

METHOD AND APPARATUS FOR MOUNTING FILE SYSTEM, DEVICE AND STORAGE MEDIUM

TECHNICAL FIELD

5 **[0001]** The present disclosure relates to the field of computer technologies, and in particular to the field of network file systems. Specifically, the present disclosure relates to a method and apparatus for mounting a file system, a device, and a storage medium.

BACKGROUND

10 **[0002]** File system mounting technology is a key component in the field of data access and sharing. Among them, the Network File System (NFS), as an important tool for sharing file directories between Linux systems, has greatly promoted flexible data access. NFS-Ganesha, an open-source software project, implements extensive support for various back-end distributed storage interfaces through file system functions in user
15 space, and exposes standard file system semantics to clients, thereby enhancing system compatibility and usability.

[0003] However, current mounting via Ganesha to access NFS still fails to meet user needs for more fine-grained and flexible file access control.

SUMMARY

20 **[0004]** The present disclosure provides a method and apparatus for mounting a file system, a device, and a storage medium.

[0005] According to an aspect of the present disclosure, a method for mounting a file system is provided, executed by a file system server, the method including:

25 **[0006]** detecting whether a mount request obtained from a client includes a target token;

[0007] if the target token is included, querying a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization record includes a target identifier of the target file system; and

[0008] feeding back the target identifier to the client, enabling the client to access the target file system using the target identifier.

[0009] According to another aspect of the present disclosure, a method for mounting a file system is provided, executed by a client, the method including:

5 **[0010]** generating a mount request including a target token, and sending the mount request to a file system server, such that when the file system server detects that the mount request obtained from the client includes the target token, the file system server queries a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization
10 record includes a target identifier of the target file system;

[0011] obtaining the target identifier fed back by the file system server; and

[0012] accessing the target file system using the target identifier.

[0013] According to another aspect of the present disclosure, an apparatus for mounting a file system is provided, configured in a file system server, the apparatus
15 including:

[0014] a token detection module, configured to detect whether a mount request obtained from a client includes a target token;

[0015] a record query module, configured to query a target authorization record associated with the target token from candidate authorization records of candidate file
20 systems if the target token is included, where the target authorization record includes a target identifier of the target file system; and

[0016] an identifier sending module, configured to feed back the target identifier to the client, enabling the client to access the target file system using the target identifier.

[0017] According to another aspect of the present disclosure, an apparatus for mounting a file system is provided, configured in a client, the apparatus including:

25 **[0018]** a mount request module, configured to generate a mount request including a target token and send the mount request to a file system server, such that when the file system server detects that the mount request obtained from the client includes the target

token, the file system server queries a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization record includes a target identifier of the target file system;

[0019] an identifier feedback module, configured to obtain the target identifier fed back by the file system server; and

[0020] a file access module, configured to access the target file system using the target identifier.

[0021] According to another aspect of the present disclosure, an electronic device is provided, the electronic device including:

[0022] at least one processor; and

[0023] a memory communicatively connected to the at least one processor; where

[0024] the memory stores instructions executable by the at least one processor, and the instructions are executed by the at least one processor to enable the at least one processor to perform the method provided in any embodiment of the present disclosure.

[0025] According to another aspect of the present disclosure, a non-transitory computer-readable storage medium storing computer instructions is provided, where the computer instructions are configured to cause a computer to perform the method provided in any embodiment of the present disclosure.

[0026] According to yet another aspect of the present disclosure, a computer program product is provided, including a computer program, which, when executed by a processor, implements the method provided in any embodiment of the present disclosure.

[0027] It should be understood that the content described in this section is not intended to identify key or essential features of the embodiments of the present disclosure, nor is it intended to limit the scope of the present disclosure. Other features of the present disclosure will become readily understood through the following description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0028] The accompanying drawings are used for better understanding of the solution and do not constitute a limitation to the present disclosure.

[0029] FIG. 1a is a flowchart of a method for mounting a file system according to an embodiment of the present disclosure;

5 **[0030]** FIG. 1b is a schematic structural diagram of a file system according to an embodiment of the present disclosure;

[0031] FIG. 2 is a flowchart of another method for mounting a file system according to an embodiment of the present disclosure;

10 **[0032]** FIG. 3a is a flowchart of yet another method for mounting a file system according to an embodiment of the present disclosure;

[0033] FIG. 3b is a schematic structural diagram of another file system according to an embodiment of the present disclosure;

[0034] FIG. 4 is a flowchart of yet another method for mounting a file system according to an embodiment of the present disclosure;

15 **[0035]** FIG. 5 is a flowchart of yet another method for mounting a file system according to an embodiment of the present disclosure;

[0036] FIG. 6 is a schematic structural diagram of an apparatus for mounting a file system according to an embodiment of the present disclosure;

20 **[0037]** FIG. 7 is a schematic structural diagram of another apparatus for mounting a file system according to an embodiment of the present disclosure;

[0038] FIG. 8 is a block diagram of an electronic device for implementing the method for mounting a file system according to an embodiment of the present disclosure.

DETAILED DESCRIPTION

25 **[0039]** FIG. 1a is a flowchart of a method for mounting a file system according to an embodiment of the present disclosure. This method is applicable to scenarios where a user is allowed to mount to a specified network file system. The method may be executed by an apparatus for mounting a file system, which may be implemented in

software and/or hardware and integrated into a file system server. The file system server may be an open-source NFS server supporting multiple versions of the NFS protocol (e.g., a Ganesha server). As shown in FIG. 1a, the method for mounting a file system in this embodiment may include the following steps.

5 **[0040]** S101 includes: detecting whether a mount request obtained from a client includes a target token.

[0041] S102 includes: if the target token is included, querying a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization record includes a target
10 identifier of the target file system.

[0042] S103 includes: feeding back the target identifier to the client, enabling the client to access the target file system using the target identifier.

[0043] Referring to FIG. 1b, an NFS file system may include a client, a file system server (e.g., a Ganesha server), a back-end storage module, and a file system
15 management module. The file system server is communicatively connected to the client, the back-end storage module, and the file system management module respectively. The back-end storage module is used to store directory information of candidate file systems and file data. Some candidate file systems in the back-end storage module may be based on the NFS protocol; others may be based on other protocols (e.g., CephFS, GlusterFS).
20 Candidate file systems based on other protocols may be integrated with the NFS server through corresponding drivers or interfaces, enabling the NFS server to share data in these candidate file systems via the NFS protocol. The database of the file system management module may pre-store candidate authorization records of candidate file systems, including at least an association relationship between candidate identifiers of
25 the candidate file systems and candidate tokens. A candidate identifier is a File System ID (fsid), and one candidate file system may have different candidate authorization records for different users (i.e., different users may use their respective candidate tokens to access the same candidate file system).

[0044] A mount request may include a request type, protocol version, server
30 address, file system directory, and mount point directory. The "file system directory" refers to the directory in the candidate file system that needs to be mounted and accessed;

the "mount point directory" refers to a directory in the local file system of the client, which serves as a mount point for the candidate file system in the candidate storage module. Once the mounting is successful, the client can access files and directories in the corresponding candidate file system in the back-end storage module by accessing the local mount point directory.

[0045] The "target token" refers to a token specified by the user in the mount request (e.g., a token input by the user). The target token may be included in the file system directory of the mount request. For example, the user may obtain candidate tokens for accessing at least one candidate file system in advance. When needing to mount to one of the candidate file systems (i.e., the target file system), the user may write the corresponding target token into the file system directory to generate a mount request, and send the mount request to the file system server.

[0046] The file system server obtains the mount request from the client and detects whether the mount request includes a target token. If the mount request includes the target token, the file system server may query the target authorization record associated with the target token from the candidate authorization records of candidate file systems, where the target authorization record includes at least the target identifier of the target file system. Therefore, if a target identifier associated with the target token is queried from the candidate authorization records, the target identifier is fed back to the client, enabling the client to access the corresponding directory in the target file system using the target identifier; and if no target identifier associated with the target token exists in the candidate authorization records, the mount request is rejected.

[0047] One user may have multiple candidate tokens for different candidate file systems; a given user may also use a given candidate token to initiate mount requests on different clients to access a given candidate file system on different clients; different users may also access different candidate file systems on a given client. By obtaining the mount request from the client and detecting whether the mount request includes a user-specified target token, querying the target identifier associated with the target token if the token is included, and feeding back the target identifier to the client, the client can access the corresponding directory in the target file system using the target identifier. In other words, by allowing the user to specify a target token in the mount request and modifying the file system server's logic for processing the mount request, flexible switching

between different candidate file systems is achieved, improving the flexibility of file system mounting.

[0048] The technical solution provided by the embodiment of the present disclosure detects whether a mount request sent by a client includes a user-specified target token. If the target token is included, the target authorization record associated with the target token is queried from the candidate authorization records of candidate file systems to obtain the target identifier associated with the target token. The target identifier is then fed back to the client, enabling the client to access the corresponding directory in the target file system using the target identifier. This achieves flexible switching between different candidate file systems and improves the flexibility of file system mounting.

[0049] FIG. 2 is a flowchart of another method for mounting a file system according to an embodiment of the present disclosure. Referring to FIG. 2, further defined based on the above embodiment, the method for mounting a file system in this embodiment may include the following steps.

[0050] S201 includes: in response to a sub-query request for the file system directory in the mount request obtained from the client, detecting whether the sub-query request includes a token field;

[0051] S202 includes: if the token field is included, extracting the value of the token field from the sub-query request to obtain the target token;

[0052] S203 includes: if the target token is included, querying a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization record includes a target identifier of the target file system;

[0053] S204 includes: feeding back the target identifier to the client, enabling the client to access the target file system using the target identifier.

[0054] The client may generate a mount request according to a user-specified target token. The client kernel may split the mount request into multiple sub-requests (e.g., a mount sub-request and sub-query (lookup) requests for each level of the file

system directory) and send the sub-requests to the file system server in logical order.

[0055] During the processing of the file system server of the mount request, a root handler needs to be obtained in the mount sub-request processing phase (e.g., a fixed root handler may be returned uniformly to simplify the processing flow). The root handler is a handle for operating the root directory, and subsequent operations of the client on the target file system may be based on this handle. In the sub-query request processing phase, whether the sub-query request includes a specified token field (e.g., whether it includes "token=") may be detected. If the token field is included, the value of the token field is used as the target token. For example, for the sub-query request "token=9ed1608d", the target token is "9ed1608d". Exemplarily, the target token may be sent to the file system management module to query whether a target identifier (e.g., fsid=108) associated with the target token exists in the candidate authorization records of candidate file systems. The target identifier is then fed back to the client, enabling the client to access the target file system using the target identifier.

[0056] By detecting and extracting the value of the specified token field as the target token during the processing of the sub-query request, and querying the target identifier associated with the target token from the candidate authorization records of candidate file systems based on the token, the file system server can efficiently and accurately provide the client with permission to access the target file system, simplifying the user access process and enhancing system security.

[0057] In an optional implementation, the target token is located in the first-level directory of the file system in the mount request.

[0058] The mount request may include information such as request type, protocol version, server address, file system directory, and mount point directory. The file system directory may have a multi-level directory structure for the file system, and the target token is placed in the first-level directory of the file system. Other levels of the file system directory may correspond to actual directories in the target file system. For example, for the file system directory "token=9ed1608d/dir1", the first-level directory "token=9ed1608d" includes the target token "9ed1608d", and the second-level directory "/dir1" is an actual directory on the target file system. By placing the target token in the first-level directory of the file system, the file system server can first parse the target

token, feed back the target identifier associated with the target token to the client, and then the client can access the actual directory in the target file system using the target identifier and the information of other optional levels of directories.

[0059] In an optional implementation, the feeding back the target identifier to the client to enable the client to access the target file system using the target identifier includes: creating a file handle according to the target identifier, feeding the file handle back to the client, and enabling the client to use the target identifier as a parent file handle for an initiated new sub-query request; and in response to the new sub-query request, accessing the corresponding directory in the target file system according to the target identifier.

[0060] Exemplarily, if the sub-query request for the file system directory in the mount request includes the target token, the target identifier associated with the target token is queried and stored in the file handle returned to the client (e.g., the target identifier may be filled back into the File System Abstraction Layer (FSAL) handle of the file handle). When the client needs to initiate a new sub-query request for other levels of the file system directory, the target identifier may be used as the parent file handle (i.e., parent directory fh) of the new sub-query request and sent to the file system server. The file system server uses the target identifier in the parent file handle to identify and locate the corresponding target file system, and further accesses the actual directory in the target file system in the back-end storage module.

[0061] Taking the file system directory "token=9ed1608d/dir1" as an example: for the first-level directory "token=9ed1608d", the file system server determines the target identifier "fsid=108" associated with the target token and stores the target identifier "fsid=108" in the file handle returned to the client. When the client initiates a new sub-query request for the second-level directory "/dir1", the target identifier "fsid=108" is used as the parent file handle of the new sub-query request. It should be noted that for subsequent sub-query requests for other levels of directories, the target identifier "fsid=108" is also used as the parent file handle of the new sub-query request.

[0062] By storing the target identifier associated with the target token in the file handle fed back to the client, and enabling the client to use the target identifier as the parent file handle for subsequent new sub-query requests for other levels of the file

system directory, the actual directory in the target file system can be quickly located and accessed without repeated verification and file system specification, thereby improving the efficiency and convenience of file access.

[0063] The technical solution provided by the embodiment of the present disclosure: when the sub-query request for the file system directory includes the target token, the file system server determines the target identifier associated with the target token, stores the target identifier in the file handle fed back to the client, and enables the client to use the target identifier as the parent file handle for subsequent new sub-query requests for other levels of the file system directory. This allows quick location and access to the actual directory in the target file system without repeated verification and file system specification, thereby improving the efficiency and convenience of file access.

[0064] FIG. 3a is a flowchart of yet another method for mounting a file system according to an embodiment of the present disclosure. Referring to FIG. 3a, further defined based on the above embodiment, the method for mounting a file system in this embodiment may include the following steps.

[0065] S301 includes: detecting whether a mount request obtained from a client includes a target token.

[0066] S302 includes: if the target token is included, querying a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization record includes a target identifier and a target operation type of the target file system.

[0067] S303 includes: feeding back the target identifier and the target operation type to the client, enabling the client to perform the corresponding target operation on the corresponding directory in the target file system using the target identifier.

[0068] The target authorization record may include not only the target identifier but also a pre-authorized target operation type (e.g., read-only permission or read-write permission). In other words, the database of the file system management module may pre-store an association relationship between candidate identifiers, candidate operation types, and candidate tokens of candidate file systems.

[0069] When detecting that the mount request includes the target token, the file system server queries the target identifier and target operation type associated with the target token from the association relationship between candidate identifiers, candidate operation types, and candidate tokens, and feeds back the target identifier and target operation type to the client. After receiving the target identifier and target operation type, the client accesses the corresponding directory in the target file system using the target identifier and performs the corresponding target operation on the directory using the target operation type. Specifically, if the target operation type is "read-only", the client only performs read operations on the corresponding directory in the target file system and rejects write operations; if the target operation type is "read-write", the client may perform both read and write operations on the corresponding directory in the target file system. By specifying the operation type in the authorization record, the operation permissions of the client on the target file system can be precisely controlled, preventing unauthorized access and greatly improving the security of file access.

[0070] The technical solution provided by the embodiment of the present disclosure detects whether a mount request sent by a client includes a user-specified target token. If the target token is included, the target authorization record associated with the target token is queried from the candidate authorization records of candidate file systems to obtain the target identifier and target operation type associated with the target token. The target identifier and target operation type are then fed back to the client, enabling the client to precisely control the operation permissions on the target file system using the target identifier and target operation type, further improving the security of file access.

[0071] In an optional implementation, the target authorization record further includes a target validity period; the method further includes: detecting whether the target token is invalid according to the target validity period; and if the target token is invalid, rejecting the mount request.

[0072] The target authorization record may include not only the target identifier but also a target validity period. In other words, the database of the file system management module may pre-store an association relationship between candidate identifiers, candidate validity periods, and candidate tokens of candidate file systems.

[0073] When detecting that the mount request includes the target token, the file system server queries the target identifier and target validity period associated with the target token from the association relationship between candidate identifiers, candidate validity periods, and candidate tokens, and detects whether the target token is invalid according to the target validity period. If the target token has expired, the mount request is rejected; if the target token is still valid, the target identifier is fed back to the client, enabling the client to access the corresponding directory in the target file system using the target identifier. By specifying the validity period in the authorization record, the file system server can precisely verify whether the target token in the mount request is still within the validity period, effectively preventing unauthorized access to the target file system using expired tokens and further improving the security of file access.

[0074] In an optional implementation, the method further includes: obtaining and storing an association relationship between the target authorization record and the target token from an administration terminal, where the target authorization record is generated in response to an access authorization application submitted by the client for the target file system.

[0075] Referring to FIG. 3b, in addition to the client, file system server, back-end storage module, and file system management module, the NFS file system may further include an administration terminal, which is communicatively connected to the client and the file system management module respectively. Exemplarily, the client may send an access authorization application for the file system to the administration terminal. The access authorization application includes at least the target file system directory to be accessed. After the administrator grants authorization, a target token is generated, the association relationship between the target token and the target authorization record is stored in the database of the file system management module, and the target token is fed back to the client for the client to access the target file system using the target token.

[0076] The access authorization application may further include a target operation type and/or a target validity period. The client may send an access authorization application including the target file system directory to be accessed, as well as the target operation type and/or target validity period to the administration terminal. The administration terminal receives the access authorization application, authenticates the client, and if the authentication is passed, generates a target access token for the

client, writes the association relationship between the target file system directory, target access token, target operation type, and target validity period into the database of the file system management module (i.e., stores the association relationship between the target access token and the target authorization record), and feeds back the target access token to the client.

[0077] The access authorization application of the client may also include at least one of a candidate file system directory, a candidate operation type, and a candidate validity period. If the administration terminal passes the authentication, it generates a candidate access token for the corresponding client, writes the association relationship between the candidate file system directory, candidate access token, candidate operation type, and candidate validity period into the database of the file system management module, and obtains the association relationship between the candidate access token and the candidate authorization record. Through the authorization application, only authorized users can access the corresponding file system directory, further improving the security and controllability of file access.

[0078] FIG. 4 is a flowchart of a method for mounting a file system according to an embodiment of the present disclosure. This method is applicable to scenarios where a user is allowed to mount to a specified network file system. The method may be executed by an apparatus for mounting a file system, which may be implemented in software and/or hardware and integrated into a client. As shown in FIG. 4, the method for mounting a file system in this embodiment may include following steps.

[0079] S401 includes: generating a mount request including a target token, and sending the mount request to a file system server, such that when the file system server detects that the mount request obtained from the client includes the target token, the file system server queries a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization record includes a target identifier of the target file system.

[0080] S402 includes: obtaining the target identifier fed back by the file system server.

[0081] S403 includes: accessing the target file system using the target identifier.

[0082] The NFS file system may include a client, a file system server, a back-end storage module, and a file system management module. The file system server is communicatively connected to the client, the back-end storage module, and the file system management module respectively. The back-end storage module is used to store directory information of candidate file systems and file data. The database of the file system management module may pre-store candidate authorization records of candidate file systems, including at least an association relationship between candidate identifiers and candidate tokens of the candidate file systems. A candidate identifier is a File System ID (fsid), and one candidate file system may have different candidate authorization records for different users.

[0083] Exemplarily, the user may obtain candidate tokens for accessing at least one candidate file system in advance. When needing to mount to one of the candidate file systems (i.e., the target file system), the user may write the corresponding target token into the file system directory to generate a mount request, and send the mount request to the file system server. The file system server detects whether the mount request includes the target token; if the mount request includes the target token, the file system server queries the target authorization record associated with the target token from the candidate authorization records of candidate file systems, where the target authorization record includes at least the target identifier of the target file system. The client obtains the target identifier fed back by the file system server and accesses the target file system using the target identifier. By writing the required target token into the mount request and sending the mount request including the target token to the file system server, the client enables the file system server to parse the target token from the mount request, query the target authorization record associated with the target token from the candidate authorization records to obtain at least the target identifier associated with the target token, and feed back the target identifier to the client. The client then accesses the corresponding directory in the target file system using the target identifier, achieving flexible switching between different candidate file systems and improving the flexibility of file system mounting.

[0084] The technical solution provided by the embodiment of the present disclosure: the client generates a mount request including a specified target token and sends the mount request to the file system server. The file system server parses the target

token from the mount request, queries the target authorization record associated with the target token from the candidate authorization records to obtain at least the target identifier associated with the target token, and feeds back the target identifier to the client. The client accesses the corresponding directory in the target file system using the target identifier, achieving flexible switching between different candidate file systems and improving the flexibility of file system mounting.

[0085] FIG. 5 is a flowchart of yet another method for mounting a file system according to an embodiment of the present disclosure. Referring to FIG. 5, further defined based on the above embodiment, the method for mounting a file system in this embodiment may include following steps.

[0086] S501 includes: generating a mount request including a target token, and sending the mount request to a file system server, such that when the file system server detects that the mount request obtained from the client includes the target token, the file system server queries a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization record includes a target identifier of the target file system.

[0087] S502 includes: obtaining a file handle including the target identifier, which is fed back by the file system server, in response to a sub-query request that targets the file system directory in the mount request and includes the target token;

[0088] S503 includes: using the target identifier as a parent file handle for an initiated new sub-query request.

[0089] S504 includes: accessing the corresponding directory in the target file system using the new sub-query request.

[0090] The client may generate a mount request according to a user-specified target token. The client kernel may split the mount request into multiple sub-requests (e.g., a mount sub-request, and sub-query (lookup) requests for levels of the file system directory) and send the sub-requests to the file system server in logical order.

[0091] The client first sends a mount sub-request to the file system server, enabling the file system server to obtain a root handler (e.g., a fixed root handler may be

returned uniformly to simplify the processing flow). Subsequent operations of the client on the target file system may be based on this handle. The client may send a sub-query request to the file system server, enabling the file system server to detect whether the sub-query request includes a specified token field (e.g., whether it includes "token="). If the token field is included, the value of the token field is used as the target token. For example, for the sub-query request "token=9ed1608d", the target token is "9ed1608d". Exemplarily, the client may send the target token to the file system management module to query whether a target identifier (e.g., fsid=108) associated with the target token exists in the candidate authorization records of candidate file systems. The file system server then feeds back the target identifier to the client, and the client accesses the target file system using the fed-back target identifier.

[0092] Exemplarily, if the sub-query request sent by the client includes the target token, the file system server queries the target identifier associated with the target token and stores the target identifier in the file handle returned to the client (e.g., the target identifier may be filled back into the File System Abstraction Layer (FSAL) handle of the file handle). When the client needs to initiate a new sub-query request for other levels of the file system directory, the target identifier may be used as the parent file handle (i.e., parent directory fh) of the new sub-query request and sent to the file system server. The file system server uses the target identifier in the parent file handle to identify and locate the corresponding target file system, and further accesses the actual directory in the target file system in the back-end storage module.

[0093] Taking the file system directory "token=9ed1608d/dir1" as an example: the client sends a sub-query request for the first-level directory "token=9ed1608d", enabling the file system server to determine the target identifier "fsid=108" associated with the target token and store the target identifier "fsid=108" in the file handle returned to the client. When the client initiates a new sub-query request for the second-level directory "/dir1", the target identifier "fsid=108" is used as the parent file handle of the new sub-query request. For subsequent sub-query requests for other levels of directories, the client also uses the target identifier "fsid=108" as the parent file handle of the new sub-query request. By obtaining the target identifier associated with the target token from the file system server and using the target identifier as the parent file handle for subsequent new sub-query requests for other levels of the file system directory, the client

can quickly locate and access the actual directory in the target file system without repeated verification and file system specification, thereby improving the efficiency and convenience of file access.

[0094] The technical solution provided by the embodiment of the present disclosure: the client may initiate a sub-query request that targets the file system directory and includes the target token, enabling the file system server to feed back the target identifier associated with the target token. The client may use the target identifier as the parent file handle for subsequent new sub-query requests for other levels of the file system directory, allowing quick location and access to the actual directory in the target file system without repeated verification and file system specification, thereby improving the efficiency and convenience of file access.

[0095] In an optional implementation, the target token is located in the first-level directory of the file system in the mount request.

[0096] The mount request may include information such as request type, protocol version, server address, file system directory, and mount point directory. The file system directory may have a multi-level directory structure for the file system, and the target token is placed in the first-level directory of the file system. Other levels of the file system directory may correspond to actual directories in the target file system. For example, for the mount request "mount -t nfs4 127.0.0.1/token=9ed1608d/dir1/home/temp", the file system directory is "token=9ed1608d/dir1", the first-level directory "token=9ed1608d" includes the target token "9ed1608d", and the second-level directory "/dir1" is an actual directory on the target file system. By placing the target token in the first-level directory of the file system, the file system server can first parse the target token, feed back the target identifier associated with the target token to the client, and then the client can access the actual directory in the target file system using the target identifier and the information of other optional levels of directories.

[0097] In an optional implementation, the target authorization record further includes a target operation type; the obtaining the target identifier fed back by the file system server includes: obtaining the target identifier and the target operation type fed back by the file system server; and according to the target operation type, allowing the

client to perform the corresponding target operation on the target file system using the target identifier.

[0098] The target authorization record may include not only the target identifier but also a pre-authorized target operation type (e.g., read-only permission or read-write permission). In other words, the database of the file system management module may pre-store an association relationship between candidate identifiers, candidate operation types, and candidate tokens of candidate file systems.

[0099] When detecting that the mount request includes the target token, the file system server queries the target authorization record associated with the target token from the association relationship between candidate tokens and candidate authorization records (the target authorization record may include the target identifier and target operation type). The client obtains the target identifier and target operation type fed back by the file system server, accesses the corresponding directory in the target file system using the target identifier, and performs the corresponding target operation on the directory using the target operation type. Specifically, if the target operation type is "read-only", the client only performs read operations on the corresponding directory in the target file system and rejects write operations; if the target operation type is "read-write", the client may perform both read and write operations on the corresponding directory in the target file system. By specifying the operation type in the authorization record, the client operation permissions on the target file system can be precisely controlled, preventing unauthorized access and greatly improving the security of file access.

[0100] In an optional implementation, the method further includes: sending an access authorization application for the target file system to an administration terminal, such that the administration terminal authorizes the generation of a target authorization record including the target token and establishes an association relationship between the target authorization record and the target token; obtaining the target token fed back by the administration terminal.

[0101] In addition to the client, file system server, back-end storage module, and file system management module, the NFS file system may further include an administration terminal, which is communicatively connected to the client and the file

system management module respectively. Exemplarily, the client may send an access authorization application for the file system to the administration terminal. The access authorization application includes at least the target file system directory to be accessed. After the administrator grants authorization, a target token is generated, the association relationship between the target token and the target authorization record is stored in the database of the file system management module, and the target token is fed back to the client for the client to access the target file system using the target token.

[0102] The access authorization application may further include a target operation type and/or a target validity period. The client may send an access authorization application including the target file system directory to be accessed, as well as the target operation type and/or target validity period to the administration terminal. The administration terminal receives the access authorization application, authenticates the client, and if the authentication is passed, generates a target access token for the client, writes the association relationship between the target file system directory, target access token, target operation type, and target validity period into the database of the file system management module (i.e., stores the association relationship between the target access token and the target authorization record), and feeds back the target access token to the client.

[0103] Furthermore, the client access authorization application includes at least one of a candidate file system directory, a candidate operation type, and a candidate validity period. If the administration terminal passes the authentication, it generates a candidate access token for the corresponding client, writes the association relationship between the candidate file system directory, candidate access token, candidate operation type, and candidate validity period into the database of the file system management module, and obtains the association relationship between the candidate access token and the candidate authorization record. In other words, the user may send an access authorization application to the administration terminal via the client. The access authorization application includes at least the candidate file system directory to be accessed, and may also include the required candidate operation type and/or candidate validity period. After the administration terminal passes the authentication (e.g., determines that the user has supervision permissions for the candidate file system), it generates a candidate token for the user for the candidate file system. The administration

terminal also feeds back the candidate token to the client (enabling the client to initiate a mount request using the corresponding token) and stores the association relationship between the candidate authorization record and the candidate token in the database of the file system management module (the candidate authorization record includes at least the candidate file system directory). Through the authorization application, only authorized users can access the corresponding file system directory, further improving the security and controllability of file access.

[0104] FIG. 6 is a schematic structural diagram of an apparatus for mounting a file system according to an embodiment of the present disclosure. This apparatus is applicable to scenarios where a user is allowed to mount to a specified network file system. The apparatus may be implemented in software and/or hardware and integrated into a file system server. As shown in FIG. 6, the apparatus for mounting a file system in this embodiment may include:

[0105] a token detection module 610, configured to detect whether a mount request obtained from a client includes a target token;

[0106] a record query module 620, configured to query a target authorization record associated with the target token from candidate authorization records of candidate file systems if the target token is included, where the target authorization record includes a target identifier of the target file system; and

[0107] an identifier sending module 630, configured to feed back the target identifier to the client, enabling the client to access the target file system using the target identifier.

[0108] In an optional implementation, the target token is located in the first-level directory of the file system in the mount request.

[0109] In an optional implementation, the token detection module 610 includes:

[0110] a token field unit, configured to detect whether a sub-query request includes a token field in response to the sub-query request for the file system directory in the mount request obtained from the client;

[0111] a token extraction unit, configured to extract the value of the token field

from the sub-query request to obtain the target token if the token field is included.

[0112] In an optional implementation, the identifier sending module includes:

[0113] a file handle unit, configured to create a file handle according to the target identifier, feed the file handle back to the client, and enable the client to use the target identifier as a parent file handle for an initiated new sub-query request;

[0114] a directory access unit, configured to access the corresponding directory in the target file system according to the target identifier in response to the new sub-query request.

[0115] In an optional implementation, the target authorization record further includes a target operation type;

[0116] the identifier sending module 630 is specifically configured to:

[0117] feed back the target identifier and the target operation type to the client, enabling the client to be allowed to perform the corresponding target operation on the corresponding directory in the target file system using the target identifier.

[0118] In an optional implementation, the target authorization record further includes a target validity period; and the apparatus 600 for mounting a file system further includes:

[0119] a token invalidation module, configured to detect whether the target token is invalid according to the target validity period; and if the target token is invalid, reject the mount request.

[0120] According to an embodiment of the disclosure, the apparatus further includes:

[0121] a record storage module, configured to obtain and store an association relationship between the target authorization record and the target token from an administration terminal, where the target authorization record is generated in response to an access authorization application submitted by the client for the target file system.

[0122] The apparatus for mounting a file system provided by the embodiment of

the present invention may execute the method for mounting a file system provided by any embodiment of the present disclosure, and has corresponding functional modules and beneficial effects for executing the method.

[0123] FIG. 7 is a schematic structural diagram of an apparatus for mounting a file system according to an embodiment of the present disclosure. This apparatus is applicable to scenarios where a user is allowed to mount to a specified network file system. The apparatus may be implemented in software and/or hardware and integrated into a client. As shown in FIG. 7, the apparatus for mounting a file system in this embodiment may include:

[0124] a mount request module 710, configured to generate a mount request including a target token and send the mount request to a file system server, such that when the file system server detects that the mount request obtained from the client includes the target token, the file system server queries a target authorization record associated with the target token from candidate authorization records of candidate file systems, where the target authorization record includes a target identifier of the target file system;

[0125] an identifier feedback module 720, configured to obtain the target identifier fed back by the file system server; and

[0126] a file access module 730, configured to access the target file system using the target identifier.

[0127] In an optional implementation, the target token is located in the first-level directory of the file system in the mount request.

[0128] In an optional implementation, the identifier feedback module 720 includes:

[0129] a file handle unit, configured to obtain a file handle including the target identifier, which is fed back by the file system server in response to a sub-query request that targets the file system directory in the mount request and includes the target token;

[0130] the file access module 730 includes:

[0131] a new request initiation module, configured to use the target identifier as a parent file handle for an initiated new sub-query request;

[0132] a file access unit, configured to access the corresponding directory in the target file system using the new sub-query request.

5 **[0133]** In an optional implementation, the target authorization record further includes a target operation type;

[0134] the identifier feedback module 720 includes:

[0135] an identifier feedback unit, configured to obtain the target identifier and the target operation type fed back by the file system server;

10 **[0136]** a target operation unit, configured to allow the client to perform the corresponding target operation on the target file system using the target identifier according to the target operation type.

[0137] In an optional implementation, the apparatus 700 for mounting a file system further includes an access authorization module, and the access authorization
15 module includes:

[0138] an authorization application unit, configured to send an access authorization application for the target file system to an administration terminal, such that the administration terminal authorizes the generation of a target authorization record including the target token and establishes an association relationship between the target
20 authorization record and the target token;

[0139] a target token unit, configured to obtain the target token fed back by the administration terminal.

[0140] The apparatus for mounting a file system provided by the embodiment of the present disclosure may execute the method for mounting a file system provided by
25 any embodiment of the present disclosure, and has corresponding functional modules and beneficial effects for executing the method.

[0141] In the technical solution of the present disclosure, the acquisition, storage, and application of user personal information all comply with relevant laws and

regulations and do not violate public order and good morals.

[0142] According to the embodiments of the present disclosure, the present disclosure further provides an electronic device, a readable storage medium, and a computer program product.

5 **[0143]** FIG. 8 is a block diagram of an electronic device for implementing the method for mounting a file system according to an embodiment of the present disclosure. FIG. 8 shows a schematic block diagram of an example electronic device that may be used to implement the embodiments of the present disclosure. The electronic device is intended to represent various forms of digital computers, such as laptops, desktops,
10 workstations, personal digital assistants, servers, blade servers, mainframes, and other suitable computers. The electronic device may also represent various forms of mobile devices, such as personal digital assistants, cellular phones, smartphones, wearable devices, and other similar computing devices. The components shown herein, their connections and relationships, and their functions are merely examples and are not
15 intended to limit the implementation of the disclosure described and/or claimed herein.

[0144] As shown in FIG. 8, the electronic device 800 includes a computing unit 801, which can perform various appropriate actions and processes according to a computer program stored in a read-only memory (ROM) 802 or a computer program loaded from a storage unit into a random access memory (RAM) 803. In the RAM 803,
20 various programs and data required for the operation of the electronic device 800 can also be stored. The computing unit 801, ROM 802, and RAM 803 are connected to each other via a bus. An input/output (I/O) interface 805 is also connected to the bus 804.

[0145] A plurality of components in the electronic device are connected to the I/O interface 805, including: an input unit 806 (e.g., a keyboard, a mouse); an output unit
25 807 (e.g., various types of displays, speakers); a storage unit 808 (e.g., a magnetic disk, an optical disk); and a communication unit 809 (e.g., a network card, a modem, a wireless communication transceiver). The communication unit 809 allows the electronic device 800 to exchange information/data with other devices via a computer network such as the Internet and/or various telecommunications networks.

30 **[0146]** The computing unit 801 may be various general-purpose and/or special-purpose processing components with processing and computing capabilities. Examples

of the computing unit 801 include, but are not limited to, a central processing unit (CPU), a graphics processing unit (GPU), various dedicated artificial intelligence (AI) computing chips, various computing units running machine learning model algorithms, a digital signal processor (DSP), and any suitable processor, controller, microcontroller, etc. The computing unit 801 executes the various methods and processes described above, such as the method for mounting a file system. For example, in some embodiments, the method for mounting a file system may be implemented as a computer software program tangibly embodied in a machine-readable medium such as the storage unit 808. In some embodiments, part or all of the computer program may be loaded and/or installed on the electronic device via the ROM 802 and/or the communication unit 809. When the computer program is loaded into the RAM 803 and executed by the computing unit 801, one or more steps of the method for mounting a file system described above may be executed. Alternatively, in other embodiments, the computing unit may be configured to execute the method for mounting a file system by any other suitable means (e.g., via firmware).

[0147] FIG. 8 shows a schematic block diagram of an example electronic device that may be used to implement the embodiments of the present disclosure. The electronic device is intended to represent various forms of digital computers, such as laptops, desktops, workstations, personal digital assistants, servers, blade servers, mainframes, and other suitable computers. The electronic device may also represent various forms of mobile devices, such as personal digital assistants, cellular phones, smartphones, wearable devices, and other similar computing devices. The components shown herein, their connections and relationships, and their functions are merely examples and are not intended to limit the implementation of the disclosure described and/or claimed herein.

[0148] Various implementations of the systems and techniques described herein may be implemented in digital electronic circuit systems, integrated circuit systems, field-programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), application-specific standard products (ASSPs), system-on-a-chip (SoC) systems, complex programmable logic devices (CPLDs), computer hardware, firmware, software, and/or combinations thereof. These various implementations may include: being implemented in one or more computer programs that can be executed and/or interpreted on a programmable system including at least one programmable processor

(which may be a dedicated or general-purpose programmable processor) that can receive data and instructions from a storage system, at least one input device, and at least one output device, and transmit data and instructions to the storage system, the at least one input device, and the at least one output device.

5 **[0149]** Program code for implementing the methods of the present disclosure may be written in any combination of one or more programming languages. These program codes may be provided to a processor or controller of a general-purpose computer, a special-purpose computer, or other programmable data processing apparatus, such that the program codes, when executed by the processor or controller, cause the
10 functions/operations specified in the flowcharts and/or block diagrams to be implemented. The program codes may be executed entirely on the machine, partially on the machine, partially on the machine and partially on a remote machine, or entirely on the remote machine or server.

15 **[0150]** In the context of the present disclosure, a machine-readable medium may be a tangible medium that can contain or store a program for use by or in connection with an instruction execution system, apparatus, or device. The machine-readable medium may be a machine-readable signal medium or a machine-readable storage medium. The machine-readable medium may include, but is not limited to, electronic, magnetic, optical, electromagnetic, infrared, or semiconductor systems, apparatuses, or
20 devices, or any suitable combination thereof. More specific examples of machine-readable storage media include one or more wire-based electrical connections, a portable computer diskette, a hard disk, a random-access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage
25 device, a magnetic storage device, or any suitable combination thereof.

30 **[0151]** To provide interaction with a user, the systems and techniques described herein may be implemented on a computer having: a display device (e.g., a cathode ray tube (CRT) or a liquid crystal display (LCD) monitor) for displaying information to the user; and a keyboard and a pointing device (e.g., a mouse or a trackball) through which the user can provide input to the computer. Other types of devices may also be used to provide interaction with the user; for example, feedback provided to the user may be any form of sensory feedback (e.g., visual feedback, auditory feedback, or tactile feedback);

and input from the user may be received in any form, including acoustic input, voice input, or tactile input.

[0152] The systems and techniques described herein may be implemented in a computing system that includes a back-end component (e.g., as a data server), or a
5 middleware component (e.g., an application server), or a front-end component (e.g., a user computer having a graphical user interface or a web browser through which the user can interact with implementations of the systems and techniques described herein), or any combination of such back-end, middleware, or front-end components. The components of the system may be interconnected by any form or medium of digital data
10 communication (e.g., a communication network). Examples of communication networks include a local area network (LAN), a wide area network (WAN), and the Internet.

[0153] A computer system may include a client and a server. The client and server are typically remote from each other and typically interact through a communication network. The relationship between the client and the server is generated
15 by computer programs running on the respective computers and having a client-server relationship with each other. The server may be a cloud server, a server of a distributed system, or a server incorporating a blockchain.

[0154] Artificial intelligence (AI) is a discipline that studies how to make computers simulate human thinking processes and intelligent behaviors (such as
20 learning, reasoning, thinking, and planning). It includes both hardware-level technologies and software-level technologies. AI hardware technologies generally include technologies such as sensors, dedicated AI chips, cloud computing, distributed storage, and big data processing; AI software technologies mainly include computer vision technologies, speech recognition technologies, natural language processing technologies,
25 machine learning/deep learning technologies, big data processing technologies, and knowledge graph technologies.

[0155] Cloud computing refers to a technical system that accesses elastic and scalable shared physical or virtual resource pools through a network, where the resources may include servers, operating systems, networks, software, applications, and storage
30 devices, and can be deployed and managed in a on-demand and self-service manner. Through cloud computing technologies, efficient and powerful data processing

capabilities can be provided for the application and model training of technologies such as AI and blockchain.

[0156] It should be understood that the various forms of processes shown above can be reordered, added, or deleted. For example, the steps described in the present disclosure can be executed in parallel, sequentially, or in a different order, as long as the desired results of the technical solution of the present disclosure can be achieved, and no limitation is imposed herein.

[0157] The specific implementations described above do not limit the protection scope of the present disclosure. It will be apparent to those skilled in the art that various modifications, combinations, sub-combinations, and substitutions can be made based on design requirements and other factors. Any modifications, equivalent substitutions, and improvements made within the spirit and principles of the present disclosure should be included in the protection scope of the present disclosure.

WHAT IS CLAIMED IS

1. A method for mounting a file system, executed by a file system server, the method comprising:

detecting whether a mount request obtained from a client includes a target token;

if the target token is included, querying a target authorization record associated with the target token from candidate authorization records of candidate file systems, wherein the target authorization record includes a target identifier of the target file system; and

feeding back the target identifier to the client, enabling the client to access the target file system using the target identifier.

2. The method according to claim 1, wherein the target token is located in a first-level directory of the file system in the mount request.

3. The method according to claim 1 or 2, wherein the detecting whether the mount request obtained from the client includes the target token comprises:

in response to a sub-query request for the file system directory in the mount request obtained from the client, detecting whether the sub-query request includes a token field; and

if the token field is included, extracting a value of the token field from the sub-query request to obtain the target token.

4. The method according to claim 3, wherein the feeding back the target identifier to the client to enable the client to access the target file system using the target identifier comprises:

creating a file handle according to the target identifier, feeding the file handle back to the client, and enabling the client to use the target identifier as a parent file handle for an initiated new sub-query request; and

in response to the new sub-query request, accessing a corresponding directory in the target file system according to the target identifier.

5 5. The method according to claim 1, wherein the target authorization record further includes a target operation type;

 the feeding back the target identifier to the client to enable the client to access the target file system using the target identifier comprises:

 feeding back the target identifier and the target operation type to the client, enabling the client to be allowed to perform a corresponding target operation on a
10 corresponding directory in the target file system using the target identifier.

6. The method according to claim 1, wherein the target authorization record further includes a target validity period;

 the method further comprises: detecting whether the target token is invalid
15 according to the target validity period; and if the target token is invalid, rejecting the mount request.

7. The method according to claim 1, further comprising:

 obtaining and storing an association relationship between the target
20 authorization record and the target token from an administration terminal, wherein the target authorization record is generated in response to an access authorization application submitted by the client for the target file system.

8. A method for mounting a file system, executed by a client, the method
25 comprising:

 generating a mount request including a target token, and sending the mount request to a file system server, such that when the file system server detects that the mount request obtained from the client includes the target token, the file system server queries a target authorization record associated with the target
30 token from candidate authorization records of candidate file systems, where the

target authorization record includes a target identifier of the target file system;
obtaining the target identifier fed back by the file system server; and
accessing the target file system using the target identifier.

5 9. The method according to claim 8, wherein the target token is located in a first-level directory of the file system in the mount request.

10 10. The method according to claim 8 or 9, wherein the obtaining the target identifier fed back by the file system server comprises:

obtaining a file handle including the target identifier, fed back by the file system server in response to a sub-query request that targets the file system directory in the mount request and includes the target token;

the accessing the target file system using the target identifier comprises:

15 using the target identifier as a parent file handle for an initiated new sub-query request; and

accessing the corresponding directory in the target file system using the new sub-query request.

20 11. The method according to claim 8, wherein the target authorization record further includes a target operation type;

the obtaining the target identifier fed back by the file system server comprises:

obtaining the target identifier and the target operation type fed back by the file system server;

25 according to the target operation type, allowing the client to perform the corresponding target operation on the target file system using the target identifier.

12. The method according to claim 8, further comprising:

30 sending an access authorization application for the target file system to an administration terminal, such that the administration terminal authorizes

generation of a target authorization record including the target token and establishes an association relationship between the target authorization record and the target token; and

obtaining the target token fed back by the administration terminal.

5

13. An apparatus for mounting a file system, configured in a file system server, the apparatus comprising:

a token detection module, configured to detect whether a mount request obtained from a client includes a target token;

10

a record query module, configured to query a target authorization record associated with the target token from candidate authorization records of candidate file systems if the target token is included, wherein the target authorization record includes a target identifier of the target file system; and

15

an identifier sending module, configured to feed back the target identifier to the client, enabling the client to access the target file system using the target identifier.

14. An apparatus for mounting a file system, configured in a client, the apparatus comprising:

20

a mount request module, configured to generate a mount request including a target token and send the mount request to a file system server, such that when the file system server detects that the mount request obtained from the client includes the target token, the file system server queries a target authorization record associated with the target token from candidate authorization records of candidate file systems, wherein the target authorization record includes a target identifier of the target file system;

25

an identifier feedback module, configured to obtain the target identifier fed back by the file system server; and

a file access module, configured to access the target file system using the target identifier.

15. An electronic device, comprising:

5

at least one processor; and

a memory communicatively connected to the at least one processor; wherein the memory stores instructions executable by the at least one processor, and the instructions are executed by the at least one processor to enable the at least one processor to perform the method according to any one of claims 1 to 12.

10

16. A non-transitory computer-readable storage medium storing computer instructions, wherein the computer instructions are configured to cause a computer to perform the method according to any one of claims 1 to 12.

15

17. A computer program product, comprising a computer program, which, when executed by a processor, implements the method according to any one of claims 1 to 12.