [instruction set architectures](https://en.wikipedia.org/wiki/Instruction_set_architecture) (ISAs). Prior to [Windows 2000](https://en.wikipedia.org/wiki/Windows_2000), Windows NT (and thus PE) supported the [MIPS](https://en.wikipedia.org/wiki/MIPS_architecture), [Alpha](https://en.wikipedia.org/wiki/DEC_Alpha), and [PowerPC](https://en.wikipedia.org/wiki/PowerPC) ISAs. Because PE is used on [Windows CE](https://en.wikipedia.org/wiki/Windows_CE), it continues to support several variants of the MIPS, [ARM](https://en.wikipedia.org/wiki/ARM_architecture) (including [Thumb](https://en.wikipedia.org/wiki/ARM_architecture#Thumb)), and [SuperH](https://en.wikipedia.org/wiki/SuperH) ISAs. [[3]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-PE_Format_(Windows)-3)

Analogous formats to PE are [ELF](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format) (used in [Linux](https://en.wikipedia.org/wiki/Linux) and most other versions of [Unix](https://en.wikipedia.org/wiki/Unix)) and [Mach-O](https://en.wikipedia.org/wiki/Mach-O) (used in [macOS](https://en.wikipedia.org/wiki/MacOS) and [iOS](https://en.wikipedia.org/wiki/IOS)).



**Contents**

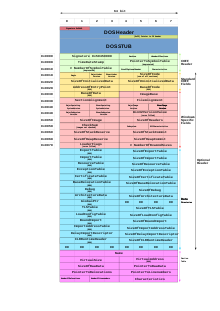
* [1History](https://en.wikipedia.org/wiki/Portable_Executable#History)
* [2Technical details](https://en.wikipedia.org/wiki/Portable_Executable#Technical_details)
  + [2.1Layout](https://en.wikipedia.org/wiki/Portable_Executable#Layout)
  + [2.2Import table](https://en.wikipedia.org/wiki/Portable_Executable#Import_table)
  + [2.3Relocations](https://en.wikipedia.org/wiki/Portable_Executable#Relocations)
* [3.NET, metadata, and the PE format](https://en.wikipedia.org/wiki/Portable_Executable#.NET,_metadata,_and_the_PE_format)
* [4Use on other operating systems](https://en.wikipedia.org/wiki/Portable_Executable#Use_on_other_operating_systems)
* [5See also](https://en.wikipedia.org/wiki/Portable_Executable#See_also)
* [6References](https://en.wikipedia.org/wiki/Portable_Executable#References)
* [7External links](https://en.wikipedia.org/wiki/Portable_Executable#External_links)

History[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=1)]

Microsoft migrated to the PE format from the 16-bit [NE](https://en.wikipedia.org/wiki/New_Executable) formats with the introduction of the [Windows NT 3.1](https://en.wikipedia.org/wiki/Windows_NT_3.1) operating system. All later versions of Windows, including Windows 95/98/ME and the [Win32s](https://en.wikipedia.org/wiki/Win32s) addition to Windows 3.1x, support the file structure. The format has retained limited legacy support to bridge the gap between [DOS](https://en.wikipedia.org/wiki/DOS)-based and NT systems. For example, PE/COFF headers still include a [DOS executable program](https://en.wikipedia.org/wiki/DOS_MZ_executable), which is by default a [DOS stub](https://en.wikipedia.org/wiki/DOS_stub) that displays a message like "This program cannot be run in DOS mode" (or similar), though it can be a full-fledged DOS version of the program (a later notable case being the Windows 98 SE installer).[[4]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-4) This constitutes a form of [fat binary](https://en.wikipedia.org/wiki/Fat_binary). PE also continues to serve the changing Windows platform. Some extensions include the .NET PE format (see below), a 64-bit version called PE32+ (sometimes PE+), and a specification for Windows CE.

Technical details[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=2)]

**Layout**[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=3)]

[](https://en.wikipedia.org/wiki/File:Portable_Executable_32_bit_Structure_in_SVG_fixed.svg)

Structure of a Portable Executable 32 bit

A PE file consists of a number of headers and sections that tell the [dynamic linker](https://en.wikipedia.org/wiki/Dynamic_linker) how to map the file into memory. An executable image consists of several different regions, each of which require different memory protection; so the start of each section must be aligned to a page boundary.[[5]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-5) For instance, typically the *.text* section (which holds program code) is mapped as execute/readonly, and the *.data* section (holding global variables) is mapped as no-execute/readwrite. However, to avoid wasting space, the different sections are not page aligned on disk. Part of the job of the dynamic linker is to map each section to memory individually and assign the correct permissions to the resulting regions, according to the instructions found in the headers.[[6]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-Peering_Inside-6)

**Import table**[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=4)]

One section of note is the *import address table* (IAT), which is used as a lookup table when the application is calling a function in a different module. It can be in the form of both [import by ordinal and import by name](https://en.wikipedia.org/wiki/Dynamic-link_library#Symbol_resolution_and_binding). Because a compiled program cannot know the memory location of the libraries it depends upon, an indirect jump is required whenever an API call is made. As the dynamic linker loads modules and joins them together, it writes actual addresses into the IAT slots, so that they point to the memory locations of the corresponding library functions. Though this adds an extra jump over the cost of an intra-module call resulting in a performance penalty, it provides a key benefit: The number of memory pages that need to be [copy-on-write](https://en.wikipedia.org/wiki/Copy-on-write) changed by the loader is minimized, saving memory and disk I/O time. If the compiler knows ahead of time that a call will be inter-module (via a dllimport attribute) it can produce more optimized code that simply results in an indirect call [opcode](https://en.wikipedia.org/wiki/Opcode).[[6]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-Peering_Inside-6)

**Relocations**[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=5)]

|  |  |
| --- | --- |
|  | This section needs to be **updated**. The reason given is: Use of [ASLR](https://en.wikipedia.org/wiki/Address_space_layout_randomization) and the trickery used to dodge the resulting problems. Please update this article to reflect recent events or newly available information. *(October 2017)* |

PE files normally do not contain [position-independent code](https://en.wikipedia.org/wiki/Position-independent_code). Instead they are compiled to a preferred [*base address*](https://en.wikipedia.org/wiki/Base_address), and all addresses emitted by the compiler/linker are fixed ahead of time. If a PE file cannot be loaded at its preferred address (because it's already taken by something else), the operating system will [*rebase*](https://en.wikipedia.org/wiki/Rebasing) it. This involves recalculating every absolute address and modifying the code to use the new values. The loader does this by comparing the preferred and actual load addresses, and calculating a [delta](https://en.wikipedia.org/wiki/Delta_encoding) value. This is then added to the preferred address to come up with the new address of the memory location. Base [relocations](https://en.wikipedia.org/wiki/Relocation_(computing)) are stored in a list and added, as needed, to an existing memory location. The resulting code is now private to the process and no longer [shareable](https://en.wikipedia.org/wiki/Shared_library), so many of the memory saving benefits of DLLs are lost in this scenario. It also slows down loading of the module significantly. For this reason rebasing is to be avoided wherever possible, and the DLLs shipped by Microsoft have base addresses pre-computed so as not to overlap. In the no rebase case PE therefore has the advantage of very efficient code, but in the presence of rebasing the memory usage hit can be expensive. This contrasts with [ELF](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format) which uses fully position-independent code and a global offset table, which trades off execution time in favor of lower memory usage.

.NET, metadata, and the PE format[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=6)]

In a .NET executable, the PE code section contains a stub that invokes the [CLR](https://en.wikipedia.org/wiki/Common_Language_Runtime) virtual machine startup entry, \_CorExeMain or \_CorDllMain in mscoree.dll, much like it was in [Visual Basic](https://en.wikipedia.org/wiki/Visual_Basic) executables. The virtual machine then makes use of .NET metadata present, the root of which, IMAGE\_COR20\_HEADER (also called "CLR header") is pointed to by IMAGE\_DIRECTORY\_ENTRY\_COMHEADER[[7]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-7) entry in the PE header's data directory. IMAGE\_COR20\_HEADER strongly resembles PE's optional header, essentially playing its role for the CLR loader.[[3]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-PE_Format_(Windows)-3)

The CLR-related data, including the root structure itself, is typically contained in the common code section, .text. It is composed of a few directories: metadata, embedded resources, strong names and a few for native-code interoperability. Metadata directory is a set of tables that list all the distinct .NET entities in the assembly, including types, methods, fields, constants, events, as well as references between them and to other assemblies.

Use on other operating systems[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=7)]

The PE format is also used by [ReactOS](https://en.wikipedia.org/wiki/ReactOS), as ReactOS is intended to be [binary-compatible](https://en.wikipedia.org/wiki/Binary_code_compatibility) with Windows. It has also historically been used by a number of other operating systems, including [SkyOS](https://en.wikipedia.org/wiki/SkyOS) and [BeOS](https://en.wikipedia.org/wiki/BeOS) R3. However, both SkyOS and BeOS eventually moved to [ELF](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format).

As the [Mono development platform](https://en.wikipedia.org/wiki/Mono_(software)) intends to be binary compatible with the Microsoft [.NET Framework](https://en.wikipedia.org/wiki/.NET_Framework), it uses the same PE format as the Microsoft implementation. The same goes for Microsoft's own cross-platform [.NET Core](https://en.wikipedia.org/wiki/.NET_Core).

On [x86](https://en.wikipedia.org/wiki/X86)(-64) [Unix-like](https://en.wikipedia.org/wiki/Unix-like) operating systems, Windows binaries (in PE format) can be executed with [Wine](https://en.wikipedia.org/wiki/Wine_(software)). The [HX DOS Extender](https://en.wikipedia.org/wiki/HX_DOS_Extender) also uses the PE format for native DOS 32-bit binaries, plus it can, to some degree, execute existing Windows binaries in DOS, thus acting like an equivalent of [Wine](https://en.wikipedia.org/wiki/Wine_(software)) for DOS.

On [IA-32](https://en.wikipedia.org/wiki/IA-32) and [x86-64](https://en.wikipedia.org/wiki/X86-64) [Linux](https://en.wikipedia.org/wiki/Linux) one can also run [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows)' [DLLs](https://en.wikipedia.org/wiki/Dynamic-link_library) under loadlibrary.[[8]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-8)

[Mac OS X 10.5](https://en.wikipedia.org/wiki/Mac_OS_X_10.5) has the ability to load and parse PE files, but is not binary compatible with Windows.[[9]](https://en.wikipedia.org/wiki/Portable_Executable#cite_note-chartier-9)

[UEFI](https://en.wikipedia.org/wiki/UEFI) and EFI firmware use Portable Executable files as well as the Windows [ABI](https://en.wikipedia.org/wiki/Application_Binary_Interface) x64 [calling convention](https://en.wikipedia.org/wiki/Calling_convention) for [applications](https://en.wikipedia.org/wiki/UEFI#Applications),

See also[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=8)]

* [EXE](https://en.wikipedia.org/wiki/EXE)
* [Executable and Linkable Format](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format)
* [Mach-O](https://en.wikipedia.org/wiki/Mach-O)
* [a.out](https://en.wikipedia.org/wiki/A.out)
* [Comparison of executable file formats](https://en.wikipedia.org/wiki/Comparison_of_executable_file_formats)
* [Executable compression](https://en.wikipedia.org/wiki/Executable_compression)
* [ar (Unix)](https://en.wikipedia.org/wiki/Ar_(Unix)) since all COFF libraries use that same format
* [Application virtualization](https://en.wikipedia.org/wiki/Application_virtualization)

References[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=9)]

* 1. [**^**](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-1) *Andersson, Henrik (2015-04-23).*[*"application/vnd.microsoft.portable-executable"*](https://www.iana.org/assignments/media-types/application/vnd.microsoft.portable-executable)*. IANA. Retrieved 2017-03-26.*
  2. [**^**](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-2) [*"UEFI Specification, version 2.4"*](http://www.uefi.org/specs/download/UEFI_2_4.pdf)*(PDF).*[[*permanent dead link*](https://en.wikipedia.org/wiki/Wikipedia:Link_rot)], a note on p.18, states that "this image type is chosen to enable UEFI images to contain Thumb and Thumb2 instructions while defining the EFI interfaces themselves to be in ARM mode."
  3. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-PE_Format_(Windows)_3-0) [***b***](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-PE_Format_(Windows)_3-1) [*"PE Format (Windows)"*](https://msdn.microsoft.com/en-us/library/windows/desktop/ms680547(v=vs.85).aspx)*. Retrieved 2017-10-21.*
  4. [**^**](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-4) E.g. Microsoft's linker has [/STUB switch](http://msdn.microsoft.com/en-us/library/7z0585h5.aspx) to attach one
  5. [**^**](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-5) [*"The Portable Executable File From Top to Bottom"*](http://www.csn.ul.ie/~caolan/pub/winresdump/winresdump/doc/pefile2.html)*. Retrieved 2017-10-21.*
  6. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-Peering_Inside_6-0) [***b***](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-Peering_Inside_6-1) [*"Peering Inside the PE: A Tour of the Win32 Portable Executable File"*](https://msdn.microsoft.com/en-us/library/ms809762.aspx)*. Retrieved 2017-10-21.*
  7. [**^**](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-7) The entry was previously used for [COM+](https://en.wikipedia.org/wiki/COM%2B) metadata in COM+ applications, hence the name
  8. [**^**](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-8) <https://github.com/taviso/loadlibrary>
  9. [**^**](https://en.wikipedia.org/wiki/Portable_Executable#cite_ref-chartier_9-0) *Chartier, David (2007-11-30).*[*"Uncovered: Evidence that Mac OS X could run Windows apps soon"*](https://arstechnica.com/journals/apple.ars/2007/11/30/uncovered-evidence-that-mac-os-x-could-run-windows-apps-soon)*. Ars Technica. Retrieved 2007-12-03. ... Steven Edwards describes the discovery that Leopard apparently contains an undocumented loader for Portable Executables, a type of file used in 32-bit and 64-bit versions of Windows. More poking around revealed that Leopard's own loader tries to find Windows DLL files when attempting to load a Windows binary.*

External links[[edit](https://en.wikipedia.org/w/index.php?title=Portable_Executable&action=edit&section=10)]

* [PE Format](https://docs.microsoft.com/en-us/windows/desktop/Debug/pe-format) (latest online document)
* [Microsoft Portable Executable and Common Object File Format Specification](https://web.archive.org/web/20081208121446/http:/www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx) (revision 8.1, [OOXML](https://en.wikipedia.org/wiki/OOXML) format)
* [Microsoft Portable Executable and Common Object File Format Specification](https://web.archive.org/web/20090126141159/http:/download.microsoft.com/download/e/b/a/eba1050f-a31d-436b-9281-92cdfeae4b45/pecoff.doc) (revision 6.0, [.doc](https://en.wikipedia.org/wiki/.doc) format)
* [The original Portable Executable article](http://msdn2.microsoft.com/en-us/library/ms809762.aspx) by [Matt Pietrek](https://en.wikipedia.org/wiki/Matt_Pietrek) ([MSDN](https://en.wikipedia.org/wiki/MSDN) Magazine, March 1994)
* [Part I. An In-Depth Look into the Win32 Portable Executable File Format](http://msdn.microsoft.com/en-us/magazine/cc301805.aspx) by [Matt Pietrek](https://en.wikipedia.org/wiki/Matt_Pietrek) ([MSDN](https://en.wikipedia.org/wiki/MSDN) Magazine, February 2002)
* [Part II. An In-Depth Look into the Win32 Portable Executable File Format](https://web.archive.org/web/20120915093039/http:/msdn.microsoft.com/en-us/magazine/cc301808.aspx) by [Matt Pietrek](https://en.wikipedia.org/wiki/Matt_Pietrek) ([MSDN](https://en.wikipedia.org/wiki/MSDN) Magazine, March 2002)
* [The .NET File Format by Daniel Pistelli](https://archive.today/20130130042959/http:/www.ntcore.com/files/dotnetformat.htm)
* [Ero Carrera's blog describing the PE header and how to walk through](http://blog.dkbza.org/)
* [PE Internals provides an easy way to learn the Portable Executable File Format](http://www.andreybazhan.com/pe-internals/)

DOS MZ executable

From Wikipedia, the free encyclopedia

[Jump to navigation](https://en.wikipedia.org/wiki/DOS_MZ_executable#mw-head)[Jump to search](https://en.wikipedia.org/wiki/DOS_MZ_executable#searchInput)

|  |  |
| --- | --- |
|  | This article **needs additional citations for**[**verification**](https://en.wikipedia.org/wiki/Wikipedia:Verifiability). Please help [improve this article](https://en.wikipedia.org/w/index.php?title=DOS_MZ_executable&action=edit) by [adding citations to reliable sources](https://en.wikipedia.org/wiki/Help:Referencing_for_beginners). Unsourced material may be challenged and removed. *Find sources:* ["DOS MZ executable"](https://www.google.com/search?as_eq=wikipedia&q=%22DOS+MZ+executable%22) – [news](https://www.google.com/search?tbm=nws&q=%22DOS+MZ+executable%22+-wikipedia) **·** [newspapers](https://www.google.com/search?&q=%22DOS+MZ+executable%22+site:news.google.com/newspapers&source=newspapers) **·** [books](https://www.google.com/search?tbs=bks:1&q=%22DOS+MZ+executable%22+-wikipedia) **·** [scholar](https://scholar.google.com/scholar?q=%22DOS+MZ+executable%22) **·** [JSTOR](https://www.jstor.org/action/doBasicSearch?Query=%22DOS+MZ+executable%22&acc=on&wc=on) *(April 2015) (*[*Learn how and when to remove this template message*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)* |

|  |  |
| --- | --- |
| **DOS MZ executable** | |
| [**Filename extension**](https://en.wikipedia.org/wiki/Filename_extension) | .[exe](https://en.wikipedia.org/wiki/EXE) |
| [**Magic number**](https://en.wikipedia.org/wiki/File_format#Magic_number) | MZ or ZM |
| **Type of format** | [Binary](https://en.wikipedia.org/wiki/Binary_file), [executable](https://en.wikipedia.org/wiki/Executable) |
| **Extended to** | [New Executable](https://en.wikipedia.org/wiki/New_Executable) [Linear Executable](https://en.wikipedia.org/wiki/Linear_Executable) [Portable Executable](https://en.wikipedia.org/wiki/Portable_Executable) |

The **DOS MZ executable** format is the [executable](https://en.wikipedia.org/wiki/Executable) [file format](https://en.wikipedia.org/wiki/File_format) used for .[EXE](https://en.wikipedia.org/wiki/EXE) files in [DOS](https://en.wikipedia.org/wiki/DOS).

The file can be identified by the [ASCII](https://en.wikipedia.org/wiki/ASCII) string "MZ" ([hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal): 4D 5A) at the beginning of the file (the "[magic number](https://en.wikipedia.org/wiki/Magic_number_(programming))"). "MZ" are the initials of [Mark Zbikowski](https://en.wikipedia.org/wiki/Mark_Zbikowski), one of leading developers of [MS-DOS](https://en.wikipedia.org/wiki/MS-DOS).[[1]](https://en.wikipedia.org/wiki/DOS_MZ_executable#cite_note-1)

The MZ DOS executable file is newer than the [COM executable format](https://en.wikipedia.org/wiki/COM_file) and differs from it. The DOS executable [header](https://en.wikipedia.org/wiki/File_header) contains [relocation](https://en.wikipedia.org/wiki/Relocation_table) information, which allows multiple segments to be loaded at arbitrary memory addresses, and it supports executables larger than 64k; however, the format still requires relatively low memory limits. These limits were later bypassed using [DOS extenders](https://en.wikipedia.org/wiki/DOS_extender).

The environment of an EXE program run by DOS is found in its [Program Segment Prefix](https://en.wikipedia.org/wiki/Program_Segment_Prefix).

EXE files normally have separate segments for the code, data, and stack. Program execution begins at address 0 of the code segment, and the stack pointer register is set to whatever value is contained in the header information (thus if the header specifies a 512 byte stack, the stack pointer is set to 200h). It is possible to not use a separate stack segment and simply use the code segment for the stack if desired.

The DS (data segment) register normally contains the same value as the CS (code segment) register and is not loaded with the actual segment address of the data segment when an EXE file is initialized; it is necessary for the programmer to set it themselves, generally done via the following instructions:

MOV AX, @DATA

MOV DS, AX

In the original DOS 1.x API, it was also necessary to have the DS register pointing to the segment with the PSP at program termination; this was done via the following instructions:

PUSH DS

XOR AX, AX

PUSH AX

Program termination would then be performed by a RETF instruction, which would retrieve the original segment address with the PSP from the stack and then jump to address 0, which contained an INT 20h instruction.

The DOS 2.x API introduced a new program termination function, INT 21h Function 4Ch which does not require saving the PSP segment address at the start of the program, and Microsoft advised against the use of the older DOS 1.x method.



**Contents**

* [1Compatibility](https://en.wikipedia.org/wiki/DOS_MZ_executable#Compatibility)
* [2See also](https://en.wikipedia.org/wiki/DOS_MZ_executable#See_also)
* [3Further reading](https://en.wikipedia.org/wiki/DOS_MZ_executable#Further_reading)
* [4References](https://en.wikipedia.org/wiki/DOS_MZ_executable#References)
* [5External links](https://en.wikipedia.org/wiki/DOS_MZ_executable#External_links)

Compatibility[[edit](https://en.wikipedia.org/w/index.php?title=DOS_MZ_executable&action=edit&section=1)]

MZ DOS executables can be run from DOS and [Windows 9x](https://en.wikipedia.org/wiki/Windows_9x)-based operating systems. 32-bit [Windows NT](https://en.wikipedia.org/wiki/Windows_NT)-based operating systems can execute them using their built-in [Virtual DOS machine](https://en.wikipedia.org/wiki/Virtual_DOS_machine) (although some graphics modes are unsupported). 64-bit versions of Windows cannot execute them. Alternative ways to run these executables include [DOSBox](https://en.wikipedia.org/wiki/DOSBox), [DOSEMU](https://en.wikipedia.org/wiki/DOSEMU), [Wine](https://en.wikipedia.org/wiki/Wine_(software)), and [Cygwin](https://en.wikipedia.org/wiki/Cygwin).

MZ DOS executables can be created by linkers, like [Digital Mars](https://en.wikipedia.org/wiki/Digital_Mars) [Optlink](https://en.wikipedia.org/wiki/Optlink), [MS linker](https://en.wikipedia.org/w/index.php?title=MS_linker&action=edit&redlink=1), [VALX](https://en.wikipedia.org/w/index.php?title=VALX&action=edit&redlink=1) or [Open Watcom](https://en.wikipedia.org/wiki/Open_Watcom)'s WLINK; additionally, [FASM](https://en.wikipedia.org/wiki/FASM) can create them directly.

See also[[edit](https://en.wikipedia.org/w/index.php?title=DOS_MZ_executable&action=edit&section=2)]

* [DOS](https://en.wikipedia.org/wiki/DOS)
* [DOS extender](https://en.wikipedia.org/wiki/DOS_extender)
* [Comparison of executable file formats](https://en.wikipedia.org/wiki/Comparison_of_executable_file_formats)
* [DOS API](https://en.wikipedia.org/wiki/DOS_API)
* [Executable compression](https://en.wikipedia.org/wiki/Executable_compression)

Further reading[[edit](https://en.wikipedia.org/w/index.php?title=DOS_MZ_executable&action=edit&section=3)]

* *Paul, Matthias R. (2002-10-07) [2000].*[*"Re: Run a COM file"*](https://groups.google.com/d/msg/alt.msdos.programmer/d7blJjY0H5M/Qu3VeTOIGVcJ)*.*[*Newsgroup*](https://en.wikipedia.org/wiki/Usenet_newsgroup)*:*[*alt.msdos.programmer*](news:alt.msdos.programmer)*.*[*Archived*](https://archive.today/20170903230312/https:/groups.google.com/forum/%23!msg/alt.msdos.programmer/d7blJjY0H5M/Qu3VeTOIGVcJ)*from the original on 2017-09-03. Retrieved 2017-09-03.*
* [[1]](https://groups.google.com/d/msg/alt.lang.asm/PNOd9zfYow0/vXbab16j4XwJ)

References[[edit](https://en.wikipedia.org/w/index.php?title=DOS_MZ_executable&action=edit&section=4)]

* 1. [**^**](https://en.wikipedia.org/wiki/DOS_MZ_executable#cite_ref-1) [Inside Windows: An In-Depth Look into the Win32 Portable Executable File Format - MSDN Magazine, February 2002](https://msdn.microsoft.com/en-us/magazine/bb985992.aspx). "Every PE file begins with a small MS-DOS executable. ... The first bytes of a PE file begin with the traditional MS-DOS header, called an IMAGE\_DOS\_HEADER. The only two values of any importance are e\_magic and e\_lfanew. ... The e\_magic field (a WORD) needs to be set to the value 0x5A4D. ... In ASCII representation, 0x5A4D is MZ, the initials of Mark Zbikowski, one of the original architects of MS-DOS."

External links[[edit](https://en.wikipedia.org/w/index.php?title=DOS_MZ_executable&action=edit&section=5)]

* [MZ DOS EXE file format](http://www.delorie.com/djgpp/doc/exe/)
* [A closer look at EXE DOS stub](https://marcin-chwedczuk.github.io/a-closer-look-at-portable-executable-msdos-stub)

COFF

From Wikipedia, the free encyclopedia

[Jump to navigation](https://en.wikipedia.org/wiki/COFF#mw-head)[Jump to search](https://en.wikipedia.org/wiki/COFF#searchInput)

|  |  |
| --- | --- |
|  | This article **needs additional citations for**[**verification**](https://en.wikipedia.org/wiki/Wikipedia:Verifiability). Please help [improve this article](https://en.wikipedia.org/w/index.php?title=COFF&action=edit) by [adding citations to reliable sources](https://en.wikipedia.org/wiki/Help:Referencing_for_beginners). Unsourced material may be challenged and removed. *Find sources:* ["COFF"](https://www.google.com/search?as_eq=wikipedia&q=%22COFF%22) – [news](https://www.google.com/search?tbm=nws&q=%22COFF%22+-wikipedia) **·** [newspapers](https://www.google.com/search?&q=%22COFF%22+site:news.google.com/newspapers&source=newspapers) **·** [books](https://www.google.com/search?tbs=bks:1&q=%22COFF%22+-wikipedia) **·** [scholar](https://scholar.google.com/scholar?q=%22COFF%22) **·** [JSTOR](https://www.jstor.org/action/doBasicSearch?Query=%22COFF%22&acc=on&wc=on) *(March 2010) (*[*Learn how and when to remove this template message*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)* |

|  |  |
| --- | --- |
| **COFF** | |
| [**Filename extension**](https://en.wikipedia.org/wiki/Filename_extension) | none, .o, .obj |
| **Developed by** | [AT&T Corporation](https://en.wikipedia.org/wiki/AT%26T_Corporation) |
| **Type of format** | [Binary](https://en.wikipedia.org/wiki/Binary_file), [executable](https://en.wikipedia.org/wiki/Executable), [object](https://en.wikipedia.org/wiki/Object_code), [shared libraries](https://en.wikipedia.org/wiki/Shared_libraries) |
| **Extended to** | [XCOFF](https://en.wikipedia.org/wiki/XCOFF), [ECOFF](https://en.wikipedia.org/wiki/ECOFF), [Portable Executable](https://en.wikipedia.org/wiki/Portable_Executable) |

The **Common Object File Format** (**COFF**) is a [format](https://en.wikipedia.org/wiki/File_format) for [executable](https://en.wikipedia.org/wiki/Executable), [object code](https://en.wikipedia.org/wiki/Object_code), and [shared library](https://en.wikipedia.org/wiki/Shared_libraries) [computer files](https://en.wikipedia.org/wiki/Computer_file) used on [Unix](https://en.wikipedia.org/wiki/Unix) systems. It was introduced in [Unix System V](https://en.wikipedia.org/wiki/Unix_System_V), replaced the previously used [a.out](https://en.wikipedia.org/wiki/A.out) format, and formed the basis for extended specifications such as [XCOFF](https://en.wikipedia.org/wiki/XCOFF) and [ECOFF](https://en.wikipedia.org/wiki/ECOFF), before being largely replaced by [ELF](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format), introduced with [SVR4](https://en.wikipedia.org/wiki/UNIX_System_V). COFF and its variants continue to be used on some [Unix-like](https://en.wikipedia.org/wiki/Unix-like) systems, on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) ([PE Format](https://en.wikipedia.org/wiki/Portable_Executable)), in [EFI](https://en.wikipedia.org/wiki/Extensible_Firmware_Interface) environments and in some embedded development systems.



**Contents**

* [1History](https://en.wikipedia.org/wiki/COFF#History)
* [2Features](https://en.wikipedia.org/wiki/COFF#Features)
  + [2.1Symbolic debugging information](https://en.wikipedia.org/wiki/COFF#Symbolic_debugging_information)
  + [2.2Relative virtual address](https://en.wikipedia.org/wiki/COFF#Relative_virtual_address)
* [3Problems](https://en.wikipedia.org/wiki/COFF#Problems)
* [4See also](https://en.wikipedia.org/wiki/COFF#See_also)
* [5Notes](https://en.wikipedia.org/wiki/COFF#Notes)
* [6References](https://en.wikipedia.org/wiki/COFF#References)
* [7External links](https://en.wikipedia.org/wiki/COFF#External_links)

History[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=1)]

The original Unix object file format [a.out](https://en.wikipedia.org/wiki/A.out_(file_format)) is unable to adequately support [shared libraries](https://en.wikipedia.org/wiki/Library_(computing)), foreign format identification[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)], or explicit address linkage[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]. As development of [Unix-like](https://en.wikipedia.org/wiki/Unix-like) systems continued both inside and outside AT&T, different solutions to these and other issues emerged.

COFF was introduced in 1983, in AT&T's [UNIX System V](https://en.wikipedia.org/wiki/UNIX_System_V) for non-VAX 32-bit platforms such as the [3B20](https://en.wikipedia.org/wiki/AT%26T_Computer_Systems)[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]. Improvements over the existing AT&T *a.out* format included arbitrary sections, explicit processor declarations, and explicit address linkage.

However, the COFF design was both too limited and incompletely specified: there was a limit on the maximum number of sections, a limit on the length of section names, included source files, and the symbolic debugging information was incapable of supporting real world languages such as [C](https://en.wikipedia.org/wiki/C_(programming_language)), much less newer languages like [C++](https://en.wikipedia.org/wiki/C%2B%2B), or new processors. All real world implementations of COFF were necessarily violations of the standard as a result. This led to numerous COFF extensions. [IBM](https://en.wikipedia.org/wiki/IBM) used the [XCOFF](https://en.wikipedia.org/wiki/XCOFF) format in [AIX](https://en.wikipedia.org/wiki/AIX_operating_system); [DEC](https://en.wikipedia.org/wiki/Digital_Equipment_Corporation), [SGI](https://en.wikipedia.org/wiki/Silicon_Graphics) and others used [ECOFF](https://en.wikipedia.org/wiki/ECOFF); and numerous SysV ports and tool chains targeting embedded development each created their own, incompatible, variations.

With the release of SVR4, AT&T replaced COFF with [ELF](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format).

While extended versions of COFF continue to be used for some Unix-like platforms, primarily in embedded systems, perhaps the most widespread use of the COFF format today is in [Microsoft](https://en.wikipedia.org/wiki/Microsoft)'s [Portable Executable](https://en.wikipedia.org/wiki/Portable_Executable) (PE) format. Developed for [Windows NT](https://en.wikipedia.org/wiki/Windows_NT), the PE format (sometimes written as PE/COFF) uses a COFF header for object files, and as a component of the PE header for executable files.[[1]](https://en.wikipedia.org/wiki/COFF#cite_note-1)

Features[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=2)]

COFF's main improvement over *a.out* was the introduction of multiple named sections in the object file. Different object files could have different numbers and types of sections.

**Symbolic debugging information**[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=3)]

The COFF symbolic debugging information consists of symbolic (string) names for program functions and variables, and line number information, used for setting breakpoints and tracing execution.

Symbolic names are stored in the COFF symbol table. Each symbol table entry includes a name, storage class, type, value and section number. Short names (8 characters or fewer) are stored directly in the symbol table; longer names are stored as an offset into the string table at the end of the COFF object.

Storage classes describe the type entity the symbol represents, and may include external variables (C\_EXT), automatic (stack) variables (C\_AUTO), register variables (C\_REG), functions (C\_FCN), and many others. The symbol type describes the interpretation of the symbol entity's value and includes values for all the [C](https://en.wikipedia.org/wiki/C_(programming_language)) data types.

When compiled with appropriate options, a COFF object file will contain line number information for each possible break point in the text section of the object file. Line number information takes two forms: in the first, for each possible break point in the code, the line number table entry records the address and its matching line number. In the second form, the entry identifies a symbol table entry representing the start of a function, enabling a breakpoint to be set using the function's name.

Note that COFF was not capable of representing line numbers or debugging symbols for included source as with header files rendering the COFF debugging information virtually useless without incompatible extensions.

**Relative virtual address**[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=4)]

When a COFF file is generated, it is not usually known where in memory it will be loaded. The [virtual address](https://en.wikipedia.org/wiki/Virtual_address) where the first byte of the file will be loaded is called image [base address](https://en.wikipedia.org/wiki/Base_address). The rest of the file is not necessarily loaded in a contiguous block, but in different **sections**.

Relative virtual addresses (RVAs) are not to be confused with standard virtual addresses. A **relative virtual address** is the [virtual address](https://en.wikipedia.org/wiki/Virtual_address) of an object from the file once it is loaded into memory, minus the base address of the file image. If the file were to be mapped literally from disk to memory, the RVA would be the same as that of the offset into the file, but this is actually quite unusual.

Note that the RVA term is only used with objects in the image file. Once loaded into memory, the image base address is added, and ordinary VAs are used.

Problems[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=5)]

The COFF file header stores the date and time that the object file was created as a 32-bit binary integer, representing the number of seconds since the [Unix epoch](https://en.wikipedia.org/wiki/Unix_epoch), 1 January 1970 00:00:00 UTC. Dates occurring after [19 January 2038](https://en.wikipedia.org/wiki/Year_2038_problem) cannot be stored in this format.

See also[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=6)]

* [Comparison of executable file formats](https://en.wikipedia.org/wiki/Comparison_of_executable_file_formats)

Notes[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=7)]

1. [**^**](https://en.wikipedia.org/wiki/COFF#cite_ref-1) [Microsoft Corporation 2006b](https://en.wikipedia.org/wiki/COFF#CITEREFMicrosoft_Corporation2006b)

References[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=8)]

* [MIPS COFF Spec](https://web.archive.org/web/20140723105157/http:/www-scf.usc.edu/~csci402/ncode/coff_8h-source.html)
* *Gircys, Gintaras (1988), Understanding and Using COFF, O'Reilly and Associates,*[*ISBN*](https://en.wikipedia.org/wiki/ISBN_(identifier))[*0-937175-31-5*](https://en.wikipedia.org/wiki/Special:BookSources/0-937175-31-5)
* *Microsoft Corporation (2006b),*[*Common Object File Format (COFF)*](https://web.archive.org/web/20061216043713/http:/support.microsoft.com/?id=121460)*(Revision 4.1 ed.), archived from*[*the original*](http://support.microsoft.com/?id=121460)*on 2006-12-16, retrieved 2007-06-02*

External links[[edit](https://en.wikipedia.org/w/index.php?title=COFF&action=edit&section=9)]

* [More on the PE Format](http://msdn.microsoft.com/en-us/magazine/cc301805.aspx) and [Public documentation](http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx) at Microsoft.com