Demartek

File System Comparison Research Report

July 2004

While most computer operating systems and file systems provide basic file management attributes that have proven adequate in the past, emerging requirements are forcing change. This paper compares the existing file management attributes of several operating systems and provides a foundation for possible new attributes to consider in light of regulatory compliance, improved security and such topics as information life-cycle management.

The operating systems and file systems discussed in this paper include

- FAT
- NTFS
- OpenVMS
- OS/400
- UNIX & Linux
- z/OS
- z/VM

© 2004 Demartek ◆ www.demartek.com ◆ Email: info@demartek.com

Introduction

As the requirements for security, management and regulatory compliance continue to grow, computer operating systems in general, and file systems in particular, need to adapt to changing requirements. We know some of the newer requirements today, and we can reasonably expect additional new requirements in the future.

Most file systems store similar basic information in their metadata. The file attributes available in most file systems have been adequate in the past, but new requirements are forcing operating system and file system providers to provide new metadata and ways to extend metadata to meet future requirements. File systems are beginning to emerge that can accommodate some of the newer requirements.

Most data centers use a variety of computing platforms and operating systems. As general management, regulatory compliance and security concerns continue to increase, it is reasonable to consider several topics across various operating systems and file systems in use today. This paper will compare the following topics across multiple operating systems and file systems:

- Security permissions
- Group membership
- Naming and character set conventions
- Auditing features

This paper concludes by discussing possible future changes to adapt to the newer needs of our time.

Background

As background, we provide a brief historical summary of the various operating systems and file systems described in this paper.

FAT – Introduced by Microsoft in 1981, it was the basic file system for MS-DOS and the early versions of Windows. FAT stands for File Allocation Table.

NTFS – Introduced by Microsoft in 1993 with the release of Windows NT, it has been the preferred file system used by Windows NT, Windows 2000, Windows XP and Windows 2003 Server. NTFS stands for NT (as in Windows NT) File System.

OpenVMS – Originally released by Digital Equipment Corp (DEC) in 1977 as VMS for the VAX platform, it was renamed to OpenVMS in the mid-1990s when it was ported to the Alpha platform. Compaq acquired Digital in 1998 and HP acquired Compaq in 2001.

OS/400 – Originally released by IBM in 1988 for the AS/400 platform, and now provided for the iSeries platform. The AS/400 was the follow-on platform for the System/36 and System/38 machines.

UNIX & Linux – Refers to various versions of UNIX and Linux. Bell Labs (AT&T) first introduced UNIX in 1969. AT&T licensed UNIX to computer science departments at several universities in the 1970s. These universities along with some private companies enhanced it over the years, each providing their own variant using various processor platforms. Linus Torvalds introduced Linux in 1991 as a UNIX-like "open source" operating system. Some variants of Linux are available for download free of charge and others are commercial products for sale.

z/OS – The IBM operating system for its zSeries mainframe platforms. It can trace its roots back to 1964 with the release of the System/360 platform. Previous names for this operating system include OS/360, OS/370, MVS and OS/390.

z/VM – The current name of the IBM VM time-sharing operating system for its zSeries mainframe, originally introduced in the 1970s running on the System/370 platform. It is a virtual operating system in that it can host other operating systems such as Linux, TPF, VSE, z/OS or VM "guests".

Security permissions

File systems generally provide for specific actions to be performed on individual files or data sets or on directories or groups of files. These actions are then associated with individual users or groups of users that can perform these functions. Not all operating systems specifically identify all these actions individually, but group some together. These actions include

- **No access:** There is no access to the resource, not even to see its name or the fact that it exists.
- **List:** Users can see the name of the resource, but no other action is permitted.
- **Read:** Users can see the name of the resource and can read the contents of the resource (usually includes copy and print).
- **Append:** Users can see the name, read the resource and append new data to the resource.
- **Change/Update:** Users can see the name, read, append and change (read and write).
- **Delete:** Users can delete the resource.
- **Execute:** Users can execute the resource as a program.
- **Permissions:** Users may grant other users the authority to alter the actions allowed.
- **Take Ownership:** Users may grant other users the authority to become the owner of the resource.
- **Custom/special:** Users can have special combinations of the above actions.

FAT

FAT was originally designed for single-user personal computers, and as a result, has very little file security. The only adjustable security permission for files is the "read-only" flag.

NTFS

NTFS has a very rich set of file security permissions that can be applied to files and directories (folders). The specific permissions include

- **List:** See the name of the resource, but no other action is permitted.
- **Read:** Read the contents of a file.
- **Read & Add:** Read the file and append new data to the file but cannot alter existing contents.
- **Change:** Read, append and write the file, but cannot delete it.
- **Delete:** Delete the file.
- **Execute:** Execute the resource as a program (usually includes Read access).
- **Permissions:** Users may grant other users the authority to alter the actions allowed.
- **Ownership:** Users may grant other users the authority to become the owner of the resource.
- **Full Control:** Includes all the above permissions.
- **Custom:** Users can have special combinations of the above actions.

OpenVMS

OpenVMS has four basic file security permissions: Read, Write, Execute and Delete.

- **Read:** Read, copy and print files and read or list directory entries. It implies the Execute permission.
- **Write:** Write or change the contents of a file, but not delete it. It also includes the permission to insert or delete an entry in the file catalog (directory).
- **Execute:** Permission to execute a file containing an executable program or command procedure. For directories, it includes the right to look up files whose names the user specifically knows.
- **Delete:** Permission to delete a file. Also requires Write access to the directory containing the file.

OS/400

OS/400 has five basic file security permissions: Read, Add, Update, Delete and Execute. Because of the unique architecture of OS/400, these security permissions can be set at the file, record or data field level.

- **Read:** Display or read the contents of an object, such as records in a file.
- **Add:** Add files or add records to a file.

- **Update:** Change records in a file.
- **Delete:** Delete files or delete records from a file.
- **Execute:** Run a program or command procedure.

OS/400 also includes an Alter permission to change the security permissions for an object.

UNIX & Linux

Most variants of UNIX and Linux have a similar set of file security permissions: Read, Write and Execute.

- **Read:** Read the contents of a file or list the files in a directory.
- **Write:** Write (including add and change/update) and delete a file.
- **Execute:** Run a program or command procedure.

Most variants of UNIX and Linux also allow manipulation of the "sticky bit" that provides for special file protections such as file deletion only by the owner of the file or directory (or "root" user), even if the Write permission is specified for other users.

z/OS

z/OS has five basic file (data set) security permissions: Read, Update, Control, Alter and Execute.

- **Read:** Read the contents of the data set.
- **Update:** Users may read and write the data set.
- **Control:** For VSAM data sets, record-level control within the data set. For non-VSAM data sets, "Control" means the same as Update.
- **Alter:** Read, write, delete, rename and move the data set. In some cases, "Alter" can provide for alteration of the security profile for that object.
- **Execute:** Load and execute a program or command procedure. Does not include read access to the program.

z/VM

z/VM has five basic file (data set) security permissions: Read, Update, Control, Alter and Execute.

- **Read:** Read the contents of the data set.
- **Update:** Users may read and write the data set.
- **Control:** For VSAM data sets, record-level control within the data set. For non-VSAM data sets, "Control" means the same as Update.
- **Alter:** Read, write, delete, rename and move the data set. In some cases, "Alter" can provide for alteration of the security profile for that object.
- **Execute:** Load and execute a program or command procedure. Does not include read access to the program.

Table 1 – Security permission comparison

	FAT	NTFS	OpenVMS	OS/400	UNIX & Linux	z/OS	z/VM
No access		✓	✓	✓	✓	✓	✓
List		✓	✓				
Read		✓	✓	✓	✓	✓	✓
Append		✓		✓			
Change/Update	✓	✓	✓	✓	✓	✓	✓
Delete		✓	✓	✓		✓	✓
Execute		✓	✓	✓	✓	✓	✓
Permissions		✓	✓	✓		✓	✓
Take Ownership		✓			*		
Custom/Special		✓					

* Some variants of UNIX & Linux have limited versions of "take ownership"

Group membership

Most operating systems provide for individual users and groups of users to be associated with particular resources. For simpler administration, security parameters can be assigned to a group, and then the parameters apply to all members of that group. Some operating systems such as OpenVMS and UNIX and Linux have not traditionally allowed users to be members of more than one group.

Access Control Lists (ACLs) provide flexibility in securing resources for file system objects such as files and directories. ACLs allow users to be members of more than one group. A file may have its own ACL or may share an ACL with other files. OpenVMS and some variants of UNIX have added support for ACLs to provide security that is more flexible.

Table 2 – Group membership and ACL comparison

	FAT	NTFS	OpenVMS	OS/400	UNIX & Linux	z/OS	z/VM
Member of more than one group	No	Yes	No	Yes	No	Yes	Yes
ACLs	No	Yes	Yes	Yes	Yes*	Yes	Yes

* Some, but not all variants

Naming and character set conventions

The information in this report applies to disk file naming only. Some systems follow different naming rules for tape files.

File naming rules among the operating systems and file systems vary widely. For example, several use a directory and sub-directory hierarchy. Most of the systems that support the directory hierarchy allow sub-directories to be nested many levels deep. The complete directory and sub-directory name is usually called the "path" or "pathname". This makes for good flexibility for the organization of files, but can result in rather long path-file names. Others have the concept of "members" of a file, which is somewhat like having a directory hierarchy exactly one level deep.

Some systems are case-sensitive with respect to directory or file names, some are "case-aware", and some are case insensitive (use only upper or lower case). Each has advantages and disadvantages. For those that are case sensitive, it is possible to have two or more very similarly named files. The names "REPORT", "report" and "RePoRt" refer to three different files. This may cause confusion or irritation for some users, but allows for some creative file naming. For case-aware systems, the above three names refer to the same file, but the system will preserve the case of the name that was used when it was created. For systems that are case insensitive that use upper case, "REPORT" is a valid file name, but "report" and "RePoRt" are not. This is simple, but not as flexible or natural for some users.

The use of special characters (those that are not letters or numbers) in file or directory names differs between systems. Some systems allow imbedded spaces in the file names. This is more natural for many end users, but not all file systems or applications can handle this. The use of the period "." character has significance in many systems. It segments file names into logical parts, indicates version numbers, or signifies a file type, such as ".txt". Other special characters may have special meaning in some situations, and therefore cannot be used as part of a file name.

The character set used also varies. Many systems use the ASCII character set. ASCII is a character set that consists of the letters used primarily by the western European languages, along with numbers, some accented letters and other special characters. Some use the EBCDIC character set, somewhat similar to ASCII, but based on the old punch cards from the early days of data processing. The characters are arranged in a different order than the ASCII characters. A more recent trend is to use Unicode when naming files. This double-byte character set has almost all characters from all languages around the world defined to it, and is a superset of ASCII.

Table 3 – File naming comparison

	FAT	NTFS	OpenVMS	OS/400	UNIX & Linux	z/OS	z/VM
Maximum file name length	11 ("8.3")[1]	256	236[2]	Varies[3]	256	44	16[4]
Member name maximum length	-	-	-	-	-	8	-
Character set	ASCII	ASCII, Unicode	ASCII, Unicode[5]	ASCII, EBCDIC, Unicode	ASCII, Unicode	EBCDIC	EBCDIC
Case-sensitive	No	Case-aware	Case-aware	Varies	Yes	No	Varies[6]
Directory hierarchy	Yes	Yes	Yes	Yes	Yes	No	No[7]
Directory separator	"\"	"\"	"."	Varies	"/"	-	-
Drive letter	Yes	Yes	No	Varies	No	No	Minidisk[8]

[1] Early versions of MS-DOS supported only the "8.3" file names and used upper case only. Long file names with mixed case were added beginning with Windows 95.
[2] OpenVMS supports file names of 236 ASCII characters or 118 Unicode characters on ODS-5 volumes. The default ODS-2 volume supports the "39.39" file naming rules.
[3] The IBM i5/OS V5R3 (current version of OS/400) natively supports the file systems used by AIX, Linux and Windows 2000, and their respective attributes.
[4] Using the CMS file system, z/VM files consist of an 8-character file "name" plus an 8-character file "type".
[5] OpenVMS supports Unicode file names on ODS-5 volumes only.
[6] The z/VM default for file names is upper case only, but this can be changed to allow mixed case using the SET INPUT command.
[7] z/VM OpenExtensions includes support for a byte file system (BFS) that is organized in a hierarchy similar to UNIX file systems.
[8] z/VM stores files on a logical disk volume known as a "minidisk". Minidisks are assigned letters, starting with the letter "A".

Auditing features

Most operating systems include tools, usually associated with security functions, for auditing file system activity. Some of these tools are native to the operating system and file system, and some are optional. These provide basic information regarding the access of resources, including the user, time of access and other pertinent information. For some systems, the auditing tools are integrated with the security subsystem and are the same tools used to set security policies. Other operating systems have separate auditing tools. Some file systems come with file system-specific auditing tools. The capabilities and specific methods for using these tools vary widely across the various operating systems and file systems.

Auditing functions typically include the ability to monitor users, resources and other objects, and can log successes, failures or both. Those wishing to audit activities may be interested in simply debugging a wayward script, setting basic security polices, or attempting to catch unauthorized or criminal activity. Auditors can be system administrators, members of a separate auditing department or members of the legal staff. Audit records typically consist of the date, time, user and resource affected. A few auditing scenarios include

- Occurrences of writes or changes to a particular file or group of files
- Instances of file deletion across all systems
- Changes to file security permissions
- Occurrences of new file creation in a specific department
- Accesses made to the audit logs

Effective auditing requires planning. As with any security function, there is a balance between the level of auditing detail possible and the system and staffing resources needed to generate, store and analyze the audit records.

Future considerations

In order to comply with various regulations in the USA and elsewhere, operating system and file system vendors are beginning to consider additions to the functions discussed in this report. The file system is an excellent place to store this new metadata, since much of the framework for managing the metadata is already there.

Some of the regulations have to do with data and records retention. Some data has a specific time for which it must be available and then must be destroyed afterwards. Destruction of data must include all copies of the data on all media, including disk, tape, optical and other forms.

Other regulations require that a "chain of custody" be maintained so that at any time (including by court order) an auditor can tell which users created, read, copied, updated or deleted any data.

Information lifecycle management systems are beginning to appear that address some of these regulations as well as other management issues. Some data needs to be classified based on the creator of the data, the department or job function of the data creator, life expectancy of the data, storage class required for the data, sensitivity or security required for the data, relation to other similar data, etc. These information lifecycle systems need a place to store much of the metadata required to perform these functions. The file system metadata is a good place for some of this data.

Several questions come to mind in reviewing these new requirements. For example, how will the automatic destruction of all copies of a particular set of data occur at the appropriate time? One method might be to encrypt all data that fits the particular category, both for disk and removable media copies. When the expiration date arrives, the encryption keys could be deleted, making all copies of the encrypted data unreadable. This avoids the work of finding all copies of data on removable media and erasing or physically destroying that media. This suggests that additional metadata involving encryption functions be available either at the application or file system level.

Another function known as the "secure erase" function may be added to file systems and backup applications in order to help improve security. This secure erase function could exploit any hardware erase functions that might be available, and use software erase functions where no hardware functions are available.

If some data must be more tightly controlled, more granular security permissions must be implemented. For example, the authority to delete a file should be separate from the authority to write or change a file. The ability to append new data to a file without altering existing data may become necessary to satisfy particular regulations or situations. We might expect to see some of the UNIX or Linux file systems adapt a more granular security

approach. New file systems or a layer above the existing file systems may provide the additional needed function. Standard features of newer file systems may include more rigorous and consistent auditing tools.

Ways to extend file system metadata should also be implemented now in order to address new requirements that will undoubtedly arise. This file system extension should be implemented so that the file system metadata can be updated in place, without massive data movement.

Reproduction guidelines: you may make copies of this document unless otherwise noted. If you quote or reference this document, you must appropriately attribute the contents and authorship to Demartek, and include the Demartek web site www.demartek.com in the attribution.

Opinions presented in this document reflect judgment at the time of publication and are subject to change. While every precaution has been taken in the preparation of this document, Demartek assumes no responsibility for errors, omissions, or damages resulting from the use of the information herein.

Products, brand names or corporate names may be trade names, service marks, trademarks or registered trademarks of their respective companies.