Suspend notifiers

(C) 2007 Rafael J. Wysocki <rjw@sisk.pl>, GPL

There are some operations that device drivers may want to carry out in their .suspend() routines, but shouldn't, because they can cause the hibernation or suspend to fail. For example, a driver may want to allocate a substantial amount of memory (like 50 MB) in .suspend(), but that shouldn't be done after the swsusp's memory shrinker has run.

Also, there may be some operations, that subsystems want to carry out before a hibernation/suspend or after a restore/resume, requiring the system to be fully functional, so the drivers' .suspend() and .resume() routines are not suitable for this purpose. For example, device drivers may want to upload firmware to their devices after a restore from a hibernation image, but they cannot do it by calling request_firmware() from their .resume() routines (user land processes are frozen at this point). The solution may be to load the firmware into memory before processes are frozen and upload it from there in the .resume() routine. Of course, a hibernation notifier may be used for this purpose.

The subsystems that have such needs can register suspend notifiers that will be called upon the following events by the suspend core:

PM_HIBERNATION_PREPARE	The system is going to hibernate or suspend, tasks will be frozen immediately.
PM_POST_HIBERNATION	The system memory state has been restored from a hibernation image or an error occured during the hibernation. Device drivers' .resume() callbacks have been executed and tasks have been thawed.
PM_RESTORE_PREPARE	The system is going to restore a hibernation image. If all goes well the restored kernel will issue a PM_POST_HIBERNATION notification.
PM_POST_RESTORE	An error occurred during the hibernation restore. Device drivers' .resume() callbacks have been executed and tasks have been thawed.
PM_SUSPEND_PREPARE	The system is preparing for a suspend.
PM_POST_SUSPEND	The system has just resumed or an error occured during the suspend. Device drivers' .resume() callbacks have been executed and tasks have been thawed.

It is generally assumed that whatever the notifiers do for PM_HIBERNATION_PREPARE, should be undone for PM_POST_HIBERNATION. Analogously, operations performed for PM_SUSPEND_PREPARE should be reversed for PM_POST_SUSPEND. Additionally, all of the notifiers are called for PM_POST_HIBERNATION if one of them fails for PM_HIBERNATION_PREPARE, and all of the notifiers are called for PM_POST_SUSPEND if one of them fails for PM_SUSPEND_PREPARE.

The hibernation and suspend notifiers are called with pm_mutex held. They are defined in the usual way, but their last argument is meaningless (it is always NULL). To register and/or unregister a suspend notifier use the functions register_pm_notifier() and unregister_pm_notifier(), respectively, defined in

include/linux/suspend.h . If you don't need to unregister the notifier, you can also use the pm_notifier() macro defined in include/linux/suspend.h .