EXTENSIBLE FILE SYSTEM

CROSS-REFERENCE TO RELATED APPLICATION

This application claims the benefit of U.S. Provisional Application No. 60/637,407, entitled FILE SYSTEM FORMAT FOR PORTABLE MEDIA, and filed on December 17, 2004. U.S. Provisional Application No. 60/637,407 is incorporated by reference herein.

BACKGROUND

Generally described, there are a number of portable computing devices, such as digital still cameras, digital video cameras, media players, mobile phones, mobile computing devices, personal digital assistants, and the like that maintain data on a storage media, such as a portable storage media. The continued development of more complex portable computing devices and larger storage capacity portable storage media places a greater demand for flexibility on the file system format used on the storage media. Current file system format approaches can become deficient in that they may provide adequate flexibility for increasing storage size capacities and/or storage media applications.

SUMMARY

An extensible file system format for portable storage media is provided. The extensible file system format includes the specification of primary and secondary directory entry types that may be custom defined. The primary and secondary directory entry types can be further classified as critical and benign directory entries.

In accordance with an aspect of the present invention, a computer-readable medium having computer-executable components for storing data is provided. The computer-readable components can include a boot parameters component for specifying boot parameters for a file system. The computer-readable components also include a file allocation table component for defining a file allocation table associated with the file system. Additionally, the computer-readable components include a primary directory entry component for specifying data in a root directory of the file system. Still further, the computer-readable components include at least one secondary entry component corresponding to the primary directory entry component. The secondary entry component defines defining meta data associated with the primary directory entry

component. The primary and secondary directory entry components can be further classified as critical or benign.

In accordance with another aspect of the present invention, a computer-readable medium having computer-executable components for storing data is provided. The computer-readable components include a boot parameters component for specifying boot parameters for a file system. The computer-readable components also include a file allocation table component for defining a file allocation table associated with the file system. Still further, the computer-readable components include a root directory component for specifying data in a root directory of the file system. Additionally, the computer-readable components include at least extensible one meta data component corresponding to the root directory entry component. The meta data component defines meta data associated with the root directory component.

In an illustrative embodiment, a file system will not mount a volume for a critical primary or root directory entry that is not recognized. The file system can ignore benign primary directory entries, critical secondary directory entries and benign secondary directory entries that are not recognized.

This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This summary is not intended to identify key features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIGURES 1A-1C are block diagrams illustrative of an illustrative environment including a portable computing device and a storage device implementing the extensible file system format in accordance with an aspect of the present invention;

FIGURE 2 is a block diagram illustrative of various volume layout components corresponding to an extensible file system format in accordance with an aspect of the present invention;

FIGURE 3 is a block diagram illustrative of an extensible file system directory structures including primary and secondary directory entry structures in accordance with an aspect of the present invention;

FIGURE 4 is a block diagram illustrative of data components for implementing a boot process block in an extensible file system format in accordance with an aspect of the present invention;

FIGURE 5 is a block diagram illustrative of data components for implementing directory entries in an extensible file system format in accordance with an aspect of the present invention

FIGURE 6 is a block diagram illustrative of data components for implementing a file name and extensions in an extensible file system format in accordance with an aspect of the present invention;

FIGURE 7 is a block diagram illustrative of data components for implementing a volume identifier in an extensible file system format in accordance with an aspect of the present invention;

FIGURE 8 is a block diagram illustrative of data components for implementing an extensible directory entry in an extensible file system format in accordance with an aspect of the present invention;

FIGURE 9 is a block diagram illustrative of data components for implementing an extensible directory entry in an extensible file system format in accordance with an aspect of the present invention;

FIGURE 10 is a block diagram illustrative of data components for implementing an access control list in an extensible file system format in accordance with an aspect of the present invention; and

FIGURE 11 is a flow diagram illustrative of a file name creation routine for an extensible file system format in accordance with an aspect of the present invention.

DETAILED DESCRIPTION

Generally described, the present invention relates to an extensible file system format and various processes associated with the extensible file system format. In an illustrative embodiment, the extensible file system format corresponds to an extensible file system format for portable storage media and various processes associated with the

extensible file system format on the portable storage media. Although the present invention will be described with regard to a portable storage media file system format, one skilled in the relevant art will appreciate that the disclosed embodiments are illustrative in nature and should not be construed as limiting. Additionally, one skilled in the relevant art will appreciate that the data structures and data layouts used in the illustrative examples may require additional information related to performance, security, and the like.

FIGURES 1A-1C are block diagrams illustrative of various operating environments 100 for the extensible file system format of the present invention. With reference to FIGURE 1A, in an illustrative embodiment, the extensible file system format is utilized to store data from a computing device, such as a mobile computing device 102, and a storage media, such as a portable storage media 104. In an illustrative embodiment, the mobile computing device 102 can correspond to any one of a variety of computing devices, including but not limited to, portable computing devices, mobile telephones, personal digital assistants, music players, media players. The portable storage media can also include, but is not limited to, hard drives, flash media, micro-drives and other storage media. In an illustrative embodiment, the extensible file system on the portable storage media 104 does not have to include any type of executable or readable software components, such as an operating environment, utilized by the mobile computing device 102. Alternatively, the extensible file system on the portable storage media 104 may include executable or readable software components used by the mobile device 102.

In an illustrative embodiment, the mobile computing device 102 may be in communication with other computing devices for collecting/exchanging data to be stored on the portable storage media 104. With reference to FIGURE 1B, the mobile computing device 102 may be in direct communication with another computing device 106 and storage media 108. In an illustrative embodiment, the direct communication can correspond to various wired and wireless communication methods. In an illustrative embodiment, the other storage media 108 is not required to be formatted in accordance with the extensible file system format of the present invention. With reference to FIGURE 1C, in a similar manner, the mobile computing device 102 may also be in communication with another computing device 110 and storage media 112, via a network

connection. In an illustrative embodiment, the network connection can correspond to local area network (LAN) and wide are network (WAN) connections.

With reference now to FIGURE 2, an illustrative embodiment volume layout 200 for an extensible file system format will be described. The volume layout 200 includes a boot parameters component 202 that include various information related to a description of the file system parameters of the partition. In an illustrative embodiment, the boot parameters component 202 can include code for bootstrapping from a defined partition, fundamental file system parameters for the defined partition, and various error checking information. A data structure for defining at least a portion of the boot parameters will be described below with regard to FIGURE 4.

The volume layout 200 also includes an extensible parameters component, designated as OEM parameters 204, that define various additional data structures used in conjunction with the file system. In an illustrative embodiment, an original equipment manufacture (OEM) may specify various extensible data structures, such as performance parameters for a storage medium, that can be defined at time of manufacture. The volume layout 200 can further include a file allocation table component 206 that defines file and directory allocations. In an illustrative embodiment, each entry in the file allocation table component 206 corresponds to a 32-bit entry that represents an allocated cluster, an unallocated cluster or an unusable cluster. The volume layout 200 can still further include series of file data components 208A-208X that correspond to the data stored according to the file system format. Various data structures for defining a portion of the file data components 208A-208X will be defined with regard to FIGURES 3-10.

Turning now to FIGURE 3, in one aspect, the file data components 208 may include one or more directory entries according to a directory structure 300. In an illustrative embodiment, directory structure 300 is organized into primary directory entries 302 and secondary directory entries 304. Each directory entry in the primary and secondary entries is typed. For example, in an illustrative embodiment, type values for the primary and secondary directory entries can correspond to a range of 1-255. Primary directory entries 302 correspond to the entries in the root directory of the file system. Secondary directory entries 304 follow a primary directory entry and are associated with

the primary directory entry. Secondary directory entries extend the metadata associated with the correlated primary directory entry.

With continued reference to FIGURE 3, in an illustrative embodiment, the primary directory entries 302 can be further classified as critical primary directory entries 306 and benign primary directory entries 308. Critical primary directory entries 306 define potentially different formats for each directory entry. In an illustrative embodiment, an operating environment will not mount a volume corresponding to the extensible file system format with an unknown critical primary directory entry, as will be described below. Examples of known primary directory entries 306 can include allocation bitmaps, up-case tables, volume labels, encryption keys, and normal directory entries. Benign primary directory entries 308 also define potential different formats for each directory entry, but can be ignored by the file system if a particular benign primary directory entry is not understood. Benign primary directory entries 308 can be associated with another cluster chain the volume. Additionally, benign primary directory entries 308 can also be associated a number of secondary directory entries 304.

In a manner similar to primary directory entries 302, secondary directory entries 304 may also be further classified as critical secondary directory entries 310 and benign secondary directory entries 312. As described above, the critical secondary directory entries 310 and benign secondary directory entries 312 are associated with a benign primary directory entry and extend the metadata associated with the primary directory entry. Both the critical secondary directory entries 310 and the benign secondary directory entries 312 can be associated with another cluster chain the volume.

To mount a corresponding to the extensible file system format, the file system implements a mount volume procedure. In an illustrative embodiment, the mount volume procedure attempts to a look at a version number for the volume. If the version number is not understood (e.g., the version number is higher), the volume will not be mounted. During a normal directory enumeration, any critical primary directory entries not known by the file system will prevent the volume from being mounted. Thereafter, various user-initiated processes, such as a file open, will cause the file system to enumerate the secondary directory entries. If the critical secondary directory entries 310 are not known by a file system, the entire directory entry will be skipped. Additionally, if benign

secondary directory entries 312 are not known by the file system, the particular unknown benign secondary directory entry will be ignored.

With reference now to FIGURE 4, a block diagram illustrative of data components 400 for implementing a boot process block in the boot parameters component 202 (FIGURE 2) will be described. The data components 400 include an OEM name component 402 for specifying a name for the file system format of the storage media. The data components 400 also include a data size descriptor component 404 for specifying various characteristics of the data stored in the file system. For example, the data size descriptor component 404 can specify a count of bytes per sector, a number of sectors per allocation unit, a FAT table offset, and a count of sectors for all data structures. The data components include an active FAT flags component 406 for specifying a number of active FATs on the file system. In an illustrative embodiment, a file system may support multiple FATs for utilization with some operating system environments. The data components 400 can further include a volume identification component 408 for identifying a volume serial number and/or version number. Still further, the data components 400 can include a file system type for specifying the file system format for the file system. One skilled in the relevant art will appreciate that the data components 400 can include a number of additional/alternative rows for implementing the above-identified components 402-410 and additional components.

Turning now to FIGURE 5, a block diagram illustrative of data components 500 for implementing directory entries in an extensible file system format will be described. Turning now to FIGURE 6, a block diagram data components 500 for implementing a file name and extensions will be described. The data components 500 include an in use component 502 for specifying whether the particular directory entry is in use. In an illustrative embodiment, the high bit of the data components will be set to "1" if the directory entry is in use. The data components 500 further include a type designation component 504 for specifying that the directory entry is associated with a normal directory entry. The data components 500 further include a secondary directory entries component 504 for specifying a number of secondary entries associated with the normal directory entry. The data components 500 also include a file attributes component 508 for specifying various file system attributes for the directory entry. Still further, the data

components 500 include a time component 510 for specifying various time information such as a creation timestamp, modification time stamp and other time information. Additionally, the data components 500 further include a time zone component 512 for specifying a time zone for the last created time stamp. One skilled in the relevant art will appreciate that the data components 500 can include a number of additional/alternative rows for implementing the above-identified components 502-512 and additional components.

Turning now to FIGURE 6, a block diagram data components 600 for implementing a file name and extensions will be described. The data components 600 include an in use component 602 for specifying whether the particular directory entry is in use. In an illustrative embodiment, the high bit of the data components will be set to "1" if the directory entry is in use. The data components 600 further include a type designation component 604 for specifying that the directory entry is associated with a file system name. The data components further include a file name length component 606 and a file name has component 608. The utilization of the file name hash component 608 will be described below. The data components 600 also include a file name component 610 for specifying the file name. One skilled in the relevant art will appreciate that the data components 600 can include a number of additional/alternative rows for implementing the above-identified components 602-610 and additional components. Additionally, file name directory entries may be extended by secondary directory entries.

Turning now to FIGURE 7, a block diagram illustrative of data components 700 for implementing a volume identifier in an extensible file system format is provided. The data components 700 include an in use component 702 for specifying whether the particular directory entry is in use. In an illustrative embodiment, the high bit of the data components will be set to "1" if the directory entry is in use. The data components 700 further include a type designation component 704 for specifying that the directory entry is associated with a volume identifier. The data components 700 further include a secondary directory entries component 706 for specifying a number of secondary entries associated with the volume identifier. The data components 700 also include a volume identifier 708, such as a global unique identifier. One skilled in the relevant art will

appreciate that the data components 700 can include a number of additional/alternative rows for implementing the above-identified components 702-708 and additional components. Additionally, in an illustrative embodiment, the data components 700 correspond to a benign directory entry that can be ignored by a file system that does not support volume identifiers.

With reference now to FIGURES 8 and 9, in an illustrative embodiment, parties, such as an OEM, may be able to define specific benign primary directory entry types 308 and benign secondary directory entry types 312. As discussed above, in the event the file system would not recognize or understand either the specific benign primary directory entry types 308 or benign secondary directory entry types 312, the file system could ignore the defined directory entry types.

With reference to FIGURE 8, a block diagram illustrative of data components 800 for implementing an extensible benign primary directory entry 308 in an extensible file system format will be described. The data components 800 include an in use component 802 for specifying whether the particular directory entry is in use. In an illustrative embodiment, the high bit of the data components will be set to "1" if the directory entry is in use. The data components 800 further include a type designation component 804 for specifying that the directory entry is a benign primary directory entry. The data components 800 further include a secondary directory entries component 806 for specifying a number of secondary entries associated with the volume identifier. The data components 800 also include a volume identifier 808, such as a global unique identifier. The data components 800 can further include additional information 810, such as verification information and a starting cluster. One skilled in the relevant art will appreciate that the data components 800 can include a number of additional/alternative rows for implementing the above-identified components 802-506 and additional components.

With reference to FIGURE 9, a block diagram illustrative of data components 900 for implementing a benign secondary directory entry in an extensible file system format will be described. The data components 900 include an in use component 902 for specifying whether the particular directory entry is in use. In an illustrative embodiment, the high bit of the data components will be set to "1" if the directory entry is in use. The

data components 900 further include a type designation component 904 for specifying that the directory entry is a benign primary directory entry. The data components 900 further include a secondary directory entries component 906 for specifying a number of secondary entries associated with the volume identifier. The data components 900 also include a volume identifier 908, such as a global unique identifier. The data components 900 can further include additional information 910, such as verification information and a starting cluster. One skilled in the relevant art will appreciate that the data components 900 can include a number of additional/alternative rows for implementing the above-identified components 902-906 and additional components.

In an illustrative embodiment, a benign primary directory entry and/or secondary directory entries may be associated with access control list (ACL) information. FIGURE 10 is a block diagram illustrative of data components 1000 for implementing an access control list in an extensible file system format. The data components 1000 include an in use component 1002 for specifying whether the particular directory entry is in use. In an illustrative embodiment, the high bit of the data components will be set to "1" if the directory entry is in use. The data components 1000 further include a type designation component 1004 for specifying that the directory entry is an ACL directory entry. The data components 1000 further include a number of ACL fields 1006, such as ACL flags, pointers to ACL databases, and the like. One skilled in the relevant art will appreciate that the data components 1000 can include a number of additional/alternative rows for implementing the above-identified components 1002-1006 and additional components.

With reference now to FIGURE 11, a file name creation routine 1100 for an extensible file system format will be described. At block 1102, a file system obtains a request to create a directory entry with a specific file name. In an illustrative embodiment, the specific file name can correspond to a naming convention, such as a digital camera picture naming convention. At block 1104, the file system generates a target name hash. At block 1106, an iterative loop is begun by examining the next directory entry hash value. An illustrative directory entry type for storing directory entry hash values is described above with regard to data components 600 (FIGURE 6).

At decision block 1108, a test is conducted to determine whether the target hash value matches the current directory entry hash value. If they do not match, the

routine 1100 returns to block 1106 (until all the directory entries have been examined. If the hash values match at decision block 1108, at block 1110, the file system obtains the full file name for the potentially matching directory entry. An illustrative directory entry type for storing directory entry full file names is described above with regard to data components 600 (FIGURE 6). At decision block 1112, a test is conducted to determine whether the target file name matches the full file name of the potentially matching directory entry. If so, the routine 1100 terminates by reporting a conflict and the file system will be required to select a new file name. If the full file does not match, the routine 1100 will return to block 1106 to continue checking hash values for all the directory entries in the file system.

In accordance with an aspect of the present invention, various additional functionality may be added through the specification of specific directory types. For example, name streams may be supported by specifying a name stream directory entry. Additionally, on-disk encryption may also be supported through the utilization of specific encryption algorithms and key exchanges. Still further, time zone conversions may be associated with directory entries to automatically convert a current time zone with a time zone with the directory entry was made.

While illustrative embodiments have been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A computer-readable medium having computer-executable components for storing data, the computer-readable components comprising:

a boot parameters component for specifying boot parameters for a file system;

a file allocation table component for defining a file allocation table associated with the file system;

a primary directory entry component for specifying data in a root directory of the file system; and

at least one secondary entry component corresponding to the primary directory entry component and defining meta data associated with the primary directory entry component.

2. The computer-readable components recited in Claim 1, wherein the primary directory entry component is a critical primary directory entry component that must be understood by the file system.

3. The computer-readable components recited in Claim 1, wherein the primary directory entry component is a benign primary directory entry component that can be ignored if unknown to the file system.

4. The computer-readable components recited in Claim 1, wherein the secondary directory entry component is a critical secondary directory entry that will cause the primary directory component and the secondary directory entry component to be ignored if unknown to the file system.

5. The computer-readable components recited in Claim 1, wherein the secondary directory entry component is a benign secondary directory entry that can be ignored if unknown to the file system.

6. The computer-readable components recited in Claim 1 further comprising two secondary entry components corresponding to the primary directory entry component and defining meta data associated with the primary directory entry

7. The computer-readable components recited in Claim 1, wherein the primary directory entry component corresponds to an allocation bitmap defining cluster availability within a storage media.

8. The computer-readable components recited in Claim 1, wherein the primary directory entry component corresponds to a volume identifier.

9. The computer-readable components recited in Claim 1, wherein the primary directory entry component corresponds to a file name identifier.

10. The computer-readable components as recited in Claim 9, wherein the file name identifier includes a full file name and a file name hash.

11. The computer-readable components recited in Claim 1, wherein the at least one secondary directory entry corresponds to an extensible secondary directory entry.

12. The computer-readable components recited in Claim 1 further comprising a manufacturer data component for specifying manufacturer data structures.

13. A computer-readable medium having computer-executable components for storing data, the computer-readable components comprising:

a boot parameters component for specifying boot parameters for a file system;

a file allocation table component for defining a file allocation table associated with the file system;

means for specifying data in a root directory of the file system and meta data associated with data in the root directory.

14. The computer-readable components recited in Claim 13, wherein the means for specifying data in a root directory of the file system and meta data associated

with data in the root directory corresponds to a primary directory entry component for specifying data in a root directory of the file system.

15. The computer-readable components recited in Claim 14, wherein the primary directory component can correspond to a critical primary directory entry.

16. The computer-readable components recited in Claim 14, wherein the primary directory component can correspond to a benign primary directory entry.

17. The computer-readable components recited in Claim 13, wherein the means for specifying data in a root directory of the file system and meta data associated with data in the root directory includes at least one secondary entry component corresponding to the data in the root directory and defining meta data associated with the primary directory entry.

18. A computer-readable medium having computer-executable components for storing data, the computer-readable components comprising:

a boot parameters component for specifying boot parameters for a file system;

a file allocation table component for defining a file allocation table associated with the file system;

a root directory component for specifying data in a root directory of the file system; and

at least extensible one meta data component corresponding to the root directory entry component and defining meta data associated with the root directory component.

19. The computer-readable components recited in Claim 18, wherein the root directory component is a critical root directory component that must be understood by the file system.

20. The computer-readable components recited in Claim 18, wherein the root directory entry component is a benign root directory entry component that can be ignored if unknown to the file system.