(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 October 2015 (15.10.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/154700 A1

(51) International Patent Classification:
G06F 11/00 (2006.01)

(21) International Application Number:
PCT/CN2015/076231

(22) International Filing Date:
10 April 2015 (10.04.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
201410145380.8 11 April 2014 (11.04.2014) CN

(71) Applicant: **TENCENT TECHNOLOGY (SHENZHEN) COMPANY LIMITED** [CN/CN]; Room 403, East Block 2, SEG Park, Zhenxing Road, Futian, Shenzhen, Guangdong 518000 (CN).

(72) Inventor: **ZHANG, Shiwei**; Room 403, East Block 2, SEG Park, Zhenxing Road, Futian, Shenzhen, Guangdong 518000 (CN).

(74) Agent: **GUANGZHOU SCIHEAD PATENT AGENT CO.. LTD**; Room 1508, Huihua Commercial & Trade Building, No. 80, XianLie Zhong Road, Yuexiu, Guangzhou, Guangdong 510070 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: METHOD AND APPARATUS FOR REPAIRING DYNAMIC LINK LIBRARY FILE

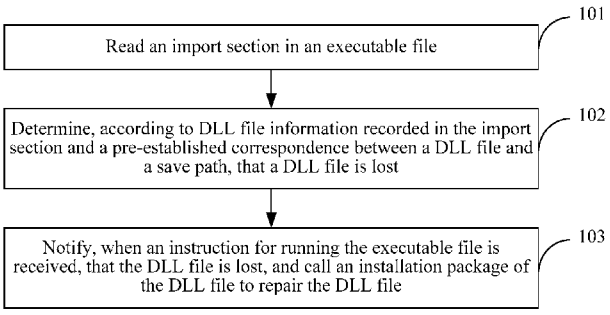


FIG. 1

(57) Abstract: A method for repairing a dynamic link library (DLL) file includes: reading an import section in an executable file; determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and notifying, when an instruction for running the executable file is received, that the DLL file is lost, and calling an installation package of the DLL file to repair the DLL file. The apparatus includes: a reading module, a determining module and a repairing module.

WO 2015/154700 A1

METHOD AND APPARATUS FOR REPAIRING DYNAMIC LINK LIBRARY FILE

FIELD OF THE TECHNOLOGY

[0001] The present disclosure relates to the field of communications, and in particular, to a method and an apparatus for repairing a dynamic link library (DLL) file.

BACKGROUND OF THE DISCLOSURE

[0002] With the development of computer technologies, there are increasingly more types of client software. In order to reduce software size so as to facilitate independent development and better release and iteration, each software company reduces interdependence between developers, and different modules are kept to be independent from each other by using a method of a DLL during software development, so that reuse of code and effective use of memory are promoted. However, meanwhile some inevitable problems are brought, for example, a version of the DLL is inconsistent with a version of an executable program, or the DLL is deleted mistakenly or renamed. All these situations cause DLL file loss when a program is started or run, thereby resulting in an abnormal end or even a crash.

[0003] At present, common software cannot manage a situation in which a DLL file of the software is lost, but lets a program end abnormally. A good software developer notifies a user of DLL file loss before installation of a program.

[0004] However, defects of the foregoing method are as follows: When a user sees a dialog box indicating DLL file loss, the user always does not know what to do; and even the user downloads a DLL file manually, the DLL file cannot be installed because the user does not know a correct save path; therefore, it is difficult to manually repair the DLL file as manual repairing has a high requirement on the user.

SUMMARY

[0005] One aspect provides a method for repairing a DLL file, including:

reading an import section in an executable file;

determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and

notifying, when an instruction for running the executable file is received, that the DLL file is lost, and calling an installation package of the DLL file to repair the DLL file.

[0006] Another aspect provides an apparatus for repairing a DLL file, including:

a reading module, configured to read an import section in an executable file;

a determining module, configured to determine, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and

a repairing module, configured to notify, when an instruction for running the executable file is received, that the DLL file is lost, and call an installation package of the DLL file to repair the DLL file.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] To describe the technical solutions of the embodiments of the present invention more clearly, the following briefly introduces the accompanying drawings required for describing the embodiments. Apparently, the accompanying drawings in the following description show only some embodiments of the present invention, and a person of ordinary skill in the art may still derive other drawings from these accompanying drawings without creative efforts.

[0008] FIG. 1 is a flowchart of a method for repairing a DLL file according to Embodiment 1 of the present invention;

[0009] FIG. 2 is a schematic diagram of an import section according to Embodiment 1 of the present invention;

[0010] FIG. 3 is a flowchart of a method for repairing a DLL file according to Embodiment 2 of the present invention;

[0011] FIG. 4 is a structural diagram of an apparatus for repairing a DLL file according to Embodiment 3 of the present invention;

[0012] FIG. 5 is another structural diagram of an apparatus for repairing a DLL file according to Embodiment 3 of the present invention; and

[0013] FIG. 6 is a structural diagram of a terminal according to Embodiment 4 of the present invention.

DESCRIPTION OF EMBODIMENTS

[0014] To make the objectives, technical solutions, and advantages of the present disclosure clearer, implementation manners of the present disclosure are further described below in detail with reference to the accompanying drawings.

Embodiment 1

[0015] Referring to FIG. 1, this embodiment provides a method for repairing a DLL file, including:

[0016] 101: Read an import section in an executable file.

[0017] Generally, the executable file includes the import section. The import section lists DLL files needed by the executable file, and symbols, including functions, variables and the like, that are referenced by the executable file from the DLL files.

[0018] For example, referring to FIG. 2, FIG. 2 is a schematic diagram of some content of an executable file notepad.exe, which includes the import section. The import section lists DLL files needed by the executable file notepad.exe, including ADVAPI32.dll, KERNEL32.dll, and the like, and functions used in the DLL files. For example, an ADVAPI32.dll file uses functions RegSetValueExW, RegQueryValueExW, RegCloseKey, and the like, which are not listed one by one herein.

[0019] 102: Determine, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost.

[0020] In this embodiment, the DLL file information at least includes a name of a DLL file, and may further include a name of an executable file to which the DLL file belongs. The pre-established correspondence may be stored in a local device in the form of a list. The correspondence may include: a name of a DLL file and a corresponding save path, or may include: a name of a DLL file, a corresponding save path, and a name of an executable file. This embodiment does not specifically limit the correspondence. The name of a DLL file may be used as an index for searching for a corresponding save path.

[0021] 103: Notify, when an instruction for running the executable file is received, that the DLL file is lost, and call an installation package of the DLL file to repair the DLL file.

[0022] Because it is determined that the DLL file is lost, a current running process of the executable file ends abnormally, that is, cannot run any longer. Notifying that the DLL file is lost

can help a user to know in time a reason for failed running of the executable file, so as to determine a subsequent operation.

[0023] In addition, a form of a pop-up dialog box may be used for notifying that the DLL file is lost, for example, a TIPS window, and notification content is displayed in the dialog box. Certainly, another manner may also be used for implementation, which is not specifically limited in this embodiment.

[0024] In this embodiment, the installation package of the DLL file may be provided by an operating system of a local device. When it is determined to perform repairing, the installation package may be automatically called to perform installation of the DLL file, to ensure that the executable file can be executed normally during running next time. A process of the installation is performed automatically. The user does not need to know where is a save path of the DLL file. Therefore, a requirement on the user is reduced greatly, and the user may automatically complete repairing of the DLL file according to a notification, thereby facilitating use for a user.

[0025] In this embodiment, the determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost may include:

searching, according to the DLL file information recorded in the import section, the pre-established correspondence between a DLL file and a save path to obtain a save path of each DLL file recorded in the import section;

determining, for the save path of each DLL file obtained through searching, whether there is a corresponding DLL file saved under the save path; and

determining, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost.

[0026] In this embodiment, before the calling an installation package of the DLL file to repair the DLL file, the method may further include:

ending a current running process of the executable file.

[0027] In this embodiment, the calling an installation package of the DLL file to repair the DLL file may include:

querying a user whether to repair the DLL file; and

calling, after the user gives a confirmation, the installation package of the DLL file to repair the DLL file.

[0028] In this embodiment, the method may further include:

determining, when the instruction for running the executable file is received again, whether repairing of the DLL file is completed; and

responding, if the repairing of the DLL file is completed, to the instruction that is received again and running the executable file; and

refusing, if the repairing of the DLL file is not completed, to respond to the instruction that is received again, or responding to the instruction that is received again after the repairing is completed.

[0029] In the foregoing method provided in this embodiment, an import section in an executable file is read; it is determined, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and it is notified, when an instruction for running the executable file is received, that the DLL file is lost, and an installation package of the DLL file is called to repair the DLL file. Therefore, intelligent awareness of loss of the DLL file is achieved, and repairing is automatically performed after a user is notified, thereby facilitating repairing of the DLL file; the user does not need to manually perform processes such as downloading and installation, thereby reducing a requirement on the user, achieving an easy operation, and facilitating use for the user. Moreover, during running of the executable file, loss of the DLL file may be determined without the need of performing an operation of loading the DLL file to an address space, and pre-determining is completed before the running, thereby improving repairing efficiency.

Embodiment 2

[0030] Referring to FIG. 3, this embodiment provides a method for repairing a DLL file, including:

[0031] 301: Read an import section in an executable file.

[0032] 302: Search, according to DLL file information recorded in the import section, a pre-established correspondence between a DLL file and a save path to obtain a save path of each DLL file recorded in the import section.

[0033] 303: Determine, for the save path of each DLL file obtained through searching, whether there is a corresponding DLL file saved under the save path.

[0034] In this embodiment, if the read import section includes multiple DLL files, save paths of the multiple DLL files are obtained through searching. Therefore, the multiple save paths

are determined one by one, so as to learn whether a DLL file is lost and which specific DLL file is lost.

[0035] 304: Determine, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost.

[0036] For any save path, if there is a corresponding DLL file saved under the save path, it may be determined that the DLL file is not lost, and if there is no corresponding DLL file saved under the save path, it may be determined that the DLL file is lost.

[0037] When there are multiple DLL files in the import section, save paths of the multiple DLL files are determined one by one. If corresponding DLL files are saved under all save paths, it may be determined that no DLL file is lost, and the executable file may be run normally. This embodiment mainly involves a scenario in which a DLL file is lost, and therefore, a scenario in which no DLL file is lost is not described further. Certainly, for the scenario in which a DLL file is lost, one or multiple DLL files may be lost. In this embodiment, information of all lost DLL files may be acquired, so as to perform repairing one by one.

[0038] 305: Notify, when an instruction for running the executable file is received, that the DLL file is lost, and end a current running process of the executable file.

[0039] Because a DLL file is lost, the executable file cannot be normally run currently, and is abnormally ended. Only after repairing of the lost DLL file succeeds, the executable file can be run normally.

[0040] 306: Query a user whether to repair the DLL file; and call, after the user gives a confirmation, an installation package of the DLL file to repair the DLL file.

[0041] In this embodiment, further, the method may further include:

determining, when the instruction for running the executable file is received again, whether repairing of the DLL file is completed; and

responding, if the repairing of the DLL file is completed, to the instruction that is received again and running the executable file; and

refusing, if the repairing of the DLL file is not completed, to respond to the instruction that is received again, or responding to the instruction that is received again after the repairing is completed.

[0042] After the repairing of the lost DLL file is completed, if the executable file is run again, the DLL file can be correctly loaded to an address space that is allocated for the executable

file, so that it is ensured that the executable file can call relevant functions in the DLL file to enable the executable file to be run correctly. During repairing of the DLL file, if an instruction for running the executable file is received, the instruction cannot be responded to. Therefore, the instruction may be refused, and the user is notified to perform running after the repairing is completed; or when the instruction for running is received again, the executable file is run automatically by default after the repairing is completed.

[0043] In the foregoing method provided in this embodiment, an import section in an executable file is read; according to DLL file information recorded in the import section, a pre-established correspondence between a DLL file and a save path is searched to obtain a save path of each DLL file recorded in the import section; for the save path, obtained through searching, of each DLL file, it is determined whether there is a corresponding DLL file saved under the save path; it is determined, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost; it is notified, when an instruction for running the executable file is received, that the DLL file is lost, and a current running process of the executable file is ended; a user is queried whether to repair the DLL file; and after the user gives a confirmation, an installation package of the DLL file is called to repair the DLL file. Therefore, intelligent awareness of loss of the DLL file is achieved, and repairing is automatically performed after a user is notified, thereby facilitating repairing of the DLL file; the user does not need to manually perform processes such as downloading and installation, thereby reducing a requirement on the user, achieving an easy operation, and facilitating use for the user. Moreover, during running of the executable file, loss of the DLL file may be determined without the need of performing an operation of loading the DLL file to an address space, and pre-determining is completed before the running, thereby improving repairing efficiency.

Embodiment 3

[0044] Referring to FIG. 4, this embodiment provides an apparatus for repairing a DLL file, including:

a reading module 401, configured to read an import section in an executable file;

a determining module 402, configured to determine, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and

a repairing module 403, configured to notify, when an instruction for running the executable file is received, that the DLL file is lost, and call an installation package of the DLL file to repair the DLL file.

[0045] Referring to FIG. 5, in this embodiment, the determining module 402 may include:

a searching unit 402a, configured to search, according to the DLL file information recorded in the import section, the pre-established correspondence between a DLL file and a save path to obtain a save path of each DLL file recorded in the import section;

a judging unit 402b, configured to: determine, for the save path of each DLL file obtained through searching, whether there is a corresponding DLL file saved under the save path; and

a determining unit 402c, configured to determine, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost.

[0046] In this embodiment, the apparatus may further include:

an ending module, configured to end a current running process of the executable file before the repairing module calls the installation package of the DLL file to repair the DLL file.

[0047] In this embodiment, the repairing module may include:

a querying unit, configured to query a user whether to repair the DLL file; and

a repairing unit, configured to call, after the user gives a confirmation, the installation package of the DLL file to repair the DLL file.

[0048] In this embodiment, the apparatus may further include:

a responding module, configured to determine, when the instruction for running the executable file is received again, whether repairing of the DLL file is completed; and respond, if the repairing of the DLL file is completed, to the instruction that is received again and run the executable file; and refuse, if the repairing of the DLL file is not completed, to respond to the instruction that is received again, or respond to the instruction that is received again after the repairing is completed.

[0049] The foregoing apparatus provided in this embodiment may execute the method provided by any of the foregoing method embodiments. Refer to descriptions in the foregoing method embodiments for a detailed process, and the details are not described herein again. The foregoing apparatus may be applied to a terminal. The terminal includes, but is not limited to, a mobile phone, a tablet computer, a computer, a notebook computer, and the like.

[0050] With the foregoing apparatus provided in this embodiment, an import section in an executable file is read; it is determined, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and it is notified, when an instruction for running the executable file is received, that the DLL file is lost, and an installation package of the DLL file is called to repair the DLL file. Therefore, intelligent awareness of loss of the DLL file is achieved, and repairing is automatically performed after a user is notified, thereby facilitating repairing of the DLL file; the user does not need to manually perform processes such as downloading and installation, thereby reducing a requirement on the user, achieving an easy operation, and facilitating use for the user. Moreover, during running of the executable file, loss of the DLL file may be determined without the need of performing an operation of loading the DLL file to an address space, and pre-determining is completed before the running, thereby improving repairing efficiency.

Embodiment 4

[0051] Referring to FIG. 6, this embodiment provides a terminal 600. The terminal 600 may include components such as a communications unit 610, a memory 620 including one or more non-volatile readable storage media, an input unit 630, a display unit 640, a sensor 650, an audio circuit 660, a wireless fidelity (WiFi) module 670, a processor 680 including one or more processing cores, and a power supply 690.

[0052] A person skilled in the art may understand that the structure of the terminal shown in FIG. 6 does not constitute a limitation to the terminal, and the terminal may include more components or fewer components than those shown in the figure, or some components may be combined, or a different component deployment may be used.

[0053] The communications unit 610 may be configured to receive and send signals during an information receiving and sending process or a call process. The communications unit 610 may be a network communications device, such as a radio frequency (RF) circuit, a router, or a modem. Particularly, when the communications unit 610 is the RF circuit, the communications unit receives downlink information from a base station, then delivers the downlink information to one or more processors 680 for processing, and sends related uplink data to the base station. Generally, the RF circuit serving as the communications unit includes, but is not limited to, an antenna, at least one amplifier, a tuner, one or more oscillators, a subscriber identity module (SIM) card, a transceiver, a coupler, a low noise amplifier (LNA), and a duplexer. In addition, the communications unit 610 may also communicate with a network and another device by wireless communication. The wireless communication may use any communications standard or protocol, which includes, but is not

limited to, Global System for Mobile communications (GSM), a General Packet Radio Service (GPRS), Code Division Multiple Access (CDMA), Wideband Code Division Multiple Access (WCDMA), Long Term Evolution (LTE), e-mail, Short Messaging Service (SMS). The memory 620 may be configured to store a software program and module. The processor 680 runs the software program and module stored in the memory 620, to implement various functional applications and data processing. The memory 620 may mainly include a program storage area and a data storage area. The program storage area may store an operating system, an application program required by at least one function (such as a sound playback function and an image display function), and the like. The data storage area may store data (such as audio data and an address book) created according to use of the terminal 600, and the like. In addition, the memory 620 may include a high speed random access memory, and may also include a non-volatile memory, such as at least one magnetic disk storage device, a flash memory, or another volatile solid-state storage device. Accordingly, the memory 620 may further include a memory controller, so as to provide access of the processor 680 and the input unit 630 to the memory 620.

[0054] The input unit 630 may be configured to receive input digit or character information, and generate a keyboard, mouse, joystick, optical, or track ball signal input related to the user setting and function control. Optionally, the input unit 630 may include a touch-sensitive surface 630a and another input device 630b. The touch-sensitive surface 630a, which may also be referred to as a touch screen or a touch panel, may collect a touch operation of a user on or near the touch-sensitive surface (such as an operation of a user on or near the touch-sensitive surface 630a by using any suitable object or accessory, such as a finger or a **stylus**), and drive a corresponding connection apparatus according to a preset program. Optionally, the touch-sensitive surface 630a may include two parts: a touch detection apparatus and a touch controller. The touch detection apparatus detects a touch position of the user, detects a signal generated by the touch operation, and transfers the signal to the touch controller. The touch controller receives the touch information from the touch detection apparatus, converts the touch information into touch point coordinates, and sends the touch point coordinates to the processor 680. Moreover, the touch controller can receive and execute a command sent from the processor 680. In addition, the touch-sensitive surface 630a may be a resistive, capacitive, infrared, or surface sound wave type touch-sensitive surface. In addition to the touch-sensitive surface 630a, the input unit 630 may further include the another input device 630b. Optionally, the another input device 630b may include, but is not limited to, one or more of a physical keyboard, a functional key (such as a volume control key or a switch key), a track ball, a mouse, and a joystick.

[0055] The display unit 640 may be configured to display information input by the user or information provided for the user, and various graphical user interfaces of the terminal 600. The graphical user interfaces may be formed by a graph, a text, an icon, a video, or any combination thereof. The display unit 640 may include a display panel 640a. Optionally, the display panel 640a may be configured by using a liquid crystal display (LCD), an organic light-emitting diode (OLED), or the like. Further, the touch-sensitive surface 630a may cover the display panel 640a. After detecting a touch operation on or near the touch-sensitive surface 630a, the touch-sensitive surface 630a transfers the touch operation to the processor 680, so as to determine the type of the touch event. Then, the processor 680 provides a corresponding visual output on the display panel 640a according to the type of the touch event. Although, in FIG. 6, the touch-sensitive surface 630a and the display panel 640a are used as two separate parts to implement input and output functions, in some embodiments, the touch-sensitive surface 630a and the display panel 640a may be integrated to implement the input and output functions.

[0056] The terminal 600 may further include at least one sensor 650, such as an optical sensor, a motion sensor, and other sensors. Optionally, the optical sensor may include an ambient light sensor and a proximity sensor. The ambient light sensor can adjust luminance of the display panel 640a according to brightness of the ambient light. The proximity sensor may switch off the display panel 640a and/or backlight when the terminal 600 is moved to the ear. As one type of motion sensor, a gravity acceleration sensor can detect magnitude of accelerations in various directions (generally on three axes), may detect magnitude and a direction of the gravity when static, and may be applied to an application that recognizes the attitude of the mobile phone (for example, switching between landscape orientation and portrait orientation, a related game, and magnetometer attitude calibration), a function related to vibration recognition (such as a pedometer and a knock), and the like. Other sensors, such as a gyroscope, a barometer, a hygrometer, a thermometer, and an infrared sensor, which may be configured in the terminal 600 are not further described herein.

[0057] The audio circuit 660, a loudspeaker 660a, and a microphone 660b may provide audio interfaces between the user and the terminal 600. The audio circuit 660 may convert received audio data into an electric signal and transmit the electric signal to the loudspeaker 660a. The loudspeaker 660a converts the electric signal into a sound signal for output. On the other hand, the microphone 660b converts a collected sound signal into an electric signal. The audio circuit 660 receives the electric signal and converts the electric signal into audio data, and outputs the audio data to the processor 680 for processing. Then, the processor 680 sends the audio data to, for example, another terminal by using the RF circuit 610, or outputs the audio data to the memory 620

for further processing. The audio circuit 660 may further include an earplug jack, so as to provide communication between a peripheral earphone and the terminal 600.

[0058] To implement wireless communication, the terminal may be configured with a wireless communications unit 670. The wireless communications unit 670 may be a WiFi module. WiFi is a short distance wireless transmission technology. The terminal 600 may help, by using the wireless communications unit 670, the user to receive and send e-mails, browse a webpage, and access streaming media, and so on, which provides wireless broadband Internet access for the user. Although FIG. 6 shows the wireless communications unit 670, it may be understood that the wireless communications unit is not a necessary component of the terminal 600, and when required, the wireless communications unit may be omitted as long as the scope of the essence of the present disclosure is not changed.

[0059] The processor 680 is the control center of the terminal 600, and is connected to various parts of the mobile phone by using various interfaces and lines. By running or executing the software program and/or module stored in the memory 620, and invoking data stored in the memory 620, the processor 680 performs various functions and data processing of the terminal 600, thereby performing overall monitoring on the mobile phone. Optionally, the processor 680 may include one or more processing cores. Preferably, the processor 680 may integrate an application processor and a modem. The application processor mainly processes an operating system, a user interface, an application program, and the like. The modem mainly processes wireless communication. It may be understood that, the foregoing modem may also not be integrated into the processor 680.

[0060] The terminal 600 further includes the power supply 690 (such as a battery) for supplying power to the components. Preferably, the power supply may be logically connected to the processor 680 by using a power management system, thereby implementing functions such as charging, discharging, and power consumption management by using the power management system. The power supply 690 may further include one or more of a direct current or alternating current power supply, a re-charging system, a power failure detection circuit, a power supply converter or inverter, a power supply state indicator, and any other components.

[0061] Although not shown in the figure, the terminal 600 may further include a camera, a Bluetooth module, and the like, which are not further described herein.

[0062] An optional structure of the terminal 600 is given above with reference to FIG. 6. One or more modules are stored in the memory and configured to be executed by one or more processors. The one or more modules have the following functions:

reading an import section in an executable file;

determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and

notifying, when an instruction for running the executable file is received, that the DLL file is lost, and calling an installation package of the DLL file to repair the DLL file.

[0063] The determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost includes:

searching, according to the DLL file information recorded in the import section, the pre-established correspondence between a DLL file and a save path to obtain a save path of each DLL file recorded in the import section;

determining, for the save path of each DLL file obtained through searching, whether there is a corresponding DLL file saved under the save path; and

determining, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost.

[0064] Before the calling an installation package of the DLL file to repair the DLL file, the functions further include:

ending a current running process of the executable file.

[0065] The calling an installation package of the DLL file to repair the DLL file includes:

querying a user whether to repair the DLL file; and

calling, after the user gives a confirmation, the installation package of the DLL file to repair the DLL file.

[0066] The one or more modules may further have the following functions:

determining, when the instruction for running the executable file is received again, whether repairing of the DLL file is completed; and

responding, if the repairing of the DLL file is completed, to the instruction that is received again and running the executable file; and

refusing, if the repairing of the DLL file is not completed, to respond to the instruction that is received again, or responding to the instruction that is received again after the repairing is completed.

[0067] With the foregoing terminal provided in this embodiment, an import section in an executable file is read; it is determined, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and it is notified, when an instruction for running the executable file is received, that the DLL file is lost, and an installation package of the DLL file is called to repair the DLL file. Therefore, intelligent awareness of loss of the DLL file is achieved, and repairing is automatically performed after a user is notified, thereby facilitating repairing of the DLL file; the user does not need to manually perform processes such as downloading and installation, thereby reducing a requirement on the user, achieving an easy operation, and facilitating use for the user. Moreover, during running of the executable file, loss of the DLL file may be determined without the need of performing an operation of loading the DLL file to an address space, and pre-determining is completed before the running, thereby improving repairing efficiency.

Embodiment 5

[0068] This embodiment of the present invention provides a non-volatile readable storage medium. The storage medium stores one or more modules. When the one or more modules are applied in a device, the device is enabled to execute instructions including the following steps:

reading an import section in an executable file;

determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and

notifying, when an instruction for running the executable file is received, that the DLL file is lost, and calling an installation package of the DLL file to repair the DLL file.

[0069] The determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost includes:

searching, according to the DLL file information recorded in the import section, the pre-established correspondence between a DLL file and a save path to obtain a save path of each DLL file recorded in the import section;

determining, for the save path of each DLL file obtained through searching, whether there is a corresponding DLL file saved under the save path; and

determining, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost.

[0070] Before the calling an installation package of the DLL file to repair the DLL file, the following step is further included:

ending a current running process of the executable file.

[0071] The calling an installation package of the DLL file to repair the DLL file includes:

querying a user whether to repair the DLL file; and

calling, after the user gives a confirmation, the installation package of the DLL file to repair the DLL file.

[0072] The foregoing device may further execute instructions including the following steps:

determining, when the instruction for running the executable file is received again, whether repairing of the DLL file is completed; and

responding, if the repairing of the DLL file is completed, to the instruction that is received again and running the executable file; and

refusing, if the repairing of the DLL file is not completed, to respond to the instruction that is received again, or responding to the instruction that is received again after the repairing is completed.

[0073] With the foregoing non-volatile readable storage medium provided in this embodiment, an import section in an executable file is read; it is determined, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and it is notified, when an instruction for running the executable file is received, that the DLL file is lost, and an installation package of the DLL file is called to repair the DLL file. Therefore, intelligent awareness of loss of the DLL file is achieved, and repairing is automatically performed after a user is notified, thereby facilitating repairing of the DLL file; the user does not need to manually perform processes such as downloading and installation, thereby reducing a requirement on the user, achieving an easy operation, and facilitating use for the user. Moreover, during running of the executable file, loss of the DLL file may be determined without the need of performing an operation of loading the DLL file to an address space, and pre-determining is completed before the running, thereby improving repairing efficiency.

[0074] A person of ordinary skill in the art may understand that all or some of the steps of the foregoing embodiments may be implemented by using hardware, or may be implemented by a program instructing relevant hardware. The program may be stored in a non-volatile readable

storage medium. The non-volatile readable storage medium may be a read-only memory, a magnetic disk, an optical disc, or the like.

[0075] The foregoing descriptions are merely preferred embodiments of the present invention, but are not intended to limit the present disclosure. Any modification, equivalent replacement, or improvement made within the spirit and principle of the present disclosure shall fall within the protection scope of the present disclosure.

CLAIMS

What is claimed is:

1. A method for repairing a dynamic link library (DLL) file, comprising:

reading an import section in an executable file;

determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and

notifying, when an instruction for running the executable file is received, that the DLL file is lost, and calling an installation package of the DLL file to repair the DLL file.

2. The method according to claim 1, wherein the determining, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost comprises:

searching, according to the DLL file information recorded in the import section, the pre-established correspondence between a DLL file and a save path to obtain a save path of each DLL file recorded in the import section;

determining, for the save path of each DLL file obtained through searching, whether there is a corresponding DLL file saved under the save path; and

determining, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost.

3. The method according to claim 1, before the calling an installation package of the DLL file to repair the DLL file, further comprising:

ending a current running process of the executable file.

4. The method according to claim 1, wherein the calling an installation package of the DLL file to repair the DLL file comprises:

querying a user whether to repair the DLL file; and

calling, after the user gives a confirmation, the installation package of the DLL file to repair the DLL file.

5. The method according to claim 1, wherein the method further comprises:

determining, when the instruction for running the executable file is received again, whether repairing of the DLL file is completed; and

responding, if the repairing of the DLL file is completed, to the instruction that is received again and running the executable file; and

refusing, if the repairing of the DLL file is not completed, to respond to the instruction that is received again, or responding to the instruction that is received again after the repairing is completed.

6. An apparatus for repairing a DLL file, comprising:

a reading module, configured to read an import section in an executable file;

a determining module, configured to determine, according to DLL file information recorded in the import section and a pre-established correspondence between a DLL file and a save path, that a DLL file is lost; and

a repairing module, configured to notify, when an instruction for running the executable file is received, that the DLL file is lost, and call an installation package of the DLL file to repair the DLL file.

7. The apparatus according to claim 6, wherein the determining module comprises:

a searching unit, configured to search, according to the DLL file information recorded in the import section, the pre-established correspondence between a DLL file and a save path to obtain a save path of each DLL file recorded in the import section;

a judging unit, configured to determine, for the save path of each DLL file obtained through searching, whether there is a corresponding DLL file saved under the save path; and

a determining unit, configured to determine, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost.

8. The apparatus according to claim 6, wherein the apparatus further comprises:

an ending module, configured to end a current running process of the executable file before the repairing module calls the installation package of the DLL file to repair the DLL file.

9. The apparatus according to claim 6, wherein the repairing module comprises:

a querying unit, configured to query a user whether to repair the DLL file; and

a repairing unit, configured to call, after the user gives a confirmation, the installation package of the DLL file to repair the DLL file.

10. The apparatus according to claim 6, wherein the apparatus further comprises:

a responding module, configured to determine, when the instruction for running the executable

file is received again, whether repairing of the DLL file is completed; and respond, if the repairing of the DLL file is completed, to the instruction that is received again and run the executable file; and refuse, if the repairing of the DLL file is not completed, to respond to the instruction that is received again, or respond to the instruction that is received again after the repairing is completed.

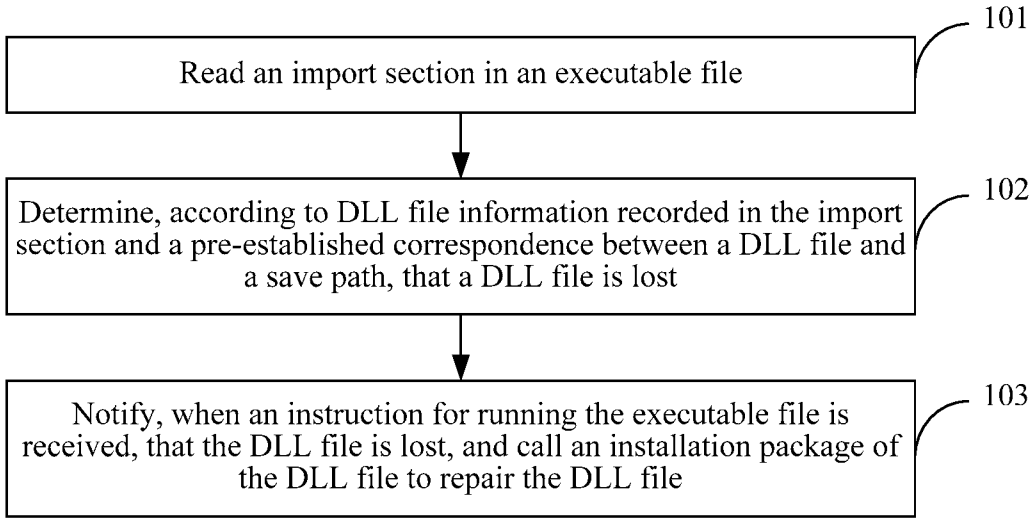


FIG. 1

```
Microsoft (R) COFF/PE Dumper Version 8.00.50727.762
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file C:\Windows\System32\notepad.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    ADVAPI32.dll
               1001000 Import Address Table
               100A234 Import Name Table
              FFFFFFFF time date stamp
              FFFFFFFF Index of first forwarder reference

      77C71C82    27E RegSetValueExW
      77C7BCD5    26E RegQueryValueExW
      77C7BED4    230 RegCloseKey
      77C71CC0    23C RegCreateKeyW
      77C7BEC4    261 RegOpenKeyExW
      77C7BEE4    180 IsTextUnicode
      77C79A61     57 CloseServiceHandle
      77C6D225    224 QueryServiceConfigW
      77C6D20D    1FB OpenServiceW
      77C6D1F5    1F9 OpenSCManagerW

    KERNEL32.dll
               100102C Import Address Table
               100A260 Import Name Table
              FFFFFFFF time date stamp
```

FIG. 2

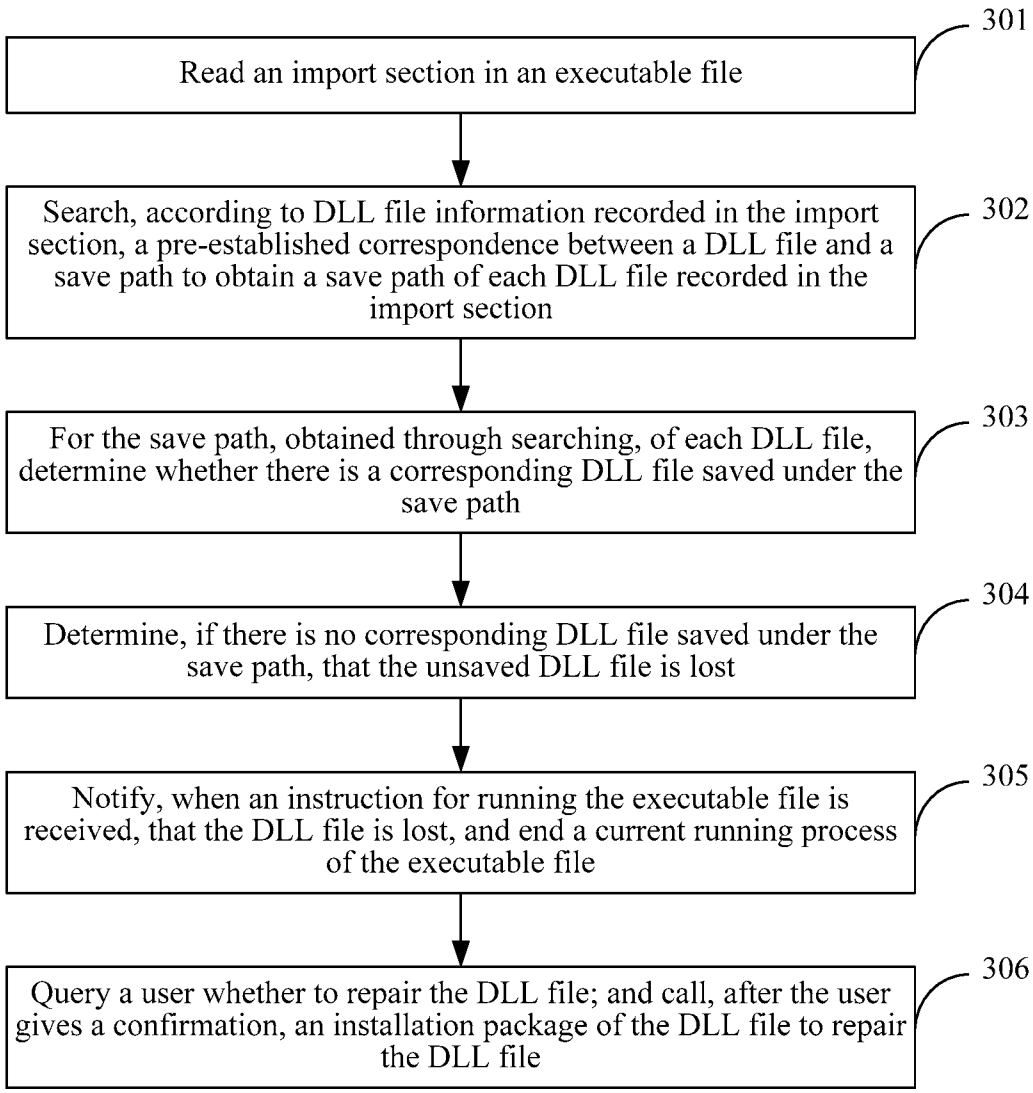

Read an import section in an executable file
301
Search, according to DLL file information recorded in the import section, a pre-established correspondence between a DLL file and a save path to obtain a save path of each DLL file recorded in the import section
302
For the save path, obtained through searching, of each DLL file, determine whether there is a corresponding DLL file saved under the save path
303
Determine, if there is no corresponding DLL file saved under the save path, that the unsaved DLL file is lost
304
Notify, when an instruction for running the executable file is received, that the DLL file is lost, and end a current running process of the executable file
305
Query a user whether to repair the DLL file; and call, after the user gives a confirmation, an installation package of the DLL file to repair the DLL file
306

FIG. 3

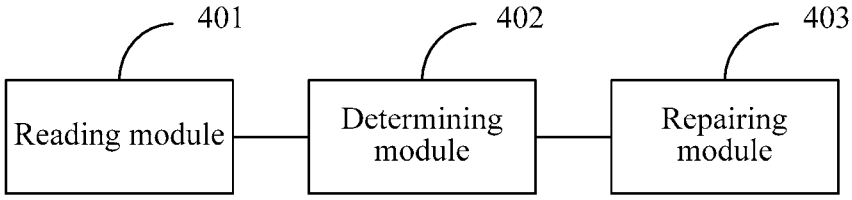

401
402
403
Reading module
Determining module
Repairing module

FIG. 4

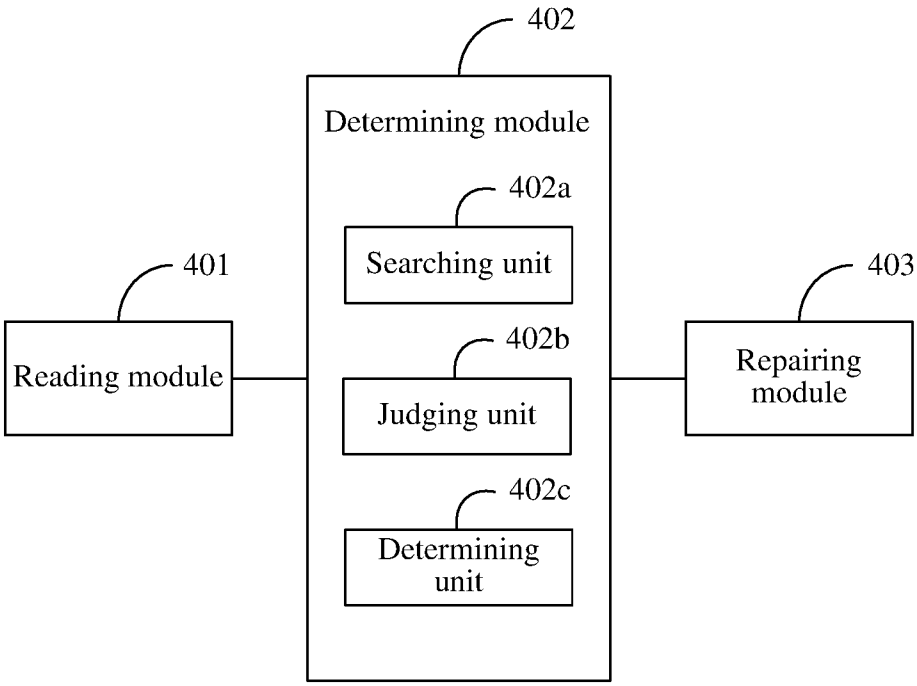

402
Determining module
402a
Searching unit
401
Reading module
402b
Judging unit
403
Repairing module
402c
Determining unit

FIG. 5

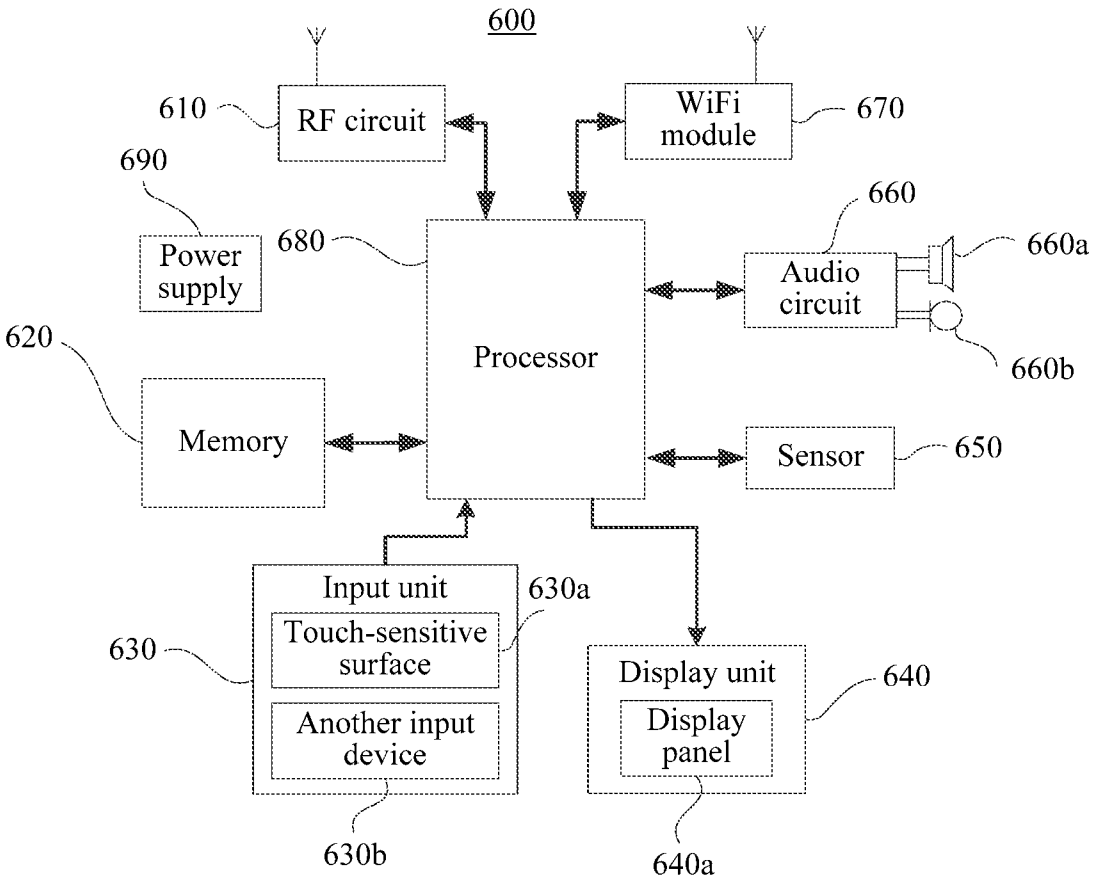

600
610
RF circuit
WiFi module
670
690
Power supply
680
Processor
660
Audio circuit
660a
660b
620
Memory
Sensor
650
Input unit
630a
630
Touch-sensitive surface
Another input device
630b
Display unit
640
Display panel
640a

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2015/076231

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/00(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT, CNKI, WPI, EPODOC, IEEE, GOOGLE: dynamic w link w library, repair+, import, section, execut+, record+, preestablish, correspondence, path, lost, table, inject, DLL

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2009049214 A1 (HONG FU JIN PRECISION INDUSTRYSHENZHENCO., LTD.ET AL) 19 February 2009 (2009-02-19) abstract description, paragraphs [0007]-[0012]	1-10
A	CN 102402455 A (PEKING UNIVERSITY FOUNDER GROUP CO., LTD. ET AL.) 04 April 2012 (2012-04-04) the whole document	1-10
A	CN 103530118 A (GUANGZHOU CHINADUO NETWORK TECHNOLOGY CO., LTD.) 22 January 2014 (2014-01-22) the whole document	1-10
A	US 2002112058 A1 (MICROSOFT CORPORATION) 15 August 2002 (2002-08-15) the whole document	1-10

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 June 2015	**01 July 2015**

Name and mailing address of the ISA/CN	Authorized officer
STATE INTELLECTUAL PROPERTY OFFICE OF THE P.R.CHINA **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088, China**	**TIAN,Minli**
Facsimile No. **(86-10)62019451**	Telephone No. **(86-10)62414425**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2015/076231

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2009049214 A1	19 February 2009	CN 101369244 A	18 February 2009
		JP 2009087327 A	23 April 2009
CN 102402455 A	04 April 2012	None	
CN 103530118 A	22 January 2014	None	
US 2002112058 A1	15 August 2002	US 2006123125 A1	08 June 2006
		US 2006168159 A1	27 July 2006
		US 2006184661 A1	17 August 2006
		US 2006123124 A1	08 June 2006