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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 June 2008 (12.06.2008)

PCT

(10) International Publication Number
WO 2008/069431 A1

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/KR2007/005029

(22) International Filing Date: 15 October 2007 (15.10.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10-2006-0124059
7 December 2006 (07.12.2006) KR

(71) Applicant *(for all designated States except US)*: **ELECTRONICS AND TELECOMMUNICATIONS RESEARCH INSTITUTE** [KR/KR]; 161 Gajeong-dong, Yuseong-gu, Daejeon 305-350 (KR).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **KIM, Hong Soog** [KR/KR]; Electronics and Telecommunications, Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon 305-350 (KR). **PARK, Nam Hoon** [KR/KR]; Electronics and Telecommunications, Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon 305-350 (KR).

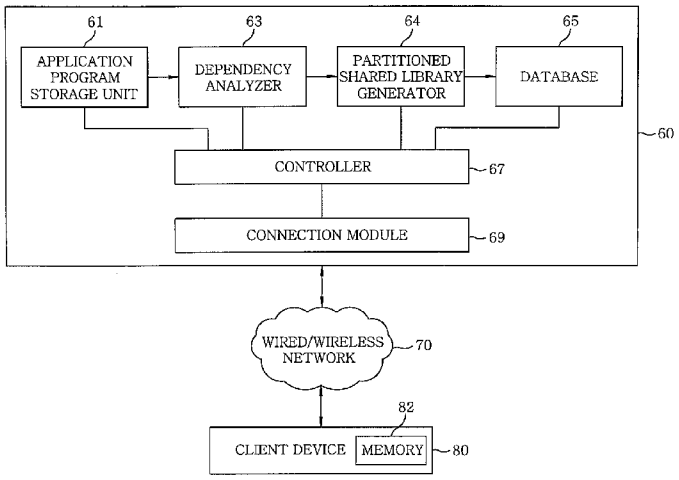
(74) Agent: **JANG, Seong Ku**; 19th Fl., Trust Tower, 275-7, Yangjae-dong, Seocho-gu, Seoul 137-130 (KR).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: DOWNLOAD SERVER AND METHOD FOR INSTALLING AND UPDATING APPLICATION PROGRAM USING PARTITIONING OF SHARED LIBRARY



(57) Abstract: A download server that transmits, when it receives a request of one or more application programs from a client device, to the client device connected to the download server via a wired or wireless network the application programs includes a dependency analyzer, a partitioned shared library generator and a controller. The dependency analyzer that analyzes a dependency of each of the application programs on existing shared libraries to detect object codes, which are actually used by corresponding application program, in the existing shared libraries; the partitioned shared library generator that generates, based on the analyzing result of the dependency analyzer, partitioned shared libraries having only the actually-used object codes; and the controller that transmits, by using a connection module, partitioned shared libraries selected from the generated partitioned shared libraries, and the application programs linked thereto to the client device via the network.

WO 2008/069431 A1

Published:
— *with international search report*

Description

DOWNLOAD SERVER AND METHOD FOR INSTALLING AND UPDATING APPLICATION PROGRAM USING PARTITIONING OF SHARED LIBRARY

Technical Field

[1] The present invention relates to an application program installation technique. In particular, the invention relates to a download server that minimizes download traffic by means of partitioning shared libraries used by an application program and using partitioned shared libraries to install and update the application program, and a method for installing and updating an application program using partitioning of shared libraries.

[2] This work was supported by the IT R&D program of MIC/IITA. [2006-S-012-01, Development of Middleware Platform Technology based on the SDR Mobile Station]

[3]

Background Art

[4] In case of using a static library, when fifty to a hundred processes, for example, are run on a system, copies of executable codes for necessary functions are needed for each process, which causes a significant waste of a memory.

[5] A shared library is an object module capable of solving such drawback in the static library. The shared library can be loaded to a certain location in a memory at runtime and linked to programs in the memory. The shared library is often called as a shared object. A file name of the shared library is ended with '.so' in most Unix systems and '.sl' in HP-UX systems. Microsoft Corporation calls the shared library as a Dynamic Link Library (DLL).

[6] To solve a problem in the static library technique that object codes shared by two or more application programs are stored in a memory repeatedly, a shared library technique minimizes the use of the memory by gathering the shared object codes into a shared library and sharing them on the memory. Accordingly, the shared library includes all object codes on which application programs are likely to have a dependency.

[7] Fig. 1 illustrates application programs having a dependency on conventional shared libraries.

[8] Referring to Fig. 1, conventional application programs App1 and App2 have a dependency on one or more shared libraries libA.so, libB.so, and libC.so. For convenience of explanation, the application programs App1 and App2 shown in Fig. 1 have a dependency on two shared libraries libA.so and libB.so and shared libraries libB.so and libC.so, respectively, however, the application programs may have a dependency on two or more libraries.

[9] In Fig. 1, the application program App1 has a dependency on object codes a1.o and a2.o in the shared library libA.so and an object code b3.o in the shared library libB.so, thus resulting in having a dependency on the libraries libA.so and libB.so that include the object codes on which the application program App1 has a dependency. Likewise, the application program App2 has a dependency on the object code b3.o in the shared library libB.so and an object code c1.o in the shared library libC.so, resulting in having a dependency on the libraries libB.so and libC.so that include the object codes on which the application program App2 has a dependency.

[10] As such, the conventional shared library results in an exaggerated dependency (i.e., the dependency on the shared library which includes the object codes on which the application programs have a dependency) in comparison to an actual dependency (the dependency on the object codes). This is because object codes on which application programs are likely to have a dependency are included in a single shared library.

[11] Accordingly, though most application programs use only some of the object codes included in the shared library, the entire shared library which also includes object codes being not used needs to be linked.

[12] In case of using such conventional shared library, non-used object codes as well as actually-used object codes in the shared library will be downloaded and unnecessarily occupy a memory of a client device in which the application programs will be installed. Further, when only some of object codes forming the application program are updated, since the updated object codes cannot be downloaded separately, the entire shared library including the updated object codes is required to be downloaded.

[13] The above-described problems are not significant when a device in which an application program is installed, e.g., a personal computer, has a sufficient memory space, or when an application program is installed via a storage medium instead of downloading. On the contrary, as in a mobile device, when an application program is installed or updated in a limited memory via downloading, a cost increase due to an unnecessary use of a communications line or a memory lack in running the application program can be generated.

[14]

Disclosure of Invention

Technical Problem

[15] In view of the above, the present invention provides a download server capable of reducing download traffic of an application program and saving memory space of a client device in which the application program is installed, by partitioning shared libraries to generate partitioned shared libraries having object codes actually used by the application program and transmitting the partitioned shared libraries and the application program linked thereto to the client device.

[16] The present invention is also provide a method for installing and updating an application program by using partitioning of shared libraries, which is capable of reducing download traffic of the application program and saving memory space of a client device in which the application program is installed, by partitioning, in response to an installation or an update request from the client device, the shared libraries to generate partitioned shared libraries having object codes actually used by the application program; linking the application program to the partitioned shared libraries; and transmitting the partitioned shared libraries and the application program to the client device.

[17]

Technical Solution

[18] In accordance with a first aspect of the present invention, there is provided a download server that transmits, when it receives a request of one or more application programs from a client device, to the client device connected to the download server via a wired or wireless network the application programs, the download server including:

[19] a dependency analyzer that analyzes a dependency of each of the application programs on existing shared libraries to detect object codes, which are actually used by corresponding application program, in the existing shared libraries;

[20] a partitioned shared library generator that generates, based on the analyzing result of the dependency analyzer, partitioned shared libraries having only the actually-used object codes; and

[21] a controller that transmits, by using a connection module, partitioned shared libraries selected from the generated partitioned shared libraries, and the application programs linked thereto to the client device via the network.

[22] In accordance with a second aspect of the present invention, there is provided a method for installing an application program using partitioning of shared libraries, wherein a download server transmits the application program to a client device via a wired or wireless network when the client device sends an installation request of the application program, the method including:

[23] determining whether the application program is linked to partitioned shared libraries which only have object codes actually used by the application program;

[24] generating, when the application program is not linked to the partitioned shared libraries, the partitioned shared libraries having only the actually-used object codes and linking the application program to the generated partitioned shared libraries; and

[25] transmitting, when the application program is linked to the partitioned shared libraries, the partitioned shared libraries and the application program from the download server to the client device.

[26] In accordance with a third aspect of the present invention, there is provided a method for updating an application program using partitioning of shared libraries, wherein a download server transmits the application program to a client device via a wired or wireless network when the client device sends an update request of the application program, the method including:

[27] determining whether the application program is linked to partitioned shared libraries which only have object codes actually used by the application program;

[28] generating, when the application program is not linked to the partitioned shared libraries, the partitioned shared libraries having only the actually-used object codes and linking the application program to the generated partitioned shared libraries; and

[29] transmitting, when the application program is linked to the partitioned shared libraries, the partitioned shared libraries and the application program from the download server to the client device.

[30]

Advantageous Effects

[31] As described above, in accordance with the present invention, a download server partitions shared libraries used by an application program, in response to an installation or update request from a client device, to generate partitioned shared libraries being in one-to-one correspondence with actually-used object codes. The application program is then linked to the partitioned shared libraries, and transmitted to the client device.

[32] Therefore, in accordance with the present invention, it is possible to reduce

download traffic of the application program and save memory space of the client device in which the application program is installed.

[33] Furthermore, in accordance with the present invention, when some of the object codes used by the application program are updated, only the updated object codes are downloaded, thus preventing unnecessary downloading and facilitating dependency management between the updated object codes and the application programs using the object codes.

[34]

Brief Description of the Drawings

[35] The objects and features of the present invention will become apparent from the following description of embodiments given in conjunction with the accompanying drawings, in which:

[36] Fig. 1 illustrates application programs having a dependency on conventional shared libraries;

[37] Fig. 2 is a block diagram schematically illustrating a client device and a download server that provides partitioned shared libraries and application programs in accordance with the present invention;

[38] Fig. 3 illustrates a flowchart of a process in which a download server generates partitioned shared libraries in accordance with the present invention;

[39] Fig. 4 illustrates an example of a dependency relationship table, stored in a download server, between application programs and partitioned shared libraries in accordance with the present invention;

[40] Fig. 5 illustrates application programs linked to partitioned shared libraries in accordance with the present invention;

[41] Fig. 6 illustrates a flowchart of a process in which a download server transmits an application program to a client device to install the application program therein using partitioned shared libraries in accordance with the present invention; and

[42] Fig. 7 illustrates a flowchart of a process in which a download server updates an application program in a client device using partitioned shared libraries in accordance with the present invention.

[43]

Best Mode for Carrying Out the Invention

[44] Hereinafter, embodiments of the present invention will be described in detail with reference to the accompanying drawings so that they can be readily implemented by

those skilled in the art. In the detailed description, "A has a dependency on B" or "A depends on B" means that "B must exist at a location accessible to A in order for A to achieve a desired purpose". Further, an "application program" is limited to an application program using shared libraries or both static libraries and the shared libraries, and does not indicates an application program using only static libraries.

[45] Fig. 2 is a block diagram schematically illustrating a client device and a download server that provides partitioned shared libraries and application programs in accordance with the present invention.

[46] A download server 60 of the present invention includes an application program storage unit 61 for storing therein application programs to be provided to a client device 80, such as a personal computer, a mobile device, and the like; a dependency analyzer 63 for analyzing a dependency of each application program on existing shared libraries to detect object codes actually used by the application program; a partitioned shared library generator 64 for partitioning, based on the analyzing result of the dependency analyzer 63, the existing shared libraries to generate partitioned shared libraries having only actually-used object codes; a database 65 for storing therein mapping information between the existing shared libraries and the partitioned shared libraries and dependency information of each application program on the existing shared libraries and the partitioned shared libraries (e.g., a version, depending object codes, and the like); and a controller 67 for controlling the components of the server to generate the partitioned shared libraries in response to a request from the client device 80 and download (transmit) the partitioned shared libraries and the application program to the client device 80.

[47] The download server 60 further includes a connection module 69 for transmitting data to the client device 80 via a wired/wireless network 70.

[48] The client device 80 stores the partitioned shared libraries and the application programs downloaded from the download server 60 in a memory 82, and installs or updates the partitioned shared libraries and the application programs.

[49] In the download server 60 and the client device 80 having the above-described configuration, when the client device 80 connected to the download server 60 sends an installation or update request of a specific application program via the wired/wireless network 70, the dependency analyzer 63 in the download server 60 analyzes the dependency of the requested application program on the existing shared libraries, and then, the partitioned shared library generator 64 generates, based on the analyzing result, the partitioned shared libraries having only the object codes actually used by the

application program. The download server 60 stores in the database 65 the information on the shared libraries on which the application program depends and the partitioned shared libraries having the object codes on which the application program actually depends.

[50] The download server 60 links the requested application program to the partitioned shared libraries to thereby make the application program use only the partitioned shared libraries, and transmits the application program and the partitioned shared libraries to the client device 80 via the wired/wireless network 70.

[51] Fig. 3 illustrates a flowchart of a process in which a download server generates partitioned shared libraries in accordance with the present invention.

[52] Referring to Fig. 3, when the client device sends an installation or update request of an application program, the download server partitions shared libraries used by the application program and links the application program thereto.

[53] First, the download server checks the application program requested by the client device and shared libraries on which the application program depends (steps S100 and S110).

[54] The download server checks object codes, on which the application program actually depends, in one of the shared libraries (step S120).

[55] The download server records information on the shared library on which the application program depends and the object codes on which the application program actually depends, in a dependency relationship table in a database (step S130).

[56] The download server determines whether there is another shared library on which the application program depends (step S140).

[57] If it is determined in the step S140 that there is another shared library on which the application program depends, the download server repeatedly performs the steps S120 to S140 on the shared library on which the application program additionally depends. If it is determined in the step S140 that there is no more shared library on which the application program depends, the download server determines whether there is another application program requested to be installed or updated by the client device (step S150).

[58] If it is determined in the step S150 that there is another application program to be installed or updated, the download server returns to perform the step S110 in which the shared libraries on which the application program depends are checked.

[59] If it is determined in the step S150 that there is no more application program to be installed or updated, the download server determines that analysis of dependency in-

formation of the application programs to be installed or updated, which is linked to the existing shared libraries, has been completed. The download server partitions the existing shared libraries to generate partitioned shared libraries being in one-to-one correspondence with the object codes recorded in the dependency relationship table (step S160).

[60] The download server links each application program recorded in the dependency relationship table to the partitioned shared libraries to make the application program use only the partitioned shared libraries (step S170).

[61] Fig. 4 illustrates an example of a dependency relationship between application programs and partitioned shared libraries in a download server in accordance with the present invention. In Fig. 4, a dependency relationship table of the application programs of Fig. 1 is illustrated.

[62] Referring to Fig. 4, the dependency relationship table, which records information on shared libraries on which two application programs App1 and App2 depend and object codes on which the application programs actually depend, includes an application program name field, a version field, a depending shared library name field, a depending object code name field, and a partitioned shared library name field.

[63] The version field indicates a version of the application program for use in distinguishing the same named application programs having different program versions. By using the version field, the latest version of the application program can be checked when installing the application program and the difference in the dependency relationship between a previous and the latest version can be analyzed when updating the application program.

[64] The partitioned shared library name field in each row indicates a name of a partitioned shared library which is obtained by extracting an object code, indicated by the depending object code name field in the same row, from a shared library indicated by the depending shared library name field in the same row.

[65] In this example, inclusion relationship between partitioned shared libraries and existing shared libraries from which the partitioned shared libraries are generated is indicated by means of the names in the partitioned shared library name field. However, this is for convenience of explanation, and the method to indicate the inclusion relationship is not limited thereto.

[66] Fig. 5 illustrates application programs linked to partitioned shared libraries in accordance with the present invention.

[67] Referring to Fig. 5, each of application programs App1 and App2 has a dependency

on a plurality of shared libraries libA_a1.so, libB_a2.so, libB_b3.so and libC_c1.so.

[68] The application program App1 has a dependency on object codes a1.o, a2.o, and b3.o in the shared libraries libA_a1.so, libB_a2.so and libB_b3.so, respectively. The application program App2 has a dependency on the object code b3.o in the shared library libB_b3.so and an object code c1.o in the shared library libC_c1.so.

[69] In Fig. 1, the application program App1 depends on the shared libraries libA.so and libB.so. However, in the present invention shown in Fig. 5, the same application program App1 depends on the partitioned shared libraries libA_a1.so, libB_a2.so, and libB_b3.so. Also, the application program App2 in Fig. 1 depends on the shared libraries libB.so and libC.so, however, the application program App2 of the present invention depends on the partitioned shared libraries libB_b3.so and libC_c1.so.

[70] As described above, in accordance with the present invention, the application program depends on the partitioned shared libraries rather than the existing shared libraries, and thus, when installing or updating an application program via downloading, instead of the existing shared libraries including object codes that are not actually used, the partitioned shared libraries having only object codes actually used by the application program are downloaded. Therefore, download traffic can be reduced, and also, memory space of the client device in which the application program is installed can be saved.

[71] Fig. 6 illustrates a flowchart of a process in which a download server transmits an application program to a client device to install the application program therein using partitioned shared libraries in accordance with the present invention.

[72] A method for transmitting the application program to be installed in the client device from the download server using partitioning of the shared library according to the present invention will now be described with reference to Fig. 6.

[73] First, the download server determines whether an application program requested to be installed by the client device is linked to the partitioned shared libraries (step S200).

[74] If it is determined in the step S200 that the application program requested to be installed is not linked to the partitioned shared libraries, the download server performs the process shown in Fig. 3 on the requested application program to generate the partitioned shared libraries (S210). That is, if the current request for installing the application program is the first request to the download server, the application program linked to the partitioned shared libraries does not exist and accordingly, the process of Fig. 3 for generating the partitioned shared libraries is carried out. On the other hand, if the application program currently requested to be installed is an application program

previously requested to be installed by another client device, the application program linked to the partitioned shared libraries exists in the download server.

[75] If it is determined in the step S200 that the application program requested to be installed is linked to the partitioned shared libraries or the step S210 has been carried out, the download server checks partitioned shared libraries required for running the requested application program, by using the dependency relationship table shown in Fig. 4 (step S220).

[76] Since the partitioned shared libraries required for running the requested application program may include the partitioned shared libraries which have been already installed in the client device, the download server checks application programs already installed in the client device which sends the installation request, and based thereon, also checks partitioned shared libraries already installed in the client device (step S230). Here, in order to check the application programs already installed in the client device, the download server may hold, for each client device, a list of the application programs which have been installed in the client device by the download server; the download server may request, when needed, the client device to transmit a list of the application programs already installed therein; or the client device may transmit, when it sends the installation request the application programs, the list of application programs already installed therein to the download server.

[77] The download server obtains partitioned shared libraries to be installed (i.e., partitioned shared libraries to be transmitted to the client device) by excluding the partitioned shared libraries already installed from the partitioned shared libraries required for running the application program (step S240).

[78] The download server then transmits to the client device the partitioned shared libraries to be installed obtained in the step S240 together with the application program requested to be installed (step S250).

[79] The client device installs the partitioned shared libraries and the application program received from the download server according to an installation method of the device.

[80] Fig. 7 illustrates a flowchart of a process in which a download server updates an application program in a client device using partitioned shared libraries in accordance with the present invention.

[81] Referring to Fig. 7, the download server receives an update request of the application program from the client device and performs an update process of the partitioned shared libraries and the application program linked thereto.

[82] First, the download server determines whether the application program requested to

be updated by the client device is linked to the partitioned shared libraries (step S300).

[83] If it is determined in the step S300 that the application program requested to be installed is not linked to the partitioned shared libraries, the download server performs the process shown in Fig. 3 on the requested application program to generate the partitioned shared libraries (step S310). That is, if the current request for updating the application program is the first installation or update request to the download server, the application program linked to the partitioned shared libraries does not exist and accordingly, the process of Fig. 3 for generating the partitioned shared libraries is carried out. On the other hand, if the currently requested application program to be installed is an application program previously requested to be installed or updated by another client device, the application program linked to the partitioned shared libraries exists in the download server.

[84] If it is determined in the step S300 that the application program requested to be updated is linked to the partitioned shared libraries or the step S310 has been carried out, the download server checks partitioned shared libraries required for running the requested application program, by using the dependency relationship table shown in Fig. 4 (step S320). To be specific, if the client device makes a request for a specific version of the application program when sending the update request, the download server checks the partitioned shared libraries required for running corresponding version of the application program. If the client device does not make a request for a specific version of the application program, the download server checks the partitioned shared libraries required for running the latest version of the application program.

[85] Since the partitioned shared libraries required for running the application program may include the partitioned shared libraries which have been already installed in the client device, the download server checks application programs already installed in the client device which sends the update request, and based thereon, also checks partitioned shared libraries already installed in the client device (step S330). Here, in order to check the application programs already installed in the client device, the download server may hold, for each client device, a list of the application programs which have been installed in the client device by the download server; the download server may request, when needed, the client device to transmit a list of the application programs already installed therein; or the client device may transmit, when it sends the update request the application programs, the list of application programs already installed therein to the download server.

[86] The download server then compares the partitioned shared libraries required for

running the application program and the partitioned shared libraries already installed in the client device, thus checking partitioned shared libraries to be added, replaced, or deleted (step S340).

[87] Here, the partitioned shared libraries to be added include partitioned shared libraries which are needed to be newly installed in the client device in order to update the application program. The partitioned shared libraries to be deleted include partitioned shared libraries that can be safely deleted from the client device since the updated application program and other application programs installed in the client device does not depend on corresponding libraries while the application program of the previous version has a dependency thereon. The partitioned shared libraries to be replaced include partitioned shared libraries which are newly updated and used only by the updated application program and the application program of the previous version. That is, since other application programs installed in the client device do not depend on the partitioned shared libraries to be replaced, the partitioned shared libraries to be replaced includes a partitioned shared libraries capable of being immediately replaced for the application program to be updated. If another application program installed in the client device has a dependency on the partitioned shared libraries on which the updated application program's previous version depends, the updated partitioned shared libraries must be included in the partitioned shared libraries to be added, instead of the partitioned shared libraries to be replaced, to assure operation of the existing application programs installed in the client device.

[88] The download server then transmits to the client device, based on the checking result, the updated application program, the partitioned shared libraries to be added or replaced, a list of the partitioned shared libraries to be replaced, and a list of the partitioned shared libraries to be deleted (step S350).

[89] Thereafter, the client device installs, with reference to the list of the partitioned shared libraries to be replaced and the list of the partitioned shared libraries to be deleted, the partitioned shared libraries and the updated application program received from the download server according to an application program update procedure of the device.

[90] While the invention has been shown and described with respect to the embodiments, it will be understood by those skilled in the art that various changes and modifications may be made without departing from the scope of the invention as defined in the following claims.

Claims

[1] A download server that transmits, when it receives a request of one or more application programs from a client device, to the client device connected to the download server via a wired or wireless network the application programs, the download server comprising:
a dependency analyzer that analyzes a dependency of each of the application programs on existing shared libraries to detect object codes, which are actually used by corresponding application program, in the existing shared libraries;
a partitioned shared library generator that generates, based on the analyzing result of the dependency analyzer, partitioned shared libraries having only the actually-used object codes; and
a controller that transmits, by using a connection module, partitioned shared libraries selected from the generated partitioned shared libraries, and the application programs linked thereto to the client device via the network.

[2] The download server of claim 1, further comprising a database for storing therein dependency information of each of the application programs on the partitioned shared libraries.

[3] The download server of claim 1, wherein, when the request from the client device is an installation request, the selected partitioned shared libraries are partitioned shared libraries to be installed in the client device.

[4] The download server of claim 3, wherein the partitioned shared libraries to be installed are obtained by excluding partitioned shared libraries already installed in the client device from partitioned shared libraries required for running the application program.

[5] The download server of claim 1, wherein, when the request from the client device is an update request, the selected partitioned shared libraries are partitioned shared libraries to be added or replaced in the client device, and the controller further transmits to the client device a list of the partitioned shared libraries to be replaced and a list of partitioned shared libraries to be deleted in the client device.

[6] The download server of claim 5, wherein the partitioned shared libraries to be added, replaced or deleted are obtained by analyzing partitioned shared libraries required for running the application program, application programs already installed in the client device, and partitioned shared libraries already installed in

the client device.

[7] A method for installing an application program using partitioning of shared libraries, wherein a download server transmits the application program to a client device via a wired or wireless network when the client device sends an installation request of the application program, the method comprising:
determining whether the application program is linked to partitioned shared libraries which only have object codes actually used by the application program;
generating, when the application program is not linked to the partitioned shared libraries, the partitioned shared libraries having only the actually-used object codes and linking the application program to the generated partitioned shared libraries; and
transmitting, when the application program is linked to the partitioned shared libraries, the partitioned shared libraries and the application program from the download server to the client device.

[8] The method of claim 7, wherein generating the partitioned shared libraries includes:
checking object codes, on which the application program actually depends, in the shared libraries on which the application program depends;
recording information on the shared libraries on which the application program depends and the object codes on which the application program actually depends, in a dependency relationship table in a database;
partitioning the shared libraries to generate the partitioned shared libraries having only the object codes on which the application program actually depends; and
linking, based on the dependency relationship table, the application program to the partitioned shared libraries to use only the partitioned shared libraries.

[9] The method of claim 7, wherein transmitting the partitioned shared libraries and the application program includes:
checking partitioned shared libraries required for running the application program;
checking application programs already installed in the client device, and based thereon, partitioned shared libraries already installed in the client device;
obtaining partitioned shared libraries to be installed by excluding the partitioned shared libraries already installed from the partitioned shared libraries required for running the application program; and

transmitting to the client device the partitioned shared libraries to be installed and the application program.

[10] A method for updating an application program using partitioning of shared libraries, wherein a download server transmits the application program to a client device via a wired or wireless network when the client device sends an update request of the application program, the method comprising:

determining whether the application program is linked to partitioned shared libraries which only have object codes actually used by the application program;

generating, when the application program is not linked to the partitioned shared libraries, the partitioned shared libraries having only the actually-used object codes and linking the application program to the generated partitioned shared libraries; and

transmitting, when the application program is linked to the partitioned shared libraries, the partitioned shared libraries and the application program from the download server to the client device.

[11] The method of claim 10, wherein generating the partitioned shared libraries includes:

checking object codes, on which the application program actually depends, in the shared libraries on which the application program depends;

recording information on the shared libraries on which the application program depends and the object codes on which the application program actually depends, in a dependency relationship table in a database;

partitioning the shared libraries to generate the partitioned shared libraries having only the object codes on which the application program actually depends; and

linking, based on the dependency relationship table, the application program to the partitioned shared libraries to use only the partitioned shared libraries.

[12] The method of claim 10, wherein transmitting the partitioned shared libraries and the application program includes:

checking partitioned shared libraries required for running the application program;

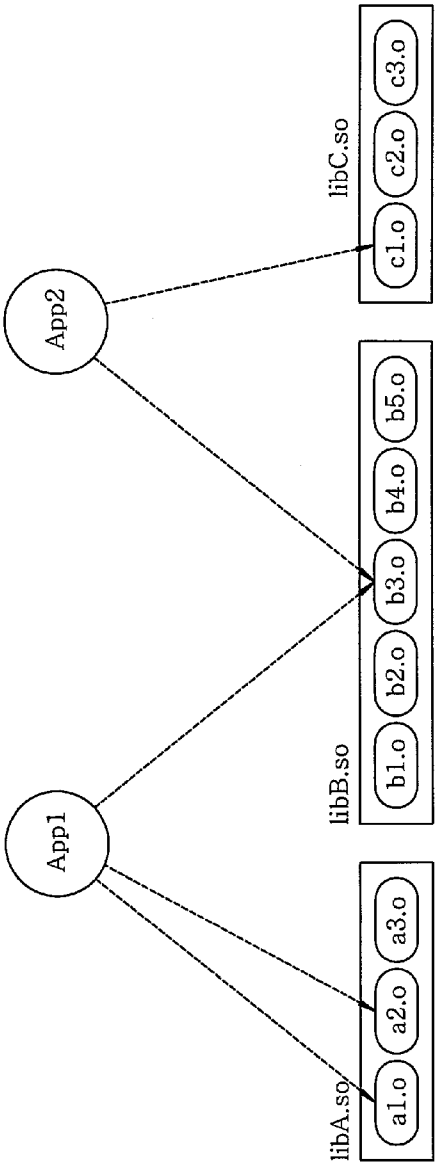
checking application programs already installed in the client device, and based thereon, partitioned shared libraries already installed in the client device;

checking partitioned shared libraries to be added, replaced, or deleted by analyzing the partitioned shared libraries required for running the application

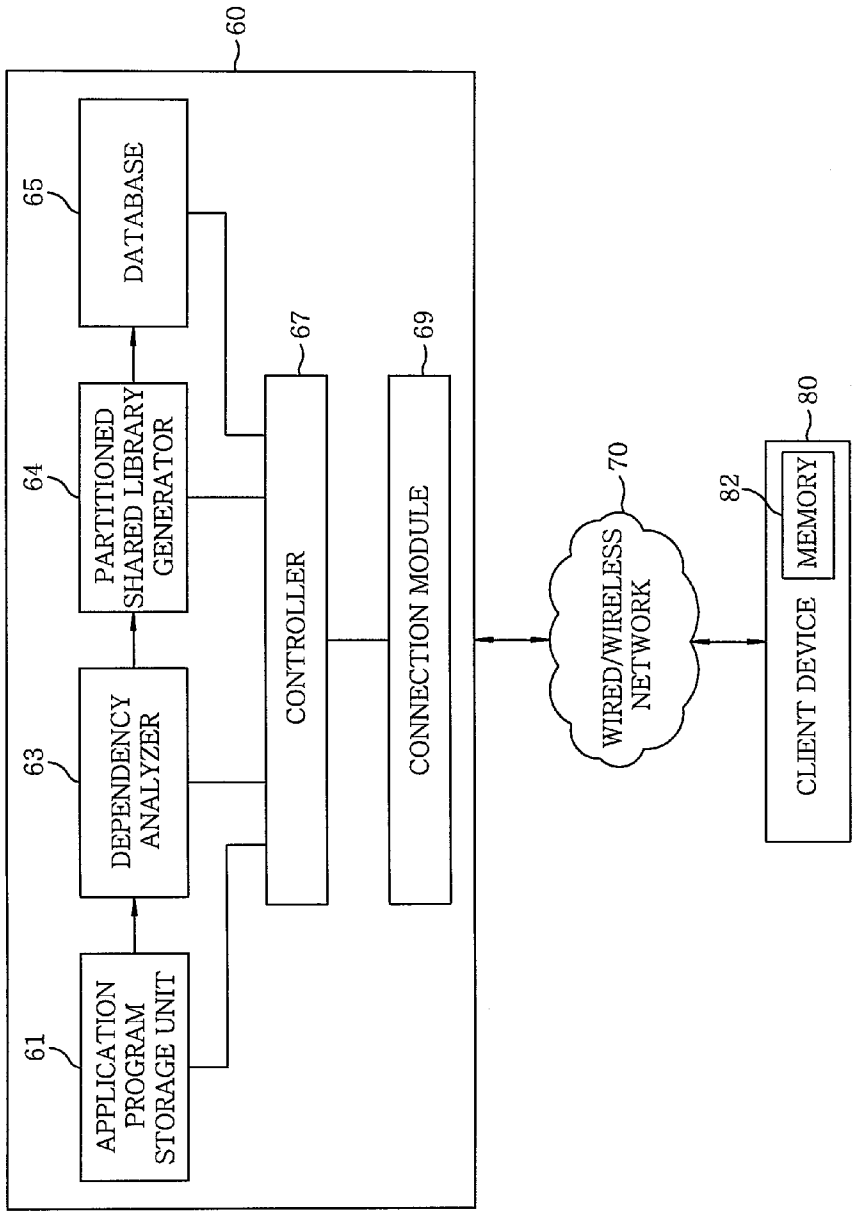
program, the application programs already installed in the client device, and the partitioned shared libraries already installed in the client device; and

transmitting to the client device the partitioned shared libraries to be added or replaced, a list of the partitioned shared libraries to be replaced, a list of the partitioned shared libraries to be deleted, and the application program.

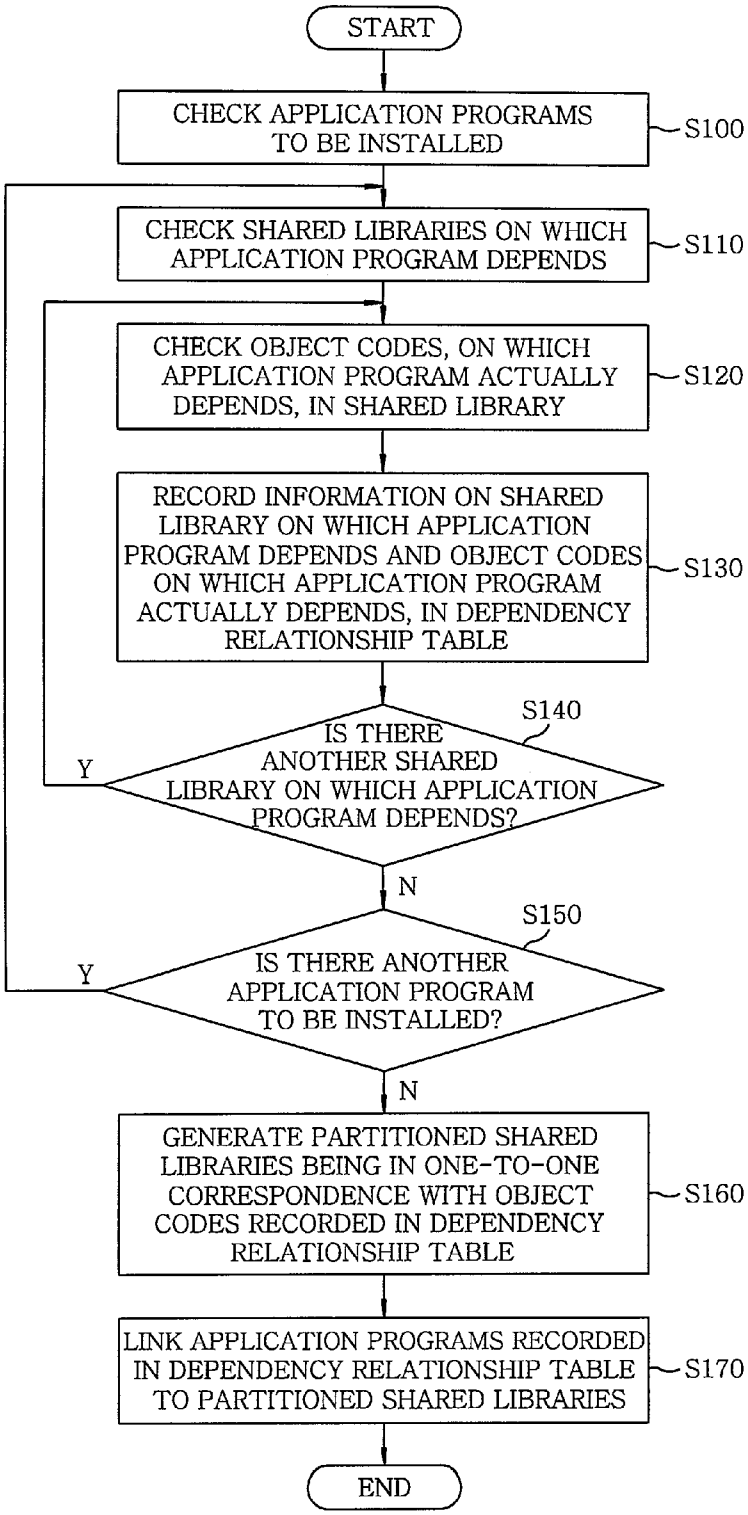
[Fig. 1]


App2
App1
libC.so
c1.o
c2.o
c3.o
libB.so
b1.o
b2.o
b3.o
b4.o
b5.o
libA.so
a1.o
a2.o
a3.o

[Fig. 2]


60
65
DATABASE
64
PARTITIONED SHARED LIBRARY GENERATOR
63
DEPENDENCY ANALYZER
61
APPLICATION PROGRAM STORAGE UNIT
67
CONTROLLER
69
CONNECTION MODULE
70
WIRED/WIRELESS NETWORK
82
80
MEMORY
CLIENT DEVICE

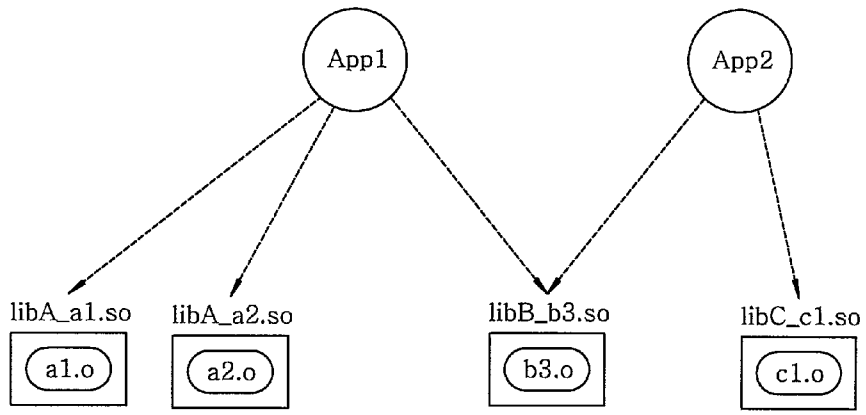
[Fig. 3]


START
CHECK APPLICATION PROGRAMS TO BE INSTALLED
S100
CHECK SHARED LIBRARIES ON WHICH APPLICATION PROGRAM DEPENDS
S110
CHECK OBJECT CODES, ON WHICH APPLICATION PROGRAM ACTUALLY DEPENDS, IN SHARED LIBRARY
S120
RECORD INFORMATION ON SHARED LIBRARY ON WHICH APPLICATION PROGRAM DEPENDS AND OBJECT CODES ON WHICH APPLICATION PROGRAM ACTUALLY DEPENDS, IN DEPENDENCY RELATIONSHIP TABLE
S130
S140
IS THERE ANOTHER SHARED LIBRARY ON WHICH APPLICATION PROGRAM DEPENDS?
Y
N
S150
IS THERE ANOTHER APPLICATION PROGRAM TO BE INSTALLED?
Y
N
GENERATE PARTITIONED SHARED LIBRARIES BEING IN ONE-TO-ONE CORRESPONDENCE WITH OBJECT CODES RECORDED IN DEPENDENCY RELATIONSHIP TABLE
S160
LINK APPLICATION PROGRAMS RECORDED IN DEPENDENCY RELATIONSHIP TABLE TO PARTITIONED SHARED LIBRARIES
S170
END

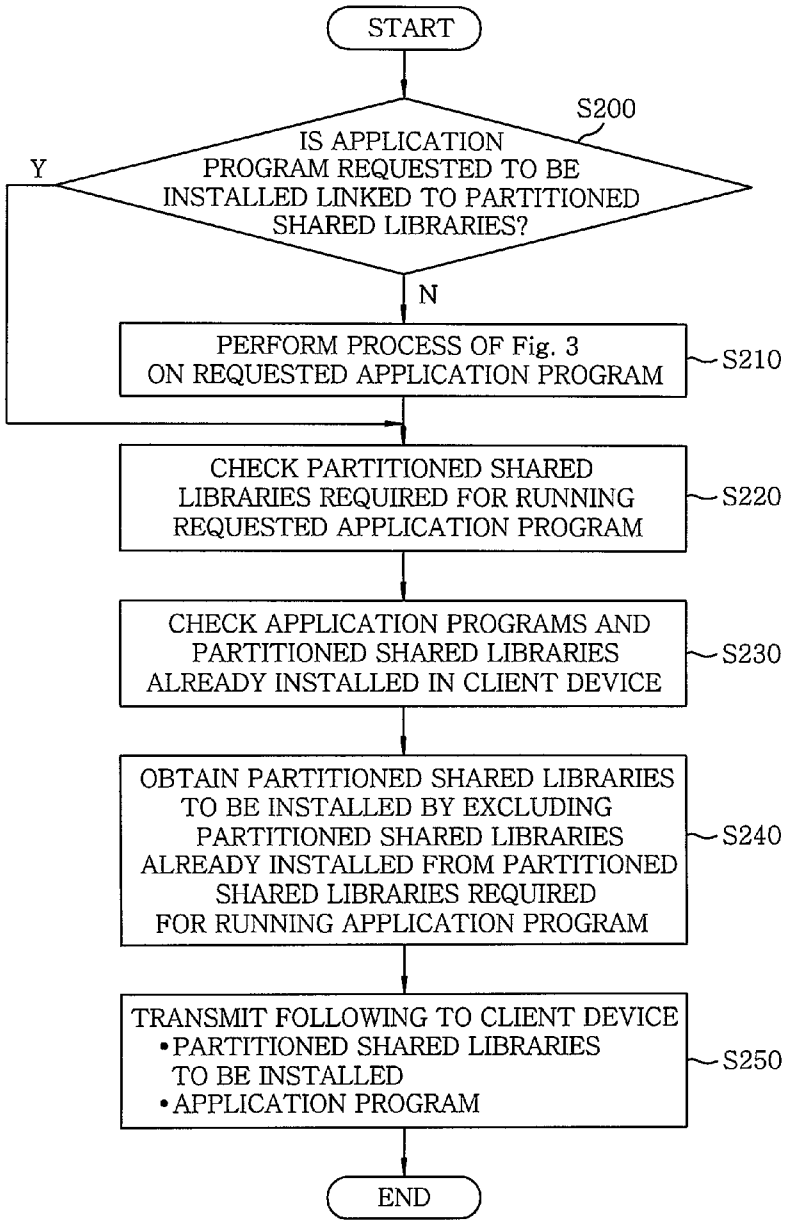
[Fig. 4]

APPLICATION PROGRAM NAME	VERSION	DEPENDING SHARED LIBRARY NAME	DEPENDING OBJECT CODE NAME	PARTITIONED SHARED LIBRARY NAME
App1	1.0	libA.so	a1.o	libA_a1.so
App1	1.0	libA.so	a2.o	libA_a2.so
App1	1.0	libB.so	b3.o	libB_b3.so
App2	1.0	libB.so	b3.o	libB_b3.so
App2	1.0	libC.so	c1.o	libC_c1.so

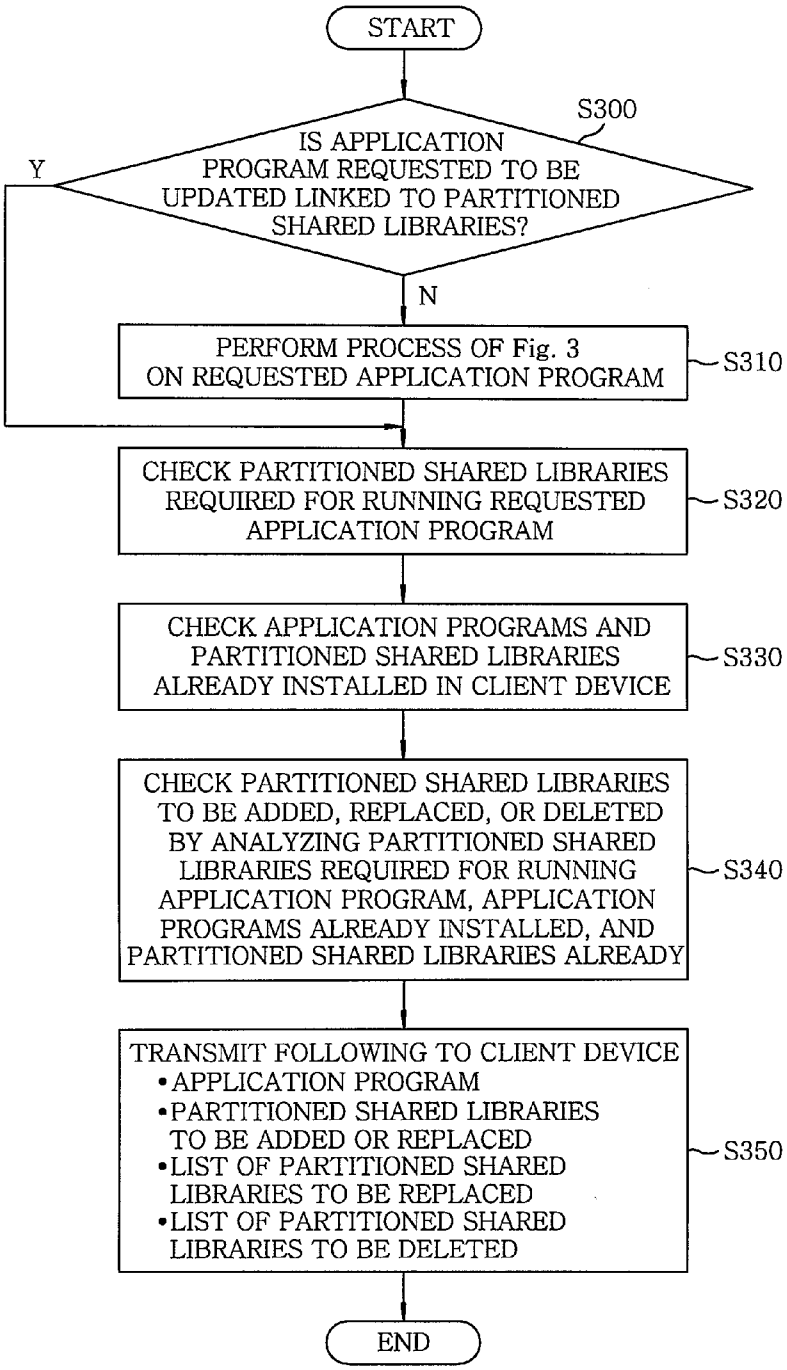
[Fig. 5]



[Fig. 6]


START
S200
IS APPLICATION PROGRAM REQUESTED TO BE INSTALLED LINKED TO PARTITIONED SHARED LIBRARIES?
Y
N
PERFORM PROCESS OF Fig. 3 ON REQUESTED APPLICATION PROGRAM
S210
CHECK PARTITIONED SHARED LIBRARIES REQUIRED FOR RUNNING REQUESTED APPLICATION PROGRAM
S220
CHECK APPLICATION PROGRAMS AND PARTITIONED SHARED LIBRARIES ALREADY INSTALLED IN CLIENT DEVICE
S230
OBTAIN PARTITIONED SHARED LIBRARIES TO BE INSTALLED BY EXCLUDING PARTITIONED SHARED LIBRARIES ALREADY INSTALLED FROM PARTITIONED SHARED LIBRARIES REQUIRED FOR RUNNING APPLICATION PROGRAM
S240
TRANSMIT FOLLOWING TO CLIENT DEVICE
• PARTITIONED SHARED LIBRARIES TO BE INSTALLED
• APPLICATION PROGRAM
S250
END

[Fig. 7]


START
S300
IS APPLICATION PROGRAM REQUESTED TO BE UPDATED LINKED TO PARTITIONED SHARED LIBRARIES?
Y
N
PERFORM PROCESS OF Fig. 3 ON REQUESTED APPLICATION PROGRAM
S310
CHECK PARTITIONED SHARED LIBRARIES REQUIRED FOR RUNNING REQUESTED APPLICATION PROGRAM
S320
CHECK APPLICATION PROGRAMS AND PARTITIONED SHARED LIBRARIES ALREADY INSTALLED IN CLIENT DEVICE
S330
CHECK PARTITIONED SHARED LIBRARIES TO BE ADDED, REPLACED, OR DELETED BY ANALYZING PARTITIONED SHARED LIBRARIES REQUIRED FOR RUNNING APPLICATION PROGRAM, APPLICATION PROGRAMS ALREADY INSTALLED, AND PARTITIONED SHARED LIBRARIES ALREADY
S340
TRANSMIT FOLLOWING TO CLIENT DEVICE
•APPLICATION PROGRAM
•PARTITIONED SHARED LIBRARIES TO BE ADDED OR REPLACED
•LIST OF PARTITIONED SHARED LIBRARIES TO BE REPLACED
•LIST OF PARTITIONED SHARED LIBRARIES TO BE DELETED
S350
END

INTERNATIONAL SEARCH REPORT

International application No.
PCT/KR2007/005029

A. CLASSIFICATION OF SUBJECT MATTER

G06F 15/16(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 8: G06F, G11B

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

Korean Utility Models and applications for Utility Models since 1975

Japanese Utility Models and applications for Utility Models since 1975

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

eKIPASS(KIPO internal) : 'library','link', 'share', 'devide', 'memory', 'applicatipn'

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	KR 1020060062240 A (ELECTRONICS AND TELECOMMUNICATIONS RESEARCH INSTITUTE) 12 June 2006 See the abstract, figures 1-5, detailed description of the invention and claims 1-6.	1-12
A	JP 2004-206221 A (HEWLETT PACKARD COMPANY) 22 July 2004 See the abstract, figures 1-2, embodiment of the invention [0011]-[0029] and claim 1.	1-12
A	US 2002/0194399 A1 (YOSHIHARU ASAKURA) 19 December 2002 See the abstract, figures 1-5, description of the embodiments and claims 1-7.	1-12
A	WO 2004/059425 A2 (MOTOROLA, INC.) 15 July 2004 See the abstract, figures 2-4, description [page 4 - 8] and claims 1-10.	1-12
A	JP 2003-216344 A (HEWLETT PACKARD COMPANY) 31 July 2003 See the abstract, figures 1-6, embodiment of the invention [0010]-[0040] and claim 1.	1-12

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 JANUARY 2008 (22.01.2008)	**22 JANUARY 2008 (22.01.2008)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
920 Dunsan-dong, Seo-gu, Daejeon 302-701,
Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

LEE, Jung Ho

Telephone No. 82-42-481-5704

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/KR2007/005029

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
KR 1020060062240 A	12.06.2006	None	
JP 2004-206221 A	22.07.2004	None	
US 2002/0194399 A1	19.12.2002	EP 01284453 A2	19.02.2003
		JP 2002373077 A	26.12.2002
WO 2004/059425 A2	15.07.2004	AU 2003300988 A1	22.07.2004
		CN 1732458 A	08.02.2006
		EP 01579341 A2	28.09.2005
		JP 18511868 A	06.04.2006
		JP 2006511868 T2	06.04.2006
		KR 1020050089072 A	07.09.2005
		US 20040123270 A1	24.06.2004
		WO 2004059425 A3	23.12.2004
JP 2003-216344 A	31.07.2003	EP 01324181 A2	02.07.2003
		EP 01324181 A3	09.11.2005
		US 07062614 B2	13.06.2006
		US 20030126309 A1	03.07.2003