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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 November 2012 (22.11.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/157906 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/KR2012/003728

(22) International Filing Date:
11 May 2012 (11.05.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/485,865	13 May 2011 (13.05.2011)	US
61/485,873	13 May 2011 (13.05.2011)	US
10-2011-0103358	11 October 2011 (11.10.2011)	KR

(71) Applicant *(for all designated States except US)*: **SAMSUNG ELECTRONICS CO., LTD.** [KR/KR]; 129, Samsung-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do 443-742 (KR).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **LIM, Geun Sik** [KR/KR]; No. 110-104, Sinyeongtong Hyundai 1-cha Apt., Banwol-dong, Hwaseong-si, Gyeonggi-do 445-330 (KR). **SUH, Sang Bum** [KR/KR]; No. 102-1009, Sinbanpo Cheongu Apt., Jamwon-dong, Seocho-gu, Seoul 137-030 (KR). **CHOI, Hyun Jin** [KR/KR]; No. 306-2301, III-ZI-UM Apt., Jamsil 3-dong, Songpa-gu, Seoul 138-223 (KR).

(74) Agent: **YOON, Dong Yol**; Yoon & Lee International Patent & Law Firm, 3rd Fl, Ace Highend Tower-5, 226, Gasan Digital 1-ro, Geumcheon-gu, Seoul 153-803 (KR).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, 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, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

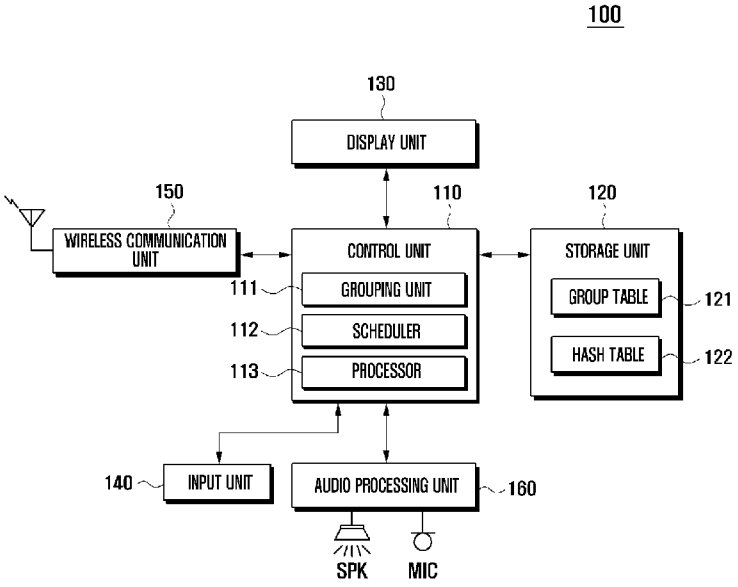
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR IMPROVING APPLICATION PROCESSING SPEED IN DIGITAL DEVICE

(57) Abstract: A method and apparatus for improving application processing speed in a digital device which improve application processing speed for a digital device running in an embedded environment where processor performance may not be sufficiently powerful by detecting an execution request for an application, identifying a group to which the requested application belongs, among preset groups with different priorities and scheduling the requested application according to the priority assigned to the identified group, and executing the requested application based on the scheduling result.

[Fig. 1]



WO 2012/157906 A2

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

Description

Title of Invention: METHOD AND APPARATUS FOR IMPROVING APPLICATION PROCESSING SPEED IN DIGITAL DEVICE

Technical Field

[1] The present invention relates generally to improvement of application processing speed and, more particularly, to a method and apparatus that can improve application processing speed for a digital device running in an embedded environment where processor performance may be not sufficiently powerful.

Background Art

[2] Due to advances in processor and memory technologies, modern digital devices are capable of executing various applications in addition to applications initially installed by manufacturers. Such a digital device may execute multiple applications at the same time. In a typical digital device, when multiple applications are simultaneously executed, the processor is allocated to the multiple applications in the same proportion.

Disclosure of Invention

Technical Problem

[3] Hence, in a state in which multiple applications are executed, when the user makes an execution request for an additional application, it may take a long time to execute the additional application because the digital device is unable to sufficiently allocate the processor resources to the additional application. This may cause user dissatisfaction with performance of the digital device. In a digital device having many installed applications, when multiple applications are executed in the background, processing speed of a user requested application can become slower if the processor is not sufficiently powerful. Hence, it is necessary to develop a means for shortening the application execution time in a digital device operating in an embedded environment lacking physical processing resources.

Solution to Problem

[4] Accordingly, the present invention has been made to solve the above mentioned problems and the present invention provides a method and apparatus that can improve application processing speed by shortening latency in application execution for a digital device running in an embedded environment where physical processing resources are not sufficient.

[5] According to an aspect of the present invention, there is provided a method and apparatus for improving application processing speed in a digital device wherein ap-

plications are classified into multiple groups according to their importance in terms of processing time and response time and different scheduling priorities are applied to the individual groups.

[6] According to an aspect of the present invention, there is further provided a method and apparatus for improving application processing speed in a digital device wherein priorities of a user requested application and other applications in execution are dynamically adjusted so that the user requested application is executed first with a reduced latency time.

[7] According to an embodiment of the present invention, there is provided a method for improving application processing speed in a digital device, including detecting an execution request for an application; identifying a group to which the requested application belongs, among preset groups with different priorities; and scheduling the requested application according to the priority assigned to the identified group, and executing the requested application based on the scheduling result.

[8] According to another embodiment of the present invention, there is provided a method for improving application processing speed in a digital device, including checking, upon detection of an execution request for a user application, whether other applications are being executed; changing, when other applications are being executed, priorities of the other applications being executed so that the user application is executed first, and performing scheduling according to the changed priorities; and executing the user application and the other applications based on the scheduling result.

[9] According to another embodiment of the present invention, there is provided an apparatus for improving application processing speed in a digital device, including a grouping unit identifying, upon detection of an execution request for an application, a group to which the requested application belongs, among preset groups with different priorities; a scheduler scheduling the requested application according to the priority assigned to the identified group; and a processor executing the requested application based on the scheduling result produced by the scheduler.

[10] According to another embodiment of the present invention, there is provided an apparatus for improving application processing speed in a digital device, including an input unit generating an execution request for a user application; and a control unit controlling a process of checking, upon detection of an execution request for a user application from the input unit, whether other applications are being executed, changing, when other applications are being executed, priorities of the other applications being executed so that the user application is executed first, performing scheduling according to the changed priorities, and executing the user application and the other applications based on the scheduling result.

Advantageous Effects of Invention

[11] According to an aspect of the present invention, there are provided a method and apparatus for improving application processing speed in a digital device. Applications are classified into multiple groups according to their importance in terms of processing time and response time, and different scheduling priorities are applied to the individual groups. Hence, the processor can be utilized in a more efficient way. In other words, the processor is utilized more by an application belonging to a group of a high priority and is utilized less by an application belonging to a group of a low priority, so that the application belonging to the high-priority group is processed first. This can be used to increase processing speed of an application directly related to the user.

[12] In addition, when the user makes an execution request for a specific application, the priority of the user requested application may be raised or priorities of other applications being executed are lowered so that the user requested application is processed first with a reduced latency time. Hence, processing speed of the user requested application can be increased.

Brief Description of Drawings

[13] The above and other aspects, features and advantages of the present invention will be more apparent from the following detailed description taken in conjunction with the accompanying drawings, in which:

[14] FIG. 1 is a block diagram illustrating a digital device according to an embodiment of the present invention;

[15] FIG. 2 is a diagram illustrating interactions between internal components of a control unit in the digital device;

[16] FIG. 3 is a flowchart illustrating a method for improving application processing speed according to another embodiment of the present invention;

[17] FIG. 4 is a flowchart illustrating a step for application-to-group assignment in the method illustrated in FIG. 3;

[18] FIG. 5 is a flowchart illustrating a method for improving application processing speed according to another embodiment of the present invention; and

[19] FIG. 6 is a diagram illustrating a scheme for changing priorities of applications.

[20] Throughout the drawings, it should be noted that like reference numbers are used to depict the same or similar elements, features, and structures.

Mode for the Invention

[21] Hereinafter, various embodiments of the present invention are described in detail with reference to the accompanying drawings. The same reference symbols are used throughout the drawings to refer to the same or like parts. Detailed descriptions of well-known functions and structures may be omitted to avoid obscuring the subject

matter of the present invention. Various changes and modifications may be made to the embodiments of the present invention without departing from the scope of the invention.

[22] In the following description, a "digital device" refers to a device having embedded system constraints, such as a Personal Digital Assistant (PDA), mobile communication terminal, smartphone, tablet computer, smart TV, camera and the like.

[23] The present invention relates to shortening latency of application execution in an embedded system environment where processor performance is not powerful by giving priority to threads requiring a high percentage of processor time. In one embodiment, to determine which application is to be scheduled first in a state in which available processor time is insufficient, applications are classified into multiple groups according to their scheduling attributes and properties and different priorities are assigned to the individual groups. In another embodiment, when the user makes an execution request for a specific application while other applications are being executed, application priorities may be changed so that the user requested application is executed first. For example, execution of the other applications may be stopped and the user requested application is executed first, or priorities of the other applications being executed are lowered, thereby increasing the percentage of processor time allocated to the user requested application.

[24] FIG. 1 is a block diagram of a digital device 100 according to an embodiment of the present invention, and FIG. 2 illustrates interactions between internal components of a control unit 110 in the digital device 100.

[25] Referring to FIGS. 1 and 2, the digital device 100 includes a control unit 110, a storage unit 120, a display unit 130, an input unit 140, a wireless communication unit 150, and an audio processing unit 160.

[26] The audio processing unit 160 performs encoding and decoding of an audio signal, and may include a codec (coder/decoder) and an audio amplifier. The audio processing unit 160 is connected to a Microphone (MIC) and a Speaker (SPK). The audio processing unit 160 converts an analog voice signal from the microphone into a digital voice signal and forward the digital voice signal to the control unit 110, and converts a digital voice signal from the control unit 110 into an analog voice signal and output the analog voice signal to the speaker. The audio processing unit 160 outputs various audio signals generated by the digital device 100 (audio signals generated by playback of MP3 files and moving image files) to the speaker. The audio processing unit 160 also outputs various audio signals generated by applications being executed.

[27] The wireless communication unit 150 establishes a communication channel with a base station for a voice call, a data call and a video call. Under control of the control unit 110, the wireless communication unit 150 sends and receives a voice signal or a

video signal to and from a corresponding terminal. The wireless communication unit 150 includes a radio frequency transmitter (not shown) for upconverting the frequency of a signal to be transmitted and amplifying the signal, a radio frequency receiver (not shown) for low-noise amplifying a received signal and downconverting the frequency of the received signal, and a duplexer (not shown) for separating a transmit signal from a receive signal. If the digital device 100 does not support wireless communication, the wireless communication unit 150 may be excluded.

[28] The display unit 130 displays various menus of the digital device 100, information input by the user, and information provided to the user. For example, the display unit 130 displays various screens such as an idle screen having multiple application icons, a message handling screen, a menu screen, a message composition screen and a call handling screen. The display unit 130 may be implemented using Liquid Crystal Display (LCD) devices, Organic Light Emitting Diodes (OLED), or Active Matrix Organic Light Emitting Diodes (AMOLED). When the display unit 130 has a touchscreen functionality, it acts as an input means.

[29] The input unit 140 includes a plurality of alphanumeric and function keys for inputting alphanumeric information and for setting various functions of the digital device 100. The input unit 140 sends input signals for requesting and controlling application execution to the control unit 110. The input unit 140 may be implemented using at least one of a keypad of buttons, a ball joystick, an optical joystick, a wheel key, a touch key, a touch pad, a touch screen and a combination thereof.

[30] The storage unit 120 stores programs and user data for realizing functions of the digital device 100. For example, the storage unit 120 stores an Operating System (OS) for booting and operating the digital device 100, and other application programs related to images, sounds and short-range wireless communication to support optional functions of the digital device 100. When the display unit 130 has a touchscreen functionality, the storage unit 120 stores key maps and menu maps for proper display operation. The key maps may correspond to various keyboard layouts including 3 by 4 and Qwerty layouts, and may include a control key map for controlling execution of an application program. The menu maps includes a menu map for controlling execution of an application. The storage unit 120 stores a group table 121 and a hash table 122.

[31] The group table 121 stores classification results of applications according to their importance in terms of processing time and response time. When applications are composed of multiple processes or threads, classification can be made on a process or thread basis. In the present invention, applications are classified into groups of “urgent tasks,” “normal tasks,” “service tasks,” “background tasks” and “idle tasks.” In the following description, entries of each group are assumed to be threads for illustration purposes.

[32] An "urgent task" is a thread that is interacting with the user through the topmost screen on the display unit 130 or is currently utilizing the processor 113. A thread in the urgent task group needs immediate processing and may be used to immediately handle user interface activities. A "normal task" is a thread that is related to the current display operation on the display unit 130 but does not need immediate processing. A thread in the normal task group is maintained in an executable state until termination. A "service task" is a middleware level thread that supports application functions and utilizes system services. A thread in the service task group may be maintained in an executable state to handle an application related function not directly evident to the user as long as resources of the processor 113 are sufficient to process urgent tasks and normal tasks. A "background task" is a thread that is invisible to the user. A thread in the background task group does not directly impact a user application. Execution of a background task is terminated when an urgent task, a normal task or a service task makes a usage request for the processor 113. Threads in the background task group may be managed according to the Least Recently Used (LRU) scheme (the most recently used thread is terminated last as it tends to be used again). An "idle task" is a thread that is unrelated to any component of an activated application. A thread in the idle task group is used to shorten the time to initiate a user requested application. That is, an idle task may act as a cache.

[33] The hash table 122 contains mappings between Thread IDs (TIDs) in the user space and Light Weight Process IDs (LWPIDs) in the kernel space.

[34] The control unit 110 controls the overall operation of the digital device 100 and signal exchange between internal components thereof, and perform data processing operations. The control unit 110 controls a process of identifying, in response to an execution request for an application, a group to which the user requested application belongs, scheduling the application according to the priority assigned to the group, and executing the application based on the determined schedule. When one or more other applications are being executed when the execution request for the user requested application is made, to process the user requested application first, the control unit 110 changes (lowers) the priorities of the other applications, performs scheduling according to the changed priorities, and controls execution of the user requested application and other applications accordingly. Here, the other applications may have a priority greater than or equal to that of the user requested application. To achieve this, the control unit 110 includes a grouping unit 111, a scheduler 112, and a processor 113.

[35] The grouping unit 111 identifies, in response to an execution request for an application, a group to which the user requested application belongs. When the user requested application is not yet assigned to a group, the grouping unit 111 adds the user requested application to one of the preset task groups according to importance in

terms of processing time and response time. When the scheduling attribute of an application is not set, the application may be assigned to a default group; and when the scheduling attribute of an application is set, the application may be assigned to one of the preset task groups according to a preset rule. When an application is composed of multiple processes or threads, group assignment may be made on a process or thread basis. An application may be assigned to one group when the application is about to be executed, and remain in the group until termination. Group classification or group assignment is described in detail with reference to FIG. 4.

[36] The scheduler 112 adjusts the execution sequence of applications and assigns sequence numbers to executable tasks. The scheduler 112 performs scheduling based on priorities assigned to the groups. That is, the scheduler 112 performs scheduling so that applications with higher priorities can be processed by the processor 113 first. As shown in FIG. 2, when multiple applications are requested for execution, the scheduler 112 performs scheduling so that those applications belonging to a group with a high priority (the urgent task group or the normal task group) can be processed first. In addition, when execution of a user application is requested, the scheduler 112 performs scheduling according to changed priorities of the user application or other applications.

[37] The processor 113 executes applications according to a schedule produced by the scheduler 112. The processor 113 sets the priority of a process or thread constituting an application by means of system calls or library calls illustrated in Table 1.

[38] Table 1

[Table 1]

<table>
<tr><th colspan="2">Content</th><th>Function name</th><th>Call</th><th>Linux Thread</th><th>NPTL</th></tr>
<tr><td rowspan="2">Normal Priority (=Regular) Range:-20~19</td><td>Process</td><td>setpriority() or nice()</td><td>System call</td><td rowspan="4">getpid()</td><td rowspan="4">gettid()</td></tr>
<tr><td>Thread</td><td>setpriority() or nice()</td><td>System call</td></tr>
<tr><td rowspan="2">Real-time Priority (=time-critical) Range:1~99</td><td>Process</td><td>sched_setscheduler() sched_setparam()</td><td>System call</td></tr>
<tr><td>Thread</td><td>pthread_setschedparam()</td><td>Library call</td></tr>
</table>

[39] Referring to Table 1, the priority of a process or thread being executed may be changed in a “normal priority” range using a system call “setpriority” or “nice”; and the priority of a thread may be changed in a “real-time priority” range using a library call “pthread_setschedparam.” Here, “normal priority” is also referred to as “dynamic

priority." Threads having a priority in the "normal priority" range may share the processor 113 in a time-sliced manner, and a thread with a smaller "nice" value (set by the "nice" function) may use the processor 113 for a longer time. The "real-time priority" is used for a thread or process with a real-time constraint. Real-time priorities are not generally used in the user space, and may be used for "watchdog" and "migration" services in the kernel space. Priority change is described in detail below.

[40] Although not shown in FIG.1, the digital device 100 may further include at least one of a camera module for capturing still or moving images, a digital broadcast receiving module for receiving digital broadcasts, a digital music player like an MP3 module and a proximity sensor module for proximity sensing. With the digital convergence trend, it should be apparent to those skilled in the art that the digital device 100 may further include a unit comparable to the above-described units, and one unit thereof may be removed or replaced with another unit.

[41] FIG. 3 is a flowchart of a method for improving application processing speed in the digital device 100.

[42] Referring to FIG. 3, the control unit 110 of the digital device 100 detects an execution request for an application in Step 301. Upon detection of an execution request for an application, the control unit 110 identifies a group to which the requested application is assigned in Step 303.

[43] When the requested application is not yet assigned to a group, the control unit 110 assigns the requested application to one of the preset task groups in Step 305. Here, the control unit 110 checks the scheduling attribute of the requested application and assigns the requested application to one of the preset task groups according to a preset rule. When the requested application is composed of multiple processes or threads, group assignment may be made on a process or thread basis. This is further described with reference to FIG. 4. After group assignment, the control unit 110 proceeds to step 307.

[44] When the requested application is already assigned to a group, the control unit 110, the control unit 110 assigns the priority of the group to the requested application in Step 307. The control unit 110 performs scheduling of the requested application according to the assigned priority and executes the requested application according to the scheduling results in Step 309. That is, the scheduler 112 performs priority-based scheduling, and the processor 113 executes applications according to the scheduling results.

[45] As described above, according to the present invention, unlike in a method in which multiple applications requested for execution are scheduled to have the same priority, an application requested for execution is assigned to one of the preset groups having different priorities according to the scheduling attribute, is scheduled based on the

priority of the assigned group, and is executed according to the scheduling results. That is, an application requiring rapid response time is assigned to a group having a high priority, enabling efficient execution management of the application.

[46] FIG. 4 is a detailed flowchart of a step for group assignment in the method illustrated in FIG. 3.

[47] Referring to FIG. 4, the grouping unit 111 of the control unit 110 detects a request for group assignment in Step 401. A request for group assignment may be made when an application is executed for the first time.

[48] The grouping unit 111 identifies the scheduling attribute of the requested application in Step 402, and checks whether the scheduling attribute is set in Step 403. When the scheduling attribute is set, the grouping unit 111 assigns the requested application to one of the preset task groups according to a preset rule in Step 409. Here, the grouping unit 111 may assign the application to a group in consideration of properties thereof such as processing time and response time. When the scheduling attribute is not set, the grouping unit 111 assigns the application to the default group in Step 407. In the event that the application is composed of multiple processes or threads, group assignment may be made on a process or thread basis.

[49] After group assignment, the application may remain in the assigned group until termination. This is because frequent group reassignment or permanent storage of the reassignment results may be an inefficient use of resources.

[50] FIG. 5 is a flowchart of a method for improving application processing speed in the digital device 100.

[51] Referring to FIG. 5, the control unit 110 detects an execution request for a user application in Step 501. Upon detection of an execution request, the control unit 110 checks whether other applications are being executed in Step 503. Here, the user application may be an application that is requested for execution from the user using a graphical user interface on the input unit 140 or the touchscreen (not shown), and the other applications may be applications that are in competition with the user application. The other applications may have a priority greater than or equal to that of the user application.

[52] When no other applications are being executed, the control unit 110 executes the user application in Step 513. When other applications are being executed, the control unit 110 lowers priorities of the other applications being executed in Step 505, and performs scheduling based on changed priorities in Step 507. In the present invention, the "pthread_setschedparam" function in the Native POSIX Thread Library (NPTL), a common library function working in the user space, may be utilized to change priorities. This is further described with reference to FIGS. 6 and 7.

[53] Based on the scheduling results, the control unit 110 controls execution of the user

application and the other applications in Step 508. As described above, as priorities of the other applications are lowered, the user application may utilize the processor 113 for a longer time. Hence, the processing speed of the user application can be improved.

[54] The control unit 110 checks whether execution of the user application is completed in Step 509. When execution of the user application is completed, the control unit 110 restores the original priorities of the other applications in Step 511.

[55] In the above description, priorities of the other applications are lowered. Alternatively, the control unit 110 may raise the priority of the user application, or may suspend execution of the other applications until execution of the user application is completed.

[56] When an application has a priority greater than or equal to a preset threshold, the control unit 110 may not change the priority of the application. For example, when an application belongs to the urgent task group (such as an application for handling reception of a call request), the control unit 110 may not lower the priority of the application.

[57] In the method illustrated in Fig. 5, when the user activates a user application, priorities of existing applications being executed are adjusted or the priority of the user application is adjusted so as to shorten processing time of the user application. This is because the user wishes to quickly view results of the activated application.

[58] FIG. 6 illustrates a scheme for changing priorities of applications.

[59] Among many scheduling policies for embedded environments with real-time constraints, while the Round Robin Scheduling Policy (SCHED_RR) is considered ideal, the First-In-First-Out scheduling policy (SCHED_FIFO) is preferred for reasons of efficiency and simplicity.

[60] In the present invention, the pthread_setschedparam function in the NPTL based on Linux Kernel 2.6 is enhanced in a manner conforming to the Portable Operating System Interface (POSIX) standard. To shorten processing time of a user requested application, the enhanced pthread_setschedparam function can be used to adjust the scheduling priority of the user requested the user requested application) to be adjusted or to change "normal priorities" of other existing applications in runtime. Here, the enhancement should be carried out so that existing threaded applications are reusable without source code modification. In addition, different priorities should be assignable to multiple threads created within one process while compatibility between interfaces of NPTL thread functions and the POSIX standard is maintained.

[61] Table 2 illustrates a pseudo code for an NPTL function that can dynamically adjust the scheduling priority of a "normal priority" thread. However, the present invention is not limited to or by such a pseudo code, and may be implemented in various ways.

[62] Table 2

[Table 2]

```
int __pthread_setschedparam(threadid, policy, param)
  pthread_t threadid;
  int policy;
  const struct sched_param   *param;
 {
/*To support priority, if used SCHED_FIFO or SCHED_RR, display   caution
message
 *(@/usr/include/linux/sched.h)

/*Default Value is normal*/
structp thread *pd = (struct pthread *) threadid;
if (policy == SCHED_OTHER){
/*The scheduling priority of the process. */
int which = PRIO_PROCESS;

/*Handling of SCHED_OTHER priority that user assigned */
if (param->sched_priority < -20 &&   param->sched_priority > 19){
   printf("ERROR! Range   of priority that you can input is -20~19 WrWn");
   return errno;
}

/*Getting LWP(thread id in the kernel) to change priority about assigned
thread id. */
if (setpriority(which, unique_kernel_tid(),   param->sched_priority)){
   perror ("setpriority() is ERROR. It seems that System not support
setpriority()Wn");
   result = errno;
}
```

[63] When the pseudo code of Table 2 is implemented, the “pthread_setschedparam” function may be invoked as shown FIG. 6. More specifically, after a requested thread is created using the “pthread_create” function, the processor 113 of the control unit 110 refers to the hash table 122 to find a thread ID (LWPID) in the kernel space corresponding to a thread ID (TID) in the user space. Here, the hash table 122 contains information on mappings between thread IDs in the user space and thread IDs in the kernel space.

[64] Thereafter, the processor 113 invokes the “setpriority” function using the found thread ID (LWPID) in the kernel space. Then, the “setpriority” function invokes a system call working in the kernel space to adjust the priority of the thread corresponding to the thread ID (LWPID). After the priority of the corresponding thread is adjusted, the scheduler 112 performs scheduling of the corresponding thread based on

the adjusted priority. In the above description, the "setpriority" function is used for priority change. Alternatively, the "nice" or "renice" function may be used for priority change. Priority change using the "setpriority," "nice" or "renice" function is well known to those skilled in the art, and a detailed description thereof is omitted.

[65] After changing the function related to thread scheduling in the NPTL, different priorities can be assigned to multiple threads created by one process as illustrated in Table 3.

[66] Table 3

[Table 3]

```
/*
*@Description:voluntary & enforced thread scheduling at non-preemptive mode
*@thread variables(pthread_t thread[max])
*If you want to affect priority about each thread of the process in the
kernel 2.6 + NPTL,
*We recommend that you use SCHED_OTHER policy based on priority-based
scheduling.
*And, If you need time slice(sharing) about threads, Use SCHED_RR Policy
*(Real-time Policy) with Same Priority using
pthread_attribute_setschedpolicy().
*/

struct sched_param  schedp;
int priority=5; /* You have to input Priority Number of between -20 and 19.
*/
memset(&schedp, 0, sizeof(schedp));
schedp.sched_priority=priority;

pthread_setschedparam(pthread_self(), SCHED_OTHER, &schedp)   //for self
thread
pthread_setschedparam(thread[i], SCHED_OTHER, &schedp)  //for other thread
```

[67] Referring to Table 3 and FIG. 6, a brief description is given of changing the priority of a thread with "pid" of "7992" by means of the "renice" function.

[68] After finding a thread ID in the kernel space using the hash table 122, the processor 113 of the control unit 110 issues a priority adjustment command "renice 8 -p 7992", which then changes the priority of the thread with "pid" of "7992" to '8'. As described above, the priority of a thread may be dynamically changed by means of the "pid" value of the thread. That is, the priority of a user application or priorities of other applications (or threads) being executed may be dynamically changed so that the user application is processed first. The "tgid" value of a thread indicates a process to which the thread belongs. As illustrated in FIG. 6, multiple threads 2, 3 and 4 are threads con-

stituting a process 1. A thread whose "tgid" value is the same as the "pid" value may be a process.

[69] According to the present invention, the method for improving application processing speed may be implemented as a computer program and may be stored in various computer readable storage media. The computer readable storage media may store program instructions, data files, data structures and combinations thereof. The program instructions may include customized instructions and existing general-purpose instructions. The computer readable storage media may include magnetic media such as a hard disk and floppy disk, optical media such as a CD-ROM and DVD, magneto-optical media such as a floptical disk, and memory devices such as a ROM and RAM. The program instructions may include machine codes produced by compilers and high-level language codes executable through interpreters.

[70] Although various embodiments of the present invention for improving application processing speed in a digital device have been described in detail herein, many variations and modifications may be made without departing from the spirit and scope of the present invention, as defined by the appended claims.

Claims

[Claim 1] A method for improving application processing speed in a digital device, comprising:
detecting an execution request for an application;
identifying a group to which the requested application belongs, among preset groups with different priorities; and
scheduling the requested application according to the priority assigned to the identified group, and executing the requested application based on the scheduling result.

[Claim 2] The method of claim 1, further comprising assigning, when the requested application is not yet assigned to a group, the requested application to one of the preset groups according to a preset rule.

[Claim 3] The method of claim 2, wherein assigning the requested application to one group comprises:
examining a scheduling attribute of the requested application; and
assigning, when the scheduling attribute is not set, the requested application to a default group, and assigning, when the scheduling attribute is set, the requested application to one of the preset groups according to the scheduling attribute.

[Claim 4] The method of claim 3, wherein assigning the requested application to one of the preset groups according to the scheduling attribute comprises:
examining importance of processing time and response time for the requested application based on the scheduling attribute; and
determining one group to which the requested application belongs according to the examination result.

[Claim 5] The method of claim 2, wherein assigning the requested application to one group comprises assigning each process and thread constituting the application to one of the preset groups.

[Claim 6] The method of claim 1, further comprising:
checking, upon detection of an execution request for a user application, whether other applications are being executed;
changing, when other applications are being executed, priorities of the other applications being executed so that the user application is executed first, and performing scheduling according to the changed priorities; and
executing the user application and the other applications based on the

scheduling result.

[Claim 7] The method of claim 6, wherein changing priorities of the other applications comprises:
lowering priorities of the other applications; and
suspending execution of the other applications until execution of the user application is completed.

[Claim 8] The method of claim 6, further comprising restoring, when execution of the user application is completed, the original priorities of the other applications.

[Claim 9] An apparatus for improving application processing speed in a digital device, comprising:
a grouping unit, identifying, upon detection of an execution request for an application, a group to which the requested application belongs, among preset groups with different priorities;
a scheduler, scheduling the requested application according to the priority assigned to the identified group; and
a processor, executing the requested application based on the scheduling result produced by the scheduler.

[Claim 10] The apparatus of claim 9, wherein the grouping unit assigns, when the requested application is not yet assigned to a group, the requested application to one of the preset groups according to a preset rule.

[Claim 11] The apparatus of claim 10, wherein the grouping unit assigns each process and thread constituting the requested application to one of the preset groups.

[Claim 12] The apparatus of claim 10, wherein the grouping unit examines a scheduling attribute of the requested application immediately before application execution, assigns, when the scheduling attribute is not set, the requested application to a default group, assigns, when the scheduling attribute is set, the requested application to one of the preset groups according to the scheduling attribute, and maintains the group assignment until application termination.

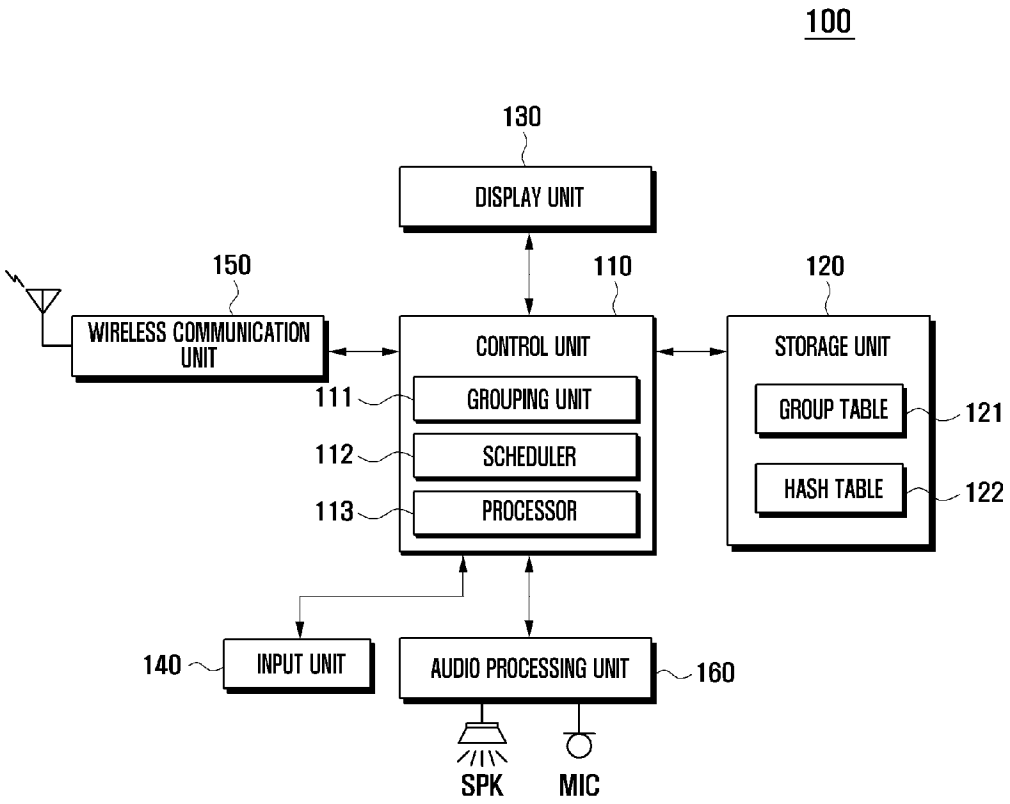
[Claim 13] The apparatus of claim 9, further comprising:
an input unit generating an execution request for a user application; and
a control unit checking, upon detection of an execution request for a user application from the input unit, whether other applications are being executed, and changing, when other applications are being executed, priorities of the other applications being executed so that the user application is executed first, and wherein the scheduler performs

scheduling according to the changed priorities, and the processor executes the user application and the other applications based on the scheduling result.

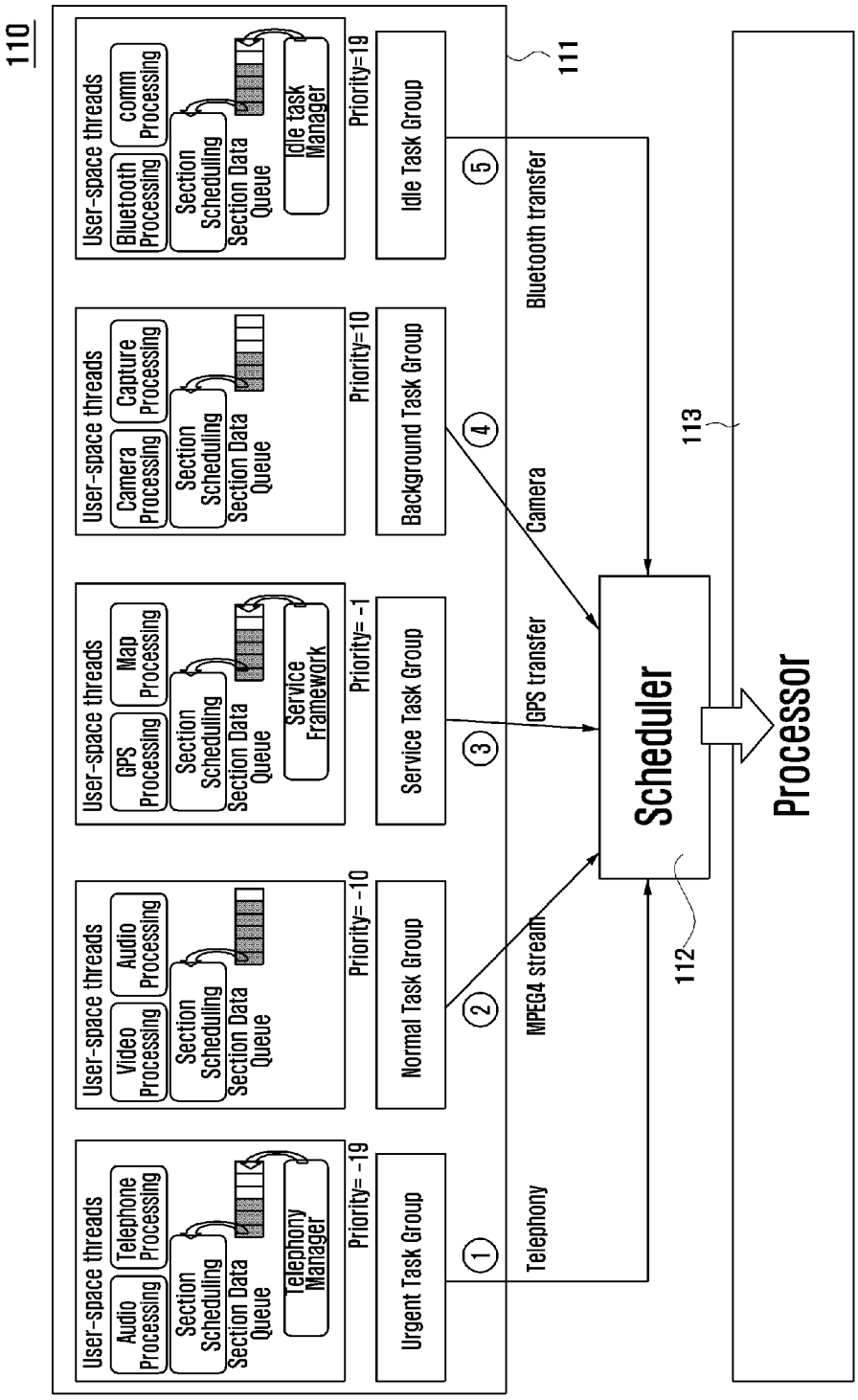
[Claim 14] The apparatus of claim 13, wherein the control unit lowers priorities of the other applications, or suspends execution of the other applications until execution of the user application is completed.

[Claim 15] The apparatus of claim 13, wherein the control unit restores, when execution of the user application is completed, the original priorities of the other applications.

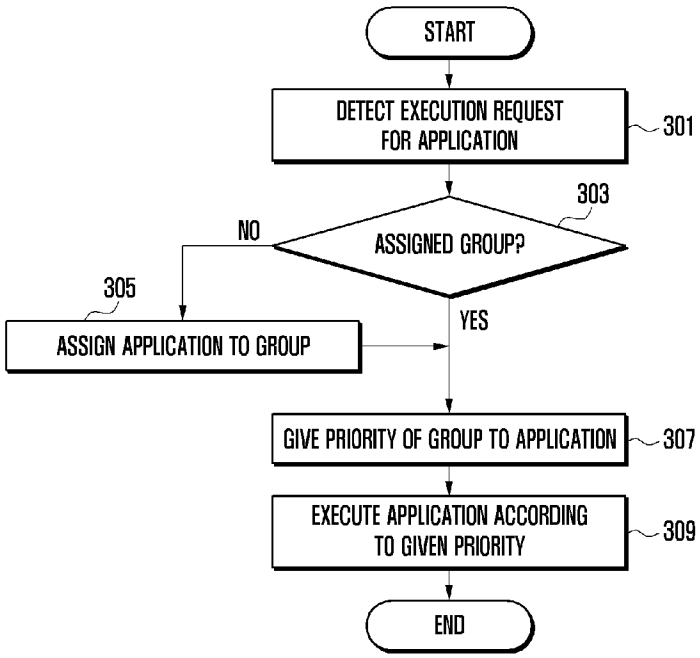
[Fig. 1]


100
130
DISPLAY UNIT
150
110
120
WIRELESS COMMUNICATION UNIT
CONTROL UNIT
STORAGE UNIT
111
GROUPING UNIT
GROUP TABLE
121
112
SCHEDULER
HASH TABLE
122
113
PROCESSOR
140
INPUT UNIT
AUDIO PROCESSING UNIT
160
SPK
MIC

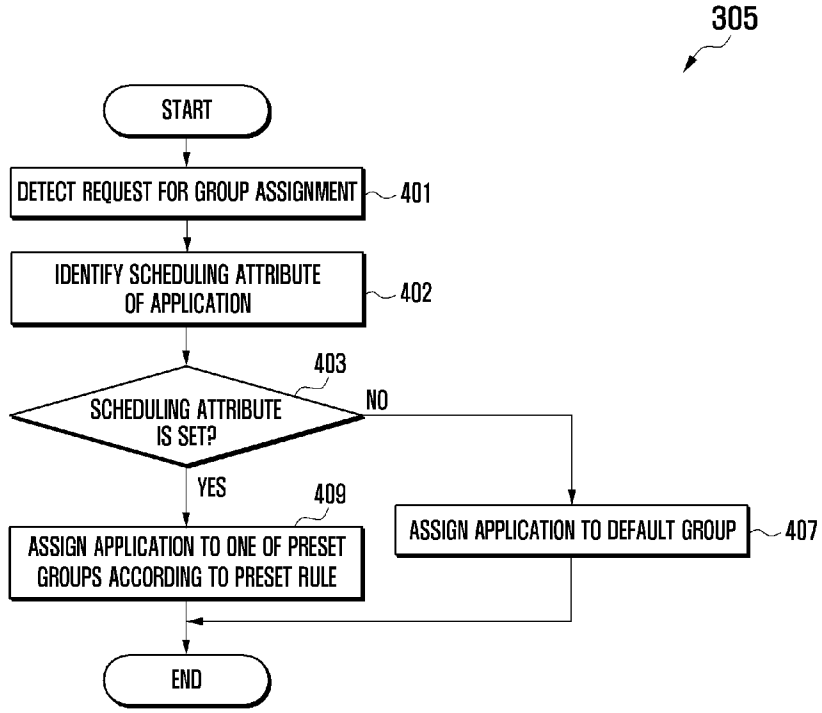
[Fig. 2]


110
111
User-space threads
Audio Processing
Telephone Processing
Section Scheduling
Section Data Queue
Telephony Manager
Priority= -19
Urgent Task Group
User-space threads
Video Processing
Audio Processing
Section Scheduling
Section Data Queue
Priority= -10
Normal Task Group
User-space threads
GPS Processing
Map Processing
Section Scheduling
Section Data Queue
Service Framework
Priority= -1
Service Task Group
User-space threads
Camera Processing
Capture Processing
Section Scheduling
Section Data Queue
Priority=10
Background Task Group
User-space threads
Bluetooth Processing
comm Processing
Section Scheduling
Section Data Queue
Idle task Manager
Priority=19
Idle Task Group
Telephony
MPEG4 stream
GPS transfer
Camera
Bluetooth transfer
Scheduler
112
113
Processor
1
2
3
4
5

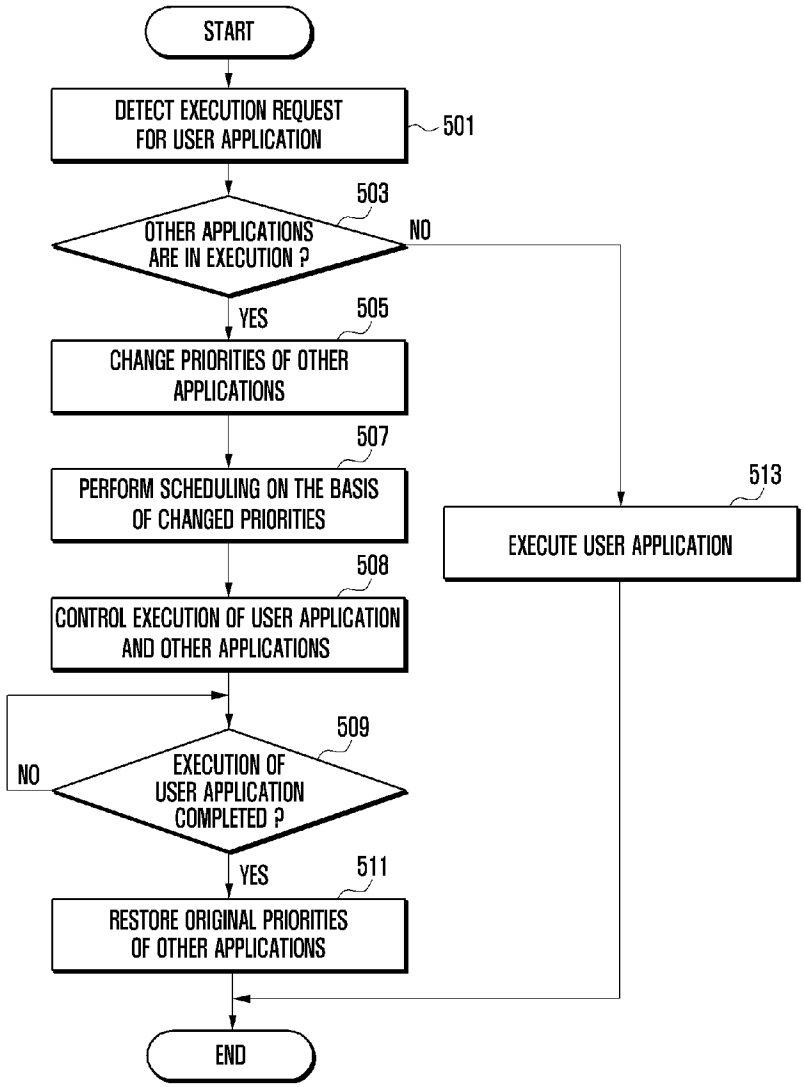
[Fig. 3]


START
DETECT EXECUTION REQUEST FOR APPLICATION
301
303
ASSIGNED GROUP?
NO
YES
305
ASSIGN APPLICATION TO GROUP
GIVE PRIORITY OF GROUP TO APPLICATION
307
EXECUTE APPLICATION ACCORDING TO GIVEN PRIORITY
309
END

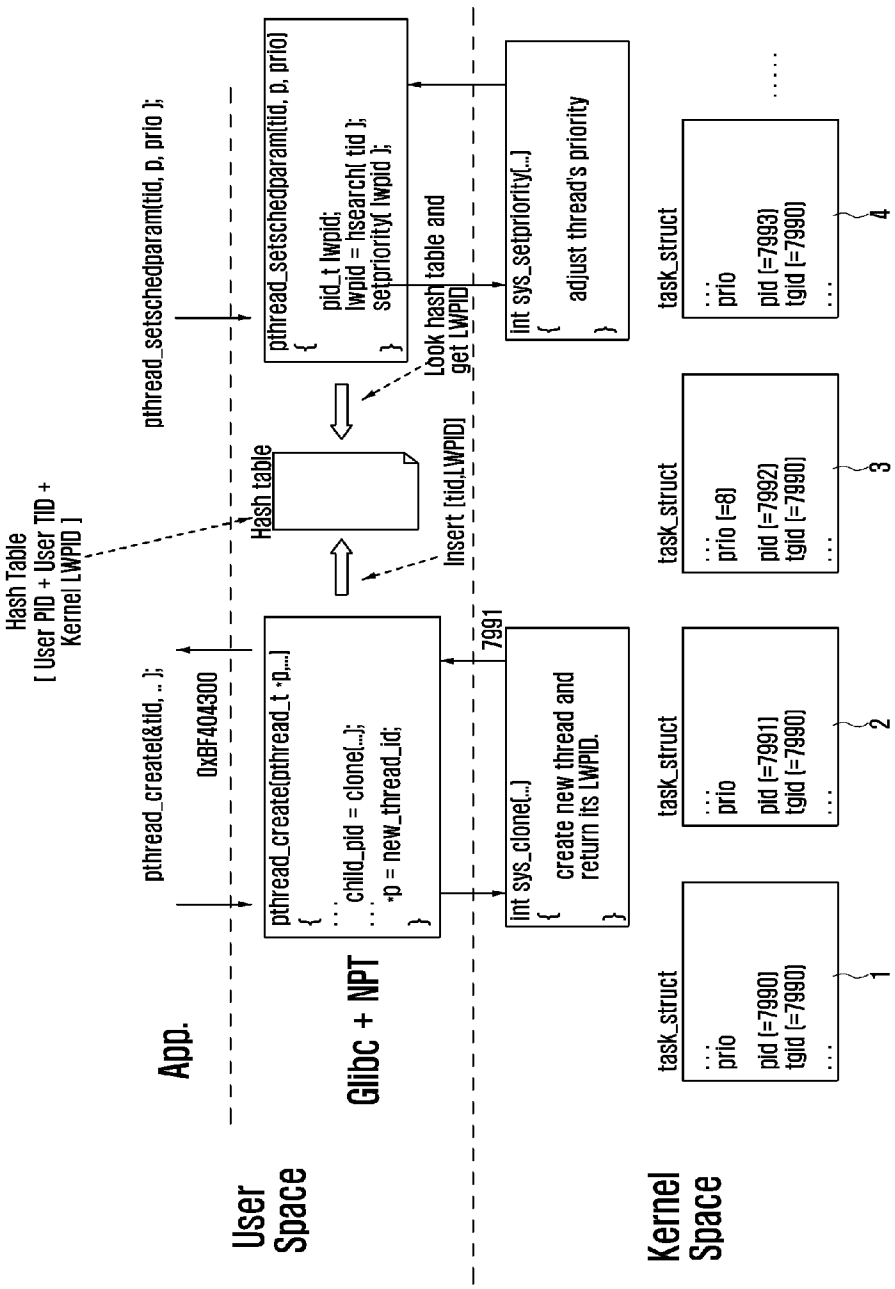
[Fig. 4]


305
START
DETECT REQUEST FOR GROUP ASSIGNMENT
401
IDENTIFY SCHEDULING ATTRIBUTE OF APPLICATION
402
403
SCHEDULING ATTRIBUTE IS SET?
NO
YES
409
ASSIGN APPLICATION TO ONE OF PRESET GROUPS ACCORDING TO PRESET RULE
ASSIGN APPLICATION TO DEFAULT GROUP
407
END

[Fig. 5]


START
DETECT EXECUTION REQUEST FOR USER APPLICATION
501
503
OTHER APPLICATIONS ARE IN EXECUTION ?
NO
YES
505
CHANGE PRIORITIES OF OTHER APPLICATIONS
507
PERFORM SCHEDULING ON THE BASIS OF CHANGED PRIORITIES
513
EXECUTE USER APPLICATION
508
CONTROL EXECUTION OF USER APPLICATION AND OTHER APPLICATIONS
509
NO
EXECUTION OF USER APPLICATION COMPLETED ?
YES
511
RESTORE ORIGINAL PRIORITIES OF OTHER APPLICATIONS
END

[Fig. 6]


User Space
Kernel Space
App.
Glibc + NPT
Hash Table
[User PID + User TID + Kernel LWPID]
pthread_create(&tid, ..);
0xBF404300
pthread_setschedparam(tid, p, prio);
Hash table
pthread_create(pthread_t *p,...)
{
...
child_pid = clone(...);
...
*p = new_thread_id;
}
pthread_setschedparam(tid, p, prio)
{
pid_t lwpid;
lwpid = hsearch(tid);
setpriority(lwpid);
}
Look hash table and get LWPID
Insert (tid,LWPID)
7991
int sys_clone(...)
{
create new thread and return its LWPID.
}
int sys_setpriority(...)
{
adjust thread's priority
}
task_struct
...
prio
pid (=7990)
tgid (=7990)
...
1
task_struct
...
prio
pid (=7991)
tgid (=7990)
...
2
task_struct
...
prio (=8)
pid (=7992)
tgid (=7990)
...
3
task_struct
...
prio
pid (=7993)
tgid (=7990)
...
4
.....