

[DESCRIPTION]

[TITLE OF INVENTION]

PROGRAM

[Technical Field]

5 [0001]

The present invention relates to a program.

Priority is claimed on Japanese Patent Application No. 2016-064497, filed March 28, 2016, the content of which is incorporated herein by reference.

[Background Art]

10 [0002]

For example, a relocation program which executes a process relating to other programs is known (refer to Patent Literature 1). The relocation program executes a process of exchanging locations on a first program located in a high-speed memory and a second program located in a low-speed memory (refer to claim 1 of Patent Literature 1).

15 [Citation List]

[Patent Literature]

[0003]

[Patent Literature 1]

Japanese Unexamined Patent Application, First Publication No. 2012-194852

20 [Summary of Invention]

[Technical Problem]

[0004]

However, in conventional programs, when a program which executes a process to a program itself (hereinafter referred to as “host program”) is generated, there are cases in which the program remains.

25

[0005]

An object of embodiments of the present invention devised in view of the aforementioned circumstances is to provide a program capable of preventing a program which executes a process to a host program from remaining.

5 [Solution to Problem]

[0006]

A program according to an embodiment of the present invention is a program for causing a computer to perform a first function of generating a first program which performs an operation to the program as a memory-mapped file and a second function of
10 executing the first program generated as the memory-mapped file.

[0007]

In the program according to one aspect of the present invention, a configuration in which the second function generates a loader program which executes the first program and causes the generated loader program to execute the first program may be
15 used.

[0008]

In the program according to one aspect of the present invention, a configuration in which the first program registers an instruction for deletion of the loader program according to an operating system may be used.

20 [0009]

In the program according to one aspect of the present invention, a configuration in which the first program performs an operation of deleting the program as the operation to the program may be used.

[0010]

25 In the program according to one aspect of the present invention, a configuration

in which the first program performs an operation of modifying the program as the operation to the program may be used.

[Advantageous Effects of Invention]

[0011]

5 According to the aforementioned program, it is possible to prevent a program which executes a process to a host program from remaining.

[Brief Description of Drawings]

[0012]

10 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 schematic images of a program according to an embodiment of the present invention.

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

 Fig. 5 is a flowchart illustrating an example of a processing procedure performed using a loader program according to an embodiment of the present invention.

20 Fig. 6 is a flowchart illustrating an example of a processing procedure performed using a self-processing program according to an embodiment of the present invention.

[Description of Embodiments]

[0013]

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

25 [0014]

[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.

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.

[0015]

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 is arbitrary.

In addition, although one network 31 is illustrated in the example of Fig. 1, the number of networks is 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 examples.

[0016]

[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.

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 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.

[0017]

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.

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).

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 disk 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.

[0018]

[Description of program according to embodiment]

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

The program executes code without outputting a program (hereinafter referred to as a “self-processing program”) which is an execution target as a physical file.

The present embodiment represents a case in which a program (deletion program) having a function of deleting a self-program (the program) is used as a

5 self-processing program. In addition, deletion of the program is executed without the deletion program remaining.

[0019]

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

10 Images (1) to (5) of a schematic process performed by the program 201 are illustrated.

(1) The program 201 generates a loader program 211. Here, the program 201 and the loader program 211 exist as normal files.

(2) The program 201 generates a self-processing program 221 as a
15 memory-mapped file. In the present embodiment, the memory-mapped file is generated using the memory 102.

(3) The loader program 211 executes the self-processing program 221.

(4) The self-processing program 221 executes a process determined to the
program 201. In the present embodiment, a process of deleting the program 201 is used
20 as the process.

(5) The self-processing program 221 registers an instruction for deletion of the loader program 211.

Meanwhile, the timing of the process of generating the loader program 211 and the timing of the process of generating the self-processing program 221 may be reversed
25 or may be simultaneous (or approximately simultaneous).

[0020]

Here, the memory-mapped file is a structure capable of storing data on a memory and sharing the data between processes. The memory-mapped file may be mapped with a physical file and may not be mapped with a physical file.

5 In the present embodiment, a usage in which the memory-mapped file is not mapped with a physical file is employed.

Further, a generally known technology, for example, may be used for the memory-mapped file.

[0021]

10 Hereinafter, the program 201 will be further described.

First, a process when the program 201 is created will be described.

A program (self-processing program 221) which performs a necessary self-processing is created. In the present embodiment, the self-processing program 221 has a function of executing deletion of the program 201 (command execution program
15 which is a program corresponding to a caller of self-processing) and deletion of a related program.

In the present embodiment, the self-processing program 221 is used to delete the program 201 and thus cannot delete a file of the program 201 until the process of the program 201 ends. Accordingly, the self-processing program 221 has a function of
20 continuing retrying at certain intervals when deletion of the program 201 fails.

[0022]

In addition, the loader program 211 is created. The loader program 211 reads data from the memory-mapped file, executes a predetermined conversion for restoration as necessary and executes binary values after the conversion as a program on a memory.

25 In addition, when the program 201 is built, binary values of the self-processing

program 221 and the loader program 211 are embedded as resources.

[0023]

Here, it is necessary to perform conversion for restoration in accordance with a format in which the binary values of the self-processing program 221 and the like as the resources are retained. Such conversion for restoration is the aforementioned
5 predetermined conversion for restoration performed by the loader program 211.

Accordingly, when the binary values of the self-processing program as the resources are retained as it is (without being converted), the aforementioned predetermined conversion for restoration performed by the loader program 211 is not necessary.

10 Meanwhile, data of a program without convention or values inversely converted under a specific condition may be used as binary values, for example. As the conversion, for example, conversion of bit inversion, conversion of lossless compression or the like may be used.

[0024]

15 In addition, in the memory-mapped file, data is generally stored in units of a page size. Accordingly, there are cases in which original binary values cannot be correctly acquired. Accordingly, the present embodiment employs a configuration in which a file size of binary values which is a reading target is received as an argument of the loader program 211 and data (binary values) corresponding to the received file size is
20 read from the memory-mapped file.

[0025]

Next, an example of a process performed in a program according to the present embodiment will be described with reference to Figs. 4 to 6.

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

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

[0026]

5 (Step S1)

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

[0027]

(Step S2)

When the program 201 is executed, the program 201 generates the loader
10 program 211 by outputting the loader program 211 as a file at a predetermined timing. The predetermined timing may be any timing. For example, the predetermined timing may be a timing designated in the program 201 or a timing when there was a predetermined instruction generated outside the program 201. In the present embodiment, the process after the step S2 is executed from the predetermined timing.

15 [0028]

Meanwhile, when data obtained by performing a predetermined conversion on binary values of the loader program 211 is retained in the program 201, the program 201 stores data (data in an executable format) obtained by performing an inverse conversion (conversion for restoration) of the predetermined conversion on the data in a file. In
20 addition, the program 201 outputs the file to a temp folder in the present embodiment.

[0029]

(Step S3)

The program 201 generates the self-processing program 221 by storing binary values of the self-processing program 221 in the memory-mapped file.

25 [0030]

Meanwhile, when data obtained by performing a predetermined conversion on the binary values of the self-processing program 221 is retained in the program 201, the program 201 may store data (data in an executable format) obtained by performing an inverse conversion (conversion for restoration) of the predetermined conversion on the data in the memory-mapped file. In this case, the inverse conversion has already been performed and thus the aforementioned predetermined conversion for restoration performed by the loader program 211 is not necessary.

[0031]

(Step S4)

The program 201 executes the loader program 211. Here, the program 201 transfers information of the name of the memory-mapped file, the binary size of the self-processing program 221 and an argument necessary to execute the self-processing program 221 to the loader program 211 to execute the loader program 211.

[0032]

(Step S5)

In the terminal device 21, the program 201 ends.

Here, when the program 201 ends before the loader program 211 reads information (self-processing program 221) from the memory-mapped file, information of the memory-mapped file is also deleted in the present embodiment. Accordingly, the program 201 waits until the loader program 211 completes the process of reading information from the memory-mapped file, and thus a configuration in which the program 201 ends after waiting for a predetermined time (allows about 2 seconds freely, for example) is employed.

[0033]

Fig. 5 is a flowchart illustrating an example of a processing procedure performed

using the loader program 211 according to an embodiment of the present invention.

[0034]

(Step S11)

The loader program 211 is executed according to the process of step S4.

5 [0035]

(Step S12)

When the loader program 211 is executed, the loader program 211 reads the self-processing program 221 from the memory-mapped file and executes the self-processing program 221.

10 [0036]

Fig. 6 is a flowchart illustrating an example of a processing procedure performed using the self-processing program 221 according to an embodiment of the present invention.

[0037]

15 (Step S21)

The self-processing program 221 is executed according to the process of step S12.

[0038]

(Step S22)

20 When the self-processing program 221 is executed, the self-processing program 221 executes a process determined for the program 201. In the present embodiment, the process is a deletion process and the self-processing program 221 deletes the program 201.

[0039]

25 (Step S23)

The self-processing program 221 registers an instruction for deletion of the loader program 211 after the aforementioned process (deletion process in the present embodiment) is completed. In the present embodiment, this registration is performed by a predetermined command, specifically, WINAPI (MoveFileEx function).

5 Accordingly, execution of a program (the self-processing program 221 in the present embodiment) which does not have the state of a file is done.

[0040]

 Here, a file for which instruction of deletion has been registered (the loader program 211 in the present embodiment) is deleted by an operating system (OS) when
10 the OS is rebooted.

 Further, the process of the self-processing program 221 operates as the process of the loader program 211 and thus the file of the loader program 211 cannot be deleted in the process of the self-processing program 221. Accordingly, the present embodiment employs a configuration in which the self-processing program 221 registers
15 the instruction for deletion of the loader program 211 in the OS. In this configuration, although the loader program 211 remains until the OS is rebooted after executing the process, the loader program 211 does not include the contents of a process to be executed and thus the contents of an executed process are not confirmed even if analysis of the loader program 211 is performed.

20 [0041]

[Summary of embodiment]

 As described above, in the program (program 201) according to the present embodiment, the program (self-processing program 221) which executes the process for the host program may not remain.

25 [0042]

Here, although the program which deletes the program 201 is used as the self-processing program in the present embodiment, a program having various functions such as modifying or updating the program 201 may be used as other configuration examples.

5 [0043]

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

As another example, the program (program 201) according to the present
15 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 interface 107.

[0044]

In addition, although a configuration in which the terminal device 21 executes
20 the program 201 is represented in the present embodiment in the terminal device, the program 201 may be executed by a device outside the terminal device 21 as another configuration example.

[0045]

Further, as another configuration example, a configuration in which the binary
25 values of the self-processing program 221 are transmitted according to network

communication (to the terminal device 21 in which the program 201 exists) instead of being embedded in the program 201 to be transferred to the program 201 or a configuration in which the binary values of the self-processing program 221 are transferred to the program 201 according to other methods may be used.

5 [0046]

As a configuration example, a program (the program 201 in the embodiment) causes a computer to perform a first function (the function of step S3 in the example of Fig. 4) of generating a first program (which is the self-processing program 221 in the embodiment and has the function of step S22 in the example of Fig. 6) which performs
10 an operation (deletion in the embodiment) to the program as a memory-mapped file and a second function (the function of step S4 in the example of Fig. 4) of executing the first program generated as the memory-mapped file.

As a configuration example, the second function generates a loader program (the loader program 211 in the embodiment) which executes the first program (the function of
15 step S2 in the example of Fig. 4) and causes the generated loader program to execute the first program (the function of step S12 in the example of Fig. 5).

As a configuration example, the first program registers an instruction for deletion of the loader program according to an operating system (OS) (the function of step S23 in the example of Fig. 6).

20 As a configuration example, the first program performs an operation of deleting the program (program 201) as the operation to the program (program 201).

As a configuration example, the first program performs an operation of modifying the program (program 201) as the operation to the program (program 201).

[0047]

25 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
5 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 disk, 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.

10 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.

15 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
20 (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
25 program recorded in a computer system, a so-called differential file (differential

program).

[0048]

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]

[0049]

- | | |
|----|----------------------------------|
| 10 | 11 Information processing system |
| | 21 to 25 Terminal device |
| | 31 Network |
| | 101 Processor |
| | 102 Memory |
| 15 | 103 Network interface |
| | 104 Operation unit |
| | 105 Display unit |
| | 106 Storage device |
| | 107 Input/output interface |
| 20 | 111 Bus |
| | 201 Program |
| | 211 Loader program |
| | 221 Self-processing program |

[CLAIMS]

[Claim 1]

A program for causing a computer to perform:

a first function of generating a first program which performs an operation to the

5 program as a memory-mapped file; and

a second function of executing the first program generated as the

memory-mapped file.

[Claim 2]

The program according to claim 1, wherein the second function generates a

10 loader program which executes the first program and causes the generated loader
program to execute the first program.

[Claim 3]

The program according to claim 2, wherein the first program registers an
instruction for deletion of the loader program according to an operating system.

15 [Claim 4]

The program according to any one of claims 1 to 3, wherein the first program
performs an operation of deleting the program as the operation to the program.

[Claim 5]

The program according to any one of claims 1 to 3, wherein the first program

20 performs an operation of modifying the program as the operation to the program.