

[DESCRIPTION]

[TITLE OF INVENTION]

PROGRAM

[Technical Field]

5 [0001]

The present invention relates to a program.

Priority is claimed on Japanese Patent Application No. 2016-064496, filed
March 28, 2016, the content of which is incorporated herein by reference.

[Background Art]

10 [0002]

For example, a technology of requiring no table which correlates versions of
projects in an application development environment is known (refer to Patent Literature
1). In this technology, when platforms have a plurality of versions, a library commonly
necessary for all the platforms is extracted and a file corresponding to a platform (a
15 version of a project) which is a target is searched for (refer to paragraphs 0020 to 0021 of
Patent Literature 1).

[Citation List]

[Patent Literature]

[0003]

20 [Patent Literature 1]

Japanese Unexamined Patent Application, First Publication No. 2015-162067

[Summary of Invention]

[Technical Problem]

[0004]

25 However, conventionally, programs respectively corresponding to a plurality of

platforms are executed, and thus there were cases in which complicated processes such as searching for a file corresponding to each platform are required.

[0005]

An object of embodiments of the present invention devised in view of the
5 aforementioned circumstances is to provide a program capable of switching execution
programs depending on an object platform.

[Solution to Problem]

[0006]

A program according to an embodiment of the present invention causes a
10 computer to perform a first function of identifying the type of a process of a target
program which is another program corresponding to an object to be operated by the
program depending on a platform in which the target program operates, a second
function of determining whether the type of the process of the target program identified
by the first function matches a first type of a process of the program, and a third function
15 of starting a second type of a process of the program matching the type of the process of
the target program and ending the first type of the program of the process when it is
determined that the type of the process of the target program does not match the first type
of the process of the program according to the second function.

[0007]

20 A program according to an embodiment of the present invention causes a
computer to perform a first function of identifying the type of a process of a target
program which is another program corresponding to an object to be operated by the
program depending on a platform in which the target program operates, a second
function of determining whether the type of the process of the target program identified
25 by the first function matches a first type of a process of the program, and a third function

of starting a second type of a process of the program matching the type of the process of the target program and ending the first type of the process of the program when it is determined that the type of the process of the target program does not match with the first type of the process of the program according to the second function.

5 [0008]

In the program according to one aspect of the present invention, a configuration in which the third function starts a process of the second type of the program and ends a process of the first type of the program may be used.

[0009]

10 In the program according to one aspect of the present invention, a configuration in which the program has resources of programs according to the process of the first type and the process of the second type may be used.

[Advantageous Effects of Invention]

[0010]

15 According to the aforementioned program, it is possible to switch execution programs depending on an object platform.

[Brief Description of Drawings]

[0011]

20 Fig. 1 is a diagram illustrating an example of a schematic configuration of an information processing system according to an embodiment of the present invention.

Fig. 2 is a diagram illustrating an example of a schematic configuration of a terminal device according to an embodiment of the present invention.

Fig. 3 is a diagram illustrating a schematic image of a program according to an embodiment of the present invention.

25 Fig. 4 is a flowchart illustrating an example of a processing procedure performed

using a program according to an embodiment of the present invention.

[Description of Embodiments]

[0012]

Embodiments of the present invention will be described in detail with reference
5 to the drawings.

[0013]

[Description of example of configuration of information processing system]

Fig. 1 is a diagram illustrating an example of a schematic configuration of an
information processing system 11 according to an embodiment of the present invention.

10 The information processing system 11 includes a plurality of terminal devices 21
to 25 and a network 31.

The terminal devices 21 to 25 are connected to the network 31.

The terminal devices 21 to 25 are computers or the like.

The network 31 is the Internet or the like.

15 [0014]

Here, although five terminal devices 21 to 25 are illustrated as the plurality of
terminal devices 21 to 25 in the example of Fig. 1, the number of terminal devices may
be arbitrary.

In addition, although one network 31 is illustrated in the example of Fig. 1, the
20 number of networks may be arbitrary.

Further, the information processing system 11 may have other configurations.

Further, although an example in which the terminal devices 21 to 25 are
connected to the network 31 is illustrated in the present embodiment, the terminal
devices 21 to 25 may not be connected to the network 31 in other configuration
25 examples.

[0015]

[Description of example of configuration of terminal device]

Fig. 2 is a diagram illustrating an example of a schematic configuration of the terminal device 21 according to an embodiment of the present invention.

5 Here, the same schematic configuration applies to the respective terminal devices 21 to 25 in the present embodiment. Accordingly, the terminal device 21 is exemplified to describe the configuration and operation examples in the present embodiment.

The terminal device 21 includes a processor 101, a memory 102, a network
10 interface 103, an operation unit 104, a display unit 105, a storage device 106, an input/output interface 107, and a bus 111 which connects these components.

[0016]

The processor 101 is composed of a central processing unit (CPU) or the like and performs a process designated in a program by executing the program.

15 The memory 102 is a volatile storage unit, is composed of a random access memory (RAM) or the like and temporarily stores data. For example, a dynamic random access memory (DRAM) may be used as a RAM.

The network interface 103 is an interface connected to an external network (the network 31 in the present embodiment).

20 The operation unit 104 is composed of one or more of a keyboard, a mouse and the like and receives an operation performed by a person (user).

The display unit 105 has a screen and displays and outputs information on the screen.

The storage device 106 is a non-volatile storage unit, is composed of a hard disc
25 or the like, for example, and stores data.

The input/output interface 107 is an interface connected to an external storage medium and the like.

[0017]

[Description of background of embodiment]

5 First, a background technology of an embodiment will be described.

For example, in programs operating on a dot network framework (. NET Framework) which is an application development and execution environment, a build type in accordance to an execution object may be designated. There is a 32-bit operating system (OS) or a 64-bit OS as an OS which becomes a platform of an

10 execution object.

[0018]

As specific examples of build types, x86, x64 and AnyCPU are represented.

<x86>

x86 operates as a 32-bit process in any one of a 32-bit OS and a 64-bit OS.

15 Further, x86 operates as WOW64 which is an emulation layer/subsystem in a 64-bit OS.

<x64>

x64 cannot operate in a 32-bit OS. x64 operates as a 64-bit process in a 64-bit OS.

<AnyCPU>

20 AnyCPU operates as a 32-bit process in a 32-bit OS and operates as a 64-bit process in a 64-bit OS.

[0019]

As described above, a 32-bit process and a 64-bit process coexist in a 64-bit OS.

As a specific example, in a generally used program such as Internet Explorer (registered

25 trademark) (hereinafter referred to as "IE"), operations of both a 32-bit process and a

64-bit process exist.

Here, certain restrictions at the OS level are applied to exchange of information or operations between a 32-bit process and a 64-bit process. Accordingly, when information of a program operating as a 64-bit process is intended to be acquired from a program operating as a 32-bit process or some operations are intended to be performed for a program operating as a 64-bit process by a program operating as a 32-bit process, for example, errors may be generated.

[0020]

In the present embodiment, a program having a function of establishing communication by operating a process (IE process) of IE during booting is described as an example.

A situation in which the aforementioned program cannot operate IE may occur when the platform in which the program operates differs from the platform in which IE operates in a 64-bit OS according to the aforementioned restrictions. As a specific example, when the program operates as a 64-bit process whereas IE operates as a 32-bit process, the program cannot access IE.

[0021]

[Description of program according to embodiment]

A program component (hereinafter referred to as “a program”) according to the present embodiment will be described.

Hereinafter, platforms of processes in which IE and the program operate are called “operation platforms.”

With respect to IE, it is known that a 32-bit process and a 64-bit process are switched simply by changing one set value in the 64-bit OS (e.g., IE11). Accordingly, IE may operate as a 32-bit process or a 64-bit process depending on an environment.

In addition, the program has a function of checking IE during operation and, when the platform (operation platform) in which IE operates does not correspond to an operation platform which is a current object of the program, rebooting depending on the platform (operation platform) in which IE is operating. Accordingly, the program may
 5 operate IE irrespective of settings of the terminal device 21 which executes the program (i.e., irrespective of whether IE operates as a 32-bit process or a 64-bit process).

[0022]

The program dynamically switches modules which operate according to the following <first combination> to <third combination>.

10 <First combination>

This is a combination of a 32-bit OS and build for AnyCPU. In this case, the program operates as a 32-bit process.

<Second combination>

This is a combination of a state in which IE operates as a 32-bit process in a
 15 64-bit OS and build for x86. In this case, the program operates as a 32-bit process.

<Third combination>

This is a combination of a state in which IE operates as a 64-bit process in a 64-bit OS and build for AnyCPU. In this case, the program operates as a 64-bit process.

[0023]

20 Fig. 3 is a diagram illustrating a schematic image of a program (program 201) according to an embodiment of the present invention.

The program 201 includes a resource 221 (hereinafter referred to as an “AnyCPU resource”) with respect to AnyCPU and a resource 222 (hereinafter referred to as an “x86 resource”) with respect to x86.

25 The program 201 has a function of generating a program (hereinafter referred to

as an “AnyCPU correspondence program”) operating corresponding to AnyCPU and a function of generating a program (hereinafter referred to as an “x86 correspondence program”) operating corresponding to x86. The AnyCPU correspondence program operates as a 32-bit process or a 64-bit process. The x86 correspondence program
5 operates as a 32-bit process.

[0024]

Fig. 4 is a flowchart illustrating an example of a processing procedure performed using a program (program 201) according to an embodiment of the present invention.

In the present embodiment, a case in which the processor 101 reads and executes
10 the program 201 stored in the storage device 106 using the memory 102 in the terminal device 21 is represented.

[0025]

First, with respect to the program 201 capable of dynamically switching operation platforms, a module built for x86 is embedded as a resource (x86 resource 222)
15 when a module (AnyCPU resource 221) having AnyCPU as an object is built.

Here, contents of substantial processes in a build for AnyCPU and a build for x86 are equivalent to each other in the present embodiment. For example, these two types of build may share a source code.

[0026]

20 (Step S1)

In the terminal device 21, the program 201 is executed.

[0027]

(Step S2)

When the program 201 is executed, the program 201 determines an operation
25 platform of a program operating in the terminal device 21, which is a target (hereinafter

referred to as a “target program”). In the present embodiment, the target program is IE.

Specifically, the program 201 acquires a shell window list which exists as a function in the terminal device 21. In addition, the program 201 sequentially checks file paths in the acquired shell window list to search for a file path which is the program name of IE. When the file path which is the program name of IE is discovered, the program 201 determines an operation platform of the discovered IE. In the present embodiment, the program 201 determines IE as a 32-bit process when the file path includes a character string “x86” and determines IE as a 64-bit process in other cases.

[0028]

On the other hand, when a file path which is the program name of IE is not discovered, the program 201 determines that IE has not been booted at that point in time, repeatedly checks file paths at timings with a certain time interval (e.g., several seconds) and waits until IE is booted.

[0029]

(Step S3)

The program 201 determines an operation platform of a program being executed caused by the program 201 (AnyCPU correspondence program or x86 correspondence program in the present embodiment).

Specifically, program names are different in the AnyCPU correspondence program and x86 correspondence program in the present embodiment. In addition the program 201 determines the operation platform on the basis of the name of the program being executed.

[0030]

(Step S4)

The program 201 determines whether the operation platform of the target

program matches the operation platform of the program being executed.

When the operation platforms match each other (YES in step S4), the process of this flow ends. In this case, the program being executed uses the discovered IE as an operation object.

5 [0031]

(Step S5)

On the other hand, when the operation platforms do not match each other (NO in step S4), the program 201 determines whether a program which is a switching destination exists.

10 Here, as cases in which the operation platforms do not match each other, there are cases in which a 64-bit OS is used in the terminal device 21 and the operation platform of the target program does not match the operation platform of the program being executed in the present embodiment.

In this case, the program 201 determines whether the program which is the
15 switching destination (the AnyCPU correspondence program or the x86 correspondence program in the present embodiment) has already existed as a file. Here, the present embodiment employs a configuration in which the AnyCPU correspondence program and the x86 correspondence program are output to the same folder and the program 201 checks the folder.

20 [0032]

The program which is the switching destination is the x86 correspondence program when the program currently being executed is the AnyCPU correspondence program. Similarly, the program which is the switching destination is the AnyCPU correspondence program when the program currently being executed is the x86
25 correspondence program.

[0033]

(Step S6)

When the program which is the switching destination does not exist (NO in step S5), the program 201 generates the program which is the switching destination. In the present embodiment, the program 201 outputs the program which is the switching destination embedded as the resource of the program 201 to a file. The program 201 outputs files having respectively determined separate names to the same folder in the AnyCPU correspondence program and the x86 correspondence program.

[0034]

10 (Step S7)

When the program which is the switching destination exists in step S5 (YES in step S5) or the program which is the switching destination is generated in step S6, the program 201 executes the program which is the switching destination as a new process.

[0035]

15 (Step S8)

Then, the program 201 ends the program which is the switching source (executed program).

[0036]

Meanwhile, when the program 201 is initially executed (in step S1), for example), the program 201 may generate one or both of the AnyCPU correspondence program and the x86 correspondence program.

[0037]

[Summary of embodiment]

As described above, in the program (program 201) according to the present embodiment, execution programs may be switched depending on an object platform and

operations may be performed in a plurality of platforms.

[0038]

Here, although IE is used as an object program in the present embodiment, various programs may be used as an object program as another configuration example.

5 In addition, although the program (program 201) according to the present embodiment has the function of operating an object program, the program may have other functions with respect to an object program as another configuration example.

[0039]

Further, the program (program 201) according to the present embodiment may
10 be stored in the storage device 106 of the terminal device 21 in advance when the terminal device 21 is shipped as an example.

As another example, the program (program 201) according to the present embodiment may be received by the terminal device 21 and stored in the storage device 106 of the terminal device 21 by being transmitted from other terminal devices (terminal
15 devices 22 to 25 in the example of Fig. 1), other server devices or the like to the terminal device through the network 31.

As another example, the program (program 201) according to the present embodiment may be stored in the storage device 106 of the terminal device 21 by being input to the terminal device 21 from a recording medium through the input/output
20 interface 107.

[0040]

In addition, although a configuration in which the terminal device 21 executes the program (program 201) according to the present embodiment, AnyCPU correspondence program and x86 correspondence program is represented in the present
25 embodiment, one or more of these programs may be executed by a device outside the

terminal device 21 as another configuration example.

[0041]

As a configuration example, a program (the program 201 in the embodiment) causes a computer (a computer constituting the terminal device 21 in the embodiment) to perform a first function (the function of step S1 and step S2 in the example of Fig. 4) of identifying the type of a process (a 64-bit process or a 32-bit process in the embodiment) of a target program (the program of IE in the embodiment) which is another program corresponding to an object to be operated by the program depending on a platform (platform corresponding to AnyCPU or x86 in the embodiment) in which the target program operates, a second function (the function of step S3 and step S4 in the example of Fig. 4) of determining whether the type of the process of the target program identified by the first function matches a first type of a process of the program, and a third function (the function of step S5 to step S8 in the example of Fig. 4) of starting a process of a second type of the program matching the type of the process of the target program and ending the process of the first type of the program when it is determined that the type of the process of the target program does not match the first type of the process of the program according to the second function.

As a configuration example, a program (the program 201 in the embodiment) causes a computer (a computer constituting the terminal device 21 in the embodiment) to perform a first function (the function of step S1 and step S2 in the example of Fig. 4) of identifying the type of a process (a 64-bit process or a 32-bit process in the embodiment) of a target program (the program of IE in the embodiment) which is another program corresponding to an object to be operated by the program depending on a platform (platform corresponding to AnyCPU or x86 in the embodiment) in which the target program operates, a second function (the function of step S3 and step S4 in the example

of Fig. 4) of determining whether the type of the process of the target program identified by the first function matches a first type of a process of the program, and a third function (the function of step S5 to step S7 in the example of Fig. 4) of generating a program file of the program matching the type of the process of the target program and starting a
5 process of a second type according to the generated program file when it is determined that the type of the process of the target program does not match the first type of the process of the program according to the second function.

As a configuration example, in the program, the third function starts a process of the second type of the program and ends a process of the first type of the program (the
10 function of step S8 in the example of Fig. 4).

As a configuration example, in the aforementioned program, the program has resources of programs according to the process of the first type and the process of the second type.

[0042]

15 <Other configuration examples>

As a configuration example, a program (the program 201 in the embodiment) causes a computer (a computer constituting the terminal device 21 in the embodiment) to perform a first function (the function of step S2 to step S4 in the example of Fig. 4) of identifying a platform and a second function (the function of step S5 to step S8 in the
20 example of Fig. 4) of performing an operation depending on the platform identified by the first function.

As a configuration example, the first function identifies a platform in which a target program (the program of IE in the embodiment) operates, and the second function executes a second program (the x86 correspondence program or the AnyCPU
25 correspondence program in the embodiment) depending on a second platform (platform

corresponding to x86 or AnyCPU in the embodiment) corresponding to the platform identified by the first function if it is determined that the platform identified by the first function is not a first platform when a first program (the AnyCPU correspondence program or the x86 correspondence program in the embodiment) depending on the first platform (platform corresponding to AnyCPU or x86 in the embodiment) is executed.

As a configuration example, the second function generates a file of the first program and a file of the second program (outputs the programs as files in the embodiments)

As a configuration example, the program (the program 201 in the embodiment) includes information of the first program and information of the second program (includes the information as resources in the embodiment).

[0043]

As described above, a program for realizing the functions of the devices (e.g., the terminal devices 21 to 25) according to the embodiment may be recorded (stored) in a computer-readable recording medium (storage medium), and a computer system may be caused to read and execute the program recorded in the recording medium to perform processes. Meanwhile, “computer system” mentioned here may be a computer system including hardware such as an operating system, peripheral devices and the like.

In addition, “computer readable recording medium” refers to writable non-volatile memories such as a flexible disc, a magneto-optical disc, a read only memory (ROM) and a flash memory, portable media such as a digital versatile disc (DVD) and storage devices such as a hard disk embedded in computer systems.

Furthermore, “computer readable storage medium” may also include a medium which holds a program for a certain time, such as a volatile memory (e.g., DRAM) in a computer system serving as a server or a client in a case in which the program is

transmitted through a network such as the Internet or a communication line such as a telephone line.

In addition, the aforementioned program may be transmitted to other computer systems from the computer system in which the program is stored in a storage device or the like through a transmission medium or according to transmitted waves in the transmission medium. Here, "transmission medium" transmitting the program refers to a medium having a function of transmitting information, such as a network (communication network) such as the Internet or a communication circuit (communication line) such as a telephone line.

Furthermore, the aforementioned program may be a program for realizing some of the above-described functions. Further, the aforementioned program may be a program capable of realizing the aforementioned functions by being combined with a program recorded in advance in a computer system, so-called a differential file (differential program).

[0044]

Meanwhile, although the present invention has been described using the embodiment, the technical scope of the present invention is not limited to the above-described embodiment. It is obvious to those skilled in the art that various modifications and employment of substitute aspects can be performed without departing from the spirit and scope of the present invention.

[Reference Signs List]

[0045]

11 Information processing system

21 to 25 Terminal device

31 Network

	101 Processor
	102 Memory
	103 Network interface
	104 Operation unit
5	105 Display unit
	106 Storage device
	107 Input/output interface
	111 Bus
	201 Program
10	221 AnyCPU resource
	222 x86 resource

[CLAIMS]

[Claim 1]

A program for causing a computer to perform:

5 a first function of identifying the type of a process of a target program which is another program corresponding to an object to be operated by the program depending on a platform in which the target program operates;

a second function of determining whether the type of the process of the target program identified by the first function matches a first type of a process of the program; and

10 a third function of starting a second type of a process of the program matching the type of the process of the target program and ending the first type of the process of the program when it is determined that the type of the process of the target program does not match the first type of the process of the program according to the second function.

[Claim 2]

15 A program for causing a computer to perform:

a first function of identifying the type of a process of a target program which is another program corresponding to an object to be operated by the program depending on a platform in which the target program operates;

20 a second function of determining whether the type of the process of the target program identified by the first function matches a first type of a process of the program; and

a third function of generating a program file of the program matching the type of the process of the target program and starting a process of a second type according to the generated program file when it is determined that the type of the process of the target
25 program does not match the first type of the process of the program according to the

second function.

[Claim 3]

The program according to claim 2, wherein the third function starts a process of the second type of the program and ends the process of the first type of the program.

5 [Claim 4]

The program according to any one of claims 1 to 3, wherein the program has resources of programs according to the process of the first type and the process of the second type.