

CODE TRANSLATION METHOD, APPARATUS, COMPUTER EQUIPMENT AND STORAGE MEDIUM

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] The present application claims priority to the Chinese application No. 2019102961237, filed on April 12, 2019. Afore-mentioned references are herein incorporated by reference in their entireties.

TECHNICAL FIELD

[0002] The present disclosure relates to the field of code translation method, apparatus, computer equipment, and storage medium.

BACKGROUND

[0003] With the development of internationalization, the developed software system will face the needs of different regions and countries in the world, so it is necessary to translate the source code of the software system from the original language to the specific language. Traditionally, business analysts typically collect translation requirements and write documents, and then hand them over to software developers to modify the source code based on the documents. However, inventors realize that the traditional process of international code translation requires business analysts to work with developers who manually extract the code segment to be translated from the code and then manually perform a large amount of translation work. High time cost and manpower cost, it is difficult to accurately and comprehensively identify all code segment to be translated through manual code internationalization translation, and it is also difficult to ensure that uniformity of

code translation, thus result in poor reliability of translation results, Code translation is inefficient.

SUMMARY

[0004] According to various embodiments of the present disclosure, a code translation method, an apparatus, a computer device, and a storage medium are provided.

[0005] A code translation method applied to a code server, the method including:

[0006] acquiring a to-be-translated analysis results set by parsing a to-be-translated source code file, the to-be-translated analysis results set including a to-be-translated code segment and a corresponding segment positioning information;

[0007] invoking a translator interface provided by a translator system, the translator system being deployed on a translation server, and translating the to-be-translated code segment through the translator system to obtain a translated code segment;

[0008] performing replacement processing on the to-be-translated source code according to the segment positioning information corresponding to the translated code segment to generate a to-be-corrected source code file;

[0009] obtaining a target translation detail report for an input of the to-be-corrected source code file, the target translation detail report including a target correction item; and

[0010] correcting the to-be-corrected source code file according to the segment positioning information corresponding to the target correction item, and generating a corrected source code file.

[0011] An apparatus for code translation including:

[0012] an acquiring module, configured to obtain a to-be-translated translation analysis result set by parsing a to-be-translated source code file, the to-be-translated analysis result set includes a code segment to be translated and corresponding segment positioning information;

[0013] a translation module, configured to call a translator interface provided by a translator system, the translator system is deployed on a translation server and translates the to-be-translated code segment through the translator system to obtain a translated code segment;

[0014] a replacement module, configured to perform replacement processing on the to-be-translated source code file according to the segment positioning information corresponding to the translated code segment to generate a to-be-translated source code file; and

[0015] a correction module, configured to obtain a target translation detail report for the to-be-corrected source code file input, the target translation detail report including a target correction item, correcting the to-be-corrected source code file according to the segment positioning information corresponding to the target correction item, and generating a corrected source code file.

[0016] A computer device including a memory and one or more processors in which computer readable instructions are stored, when the computer readable instructions are executed by the one or more processors, causing the one or more processors to perform the following steps:

[0017] acquiring a to-be-translated analysis results set by parsing the to-be-translated source code file, the to-be-translated analysis results set including a to-be-translated code segment and corresponding segment positioning information;

[0018] invoking a translator interface provided by a translator system, the translator

system being deployed on a translation server, and translating the to-be-translated code segment through the translator system to obtain a translated code segment;

[0019] performing replacement processing on the to-be-translated source code file according to the segment positioning information corresponding to the translated code segment to generate a to-be-corrected source code file;

[0020] obtaining a target translation detail report for the to-be-corrected source code file input, the target translation detail report including a target correction item; and

[0021] correcting the to-be-corrected source code file according to the segment positioning information corresponding to the target correction item, and generating a corrected source code file.

[0022] One or more non-transitory computer-readable storage medium storing computer-readable instructions that, when executed by one or more processors, cause the one or more processors to perform the following steps:

[0023] acquiring a to-be-translated analysis results set by parsing the to-be-translated source code file, the to-be-translated analysis result set including a to-be-translated code segment and corresponding segment positioning information;

[0024] invoking a translator interface provided by a translator system, the translator system being deployed on a translation server, and translating the to-be-translated code segment through the translator system to obtain a translated code segment;

[0025] performing replacement processing on the to-be-translated source code file according to the segment positioning information corresponding to the translated code segment to generate a to-be-corrected source code file;

[0026] obtaining a target translation detail report for the to-be-corrected source

code file input, the target translation detail report including a target correction item; and

[0027] correcting the to-be-corrected source code file according to the segment positioning information corresponding to the target correction item, and generating a corrected source code file.

[0028] Details of one or more embodiments of the present application are set forth in the following drawings and description. Other features and advantages of this application will become apparent from the specification, drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0029] In order to more clearly explain that technical solution in the embodiments of the present application, the following brief description will be given of the drawings required to be used in the embodiments. It is obvious that the drawings in the following description are only some embodiments of the present application. For those of ordinary skills in the art, other drawings may also be obtained from these drawings without the effort of creative labor.

[0030] FIG. 1 is an application scenario diagram of a code translation method according to one or more embodiments;

[0031] FIG. 2 is a flow diagram of a code translation method according to one or more embodiments;

[0032] FIG. 3 is a flow diagram of a code translation method according to another embodiment;

[0033] FIG. 4 is a block diagram of a structure of an apparatus for code translation in accordance with one or more embodiments;

[0034] FIG. 5 is an internal block diagram of a computer device in accordance with one or more embodiments.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0035] The present disclosure will be clearly and completely described in combination with the accompanying drawings of the embodiments such that the purposes, the technical solutions and advantages of the embodiments will be more apparent. Apparently, the embodiments described are part, rather than all, of the embodiments of the invention. It should be understood that any and all embodiments derived by a person skilled in the art based on the embodiments of the invention and without creative effort shall be considered within the scope of the invention.

[0036] The code translation method provided in the present application can be applied to an application environment as shown in FIG. 1. The code server 102 communicates with the translation server 104 via a network. The code server 102 and the translation server 104 may be implemented as a separate server or a server cluster composed of a plurality of servers. After parsing the source code file to be translated, the code server 102 may obtain a translation analysis result set including a code segment to be translated and corresponding segment positioning information. The code server 102 may call the translator interface to send the translated code fragment to the translation server 104 corresponding to the translation for translation to obtain the translated code fragment. After receiving the translated code segment returned by the translation server 104, the code server 102 may perform replacement processing on the source code file to be translated according to the segment

positioning information corresponding to the translated code segment to generate a source code file to be corrected. After acquiring the target translation detail report for the source code file input to be corrected, the code server 102 may also perform correction processing on the source code file to be corrected according to the target correction items contained in the target translation detail report, generate the corrected source code file.

[0037] In one embodiment, as shown in FIG. 2, a code translation method is provided, which is illustrated by way of an example as being applied to the code server 102 in FIG. 1, including the steps of:

[0038] Step 202: acquiring the analysis result set to be translated by parsing the source code file to be translated. The result of the analysis to be translated includes a code segment to be translated and corresponding segment positioning information.

[0039] Source code files to be translated are source code files that need to be translated. Source code files refer to uncompiled text files written in accordance with certain programming language specifications. The source code file to be translated may be pre-stored in the database of the code server, and the code server may directly obtain the source code file to be translated from the local database. The code server may also receive the code file to be translated uploaded by the terminal or the source code file to be translated sent by other servers. The to-be-translated analysis result set refers to a set of to-be-translated analysis results respectively corresponding to a plurality of to-be-translated code segments obtained after the source code file to be translated is parsed. The result of the analysis to be translated includes the code fragments to be translated and the related information in the source code file to be translated. The set of analysis results to be translated may contain one or more code segments to be translated. The segment positioning information refers to the positioning information of the code segment to be translated

in the source code file to be translated.

[0040] In one embodiment, after a source code file to be translated is parse by a parser to obtain a code segment to be translated, corresponding segment location information is divide in a code to be processed where that code segment to be translated is located. For example, the code segment to be translated in a specific syntax corresponds to a specific start character string and an end character string, and the start positioning information of the start character string in the code to be processed and the end positioning information of the end character string in the code to be processed can be detected, respectively. The start positioning information and the end positioning information are used as segment positioning information. In addition, a plurality of code segment may be divided and locate in that code to be processed in advance. For example, a code having a preset character length in the code to be processed is use as a code segment, and when a specific character string is detected in the code segment, the code fragment may be taken as a code fragment to be translated.

[0041] Step 204: the translator interface is called to translate the code segment to be translated to obtain the translated code segment.

[0042] A translator interface refers to an interface provided by a translator system that can be used to implement translation functions by invocation. The translator system may be deployed on a translation server. Translator systems include, but are not limited to, one or more of a Baidu translation or a Google translation. The translator system may translate the code segment to be translated into a translated code segment of the corresponding target language type based on a translation policy. Target language types include, but are not limited to, one or more of Chinese, Khmer, Thai, Burmese, and Lao languages.

[0043] In one embodiment, when a set of analysis result to be translated contains a plurality of code segments to be translated, a translation request may be generate based on each code segment to be translated, one by one, and sent to a translation server through a translator interface for translation. In addition, a translation request can be generated accord to a plurality of code segments to be translated, and that translation processing of the plurality of code segment to be translated can be implemented at one time, so as to reduce the number of calls of the overall translation request and improve the execution stability of the translation of the plurality of code segments to be translated.

[0044] Step 206: performing replacement processing on the source code file to be translated according to the segment positioning information corresponding to the translated code segment to generate a source code file to be corrected.

[0045] The corresponding code segment to be translated may be found in the source code file to be translated based on the segment positioning information, so that the corresponding code segment to be translated may be replaced with the corresponding code segment to be translated. The source code file to be translated may contain a plurality of pieces of code to be translated, and the translated source code file may be obtained after detecting that all pieces of code to be translated in the source code file to be translated have been replaced. Rewriting the translated source code file to generate the source code file to be corrected.

[0046] Step 208: obtaining a target translation detail report for the source code file input to be corrected; Target translation details report contains target correction items.

[0047] The target translation detail report refers to a report which records the specific translation details of the source code file to be translated and the translated source code file after being translated, and the specific translation details that need to

be corrected. The target translation detail report may contain multiple target correction items, one corresponding to a piece of code to be corrected in the source code file to be corrected. Each target correction item includes a code fragment to be corrected, a corrected code fragment, fragment positioning information, and may also include a context code fragment, etc. Since the direct replacement process of the translated code segment at the time of automatic translation does not take into account the context of the code segment to be translated in the source code file to be translated, it may lead to an inaccurate translation. Therefore, it is necessary to correct the source code file with correction after automatic translation.

[0048] Step 210: performing correction processing on the source code file to be corrected according to the segment positioning information corresponding to the target correction item, and generate a source code file after correction.

[0049] Locating a code segment to be corrected in the source code file to be corrected based on segment positioning information corresponding to each target correction item in the target translation detail report, thereby performing correction processing on the located code segment to be corrected based on the target correction item. A corrected source code file can be generated by performing correction processing on all pieces of code to be corrected in the source code file to be corrected.

[0050] In that code translation method, after the code server perform parsing processing on the source code file to be translated, a result set of the analysis result to be translated including the code segment to be translated and the corresponding segment positioning information can be obtained. The code server may call the translator interface to send the translated code fragment to the translation server corresponding to the translation for translation to obtain the translated code fragment. After receiving the translated code segment, the source code file to be corrected may be generated by performing replacement processing on the source code file to be

translated according to the segment positioning information corresponding to the translated code segment. After obtaining the target translation detail report for the input of the source code file to be corrected, the source code file to be corrected can be corrected according to the target correction items contained in the target translation detail report, and the corrected source code file can be generated. Through the automatic analysis, translation and correction of source code files to be translated, the comprehensiveness, unity and reliability of code translation can be guaranteed, and the efficiency of code translation can be improved.

[0051] In one embodiment, acquiring that analysis result set to be translated, which is obtain by analyzing the source code file to be translated, includes obtain the source code file to be translated; reading a source code directory in a source code file to be translated; in that source code directory to be translate, a plurality of initial file names are contain; traversing an initial code sub-file corresponding to each initial file name to filter out a code sub-file to be translated; the subfile of the code to be translated matches the preset language type, and the subfile of the code to be translated includes a translation code segment; searching for a parser that matches a preset language type; performing parsing processing on the code subfile to be translated by the parser to generate a set of analysis results to be translated.

[0052] A source code directory refers to a directory in which initial file names of a plurality of initial code subfiles are stored. The initial code subfile refers to the source code file that has the smallest granularity in the source code file to be translated and cannot be further expanded. The code subfile to be translated refers to an initial code subfile that matches the preset language type and needs to be translated into the target language type. The code subfile to be translated contains the code fragment to be translated. By filtering out only the subfiles of the code to be translated that contain the code fragments to be translated, invalid processing of

the initial code subfiles that need not be translated is avoided to save system resources.

[0053] A plurality of parsers corresponding to various preset language types can be preset and installed in the code server, and the sub-file of the code to be translated can be processed by the matching parsers. The parser can be either a stand-alone application or a plug-in running under a particular system platform. For example, the parser can be a Babel parser (a JavaScript source code parser). Different parsers may also be preset for code subfiles to be translated for different file format types. After detecting the file name suffix of the code subfile to be translated, a plurality of parser sets corresponding to the file format type of the code subfile to be translated may be found based on the file format type corresponding to the file name suffix of the code subfile to be translated. Looks for a parser from the parser collection that matches the default language type of the code subfile to be translated. For example, the suffixes can be ".js" (JavaScript, a straightforward scripting language) and ".jsx" (JavaScript XML, a class XML syntax that builds tags inside the React build) respectively. Subfiles of code to be translated are presets to different parsers. For source code subfiles belonging to different types to be translated, a specific parsing process is performed by a specific parser, so that the parsing result can be made more accurate.

[0054] In one embodiment, an initial code sub-file corresponding to each initial file name is traversed to filter out a code sub-file to be translated, including obtaining regular expressions respectively corresponding to a plurality of preset language types; traversing an initial code sub-file corresponding to each initial file name, and matching each initial code sub-file with a regular expression; determining a preset language type matched by each initial code sub-file according to the matched regular expression; filtering out the initial code subfile containing any one of the default

language types as the code subfile to be translated.

[0055] A regular expression is a logical formula for manipulating a string, which consists of a "rule string," a "rule string," composed of predefined specific characters and combinations of these specific characters. A filter logic that can be used to express a string. Each of the preset language types has a corresponding one or more regular expressions through which it can be known whether or not there are pieces of code to be translated that need to be translated in the code to be processed. In order to avoid parsing that initial code sub-file that do not contain the code segment to be translated, reduce the workload of parsing, and improve the parsing efficiency.

[0056] In one embodiment, that analysis result set to be translated further includes a context code segment corresponding to the code segment to be translated, calling the translator interface to translate the code segment to be translated, and obtain the translated code segment, including: finding a code processor that matches the code segment to be translated; identifying, by a code processor, a target language type of a code segment to be translated and a translation policy according to that context code segment; calling the translator interface by the code processor to obtain a translated code segment of the target language type corresponding to the code segment to be translated, which is translated based on the translation policy.

[0057] A code processor refers to a processor for performing operations such as syntax identification, risk identification, translation processing, and status annotation of a code segment to be translated. Different code processors may be provided for the code segments to be translated of different programming language types in advance, thereby enabling the code processor to perform accurate operations for the translated code segments more specifically, so as to realize more accurate code segment translation. The code processor may analyze the syntax of the code segment to be translated in conjunction with the context code segment to identify the

translation policy corresponding to the code segment to be translated. Different translation strategies may be preset for different code segments to be translated. The translation policy includes whether or not to perform a replacement operation, whether or not to perform a different translation operation corresponding to a code segment to be translated of a different syntax, and the like. The resulting context code snippet can be used to parse the translated code snippet more accurately.

[0058] In one of that embodiment, searching for a code processor that matches the code segment to be translated includes identifying a programming language type of the code segment to be translated; searching a set of code processors corresponding to a programming language type; matching the code segment to be translated with a matching condition corresponding to each code processor in the code processor set; using the code processor corresponding to the matching condition as a code processor matching the code segment to be translated.

[0059] A corresponding set of code processors may be predefined for each programming language type. Programming language types include, but are not limited to, JS (JavaScript), JSX (JavaScript XML, a class XML syntax that builds tags inside React builds), and the like, and are not limited thereto. Each code processor has a custom match condition. The matching condition includes, but is not limited to, a condition that the code fragment to be translated relates to a character string type, a condition that the code fragment to be translated relates to a business logic, and a condition that the code fragment to be translated relates to a configuration item attribute in a configuration file, and is not limited thereto. Depending on the matching condition, a code processor that matches the translated code fragment may be found from the set of code processors.

[0060] In one embodiment, a target translation detail report for input to a source code file to be corrected is obtain, including: according to a code segment to be

translated and a corresponding context code segment, generating an initial translation detail report with the translated code segment and segment positioning information; obtaining a corrected code fragment based on an initial translation detail report input; using each code segment to be corrected and the corresponding corrected code segment as a target correction item; generate a target translation detail report based on multiple target correction projects.

[0061] The initial translation details report refers to the report which records the translation details of the source code file to be translated after automatic translation processing. Translation details include, but are not limited to, code segments to be translated and corresponding context code segments, translated code segments, segment positioning information, and the like. The developer can input the corrected code snippet based on the translation details in the initial translation details report. A code fragment to be corrected refers to a code fragment that is incorrectly translated in a source code file to be corrected so that correction processing is required. The developer can mark the translated code snippet that needs to be corrected as the code snippet to be corrected. A corrected code fragment is a code fragment used to correct a code fragment to be corrected. The developer does not need to open the source code file to be corrected, only needs to translate the detail report according to the target, and can manually check the translated code fragment by referring to the context code fragment. In order to more accurately filter out that code snippet to be corrected from the translate code snippet, which reduces the time for the developer to browse through and find the code.

[0062] In one embodiment, when the target correction items are plural, extracting a code segment to be corrected and a corresponding corrected code segment corresponding to each target correction item in the target translation detail report; searching the source code file to be corrected for the code segment to be corrected

according to the segment positioning information corresponding to each target correction item; replacing the code segment to be corrected in the source code file to be corrected with the corresponding corrected code segment; when it is detected that the correction processing corresponding to all the target correction items is completed, the source code file to be corrected after the correction processing is rewritten to obtain the corrected source code file.

[0063] In one embodiment, that initial translation detail report may also contain a risk level corresponding to each of the translate code segments. Risk levels include but are not limited to low risk levels, potential risk levels and high risk levels. After checking the context code segment, it is found that the segment to be translated is a part of a string splice, which indicates that the automatic translation in this case is often incorrect, and the segment to be translated can be marked as a potential risk level. Or when the translation result is checked and it is found that the character string of the original language type is still contained in the translated code segment, the translator system does not perform an accurate translation, and the code segment to be translated can be marked as a potential risk level. When a translator interface call failure is detected resulting in a translation not being completed, the code segment to be translated may be marked as a high risk level. Or when the code processor matching the code segment to be translated is not found, the code segment to be translated may be marked as a high risk level. Developers can focus on verifying post-translation code segments with high risk levels to quickly identify the code segments to be corrected to enter the post-correction code segments for the code segments to be corrected.

[0064] In one embodiment, receiving a corrected code segment input base on that code segment to be corrected and the corresponding segment positioning information, includes obtaining a code jump instruction; the code jump instruction corresponds

to a code segment to be corrected; obtaining a code file to be corrected to which a code fragment to be corrected belongs based on a code jump instruction; displaying the source code file to be corrected according to the segment positioning information corresponding to the code segment to be corrected; detecting a code correction operation acting on the displayed source code file to be corrected; a corrected code fragment based on a code correction operation is extracted.

[0065] When the developer cannot determine the corrected code segment of the code segment to be corrected by initially translating the detail report, for example, when a code logic problem or a string stitching problem occurs, the developer can trigger a code jump instruction. The code server may open the display source code file to be corrected according to the fragment positioning information based on the code jump instruction, and jump to the portion containing the uncertain code fragment to be corrected. The code server may also label the portion of the code snippet to be corrected in the source code file to be corrected so that the developer can quickly browse the context snippet of the code snippet to be corrected to perform a code correction operation on the source code file to be corrected. The code correction operation may be that the developer modifies the code directly in the source code file to be corrected, and the code server extracts the code fragment to be corrected and the corrected code fragment compared to the source code file to be corrected before and after the modification. It can also be that the developer selects the portion of the source code file to be corrected that needs to be modified as the code fragment to be corrected, so that the developer can enter the corrected code fragment in the corresponding position of the target translation detail report. Through the fast code jump, the time for the developer to find the code fragment to be corrected in the source code file to be corrected is saved, and the generation efficiency of the target translation detail report is improved.

[0066] In one embodiment, as shown in FIG. 3, another code translation method is provided, which is illustrated by way of example as being applied to the code server 102 in FIG. 1, including the steps of:

[0067] In step 302, a source code file to be translate is obtained.

[0068] Step 304: reading the source code directory in the source code file to be translated; the source code directory to be translated contains multiple initial file names.

[0069] Step 306, traversing an initial code sub-file corresponding to each initial file name to filter out a code sub-file to be translated; the subfile of the code to be translated matches the preset language type, and the subfile of the code to be translated contains a translation code fragment.

[0070] Step 308: searching for a parser that matches the preset language type.

[0071] In step 310, the parser performs parsing processing on the code subfile to be translated to generate a result set to be translated. The analysis result set to be translated includes a code segment to be translated and corresponding segment positioning information and a context code segment.

[0072] Step 312: searching for a code processor that matches the code segment to be translated.

[0073] Step 314: identifying the target language type and translation policy of the code segment to be translated according to the context code segment by the code processor.

[0074] Step 316: a translator interface is called by a code processor to obtain a translated code segment of a target language type corresponding to the code segment to be translated, which is translated based on a translation policy.

[0075] Step 318: performing replacement processing on the source code file to be translated according to the segment positioning information corresponding to the translated code segment to generate a source code file to be corrected.

[0076] Step 320: generating an initial translation detail report according to the code segment to be translated and the corresponding context code segment, the translated code segment and the segment positioning information.

[0077] In step 322, a corrected code fragment based on the initial translation detail report input is obtained.

[0078] In step 324, each code segment to be corrected and the corresponding corrected code segment are taken as target correction items.

[0079] Step 326: generating a target translation detail report according to the plurality of target correction items.

[0080] Step 302: performing correction processing on the source code file to be corrected according to the segment positioning information corresponding to the target correction item, and generate a source code file after correction.

[0081] In that code translation method, after the code server perform parsing processing on the source code file to be translated, a result set of the analysis result to be translated includes the code segment to be translated and the corresponding segment positioning information can be obtained. The code server may call the translator interface to send the translated code fragment to the translation server corresponding to the translation for translation to obtain the translated code fragment. After receiving the translated code segment, the source code file to be translated can be replaced according to the segment positioning information corresponding to the translated code segment to generate a source code file to be corrected, and an initial translation detail report can be obtained; enabling the developer to input the target

translation detail report containing the target correction project based on the initial translation detail report, and generating the corrected source code file after the correction processing of the source code file to be corrected based on the target correction project. Through the automatic analysis, translation and correction of source code files to be translated, the comprehensiveness, unity and reliability of code translation can be guaranteed, and the efficiency of code translation can be improved.

[0082] It should be understood that although the steps in the flow charts of FIGS. 2 and 3 are shown sequentially as indicated by arrows, these steps are not necessarily performed sequentially as indicated by arrows. Unless expressly stated herein, the execution of these steps is not strictly sequence limited, and these steps may be performed in other sequences. Moreover, at least a portion of the steps in FIGS. 2 and 3 may include a plurality of sub-steps or stages, which are not necessarily performed at the same time, but may be performed at different times. The order of execution of these sub-steps or stages is also not necessarily sequential, but may be performed in turn or alternately with at least a portion of the sub-steps or stages of other steps or other steps.

[0083] In one embodiment, as shown in FIG. 3, a code translation apparatus 400 is provided, including an acquiring module 402, configured to obtain a set of analysis results to be translated by parsing a source code file to be translated. The result of the analysis to be translated includes a code segment to be translated and corresponding segment positioning information. A translation module 404, configured to invoke a translator interface to translate a code segment to be translated to obtain a translated code segment. A replacement module 406, configured to perform replacement processing on the source code file to be translated according to the segment positioning information corresponding to the translated code segment, and generate a source code file to be corrected. A correction module 408,

configured to obtain a target translation detail report for source code file input to be corrected. Target translation details include target correction items in the target translation details statement. The source code file to be corrected is corrected according to the segment positioning information corresponding to the target correction project, and the source code file after correction is generated.

[0084] In one embodiment, that acquisition module 402 is further configured to acquire a source code file to be translated; reading a source code directory in a source code file to be translated; in that source code directory to be translated, a plurality of initial file names are contained; traversing an initial code sub-file corresponding to each initial file name to filter out a code sub-file to be translated; the subfile of the code to be translated matches the preset language type, and the subfile of the code to be translated includes a translation code segment; searching for a parser that matches a preset language type; performing parsing processing on the code subfile to be translated by the parser to generate a set of analysis results to be translated.

[0085] In one embodiment, the acquiring module 402 is further configured to obtain regular expressions respectively corresponding to a plurality of preset language types; traversing an initial code sub-file corresponding to each initial file name, and matching each initial code sub-file with a regular expression; determining a preset language type matched by each initial code sub-file according to the matched regular expression; filtering out the initial code subfile containing any one of the default language types as the code subfile to be translated.

[0086] In one embodiment, that analysis result set to be translated further includes a context code segment corresponding to the code segment to be translated, and the translation module 404 is further configured to find a code processor that matches the code segment to be translated; identify, by a code processor, a target language type of a code segment to be translated and a translation policy according to that context code

segment; the translator interface is called by the code processor to obtain a translated code segment of the target language type corresponding to the code segment to be translated, which is translated based on the translation policy.

[0087] In one embodiment, the translation module 404 is further configured to identify a programming language type of the code segment to be translated; searching a set of code processors corresponding to a programming language type; matching the code segment to be translated with a matching condition corresponding to each code processor in the code processor set; the code processor corresponding to the matching condition is used as a code processor matching the code segment to be translated.

[0088] In one embodiment, the correction module 408 is further configured to generate an initial translation detail report based on the code segment to be translated and the corresponding context code segment, the translated code segment and the segment location information; obtaining a corrected code fragment based on an initial translation detail report input; using each code segment to be corrected and the corresponding corrected code segment as a target correction item; generating a target translation detail report based on multiple target correction projects.

[0089] In one embodiment, the correction module 408 is further configured to obtain a code jump instruction; the code jump instruction corresponds to a code segment to be corrected; obtaining a code file to be corrected to which a code fragment to be corrected belongs based on a code jump instruction; displaying the source code file to be corrected according to the segment positioning information corresponding to the code segment to be corrected; detecting a code correction operation acting on the displayed source code file to be corrected; a corrected code fragment based on a code correction operation is extracted.

[0090] For the specific definition of the code translation apparatus, reference may

be made to the above definition of the code translation method, which will not be described in detail herein. The various modules in the above-described code translation apparatus may be implemented in whole or in part by software, hardware, and combinations thereof. The above modules may be embedded in or independent of the processor in the computer device in the form of hardware, or may be stored in a memory in the computer device in the form of software so that the processor calls to perform the operations corresponding to the above modules.

[0091] In one embodiment, a compute device is provided, which may be a server and whose internal structure diagram may be as shown in FIG. 5. The computer device includes a processor, a memory, a network interface, and a database connected through a system bus. A processor of the computer device is used to provide computing and control capabilities. The memory of the computer device includes a non-transitory storage medium, an internal memory. The non-transitory storage medium stores an operating system, computer readable instructions, and a data library. The internal memory provides an environment for the operation of operating systems and computer readable instructions in non-transitory storage media. The database of the computer device is used to store data such as the analysis result set to be translated. The network interface of the computer device is used to communicate with an external terminal through a network connection. The computer readable instructions are executed by a processor to implement a code translation method.

[0092] It will be understood by those skilled in the art that the structure shown in fig. 5 is merely a block diagram of a portion of the structure associated with the present application and does not constitute a limitation on the computer device to which the present application applies. A particular computer device may include more or fewer components than shown in the figures, or combine certain components, or have a different arrangement of components.

[0093] In one embodiment, a compute device is provided that include a memory and one or more processors in which computer readable instructions are store, when that computer readable instructions are executed by the one or more processors, causing the one or more processors to implement the steps of the code fragment positioning method in the various embodiments described above.

[0094] In one embodiment, there is provide one or more non-transitory computer-readable storage media store computer-readable instructions that are executed by one or more processor, causing the one or more processors to implement the steps of the code fragment positioning method in the various embodiments described above.

[0095] One of ordinary skill in the art will appreciate that all or part of the flow in the method of implementing the embodiments described above may be accomplished by computer readable instructions to instruct related hardware. The computer-readable instructions may be stored in a non-transitory computer-readable storage medium, and the computer-readable instructions, when executed, may include the flow of embodiments of the methods described above. Any reference to memory, storage, database, or other media used in the various embodiments provided herein may include non-transitory and / or transitory memory. The non-transitory memory may include read-only memory (ROM), programmable ROM (PROM), electrically programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM), or flash memory. The transitory memory may include a random access memory (RAM) or an external cache memory. By way of illustration and not limitation, RAM is available in a variety of forms, such as static RAM (SRAM), dynamic RAM (DRAM), synchronous DRAM (SDRAM), dual data rate SDRAM (DDRSDRAM), enhanced SDRAM (ESDRAM), synch link DRAM (SLDRAM), memory Bus (Rambus) direct RAM (RDRAM), direct memory bus dynamic RAM

(DRDRAM), and memory bus dynamic RAM (RDRAM) and the like.

[0096] The various technical features of the above embodiments may be combined arbitrarily, and in order to simplify the description, all possible combinations of the various technical features in the above embodiments are not described, however, as long as there is no contradiction in the combination of these technical features, Should be considered as the scope of this specification.

[0097] The foregoing implementations are merely specific embodiments of the present disclosure, and are not intended to limit the protection scope of the present disclosure. It should be noted that any variation or replacement readily figured out by persons skilled in the art within the technical scope disclosed in the present disclosure shall all fall into the protection scope of the present disclosure. Therefore, the protection scope of the present disclosure shall be subject to the protection scope of the claims.

What is claimed is:

1. A method for code translation applied to a code server, comprising:

acquiring a to-be-translated analysis results set obtained by parsing a to-be-translated source code file, the to-be-translated analysis results set comprising a to-be-translated code segment and a corresponding segment positioning information;

calling a translator interface provided by a translator system, the translator system being deployed on a translation server, and obtaining a translated code segment through translating the to-be-translated code segment by the translation system;

performing replacement processing on the to-be-translated source code file according to the segment positioning information corresponding to the translated code segment, so as to generate a to-be-corrected source code file;

acquiring a target translation detail report for an input of the to-be-corrected source code file, the target translation detail report comprising a target correction item; and

correcting the to-be-corrected source code file according to the segment positioning information corresponding to the target correction item, so as to generate a corrected source code file.

2. The method according to claim 1, wherein the acquiring of a to-be-translated analysis results set obtained by parsing a to-be-translated source code file comprises:

acquiring the to-be-translated source code file;

reading a source code directory in the to-be-translated source code file; the to-be-translated source code directory comprising a plurality of initial file names;

traversing an initial code sub-file corresponding to each of the initial file names to screen a to-be-translated code sub-file, the to-be-translated code sub-file being matched with a preset language type, and the to-be-translated code sub-file comprising a translation code segment;

looking up a parser matched with the preset language type; and

acquiring the segment positioning information of the to-be-translated code segment in the to-be-translated code sub-file by the parser, extracting the to-be-translated code segment based on the segment positioning information, and generating the to-be-translated analysis results set according to the to-be-translated code segment and the segment positioning information.

3. The method according to claim 2, wherein the traversing of an initial code sub-file corresponding to each of the initial file names to screen a to-be-translated code sub-file comprises:

acquiring regular expressions respectively corresponding to a plurality of preset language types;

traversing the initial code sub-file corresponding to each of the initial file names, and matching each initial code sub-file with the regular expression;

determining the preset language type matched with each initial code sub-file according to the matched regular expression; and

screening the initial code sub-file containing any one of the preset language types as the to-be-translated code sub-file.

4. The method according to claim 1, wherein the to-be-translated analysis results set

further comprises a context code segment corresponding to the to-be-translated code segment, and the calling of a translator interface provided by a translator system comprises:

- looking up a code processor matched with the to-be-translated code segment;

- analyzing, by the code processor, the syntax of the to-be-translated code segment according to the context code segment to identify a target language type and a translation policy of the to-be-translated code segment; and

- calling, by the code processor, the translator interface provided by the translator system, the translator system being deployed on the translation server, the translator system translating the to-be-translated code segment into a translated code segment of a corresponding target language type based on the translation policy.

5. The method according to claim 4, wherein the looking up of a code processor matched with the to-be-translated code segment comprises:

- identifying a programming language type of the to-be-translated code segment based on the regular expression;

- looking up a code processor set corresponding to the programming language type;

- matching the to-be-translated code segment with a matching condition corresponding to each code processor in the code processor set; and

- serving the code processor corresponding to the matched matching condition as the code processor matched with the to-be-translated code segment.

6. The method according to claim 4, wherein the acquiring of a target translation

detail report for an input of the to-be-corrected source code file comprises:

generating an initial translation detail report according to the to-be-translated code segment and the corresponding context code segment, the translated code segment and the segment positioning information;

acquiring a corrected code segment inputted by a terminal based on the initial translation detail report;

serving each of the to-be-corrected code segments, the corresponding corrected code segment and the context code segment as a target correction item; and

generating the target translation detail report based on the plurality of target correction items.

7. The method according to claim 5, wherein the acquiring of a corrected code segment inputted by a terminal based on the to-be-corrected code segment and the corresponding segment positioning information comprises:

acquiring a code jump instruction, the code jump instruction corresponding to the to-be-corrected code segment;

acquiring a to-be-corrected code file to which the to-be-corrected code segment belongs based on the code jump instruction;

displaying the to-be-corrected source code file by jumping according to the segment positioning information corresponding to the to-be-corrected code segment;

detecting a code correction operation acting on the displayed to-be-corrected source code file; and

extracting the obtained corrected code segment based on the code correction operation.

8. An apparatus for code translation comprising:

an acquiring module, configured to obtain a to-be-translated translation analysis result set by parsing a to-be-translated source code file, the to-be-translated analysis result set comprising a code segment to be translated and corresponding segment positioning information;

a translation module, configured to call a translator interface provided by a translator system, wherein the translator system is deployed on a translation server and translates the to-be-translated code segment through the translator system to obtain a translated code segment;

a replacement module, configured to perform replacement processing on the to-be-translated source code file according to the segment positioning information corresponding to the translated code segment to generate a to-be-translated source code file; and

a correction module, configured to obtain a target translation detail report for the to-be-corrected source code file input, the target translation detail report comprising a target correction item, correcting the to-be-corrected source code file according to the segment positioning information corresponding to the target correction item, and generating a corrected source code file.

9. A computer device including a memory and one or more processors in which computer readable instructions are stored, when the computer readable instructions are executed by the one or more processors, causing the one or more processors to perform the following steps:

obtaining a to-be-translated analysis results set by parsing the to-be-translated source code file, the to-be-translated analysis results set comprising a to-be-translated code segment and corresponding segment positioning information;

invoking a translator interface provided by a translator system, the translator system being deployed on a translation server, and translating the to-be-translated code segment through the translator system to obtain a translated code segment;

performing replacement processing on the to-be-translated source code file according to the segment positioning information corresponding to the translated code segment to generate a to-be-corrected source code file;

obtaining a target translation detail report for the to-be-corrected source code file input, the target translation detail report comprising a target correction item; and

correcting the to-be-corrected source code file according to the segment positioning information corresponding to the target correction item, and generating a corrected source code file.

10. The computer device according to claim 9, wherein when the computer-readable instructions are executed by the one or more processors, the one or more processors further performing the following steps:

obtaining a to-be-translated source code file to be translated;

reading a source code directory in the to-be-translated source code file, the to-be-translated source code directory comprising a plurality of initial file names;

traversing an initial code sub-file corresponding to each of the initial file names to filter out a to-be-translated code sub-file, the to-be-translated subfile of the code matching a preset language type, and the to-be-translated subfile of the code

comprising a translation code segment;

searching for a parser that matches the preset language type; and

acquiring segment positioning information of a to-be-translated code segment in the to-be-translated code sub-file by the parser, extracting a to-be-translated code segment based on the segment positioning information, and extracting the to-be-translated code segment according to the to-be-translated code segment and the segment positioning information, generating a to-be-translated analysis results set.

11. The computer device according to claim 10, wherein when the computer-readable instructions are executed by the one or more processors, the one or more processors further perform the following steps:

obtaining regular expressions respectively corresponding to a plurality of preset language types;

traversing an initial code sub-file corresponding to each of the initial file names, and matching each initial code sub-file with the regular expression;

determining a preset language type matched by each initial code sub-file according to the matched regular expression; and

filtering out the initial code subfile containing any one of the default language types as the to-be-translated code subfile.

12. The computer device according to claim 9, wherein when the computer-readable instructions are executed by the one or more processors, the one or more processors further perform the following steps:

finding a code processor that matches the to-be-translated code segment;

analyzing, by the code processor, the syntax of the to-be-translated code segment according to the context code segment to identify a target language type and a translation policy of the to-be-translated code segment; and

invoking, by the code processor, a translator interface provided by a translator system, the translator system being deployed on a translation server, the translator system translates the to-be-translated code segment into a translated code segment of a corresponding target language type based on the translation policy.

13. The computer device according to claim 12, wherein when the computer-readable instructions are executed by the one or more processors, the one or more processors further perform the following steps:

identifying a programming language type of the to-be-translated code segment based on a regular expression;

searching for a set of code processors corresponding to the programming language type;

matching the to-be-translated code segment with a matching condition corresponding to each code processor in the code processor set; and

using a code processor corresponding to the matched matching condition as a code processor matching the to-be-translated code segment.

14. The computer device according to claim 12, wherein when the computer-readable instructions are executed by the one or more processors, causing the one or more processors to further perform the following steps:

generating an initial translation detail report according to the to-be-translated

code segment and the corresponding context code segment, the translated code segment and the segment positioning information;

obtaining a corrected code segment input through a terminal based on the initial translation detail report;

using each of the to-be-corrected code segments, the corresponding corrected code segment and the context code segment as a target correction item; and

generating a target translation detail report based on the plurality of target correction items.

15. The method according to claim 13, wherein the obtaining of the corrected code segment inputted through the terminal based on the to-be-corrected code segment and the corresponding segment positioning information comprises:

obtaining a code jump instruction, the code jump instruction corresponding to the to-be-corrected code segment;

obtaining a to-be-corrected code file to which the to-be-corrected code segment belongs based on a code jump instruction;

displaying the to-be-corrected source code file according to the segment positioning information corresponding to the to-be-corrected code segment;

detecting a code correction operation acting on the displayed to-be-corrected source code file; and

extracting a corrected code segment based on the code correction operation.

16. One or more non-transitory computer-readable storage medium storing

computer-readable instructions that, when executed by one or more processors, cause the one or more processors to perform the following steps:

- obtaining a to-be-translated analysis results set by parsing the to-be-translated source code file, the to-be-translated analysis result set comprising a to-be-translated code segment and corresponding segment positioning information;

- invoking a translator interface provided by a translator system, the translator system being deployed on a translation server, and translating the to-be-translated code segment through the translator system to obtain a translated code segment;

- performing replacement processing on the to-be-translated source code file according to the segment positioning information corresponding to the translated code segment to generate a to-be-corrected source code file;

- obtaining a target translation detail report for the to-be-corrected source code file input, the target translation detail report comprising a target correction item; and

- correcting the to-be-corrected source code file according to the segment positioning information corresponding to the target correction item, and generating a corrected source code file.

17. The computer-readable storage medium according to claim 16, wherein the computer-readable instructions, when executed by one or more processors, cause the one or more processors to further perform the following steps:

- obtaining a to-be-translated source code file;

- reading a source code directory in the to-be-translated source code file, the to-be-translated source code directory comprising a plurality of initial file names;

- traversing an initial code sub-file corresponding to each of the initial file names

to filter out a to-be-translated code sub-file, the to-be-translated subfile of the code matches a preset language type, and the to-be-translated subfile of the code comprising a translation code segment;

searching for a parser that matches the preset language type; and

acquiring segment positioning information of a to-be-translated code segment in the to-be-translated code sub-file by the parser, extracting a to-be-translated code segment based on the segment positioning information, and extracting the to-be-translated code segment according to the to-be-translated code segment and the segment positioning information, generate a to-be-translated analysis results set.

18. The computer-readable storage medium according to claim 17, when the computer-readable instructions are executed by one or more processors, cause the one or more processors to further perform the following steps: obtaining regular expressions respectively corresponding to a plurality of preset language types;

obtaining regular expressions respectively corresponding to a plurality of preset language types;

traversing an initial code sub-file corresponding to each of the initial file names, and matching each initial code sub-file with the regular expression;

determining a preset language type matched by each initial code sub-file according to the matched regular expression; and

filtering out the initial code subfile containing any one of the default language types as the to-be-translated code subfile.

19. The computer-readable storage medium according to claim 16, the

computer-readable instructions, when executed by one or more processors, cause the one or more processors to further perform the following steps

- finding a code processor that matches the to-be-translated code segment;

- analyzing, by the code processor, the syntax of the to-be-translated code segment according to the context code segment to identify a target language type and a translation policy of the to-be-translated code segment; and

- invoking, by the code processor, a translator interface provided by a translator system, the translator system being deployed on a translation server, the translator system translates the to-be-translated code segment into a translated code segment of a corresponding target language type based on the translation policy.

20. The computer-readable storage medium according to claim 19, the computer-readable instructions, when executed by one or more processors, cause the one or more processors to further perform the following steps

- identifying a programming language type of the to-be-translated code segment based on a regular expression;

- searching for a set of code processors corresponding to the programming language type;

- matching the to-be-translated code segment with a matching condition corresponding to each code processor in the code processor set; and

- using a code processor corresponding to the matched matching condition as a code processor matching the to-be-translated code segment.