Resource fork

From Wikipedia, the free encyclopedia

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

The **resource fork** is a [fork or section](https://en.wikipedia.org/wiki/Fork_(file_system)) of a [file](https://en.wikipedia.org/wiki/Computer_file) on [Apple](https://en.wikipedia.org/wiki/Apple_Inc.)'s [classic Mac OS](https://en.wikipedia.org/wiki/Classic_Mac_OS) [operating system](https://en.wikipedia.org/wiki/Operating_system), which was also carried over to the modern [macOS](https://en.wikipedia.org/wiki/MacOS) for compatibility, used to store structured data along with the unstructured data stored within the **data fork**.

A resource fork stores information in a specific form, containing details such as icon bitmaps, the shapes of windows, definitions of menus and their contents, and application code ([machine code](https://en.wikipedia.org/wiki/Machine_code)). For example, a word processing file might store its text in the data fork, while storing any embedded images in the same file's resource fork. The resource fork is used mostly by [executables](https://en.wikipedia.org/wiki/Executable), but every file is able to have a resource fork.



**Contents**

* [1The Macintosh file system](https://en.wikipedia.org/wiki/Resource_fork#The_Macintosh_file_system)
* [2Resource identifiers](https://en.wikipedia.org/wiki/Resource_fork#Resource_identifiers)
* [3Editing resource forks](https://en.wikipedia.org/wiki/Resource_fork#Editing_resource_forks)
* [4How a resource fork is accessed](https://en.wikipedia.org/wiki/Resource_fork#How_a_resource_fork_is_accessed)
* [5Data types in a resource fork](https://en.wikipedia.org/wiki/Resource_fork#Data_types_in_a_resource_fork)
* [6Major resource types](https://en.wikipedia.org/wiki/Resource_fork#Major_resource_types)
* [7Major resource editors](https://en.wikipedia.org/wiki/Resource_fork#Major_resource_editors)
* [8Compatibility problems](https://en.wikipedia.org/wiki/Resource_fork#Compatibility_problems)
* [9Other operating systems](https://en.wikipedia.org/wiki/Resource_fork#Other_operating_systems)
* [10See also](https://en.wikipedia.org/wiki/Resource_fork#See_also)
* [11References](https://en.wikipedia.org/wiki/Resource_fork#References)
* [12External links](https://en.wikipedia.org/wiki/Resource_fork#External_links)

The Macintosh file system[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=1)]

*Further information:*[*Hierarchical File System*](https://en.wikipedia.org/wiki/Hierarchical_File_System)

Originally conceived and implemented by programmer [Bruce Horn](https://en.wikipedia.org/wiki/Bruce_Horn), the resource fork was used for three purposes with [Macintosh file system](https://en.wikipedia.org/wiki/Hierarchical_File_System):

* It was used to store all graphical data on disk until it was needed, then retrieved, drawn on the screen, and thrown away. This software variant of [virtual memory](https://en.wikipedia.org/wiki/Virtual_memory) helped Apple to reduce memory requirements from 1 MB in the [Apple Lisa](https://en.wikipedia.org/wiki/Apple_Lisa) to 128 KB in Macintosh.
* Because all the pictures and text were stored separately in a resource fork, it could be used to allow a non-programmer to translate an application for a foreign market, a process called [internationalization and localization](https://en.wikipedia.org/wiki/Internationalization_and_localization).
* It could be used to distribute nearly all of the components of an application in a single file, reducing clutter and simplifying application installation and removal.

The resource fork is implemented in all of the [file systems](https://en.wikipedia.org/wiki/File_system) used for system drives on the [Macintosh](https://en.wikipedia.org/wiki/Macintosh) ([MFS](https://en.wikipedia.org/wiki/Macintosh_File_System), [HFS](https://en.wikipedia.org/wiki/Hierarchical_File_System) and [HFS Plus](https://en.wikipedia.org/wiki/HFS_Plus)). The presence of a resource fork makes it easy to store a variety of additional information, such as allowing the system to display the correct [icon](https://en.wikipedia.org/wiki/Icon_(computing)) for a file and open it without the need for a [file extension](https://en.wikipedia.org/wiki/File_extension) in the file name. While access to the data fork works like file access on any other operating system — pick a file, pick a byte offset, read some data — access to the resource fork works more like extracting structured records from a [database](https://en.wikipedia.org/wiki/Database). ([Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) also has a concept of "[resources](https://en.wikipedia.org/wiki/Resource_(Windows))", but these are completely unrelated to resources in Mac OS.)

The resource fork is sometimes used to store the [metadata](https://en.wikipedia.org/wiki/Metadata) of a file, although it can also be used for storing the actual data, as was the case with font files in the classic Mac operating systems. Note that the Macintosh file systems also have a separate area for metadata distinct from either the data or resource fork. Being part of the catalogue entry for the file, it is much faster to access this. However, the amount of data stored here is minimal, being just the creation and modification timestamps, the file type and creator codes, fork lengths, and the file name. Some files have only a resource fork. Classic [68k](https://en.wikipedia.org/wiki/68k) applications are one example, where even the executable code is contained in resources of type 'CODE'. Later [PowerPC](https://en.wikipedia.org/wiki/PowerPC) binaries store the executable code in the data fork.

As resource forks are supported only on the file systems HFS and HFS Plus, they cannot be used on [operating systems](https://en.wikipedia.org/wiki/Operating_system) which use other file systems. At present, HFS is supported only by the Macintosh operating system, which means that only machines running Mac OS can use resource forks. Even in a Mac OS system, resource forks cannot be used if the [Unix File System](https://en.wikipedia.org/wiki/Unix_File_System) has been [installed](https://en.wikipedia.org/wiki/Installation_(computer_programs)). In the HFS Plus file system, which is currently the system most commonly used under Mac OS, settings can be made to allow other forks in addition to the data and resource forks, to create a "multi-fork" application. However, as forks can make it difficult to exchange files with other operating systems, this feature is not in common use. Even in macOS, resource forks are seldom used anymore.

Currently, macOS supports resource forks on Windows [SMB](https://en.wikipedia.org/wiki/Server_Message_Block) shares by creating a hidden file with the characters ".\_" added at the beginning of the file name, in the same directory as the data fork file.

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

Each resource has an [OSType](https://en.wikipedia.org/wiki/OSType) identifier (a four byte value) and an ID (a [signed](https://en.wikipedia.org/wiki/Signedness) 16-bit [word](https://en.wikipedia.org/wiki/Word_(computer_architecture))), as well as an optional name. There are standardised resource types for [dialog boxes](https://en.wikipedia.org/wiki/Dialog_box) ('DITL'), images ('[PICT](https://en.wikipedia.org/wiki/PICT)'), sounds ('snd ') — and even for [executable](https://en.wikipedia.org/wiki/Executable) binaries ('CODE') which, until the advent of the [PowerPC](https://en.wikipedia.org/wiki/PowerPC) [processor](https://en.wikipedia.org/wiki/Central_processing_unit), were without exception stored in the resource fork. Subroutines for rendering [windows](https://en.wikipedia.org/wiki/Window_(computing)) are stored in their own type of resources ('WDEF'), subroutines for rendering menus in theirs ('MDEF'), and if there is a type of data you think does not fit any of the standardised categories, you can just as well use a type of your own (e.g. 'John') — actually any four characters or 32-bit value can serve as a resource type. This arrangement enabled users to easily customise not only individual applications but also the operating system itself, using tools such as [ResEdit](https://en.wikipedia.org/wiki/ResEdit) to modify the resources of an application file or any of the system files.

Within an application or other code, resources can be loaded simply using a combination of their type, ID or name, without regard to how and where they are stored in the resource fork. The client is returned a [Handle](https://en.wikipedia.org/wiki/Reference_(computer_science)) to the loaded resource which can then be accessed like any other heap-based data. The OS component that facilitates this is the *Resource Manager*. In addition to abstracting the details of the data storage from the data itself, the Resource Manager also arranges sets of open resource forks into a stack, with the most recently opened file on top. When trying to load a resource, it will look in the top of the stack first, (perhaps the current document's resource fork), then the next one down (the application's resource fork), then the next one (system resource forks). This arrangement is very powerful — it permits local resources to override more global ones lower down — so an application can provide its own icons or fonts in place of the standard system ones, for example. It also allows an application to load resources from the system using the same API as any other resource, without regard to where or how that resource is stored — to the application, all resources are equally available and easy to use. The system reserves resource IDs in a certain range to help avoid resource conflicts arising from this. Resource Manager APIs allow the programmer to manipulate the stack and modify the search behaviour.

Editing resource forks[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=3)]

*"Resource editor" and "Resource editing" redirect here. For editing resources in Microsoft Windows, see*[*Resource editor (Windows)*](https://en.wikipedia.org/wiki/Resource_editor_(Windows))*.*

As the resource fork can be edited with a resource editor such as [ResEdit](https://en.wikipedia.org/wiki/ResEdit), it can be used to [localize](https://en.wikipedia.org/wiki/Internationalization_and_localization) and customize [software](https://en.wikipedia.org/wiki/Software). In addition, most resource editors allow visual editing of data. In [macOS](https://en.wikipedia.org/wiki/MacOS), it is possible to use resources when developing an application. However, if the application may need to be used in [UFS](https://en.wikipedia.org/wiki/Unix_File_System), it is also possible to configure it so that the entire resource fork is moved to the data fork, using the Raw Resource File setting. The [integrated development environments](https://en.wikipedia.org/wiki/Integrated_development_environment) distributed for free by [Apple Inc.](https://en.wikipedia.org/wiki/Apple_Inc.), which include [MPW](https://en.wikipedia.org/wiki/Macintosh_Programmer%27s_Workshop) and [Apple Developer's Tools](https://en.wikipedia.org/wiki/Xcode), include a [compiler](https://en.wikipedia.org/wiki/Compiler) called Rez. This uses a dedicated language, also called Rez, which can be used to create a resource fork by compiling [source code](https://en.wikipedia.org/wiki/Source_code). A decompiler, DeRez, which can be used to change a resource fork back into Rez code is also included.

In the structure of the resource fork, there is a piece of data called a "resource map" which stores the positions of resource data items. This can be used to allow [random access](https://en.wikipedia.org/wiki/Random_access) to resource data based on the defined IDs and names. The resource fork can be thought of as consisting of essentially two objects, the resource map and the resource data itself, but in fact each data type is a hierarchical structure which stores multiple items of data. The format in which the information in the resource data is stored is defined based on the types of information, which are known as "resource types." Resource data often makes references to other types of data.

In macOS, forks are named *file*/..namedfork/*forkname*, *e.g.*, the resource fork of the file IMG\_0593.jpg is IMG\_0593.jpg/..namedfork/rsrc. The ls command supports a -l@ option which lists a file's forks.

How a resource fork is accessed[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=4)]

Resource forks appear as the [extended attribute](https://en.wikipedia.org/wiki/Extended_file_attributes) com.apple.ResourceFork.[[1]](https://en.wikipedia.org/wiki/Resource_fork#cite_note-1)

Previously resource forks were accessed via the 'Resource Manager' [API](https://en.wikipedia.org/wiki/API). This API is now deprecated.[[2]](https://en.wikipedia.org/wiki/Resource_fork#cite_note-2)

Under the deprecated API:

1. When a resource fork is accessed, data including the start position and length of the resource data and resource map is read in from the header.
2. If a resource type to read in has been specified, a check is performed to make sure that type is present in the resource list, and the number of items of data containing that type and their offsets in the resource reference list from the start position of the resource map is found.
3. The resource ID, the offset of the resource name, the resource properties, and the offset of the data from the start position of the resource data is found.
4. If resource data with the specified ID or name is present in the resource data, the offset obtained above is accessed, the data length is found, and all the data stored there is read in, and returned as the return value.

File Manager APIs such as PBOpenRF() also allowed access to the raw resource fork; however, they should be used only for applications such as copying a file – Apple strongly warns against using the resource fork as a "second data fork."

From the [POSIX](https://en.wikipedia.org/wiki/POSIX) interface, the resource fork could be accessed as *filename*/..namedfork/rsrc or as *filename*/rsrc; the shorter form was deprecated in [Mac OS X v10.4](https://en.wikipedia.org/wiki/Mac_OS_X_v10.4) and removed completely in [Mac OS X v10.7](https://en.wikipedia.org/wiki/Mac_OS_X_v10.7).[[3]](https://en.wikipedia.org/wiki/Resource_fork#cite_note-3)

Data types in a resource fork[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=5)]

The smallest elements making up a resource fork are called data types. There are several data types. After a resource fork is accessed, its contents can be found by reading it in as appropriate for the data types defined in advance. Placing definitions inside the program stating how data is to be treated makes it possible to store resources called TMPL resources as well. Using this method increases the visibility of the data when viewed with a program such as ResEdit, making later editing simpler. As the Macintosh platform originated with Motorola-based processors (68k and PPC), the data is serialized to disk in [big endian](https://en.wikipedia.org/wiki/Endianness) format.

The following is a list of the major data types, in alphabetical order.

|  |  |  |
| --- | --- | --- |
| **Data type** | **actual name** | **Description** |
| BBIT | binary bit | Represents a single boolean bit (true or false). Normally the number of BBITs must be a multiple of 8. |
| BOOL | boolean | Represents a boolean value. It consists of 2 bytes; 256 is true, and 0 is false. |
| CHAR | character | Represents a one-byte character. |
| CSTR | C string | Represents a string of the form used in the [C programming language](https://en.wikipedia.org/wiki/C_(programming_language)): a [null-terminated string](https://en.wikipedia.org/wiki/Null-terminated_string) of bytes. |
| DLNG | decimal long word integer | A decimal long word (4 byte integer). Represents values between approximately − 2.1 billion and 2.1 billion. |
| HEXD | hex dump | Indicates that the data from this position to the end is hexadecimal. This is used to represent code resources or compressed data. |
| HLNG | long word hexadecimal | This data is treated as a 4 byte hexadecimal value. It is used, among other things, to represent integers greater than 2.1 billion, such as unsigned long values in C. |
| PSTR | Pascal string | Represents a Pascal string, with the first byte giving the length of the string. |
| TNAM | type name | A string representing a value such as a [creator code](https://en.wikipedia.org/wiki/Creator_code), which is always 4 bytes long. |
| RECT | rectangle | Represents the coordinates of the corners of a rectangle (top, left, bottom, right). Always 8 bytes long. |

Major resource types[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=6)]

The type codes below, like the above datatypes, are used as type identifiers for more than resource forks themselves: they are used to identify file themselves, to describe data in the clipboard, and much more.

Note that types must be 4 bytes long, so types like snd and STR actually have a space (0x20) at the end.

|  |  |  |
| --- | --- | --- |
| **Name of resource type** | **actual name** | **Description** |
| alis | alias | Stores an alias to another file, in a resource fork of a file whose "alias" attribute bit is set |
| ALRT | alert | Defines the shape of an application alert box |
| APPL | application | Stores application information |
| BNDL | bundle | Defines data such as a file type icon used in an application |
| cicn | color icon | Defines a color icon used in data |
| clut | color look-up table | Defines a color palette used in data |
| CNTL | control | Defines the details of a component positioned in a window |
| CODE | code resource | Stores the machine code for the program |
| CURS | cursor | Defines the shape of a monochrome cursor (8 × 8 bit square) |
| DITL | dialog item list | Defines a component of a window |
| DLOG | dialog | Defines the shape of a dialog box for an application |
| FREF | file reference | Defines a file type handled by an application |
| hfdr | icon balloon help | Defines the contents and shape of the balloon help displayed when the cursor hovers over the file in the Finder |
| icl8 | 8 bit icon list | Defines an icon displayed in the Finder |
| icns | 32 bit icon list | Defines an icon displayed in the Finder |
| ICON | icon | Defines a monochrome item used in data |
| kind | file description | Defines a description of a file type |
| MBAR | menu bar | Defines a menu and menu bar for an application |
| MDEF | menu definition | Defines a menu for an application. Can also be used to define menus with complex shapes such as color palettes. |
| MENU | menu | Defines the menu items in an application |
| MooV | movie | Stores a QuickTime movie |
| open | open | Defines a file type which the application can open |
| PICT | picture | Stores a PICT image contained in the file |
| PREF | preference | Stores the environment settings for an application |
| snd | sound | Stores a sound used in the file |
| STR | string | Stores a string or hexadecimal data used in the file |
| STR# | string list | Stores multiple strings used in the file |
| styl | style | Defines style information, such as the font, color and size of text |
| TEXT | text | Stores text |
| TMPL | template | Defines the format for the resource data |
| vers | version | Defines the [version](https://en.wikipedia.org/wiki/Software_versioning) or region of use of the file |
| WDEF | window definition | Defines a window for the application. Windows of an unspecified shape can also be defined. |
| WIND | window | Defines the shape of an application window |

Major resource editors[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=7)]

[**ResEdit**](https://en.wikipedia.org/wiki/ResEdit)

Distributed free of charge by Apple. Can be used for visual editing of resource data. If the structure of data is known, it can display a range of different types of data in a visual format. Does not run on modern macOS.

**Resorcerer**

Expensive, but popular, as it can be used for visual editing of many more types of data than ResEdit.

**HexEdit**

A binary editor, which in fact is normally used more for editing the data fork rather than the resource fork.

[**ResKnife**](https://en.wikipedia.org/wiki/ResKnife)

Open‐source editor for [Mac OS X](https://en.wikipedia.org/wiki/Mac_OS_X); no longer maintained.

**Rezycle**

A macOS tool that extracts resources from a resource fork into separate binary files while converting many types into formats suitable for modern development.

**resource\_dasm**

An open-source resource extractor for macOS, also capable of converting many resources into modern formats.

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

The complexity of programming with resource forks has led to compatibility problems when accessing other file systems via file sharing protocols such as [AFP](https://en.wikipedia.org/wiki/Apple_Filing_Protocol), [SMB](https://en.wikipedia.org/wiki/Server_Message_Block), [NFS](https://en.wikipedia.org/wiki/Network_File_System) and [FTP](https://en.wikipedia.org/wiki/FTP), when storing to non-HFS volumes, or when transmitting files to other systems in other ways (such as via email). The AFP protocol natively supports Resource Forks, and so resource forks are typically transmitted to these volumes as-is, and stored by the server transparently to clients. The SMB protocol supports a file metadata system similar to Macintosh forks known as [Alternate Data Streams](https://en.wikipedia.org/wiki/Alternate_data_streams#Microsoft) (ADSes hereafter). macOS did not support storing resource forks in ADSes on SMB volumes by default until [Mac OS X v10.6](https://en.wikipedia.org/wiki/Mac_OS_X_v10.6). In previous versions of the OS, including upgraded versions of 10.6, this feature can be enabled with a param change or by creating a special file.[[4]](https://en.wikipedia.org/wiki/Resource_fork#cite_note-4)

Networked file sharing protocols such as NFSv3 and FTP do not have a concept of file metadata, and so there is no way to natively store resource forks. This is also true when writing to certain types of local file systems, including UFS, and on SMB volumes where Alternate Data Stream support is not enabled. In those cases, macOS stores metadata and resource forks using a technique called [AppleDouble](https://en.wikipedia.org/wiki/AppleSingle_and_AppleDouble_formats), in which the data fork is written as one file, and the resource fork and metadata are written as an entirely separate file preceded by a ".\_" naming convention. For example: **ExampleFile.psd** would contain the data fork, and **.\_ExampleFile.psd** would contain the resource fork and metadata.

Compatibility problems can arise because macOS will handle storage of resource forks differently, depending on macOS version, settings, and file system type. For example, on an SMB network with a mixture of 10.5 and 10.6 clients. A freshly installed 10.6 client will look for and store resource forks on an SMB volume in ADSes, but the 10.5 client will (by default) ignore ADSes and use [AppleDouble](https://en.wikipedia.org/wiki/AppleSingle_and_AppleDouble_formats) format to handle forks. If a fileserver supports both AFP and NFS, then clients using NFS will store files in [AppleDouble](https://en.wikipedia.org/wiki/AppleSingle_and_AppleDouble_formats) format, whereas AFP users will stored the resource fork natively. In those cases, compatibility can sometimes be maintained by forcing clients to use, or not use, [AppleDouble](https://en.wikipedia.org/wiki/AppleSingle_and_AppleDouble_formats) format.

Many fileservers providing AFP support do not natively support resource forks on their local file systems. In those cases the forks may be stored in special ways, such as specially named files, special directories, or even Alternate Data Streams.

Another challenge is preserving resource forks when transmitting files using non-resource fork-aware applications or with certain transfer methods, including email and FTP. A number of file formats, such as [MacBinary](https://en.wikipedia.org/wiki/MacBinary) and [BinHex](https://en.wikipedia.org/wiki/BinHex), have been created to handle this. Command-line system tools SplitForks and FixupResourceForks allow manual flattening and merging of resource forks. In addition, a file server seeking to present file systems to Macintosh clients must accommodate the resource fork as well as the data fork of files; [UNIX](https://en.wikipedia.org/wiki/UNIX) servers providing AFP support usually implement this with hidden directories.

Older applications written with the [Carbon API](https://en.wikipedia.org/wiki/Carbon_API) have a potential issue when being ported to the current [Intel](https://en.wikipedia.org/wiki/Intel) Macs. While the Resource Manager and operating system know how to deserialize data correctly for common resources like "snd " or "moov", resources created using TMPL resources have to be byte swapped manually to ensure file interoperability between PPC and Intel-based versions of an application. (While the resource map and other implementation details are big endian, the Resource Manager by itself doesn't have any knowledge of the contents of a generic resource, and so cannot perform the byte swapping automatically.)

Until the advent of [Mac OS X v10.4](https://en.wikipedia.org/wiki/Mac_OS_X_v10.4), the standard UNIX command line utilities in macOS (such as cp and mv) did not respect resource forks. To copy files with resource forks, one had to use ditto or CpMac and MvMac.

Other operating systems[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=9)]

The concept of a *resource manager* for graphics objects, to save memory, originated in the OOZE package on the [Xerox Alto](https://en.wikipedia.org/wiki/Xerox_Alto) in Smalltalk-76.[[5]](https://en.wikipedia.org/wiki/Resource_fork#cite_note-5) The concept is now largely universal in all modern operating systems. However, the concept of the resource fork remains peculiar to the Macintosh. Most operating systems used a binary file containing resources, which is then "tacked onto" the end of an existing program file. This solution is used on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) for instance, and similar solutions are used with the [X Window System](https://en.wikipedia.org/wiki/X_Window_System), although the resources are often left as a separate file.

The [Windows NT](https://en.wikipedia.org/wiki/Windows_NT) [NTFS](https://en.wikipedia.org/wiki/NTFS) can support forks (and so can be a file server for Mac files), the native feature providing that support is called an [alternate data stream](https://en.wikipedia.org/wiki/Fork_(file_system)). Windows operating system features (such as the standard Summary tab in the Properties page for non-Office files) and Windows applications are use them and Microsoft was developing a [next-generation file system](https://en.wikipedia.org/wiki/WinFS) that has this sort of feature as basis.

Early versions of the [BeOS](https://en.wikipedia.org/wiki/BeOS) implemented a database within the file system, which could be used in a manner analogous to a resource fork. Performance issues led to a change in later releases to a system of complex file system attributes. Under this system resources were handled in a fashion somewhat more analogous to the Mac.

[AmigaOS](https://en.wikipedia.org/wiki/AmigaOS) does not use forked files. Its [executable files](https://en.wikipedia.org/wiki/Executable_file) are internally divided into a modular structure of large pieces ([hunk](https://en.wikipedia.org/wiki/Amiga_Hunk)) capable of storing code, data, and additional information. Similarly, data and project files have a [chunk](https://en.wikipedia.org/wiki/Chunk_(information)) structure codified in the [IFF](https://en.wikipedia.org/wiki/Interchange_File_Format) standard. Other file types are stored similarly to other operating systems. Though not strictly a resource fork, [AmigaOS](https://en.wikipedia.org/wiki/AmigaOS) stores meta data in files known as **.info** files. .info files can be identified by the .info extension; for example, if you save a project to a disk, two files will be saved, MyProject and MyProject.info. MyProject would be the actual project data and MyProject.info would contain the project icon, information regarding which program is needed to open the project (since there is no [application binding](https://en.wikipedia.org/wiki/Application_binding) in AmigaOS), special project options and any user comments. .info files are invisible on the Amiga's desktop ([Workbench](https://en.wikipedia.org/wiki/Workbench_(AmigaOS))). The icon on the desktop, taken from the .info itself, is the [interface metaphor](https://en.wikipedia.org/wiki/Interface_metaphor) through which the user interacts both with the project itself and its associated .info file. A dialog box accessible by right-clicking the icon allows the user to see and modify the metadata present in the .info file. .info files can be seen as individual files in the [Command line interface](https://en.wikipedia.org/wiki/Command_line_interface) or a [File manager](https://en.wikipedia.org/wiki/File_manager). Modern AmigaOS clones ([AROS](https://en.wikipedia.org/wiki/AROS_Research_Operating_System), [MorphOS](https://en.wikipedia.org/wiki/MorphOS) and [AOS4](https://en.wikipedia.org/wiki/AmigaOS_4)) inherit the structure (complete with metadata) of the .info files of older AmigaOS versions, and can also accept standard [PNG](https://en.wikipedia.org/wiki/Portable_Network_Graphics) graphic files as icon bitmaps in their .info files.

[NeXT](https://en.wikipedia.org/wiki/NeXT) operating systems [NeXTSTEP](https://en.wikipedia.org/wiki/NeXTSTEP) and [OPENSTEP](https://en.wikipedia.org/wiki/OPENSTEP), their successor, [macOS](https://en.wikipedia.org/wiki/MacOS), and other systems like [RISC OS](https://en.wikipedia.org/wiki/RISC_OS) implemented another solution. Under these systems the resources are left in an original format, for instance, pictures are included as complete [TIFF](https://en.wikipedia.org/wiki/TIFF) files instead of being encoded into some sort of container. These resources are then placed in a directory along with the executable code and "raw data". The directory (called a "[bundle](https://en.wikipedia.org/wiki/Bundle_(macOS))" or "[application directory](https://en.wikipedia.org/wiki/Application_directory)") is then presented to the user as the application itself. This solution provides all of the same functionality as the resource fork, but allows the resources to be easily manipulated by any application – a "resource editor" (like [ResEdit](https://en.wikipedia.org/wiki/ResEdit)) is not needed. From the [command line interface](https://en.wikipedia.org/wiki/Command_line_interface), the bundle appears to be a normal directory. This approach was not an option on the [classic Mac OS](https://en.wikipedia.org/wiki/Classic_Mac_OS), since the file system ([MFS](https://en.wikipedia.org/wiki/Macintosh_File_System)) did not support separate catalog directories. When catalog file support was included in Mac OS, with the HFS filesystem, the resource fork was retained. macOS does retain the classic Resource Manager [API](https://en.wikipedia.org/wiki/Application_programming_interface) as part of its [Carbon](https://en.wikipedia.org/wiki/Carbon_(API)) libraries for backward compatibility. However, the resources themselves can now be stored in separate data files within the file system — the Resource Manager now hides this implementation change from the client code.

See also[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=10)]

* [Fork (file system)](https://en.wikipedia.org/wiki/Fork_(file_system))

References[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=11)]

* 1. [**^**](https://en.wikipedia.org/wiki/Resource_fork#cite_ref-1) [*"Mac OS X Resource Forks"*](http://jonsview.com/mac-os-x-resource-forks)*. Retrieved 2012-10-22.*
  2. [**^**](https://en.wikipedia.org/wiki/Resource_fork#cite_ref-2) [*"Resource Manager Reference"*](https://developer.apple.com/library/mac/#documentation/Carbon/Reference/Resource_Manager/Reference/reference.html)*. Retrieved 2012-10-22.*
  3. [**^**](https://en.wikipedia.org/wiki/Resource_fork#cite_ref-3) [*"Using Pathnames"*](https://web.archive.org/web/20021218054520/http:/developer.apple.com/techpubs/macosx/Essentials/Performance/FilesNetworksThreads/Using_Pathnames.html)*. developer.apple.com. 2002-12-18. Archived from the original on 2002-12-18. Retrieved 2002-12-18.*
  4. [**^**](https://en.wikipedia.org/wiki/Resource_fork#cite_ref-4) [*"OS X v10.5, v10.6: About named streams on SMB-mounted NAS, OS X, and Windows servers"*](http://support.apple.com/kb/HT4017)*. Retrieved 2010-04-19.*
  5. [**^**](https://en.wikipedia.org/wiki/Resource_fork#cite_ref-5) [*"The Early History of Smalltalk"*](http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html)*. Retrieved 2008-07-24.*

External links[[edit](https://en.wikipedia.org/w/index.php?title=Resource_fork&action=edit&section=12)]

* [Description of the Resource File Format](https://developer.apple.com/legacy/mac/library/documentation/mac/pdf/MoreMacintoshToolbox.pdf#page=151)
* [Apple Developer Resource Library: Resource Manager Reference](https://developer.apple.com/documentation/Carbon/Reference/Resource_Manager/index.html)
* [Apple Developer Resource Library: Resource Management, Bundles](https://developer.apple.com/documentation/CoreFoundation/Conceptual/CFBundles/index.html)
* [The Grand Unified Model](https://web.archive.org/web/20061023221041/http:/www.folklore.org/StoryView.py?project=Macintosh&story=The_Grand_Unified_Model.txt) – History of the resource fork, from folklore.org
* [Rezycle](http://evolutioninteractive.com/rezycle/) – Resource extraction tool
* [Mac OS X Services](http://www.gingerbeardman.com/services/) – A Mac OS X Service to delete the resource fork of a file through context menu
* [What's up with Mac OS X Resource forks, Extended Attributes, NTFS Streams and Dot-Underscore files?](http://support.grouplogic.com/?p=1496)
* [When I save a file using SMB protocol, what information is saved in the “dot-underscore” (.\_) files? How is this information stored on an NTFS file system?](http://support.grouplogic.com/?p=1678)