

APPLICATION CRASH HANDLING METHOD AND APPARATUS, ELECTRONIC DEVICE AND READABLE STORAGE MEDIUM

[0001] The present disclosure claims the priority to the Chinese patent application with the filing No. 2024101725789 filed on February 7, 2024, and entitled “APPLICATION CRASH HANDLING METHOD AND APPARATUS, ELECTRONIC DEVICE AND READABLE STORAGE MEDIUM”, and the contents disclosed in the above Chinese patent application are incorporated herein by reference in entirety as a part of the present disclosure.

TECHNICAL FIELD

[0002] The present disclosure relates to the technical field of computers, and specifically to an application crash handling method and apparatus, an electronic device and a readable storage medium.

BACKGROUND ART

[0003] If an application crashes after a switch is turned on in a startup phase, the application cannot obtain new configuration from a configuration platform. If the crash is not promptly handled, it may cause repeated crashes of the application at startup, and might lead to serious user churn in more severe cases.

SUMMARY

[0004] The present disclosure aims at providing an application crash handling method and apparatus, an electronic device and a readable storage medium, which can address frequent crashes of an application in a startup phase.

[0005] In the first aspect, the present disclosure provides an application crash handling method, including: acquiring flag bit information of an application at startup of the application, where the flag bit information is read at the startup of the application through preset event tracking; and if the flag bit information of the application is a first value, switching the application from a current configuration to a historical configuration, and executing the historical configuration, where the flag bit information being the first value characterizes that the application crashed in a last startup phase.

[0006] In the method of embodiments of the present disclosure, by setting the flag bit information, whether a crash occurred in the last startup can be found at the startup, and if the crashed occurred, the configuration can be promptly enabled and switched, and the historical

configuration is executed, so that the probability of recurrent crash can be reduced, thus improving user experience.

[0007] In optional embodiments, the method further includes: identifying, through application monitoring, a first functional module that caused the application to crash in the last startup phase; and sending a handling request for the first functional module to a configuration platform, where the handling request is configured to instruct the configuration platform to cease issuing a configuration item corresponding to the first functional module.

[0008] In the above embodiments, the functional module that crashed can be promptly disabled, and the functional module can be prevented from causing recurrent crash, thus reducing a crash frequency of the application and also improving user experience.

[0009] In optional embodiments, the method further includes: for a second functional module corresponding to a configuration obtained by the application that crashed in the last startup phase, executing a latest configuration of the second functional module after waiting for a limited time limit, where the second functional module is an updated functional module corresponding to the configuration obtained in the last startup phase, and the second functional module is different from the first functional module.

[0010] In optional embodiments, the limited time limit includes: a real-time timestamp and a limited duration.

[0011] In optional embodiments, the limited time limit includes: a countdown timer.

[0012] In the above embodiments, the limited time limit is set for the latest configuration during potential crash, thereby restricting activation of the latest configuration to implement an operation for the crash during the startup of the application, and preventing recurrent crash of the application.

[0013] In optional embodiments, the application is restarted upon identification of a crash of the application through application monitoring at the startup of the application.

[0014] In the above embodiments, at the startup of the application, if the crash is detected, the application can be automatically restarted, so that situations that the user experiences crashes can be reduced.

[0015] In optional embodiments, the method further includes: acquiring a latest configuration of the application if the flag bit information of the application is a second value, where the flag bit information being the second value characterizes that the application did no crash in the last startup phase.

[0016] In the above embodiments, even if the flag bit information characterizes that the application did no crash last time, whether the latest configuration is restricted still can be

identified, so that the probability of crashes can be reduced, and the user experience can be improved.

[0017] In optional embodiments, the method further includes: updating the flag bit information to the first value if a crash of the application is detected, at the startup of the application; and updating the flag bit information to the second value if the application does not crash and the latest configuration of the application is obtained from the configuration platform at the startup of the application.

[0018] In optional embodiments, a configuration file of the application is read locally if the flag bit information of the application is not obtained at the startup of the application, and the configuration file is executed.

[0019] In optional embodiments, the method further includes: reading a third functional module from a configuration file of the application, where the third functional module is a functional module undergoing update in the application, and the third functional module includes the second functional module, and excludes the first functional module; and loading a historical configuration of the third functional module and executing the historical configuration of the third functional module, if current time falls within a limited time limit of the third functional module, where the limited time limit of the third functional module is a time limit set when acquiring a current configuration of the third functional module.

[0020] In the second aspect, the present disclosure provides an application crash handling apparatus, including an acquisition module, configured to acquire flag bit information of an application at startup of the application, where the flag bit information is read at the startup of the application through preset event tracking; and an execution module, configured to switch the application from a current configuration to a historical configuration if the flag bit information of the application is a first value, and execute the historical configuration, where the flag bit information being the first value characterizes that the application crashed in a last startup phase.

[0021] In the third aspect, the present disclosure provides an electronic device, including: a processor and a memory, where the memory stores machine-readable instructions executable by the processor, and when the electronic device runs, and when the machine-readable instructions, when executed by the processor, implements steps of the method according to any one of the preceding embodiments are implemented.

[0022] In the fourth aspect, the present disclosure provides a computer-readable storage medium, storing a computer program, where the computer program, when executed by a processor, implements steps of the method according to any one of the preceding embodiments.

BRIEF DESCRIPTION OF DRAWINGS

[0023] In order to more clearly illustrate technical solutions of embodiments of the present disclosure, drawings which need to be used in the embodiments will be briefly introduced below. It should be understood that the drawings merely show some embodiments of the present disclosure, and thus should not be considered as limitation to the scope. Those ordinarily skilled in the art still could obtain other relevant drawings according to these drawings, without using any inventive efforts.

[0024] FIG. 1 is a schematic diagram of a runtime environment of an application crash handling method provided by embodiments of the present disclosure;

[0025] FIG. 2 is a block diagram of an electronic device provided by embodiments of the present disclosure;

[0026] FIG. 3 is a flowchart of the application crash handling method provided by embodiments of the present disclosure; and

[0027] FIG. 4 is a schematic diagram of functional modules of an application crash handling apparatus provided by embodiments of the present disclosure.

DETAILED DESCRIPTION OF EMBODIMENTS

[0028] Technical solutions in embodiments of the present disclosure will be described below in conjunction with drawings in the embodiments of the present disclosure.

[0029] It should be noted that like reference signs and letters represent like items in the following drawings, and thus, once a certain item is defined in one drawing, it is unnecessary to further define and explain the same in subsequent drawings. Moreover, in the description of the present disclosure, the terms such as “first” and “second” are merely used for distinguishing the description, but should not be construed as indicating or implying importance in the relativity.

[0030] At present, many new functions will be added to an application in an update iteration process. Due to uncertainty of new functions and often inability to test them comprehensively, certain a bug may remain undetected, thus bringing the bug online, thereby potentially causing an online crash (Crash) issue, and negatively affecting user experience. Given that a task executed in a startup phase of an application is critical foundation library initialization, a crash occurring in this phase is highly likely to cause subsequent crash or function exception, and if it is left unhandled, it will cause repeated crashes of the application at startup, thus bringing very bad influence and experience to the user.

[0031] Based on this, a current approach to handle crashes of the application in the startup phase is to add trycatch protection in a code. Trycatch is an error handling mechanism for capturing and handling exceptions in programs. A code that may trigger exceptions is written in a try block. In the event of an exception in the try block, the program will jump to a catch block. A code for handling exceptions, such as typing error message or executing other operations, is written in the catch block. This handling mechanism can prevent program crashes and provide better error handling and user experience. However, the trycatch protection is only applicable to protection of Java-layer code.

[0032] In view of this, the present disclosure provides an application crash handling method and apparatus, an electronic device and a readable storage medium, which can reduce a frequency of occurrence of crashes of an application at startup. Further, there are enabling/disabling options for various functions in a configuration platform. When an online Crash monitoring platform detects Crash, the configuration platform promptly issues a disabling configuration for a related function, and by pulling the latest configuration and switching to an original logic, an application end avoids Crash caused by executing a new configuration of the function, thus reducing the probability of Crash occurring at startup of the application next time.

[0033] In order to facilitate understanding embodiments, a runtime environment of an application crash handling method disclosed in the embodiments of the present disclosure is first introduced in detail.

[0034] As shown in FIG. 1, it is a schematic diagram of the runtime environment of the application crash handling method provided by embodiments of the present disclosure. The runtime environment of the application crash handling method includes a server 110 and an electronic device 120. A configuration platform runs in the server 110, and an application runs in the electronic device 120. The server 110 communicates with one or more electronic devices 120 over a network for data communication or interaction. The server 110 may be a network server, a database server, and the like. The electronic device 120 may be a personal computer (PC), a tablet computer, a smart mobile phone, a personal digital assistant (PDA), etc.

[0035] As shown in FIG. 2, it is a block diagram of the electronic device. The electronic device 120 may include a memory 121, a storage controller 122, a processor 123, a peripheral interface 124, an input-output unit 125, and a display unit 126. Those ordinarily skilled in the art could understand that the structure shown in FIG. 2 is merely schematic and does not limit a structure of the electronic device 120. For example, the electronic device 120 further may

include more or fewer components than that shown in FIG. 2, or a different configuration than that shown in FIG. 2.

[0036] Various elements of the memory 121, the storage controller 122, the processor 123, the peripheral interface 124, the input-output unit 125 and the display unit 126 are electrically connected to each other directly or indirectly to realize data transmission or interaction. For example, these elements may be electrically connected to each other via one or more communication buses or signal lines. The above processor 123 is configured to execute executable modules stored in the memory.

[0037] In the above, the memory 121 can be, but is not limited to, a random access memory (RAM for short), a read-only memory (ROM for short), a programmable read-only memory (PROM for short), an erasable programmable read-only memory (EPROM for short), an electrically erasable programmable read-only memory (EEPROM for short), and the like. In the above, the memory 121 is configured to store program, the processor 123 executes the program upon receiving an execution instruction. The method executed by the electronic device 120 defined by a process disclosed in any embodiment of the present disclosure can be applied to the processor 123, or implemented by the processor 123.

[0038] The above processor 123 can be an integrated circuit chip with signal processing capability. The above processor 123 can be a general-purpose processor, including a central processing unit (CPU for short), a network processor (NP for short), etc., and can also be a digital signal processor (DSP for short), an application specific integrated circuit (ASIC for short), a field-programmable gate array (FPGA for short) or other programmable logic devices, discrete gates, transistor logic devices, or discrete hardware components that can realize or implement various methods, steps and logic blocks disclosed in the embodiments of the present disclosure. The general-purpose processor may be a microprocessor or the processor may also be any conventional processor and so on.

[0039] The above peripheral interface 124 couples various input/output devices to the processor 123 and the memory 121. In some embodiments, the peripheral interface 124, the processor 123 and the memory controller 122 may be realized in a single chip. In some other embodiments, they can be implemented separately by independent chips.

[0040] The above input-output unit 125 is configured for the user to input data. The input-output unit 125 may be, but is not limited to, a mouse, a keyboard, and the like.

[0041] The above display unit 126 provides an interactive interface (for example, a user operation interface) between the electronic device 120 and the user or is configured to display image data for user reference. In the present embodiment, the display unit may be a liquid

crystal display or a touch display. If the display unit is a touch display, it may be a capacitive touch screen, a resistive touch screen or the like that supports single-touch and multi-touch operations. Supporting single-touch and multi-touch operations means that the touch display can sense a touch operation simultaneously generated at one or more positions on the touch display, and transfer the sensed touch operations to the process for calculation and processing.

[0042] The electronic device 120 in the present embodiment can be configured to execute various steps in various methods provided by the embodiments of the present disclosure. An implementation process of the application crash handling method is described below through several embodiments.

[0043] Referring to FIG. 3, it is a flowchart of the application crash handling method provided by embodiments of the present disclosure. The application crash handling method in the present embodiment can be applied to an electronic device in which an application runs. A specific flow shown in FIG. 3 is described in detail below.

[0044] Step 210, acquiring flag bit information of the application at startup of the application.

[0045] In the above, the flag bit information is read at the startup of the application through preset event tracking.

[0046] Optionally, the preset event tracking may be located at a starting position of a code of the application. For example, the starting position of the code may represent a position of a first line of code when the application is executed.

[0047] The flag bit information being a first value characterizes that the application crashed at last startup.

[0048] Values of the first value and a second value of the flag bit information can be set as required, and the first value and the second value can be specifically set according to usage habits. For example, the first value can take a value flag1, and the second value can take a value flag2. For another example, the first value can also take a value 1, and the second value can also take a value 0.

[0049] In other cases, if the flag bit information of the application is null, or the application has no flag bit information, it may indicate that the application currently is started in new user equipment, and then a local default configuration file of the current device can be directly read.

[0050] Optionally, if the flag bit information of the application is not assigned a value, an initial value can be set for the flag bit information of the application, where the initial value may be a value distinct from the first value and the second value. For example, the initial value may take a value flat0.

[0051] Optionally, if the flag bit information of the application is null, the flag bit information can be set with a value based on whether the application crashes at this startup, where if no crash occurs, the flag bit information can be set with the second value; and if a crash occurs, the flag bit information can be set with the first value.

[0052] Step 220, if the flag bit information of the application is the first value, switching the application from a current configuration to a historical configuration, and executing the historical configuration.

[0053] If the flag bit information of the application is the first value, it may indicate that a crash occurred at the last startup of the application in the current device. In order to avoid a crash triggered by acquiring and executing a latest configuration this time, the historical configuration of the application can be executed at this startup.

[0054] In the above, the historical configuration can represent a configuration that enables successful startup and successful running of the application in the past. Exemplarily, the historical configuration can be a configuration used for running of the application before the crash of the application is caused.

[0055] In the present embodiment, at the startup of the application, upon detection of a crash of the application and identification of the flag bit information, the flag bit information can be updated to the first value.

[0056] Given that crash handling can be performed based on the flag bit information after the flag bit information is identified, update timing of the flag bit information is determined to be after the identification of the flag bit information.

[0057] At the startup of the application, if the application does not crash and the latest configuration of the application is obtained from the configuration platform, the flag bit information is updated to the second value. The second value can be used to mark that the startup of the application does not crash.

[0058] By the above setting method, when it has been detected that the application crashed last time, in order to avoid repeated occurrence of crash, the latest configuration of the application may not be executed, and for the user, the repeated occurrence of crash at the startup of the application can be avoided, and a customer churn rate caused by the application crash can also be reduced.

[0059] Given that a functional module causing the crash may be persistent and will not disappear due to restart of the application, for this situation, the application crash handling method provided by the present disclosure can further include step 230 and step 240.

[0060] Step 230: identifying, through application monitoring, a first functional module that caused the application to crash in a last startup phase.

[0061] Exemplarily, a startup situation of the application can be monitored through the application monitoring. When the monitoring detects that it is the first functional module that caused the application to crash, step 240 is executed.

[0062] Exemplarily, the first functional module may be one functional module in the application, for example, the first functional module may be a login functional module, a data acquisition functional module, etc.

[0063] Step 240, sending a handling request for the first functional module to the configuration platform.

[0064] The handling request is configured to instruct the configuration platform to cease issuing a configuration item associated with the first functional module. For example, upon receiving the handling request for the first functional module, the configuration platform can disable the first functional module, so that when the electronic device sends a configuration acquisition request for the application next time, a configuration item of corresponding update of the first functional module is not issued any longer.

[0065] Optionally, upon monitoring the crash, “disable” can be selected in the configuration platform.

[0066] By the above method, the functional module that caused crash can be promptly disabled, and the configuration platform will no longer issue an updated configuration of this functional module, thereby reducing the occurrence of frequent crashes caused by exception of a single functional module.

[0067] If the application crashes at startup, it is likely attributed to exceptions existing in updated configurations of some functional modules, or delayed activation of updated configurations of the functional modules. Based on this, the application crash handling provided by embodiments of the present disclosure can further include: for a second functional module associated with a configuration obtained by the application that crashed in the last startup phase, executing a latest configuration of the second functional module after waiting for a limited time limit.

[0068] In the above, the second functional module is a corresponding updated functional module in the configuration obtained in the last startup phase, and the second functional module is different from the first functional module.

[0069] Exemplarily, for the application that crashed at the last startup, the second functional module associated with the latest configuration obtained at the last startup can be set with the limited time limit.

[0070] The limited time limit can be used to define effective timing of the updated configuration. If timing of the updated configuration of the functional module still falls within the limited time limit, it can be indicated that the updated configuration of the functional module has not yet taken effect, and then the updated configuration of the functional module can be used after the updated configuration of the functional module takes effect.

[0071] In an embodiment, the limited time limit can be enforced by a real-time timestamp and a limited duration. The above step of setting the limited time limit for the functional module associated with the current latest configuration of the application that crashed may include: setting a real-time timestamp and a limited duration for the second functional module of the application.

[0072] For the second functional module, when the second functional module generates the current latest configuration or obtains the current latest configuration, the real-time timestamp and the limited duration are set for the second functional module.

[0073] The real-time timestamp may be real-time timing for generating the current latest configuration or obtaining the current latest configuration.

[0074] The limited duration can be set as required, for example, the limited duration is set to be 5 minutes, 6 minutes, 7 minutes, etc.

[0075] When the second functional module needs to be used and when the functional module of the application is read from a configuration file, whether the functional module has a real-time timestamp is identified first, and upon identification of the real-time timestamp of the second functional module, whether a time difference between current time and the real-time timestamp is greater than the limited duration is determined, where if the time difference between the current time and the real-time timestamp is greater than the limited duration, the updated configuration of the functional module can be enabled, and if the time difference between the current time and the real-time timestamp is not greater than the limited duration, the updated configuration of the second functional module may not be enabled, and the historical configuration of the second functional module is still used, thus reducing the probability of crash in the startup phase caused by the second functional module.

[0076] In an embodiment, the limited time limit can be implemented by a countdown timer. The above step of setting the limited time limit for the functional module associated with the current latest configuration of the application that crashed may include setting a countdown

timer for an updated functional module associated with the current latest configuration of the application.

[0077] For the second functional module, upon obtaining the latest configuration of the second functional module, the countdown timer is set for the target functional module.

[0078] At the startup of the application, when reading the functional module of the application from the configuration file, whether the functional module has a countdown timer is identified first. Upon identification of the countdown timer of the functional module, whether the countdown timer has ended is then determined, where if the countdown timer has ended, the updated configuration of the functional module is enabled, and if the countdown timer has not ended, the updated configuration of the functional module is not enabled, and the historical configuration of this module is still used, thus reducing crashes caused by the functional module in the startup phase.

[0079] By setting the limited time limit, the effective timing of the updated configuration of the functional module in the application can be delayed, thereby reducing the crashes caused by delayed activation of the updated configuration in the startup phase.

[0080] If the application crashed at the last startup, the probability of crashes at subsequent startup can be reduced through the above processing, thereby reducing the occurrence of crashes of the application. It can be understood that, if no crash occurred at the last startup, the configuration file can be obtained according to a conventional procedure. Based on this, the method provided by embodiments of the present disclosure further includes: step 250, acquiring the latest configuration of the application if the flag bit information of the application is the second value.

[0081] In the above, the flag bit information of the application being the second value characterizes that the application did no crash at the last startup, where no crash at the last startup of the application can indicate that the application successfully obtained the latest configuration from the configuration platform at the last startup.

[0082] Optionally, each time any functional module of the application is enabled, whether the functional module has a limited time limit can be determined first. Based on this, the application crash handling method provided by embodiments of the present disclosure may further include step 260 to step 270.

[0083] Step 260, reading a third functional module from the configuration file of the application.

[0084] In the above, the third functional module refers to a functional module undergoing update in the application.

[0085] Given that the first functional module in the preceding may not be executed temporarily, the third functional module may include the above second functional module, and excludes the above first functional module.

[0086] If the third functional module is set with the limited time limit and current time falls within the limited time limit, it can be indicated that the latest configuration of the third functional module has not taken effect yet, and then step 270 can be executed.

[0087] Step 270, if the current time falls within the limited time limit of the third functional module, loading a historical configuration of the third functional module, and executing the historical configuration of the third functional module.

[0088] In the present embodiment, the limited time limit of the third functional module is a time limit set when acquiring the current configuration of the third functional module.

[0089] Optionally, the historical configuration of the third functional module may be a configuration of the third functional module used at last normal startup of the application.

[0090] In some cases, some unknown crashes may occur to the application in the startup phase, when the above handling method cannot solve the crashes, the application crash handling method provided by embodiments of the present disclosure may further include: restarting the application if a crash of the application is identified by the application monitoring, at the startup of the application.

[0091] Optionally, upon detection of a program crash by the application monitoring, the number of restart times can be set as required, and within the limit of the number of restart times, upon detection of the program crash, the application can be restarted. If the crash is still detected after the program crash is detected and the application is continuously restarted for the number of restart times, a prompt can be output to notify the user of failure of the application.

[0092] Optionally, a fault content existing in the application can also be output to prompt the user. For example, the fault content may be incorrect application version, incorrect data source, etc.

[0093] By automatically restarting the application, startup of the application can be realized without the user perceiving the crash of the application, only a slightly extended startup period can be perceived, and there is no need to purge application caches, uninstall and reinstall the application or carry out other complex handlings, thereby reducing the user churn rate.

[0094] In the method provided by embodiments of the present disclosure, by identifying the flag bit information and adaptively adjusting a startup policy of the application based on a rated value of the flag bit information, the situation where application cache clearing or uninstall and

reinstall are needed due to repeated startup crashes of the application can be alleviated, and the situation of user churn can be avoided. An operation of automatic restart can be executed after the occurrence of crash, making the crash virtually undetectable from a user perspective, except longer startup time than usual, thereby improving the user experience. For Native layer that cannot be protected by trycatch the solution provided by embodiments of the present disclosure adds a significantly effective protection mechanism, resulting in fewer startup crashes of the application. Moreover, all logics of the method provided by embodiments of the present disclosure can be completed in the application, without secondary backend modification, thereby reducing implementation difficulty. Furthermore, the method provided by embodiments of the present disclosure imposes less impact on time consumption and performance of the startup of the application.

[0095] Based on the same application concept, embodiments of the present disclosure further provide an application crash handling apparatus corresponding to the application crash handling method. As the principle of the apparatus in the embodiments of the present disclosure for solving the problem is similar to that of the preceding application crash handling method, for implementation of the apparatus in the present embodiment, reference can be made to the description in the embodiments of the above method, which will not be repeated herein.

[0096] Referring to FIG. 4, it is a schematic diagram of functional modules of the application crash handling apparatus provided by embodiments of the present disclosure. Various modules in the application crash handling apparatus in the present embodiment are configured to execute various steps in the above method embodiments. The application crash handling apparatus includes: an acquisition module 310 and an execution module 320, where contents of various modules are as follows: the acquisition module 310 is configured to acquire flag bit information of an application at startup of the application, where the flag bit information is read at the startup of the application through preset event tracking; and the execution module 320 is configured to switch the application from a current configuration to a historical configuration if the flag bit information of the application is a first value, and execute the historical configuration, where the flag bit information being the first value characterizes that the application crashed in a last startup phase.

[0097] In a possible embodiment, the application crash handling apparatus provided by the present embodiment can further include: a determination module, configured to identify a first functional module that caused the application to crash in the last startup phase through application monitoring; and a request module, configured to send a handling request for the first functional module to the configuration platform, where the handling request is configured

to instruct the configuration platform to cease issuing a configuration item associated with the first functional module.

[0098] In a possible embodiment, the application crash handling apparatus provided by the present embodiment can further include an execution module, configured to execute a latest configuration of a second functional module associated with a configuration obtained by the application that crashed in the last startup phase after waiting for a limited time limit for the second functional module, where the second functional module is a corresponding updated functional module in the configuration obtained in the last startup phase, and the second functional module is different from the first functional module.

[0099] In a possible embodiment, the application crash handling apparatus provided by the present embodiment can further include: a restarting module, configured to restart the application upon identification of the crash of the application through the application monitoring at the startup of the application.

[0100] In a possible embodiment, the application crash handling apparatus provided by the present embodiment can further include: acquiring the latest configuration of the application if the flag bit information of the application is the second value, where the flag bit information of being the second value characterizes that the application did no crash in the last startup phase.

[0101] In a possible embodiment, the application crash handling apparatus provided by the present embodiment can further include: an updating module, configured to update the flag bit information to the first value upon detection of a crash of the application at the startup of the application; and update the flag bit information to the second value if the application does not crash and the latest configuration of the application is obtained from the configuration platform at the startup of the application.

[0102] In a possible embodiment, the execution module of the application crash handling apparatus provided by the present embodiment can be further configured to read a configuration file of the application locally if the flag bit information of the application is not obtained at the startup of the application, and execute the configuration file.

[0103] In a possible embodiment, the application crash handling apparatus provided by the present embodiment further may include: a reading module, configured to read a third functional module from the configuration file of the application, where the third functional module refers to a functional module undergoing update in the application, and the third functional module includes the second functional module, and excludes the first functional module.

[0104] The execution module is configured to load a historical configuration of the third functional module and execute the historical configuration of the third functional module if current time falls within a limited time limit of the third functional module, where the limited time limit of the third functional module is a time limit set when acquiring the current configuration of the third functional module.

[0105] Besides, embodiments of the present disclosure further provide a computer-readable storage medium, where the computer-readable storage medium stores a computer program, and the computer program, when executed by the processor, implements the steps of the above application crash handling method in the above method embodiments.

[0106] A computer program product of the application crash handling method provided by the embodiments of the present disclosure includes a computer-readable storage medium in which a program code is stored, and instructions included in the program code can be used to execute the steps of the application crash handling method in the above method embodiments. Reference can be made to the above method embodiments for specific implementation, which will not be repeated herein.

[0107] In the several embodiments provided by the present disclosure, it should be understood that the apparatus and method disclosed can be implemented in other ways. The apparatus embodiments described above are merely illustrative. For example, flowcharts and block diagrams in the drawings show system structures, functions and operations that can be implemented in the apparatus, method and computer program products according to multiple embodiments of the present disclosure. In this regard, each block in the flowcharts or the block diagrams may represent a part of a module, a program segment or a code, which contains one or more executable instructions configured to realize a specified logical function. It also should be noted that in some implementation modes as substitutions, the functions marked in the blocks can also take effect in an order different than that marked in the drawings. For example, two continuous blocks actually can be executed in parallel or in a reverse order, which depends on the function involved. It also should be noted that each block or combination of the blocks in the block diagram and/or the flowchart can be realized by a dedicated hardware-based system used to execute a specified function or action, or can be realized by a combination of dedicated hardware and computer instructions.

[0108] Besides, various functional modules in various embodiments of the present disclosure can be integrated together to form one independent part, the various modules may also exist independently, or two or more modules may be integrated to form one independent part.

[0109] If the function is realized in a form of software functional module and is sold or used as an individual product, it may be stored in one computer-readable storage medium. Based on such understanding, the technical solutions of the present disclosure in essence or parts making contribution to the prior art or parts of the technical solutions can be embodied in the form of a software product, and this computer software product is stored in a storage medium, including several instructions for making a computer device (which may be a personal computer, a server or a network device, etc.) execute all or part of the steps of the methods of the embodiments of the present disclosure. The aforementioned storage medium includes various media in which program codes can be stored, such as USB disk, mobile hard disk, read-only memory (ROM), random access memory (RAM), diskette and compact disk. It should be indicated that in the present text, relational terms such as first and second are merely used for distinguishing one entity or operation from another entity or operation, while it is not required or implied that these entities or operations necessarily have any such practical relation or order. Moreover, the terms such as “comprise”, “contain” or any other variants thereof are intended to encompass non-exclusive inclusion, such that a process, a method, an article, or a device including a series of elements includes not only those elements, but also includes other elements that are not explicitly listed, or further includes elements inherent in such a process, method, article, or device. Without more restrictions, an element defined with wordings “including...” does not exclude presence of other same elements in the process, method, article or device including said element.

[0110] The above-mentioned are merely for preferred embodiments of the present disclosure, but are not intended to limit the present disclosure. For those skilled in the art, various modifications and changes could be made to the present disclosure. Any modifications, equivalent substitutions, improvements and so on made within the spirit and principle of the present disclosure should be covered within the scope of protection of the present disclosure. It should be noted that like reference signs and letters represent like items in the following drawings, and thus, once a certain item is defined in one drawing, it is unnecessary to further define and explain the same in subsequent drawings.

[0111] The above-mentioned are merely for specific embodiments of the present disclosure, but the scope of protection of the present disclosure is not limited thereto, and any variation or substitution that would be readily conceivable to those skilled in the art within the technical scope disclosed in the present disclosure should fall within the scope of protection of the

present disclosure. Therefore, the scope of protection of the present disclosure should be determined by the scope of protection of the claims.

CLAIMS

What is claimed is:

1. An application crash handling method, comprising:
acquiring flag bit information of an application at startup of the application, wherein the flag bit information is read at the startup of the application through preset event tracking; and
if the flag bit information of the application is a first value, switching the application from a current configuration to a historical configuration, and executing the historical configuration, wherein the flag bit information being the first value characterizes that the application crashed in a last startup phase.
2. The method according to claim 1, wherein the method further comprises:
identifying, through application monitoring, a first functional module that caused the application to crash in a last startup phase; and
sending a handling request for the first functional module to a configuration platform, wherein the handling request is configured to instruct the configuration platform to cease issuing a configuration item corresponding to the first functional module.
3. The method according to claim 2, wherein the method further comprises:
for a second functional module corresponding to a configuration obtained by the application that crashed in the last startup phase, executing a latest configuration of the second functional module after waiting for a limited time limit, wherein the second functional module is an updated functional module corresponding to the configuration obtained in the last startup phase, and the second functional module is different from the first functional module.
4. The method according to claim 1, wherein the method further comprises:
restarting the application if a crash of the application is identified by application monitoring, at the startup of the application.
5. The method according to claim 1, wherein the method further comprises:
acquiring a latest configuration of the application if the flag bit information of the application is a second value, wherein the flag bit information being the second value characterizes that the application did no crash in a last startup phase.

6. The method according to any one of claims 1–5, wherein the method further comprises:

updating the flag bit information to the first value if a crash of the application is detected, at the startup of the application; and

updating the flag bit information to a second value if the application does not crash and a latest configuration of the application is obtained from a configuration platform, at the startup of the application.

7. The method according to any one of claims 1–5, wherein the method further comprises:

reading a configuration file of the application locally if the flag bit information of the application is not obtained, at the startup of the application, and executing the configuration file.

8. The method according to any one of claims 1–5, wherein the method further comprises:

reading a third functional module from a configuration file of the application, wherein the third functional module is a functional module undergoing update in the application, and the third functional module comprises a second functional module and excludes a first functional module; and

loading a historical configuration of the third functional module, and executing the historical configuration of the third functional module, if current time falls within a limited time limit of the third functional module, wherein the limited time limit of the third functional module is a time limit set when acquiring a current configuration of the third functional module.

9. An application crash handling apparatus, comprising:

an acquisition module, configured to acquire flag bit information of the application at startup of an application, wherein the flag bit information is read at the startup of the application through preset event tracking; and

an execution module, configured to switch the application from a current configuration to a historical configuration if the flag bit information of the application is a first value, and execute the historical configuration, wherein the flag bit information being the first value characterizes that the application crashed in a last startup phase.

10. An electronic device, comprising: a processor and a memory, wherein the memory stores machine-readable instructions executable by the processor, and when the electronic device runs, the machine-readable instructions, when executed by the processor, implement steps of the method according to any one of claims 1 to 8.

11. A computer-readable storage medium, storing a computer program, wherein the computer program, when executed by a processor, implements steps of the method according to any one of claims 1 to 8.