МЫ С LINUX ЭТИМ ЛОЖИМСЯ И С LINUX ЭТИМ ВСТАЕМ – ПОСЛЕДСТВИЯ ГИБЕРНАЦИИ

АНТОН БОРИСОВ

Сегодня мы рассмотрим такой вопрос, как программная гибернация (hibernation) в ОС Linux. Поддержка засыпания была начата для серий 2.4 Linux, но т.к. время не стоит на месте, на сегодняшний момент выпущена новая серия 2.6 (где, в частности, переработана подсистема управления энергосбережением), поэтому и мы будем работать именно под этой серией. Теперь в 2.6.xx Linux-ядрах существует полноценная поддержка, то, что раньше было только в экспериментальном исполнении. Для приверженцев 2.4.xx ядер рекомендуется наложить патчи, чтобы получить то, что по умолчанию поставляется в новой ветке.

Ну что же, краткая предыстория рассказа о гибернации. Первый раз мне пришлось столкнуться с этой увлекательной функцией в 2002 году, когда вышли первые релизы Windows XP Professional. Любопытство одержало верх, и некоторое время я проработал в данной ОС, тем паче, что задачи, решаемые в тот момент, требовали среду разработки именно Windows-платформы. Кроме отличного дизайна интерфейса в системе была функция сбережения питания, так называемое hibernate state.

Это означает, что в любой момент времени можно перевести систему вместе со всеми приложениями, которые работают, в состояние, когда все ОЗУ персоналки записывается на диск и производится отключение питания. При этом в момент включения питания, ОЗУ, которое было записано на диск, переписывается обратно в память (точнее не ОЗУ, а его содержимое) и передается управление на тот код, который работал перед самым отключением питания. Чем эта схема любопытна? В первую очередь тем, что мы не закрываем/открываем заново приложения, состояния в программных продуктах остаются неизменными. Во-вторых, не требуется вспоминать, что же конкретно мы делали в предыдущий день, все остается на рабочем столе, как и в прошлый раз. Тем самым, уходя с работы и переведя систему в такое состояние, мы не беспокоимся, а выдержат ли наши бесперебойники, а не сгорит ли рабочее место из-за повышенной температуры, в общем, масса хлопот с плеч долой. Тем более, если мы работаем с ноутбуком, где в первую очередь стоит энергосбережение. Поэтому возможность эту мы запомнили, поставили под контроль.

Что же касается Windows XP, то может быть из-за того, что версия была слишком сырой на тот момент или просто из-за того, что за 100 рублей Genuine Windows-продукты не купить, а получать извещения о завершенных приложениях необходимостив этой связи направить письмо в службу поддержки, совершенно меня стали огорчать, пришлось совмещать полезное с приятным или даже, наоборот, приятное с полезным, slackware с VMware+Windows 2000 Pro.

Это была преамбула, нас же интересует реализация режима сна именно в Linux.

Так что история начинается.

Для тех, кто ни разу не сталкивался с гибернацией (засыпанием), следует ознакомиться с документацией, которая идет вместе с 2.6 ядром (директория /usr/src/linux/Documentation/power). Не стану подробно останавливаться на всех файлах. Отмечу, что «swsusp.txt» – начальная точка для понимания проблемы.

Итак, что в первую очередь следует сделать? Подготовить swap-раздел таким образом, чтобы его размер превышал в 2 раза объем установленной RAM в системе. Если вы опытный администратор, то именно так вы и поступаете, не мне вас учить. Для чего это нужно?

Именно в swap-раздел производится сброс содержимого RAM в процессе засыпания и дальнейшее его восстановление во время пробуждения.

Второе, добавляем в /etc/lilo.conf следующую строку:

```
append="apm=power-off resume=/dev/hdc5 acpi=force"
```

Это означает, что при старте ядру будут переданы данные параметры.

Если ваш box изготовлен не позднее 1999 года, то «acpi=force», как правило, не требуется. Однако на моей рабочей машине, приходится добавлять, ибо без нее ACPI-подсистема не хочет подниматься.

Сочетание «resume=/dev/hdc5» означает, что раздел для гибернации называется /dev/hdc5 (он же swap-раздел). В дальнейшем мы увидим, что во время нормальной работы сигнатура swap-раздела меняется на свою нормальную сигнатуру SWAPSPACE2. (В спящем режиме она устанавливается как S2SUSP).

Не забудем установить заново lilo-загрузчик:

```
lilo -v
```

Теперь мы готовы для засыпания. Перезагрузимся, чтобы новые параметры были знакомы ядру. reboot. Итак, производим следующую простую операцию:

```
echo 4 > /proc/acpi/sleep
```

(за цифрами обращайтесь к документации, прилагаемой к исходникам ядра – ссылка была дана выше в статье). Таким образом мы не совершаем какие-то магические пасы, а через псевдофайловую систему proc записываем данные в служебные структуры ядра.

Заметьте, что если ACPI-подсистема не стартовала, то директории /proc/acpi у нас в системе не будет.

Далее ядро начинает сбрасывать дамп памяти в swap-раздел. При этом в syslog появляются следующие сообщения:

```
Jan 11 22:15:07 athlon kernel: Stopping tasks: =======exiting...exiting...=======|
Jan 11 22:15:07 athlon kernel: Freeing memory: .........|
Jan 11 22:15:07 athlon kernel: hdd: start_power_step(step: 0)
Jan 11 22:15:07 athlon kernel: hdd: completing PM request, suspend
Jan 11 22:15:07 athlon kernel: hdc: start_power_step(step: 0)
Jan 11 22:15:07 athlon kernel: hdc: completing PM request, suspend
Jan 11 22:15:07 athlon kernel: hdb: start_power_step(step: 0)
Jan 11 22:15:07 athlon kernel: hdb: completing PM request, suspend
Jan 11 22:15:07 athlon kernel: hda: start_power_step(step: 0)
Jan 11 22:15:07 athlon kernel: hda: completing PM request, suspend
Jan 11 22:15:07 athlon kernel: /critical section: Counting pages to copy[nosave c04a6000]
(pages needed: 3642+512=4154 free: 61889)
Jan 11 22:15:07 athlon kernel: Alloc pagedir
Jan 11 22:15:07 athlon kernel: [nosave c04a6000]<4>Freeing prev allocated pagedir
Jan 11 22:15:07 athlon kernel: Debug: sleeping function called from invalid context at
inc lude/asm/semaphore.h:119
Jan 11 22:15:07 athlon kernel: in_atomic():1, irqs_disabled():0
Jan 11 22:15:07 athlon kernel: Call Trace:
Jan 11 22:15:07 athlon kernel:  [<c011d6db>] __might_sleep+0xab/0xd0
Jan 11 22:15:07 athlon kernel:  [<c010b5c8>] common_interrupt+0x18/0x20
Jan 11 22:15:07 athlon kernel:  [<c0291b32>] device_resume+0x22/0x50
Jan 11 22:15:07 athlon kernel:  [<c013839c>] drivers_resume+0x3c/0x40
Jan 11 22:15:07 athlon kernel:  [<c0138699>] do_magic_resume_2+0x79/0xe0
Jan 11 22:15:07 athlon kernel:  [<c0339adf>] do_magic+0x11f/0x140
Jan 11 22:15:07 athlon kernel:  [<c01388bb>] do_software_suspend+0x6b/0x90
Jan 11 22:15:07 athlon kernel:  [<c025a4aa>] acpi_system_write_sleep+0xc3/0xe6
Jan 11 22:15:07 athlon kernel:  [<c015652d>] vfs_write+0xad/0x120
Jan 11 22:15:07 athlon kernel:  [<c015663f>] sys_write+0x3f/0x60
Jan 11 22:15:07 athlon kernel:  [<c010b45b>] syscall_call+0x7/0xb
```

После этого происходит отключение питания.

На что следует обратить внимание. Когда вы в следующий раз включаете машину, то вы имеете полное право подать параметр «noresume» ядру, которое запрещает восстанавливать состояние гибернации. Делать этого не рекомендуется, так как содержимое на вашем корневом разделе может помахать вам ручкой. Во всяком случае не все содержимое, а только последние изменения. У меня используется ext3 файловая система, поэтому этот вывод был сделан из сообщений, посылаемых обработчиком файловой системы.

Если у вас не было swap-раздела до засыпания, то заснуть вам также не удастся. Так что имейте это в виду. Простое правило – между засыпанием и просыпанием не пишите на раздел. На swap-раздел уж точно писать не надо, т.к. сигнатуры (SWAPSPACE2), показывающей, что там именно swap, нет. Поэтому подключить его пока не получится. А специально проинициализировать его придется, вызвав команду:

```
/sbin/mkswap /dev/hdc5
```

Повторюсь, что данные действия не являются обязательными. Мы исследуем то, что может быть при обстоятельствах, когда жесткий диск изъят из системы в состоянии «сна» (гибернации).

Теперь мы приходим из отпуска – включаем рубильник.

```
Jan 19 05:13:07 athlon kernel: hda: Wakeup request inited, waiting for !BSY...
Jan 19 05:13:07 athlon kernel: hda: start_power_step(step: 1000)
Jan 19 05:13:07 athlon kernel: blk: queue c135a200, I/O limit 4095Mb (mask 0xffffffff)
Jan 19 05:13:07 athlon kernel: hda: completing PM request, resume
Jan 19 05:13:07 athlon kernel: hdb: Wakeup request inited, waiting for !BSY...
Jan 19 05:13:07 athlon kernel: hdb: start_power_step(step: 1000)
Jan 19 05:13:07 athlon kernel: hdb: completing PM request, resume
Jan 19 05:13:07 athlon kernel: hdc: Wakeup request inited, waiting for !BSY...
Jan 19 05:13:07 athlon kernel: hdc: start_power_step(step: 1000)
Jan 19 05:13:07 athlon kernel: blk: queue c1334600, I/O limit 4095Mb (mask 0xffffffff)
Jan 19 05:13:07 athlon kernel: hdc: completing PM request, resume
Jan 19 05:13:07 athlon kernel: hdd: Wakeup request inited, waiting for !BSY...
Jan 19 05:13:07 athlon kernel: hdd: start_power_step(step: 1000)
Jan 19 05:13:07 athlon kernel: blk: queue c1334200, I/O limit 4095Mb (mask 0xffffffff)
Jan 19 05:13:07 athlon kernel: hdd: completing PM request, resume
Jan 19 05:13:07 athlon kernel: Fixing swap signatures... <3>bad: scheduling while atomic!
Jan 19 05:13:07 athlon kernel: Call Trace:
Jan 19 05:13:07 athlon kernel:  [<c011c1f6>] schedule+0x566/0x570
Jan 19 05:13:07 athlon kernel:  [<c0293c82>] generic_unplug_device+0x72/0x80
Jan 19 05:13:07 athlon kernel:  [<c0293e1d>] blk_run_queues+0xad/0xc0
Jan 19 05:13:07 athlon kernel:  [<c011d1be>] io_schedule+0xe/0x20
Jan 19 05:13:07 athlon kernel:  [<c013a962>] wait_on_page_bit+0xa2/0xd0
Jan 19 05:13:07 athlon kernel:  [<c011dad0>] autoremove_wake_function+0x0/0x50
Jan 19 05:13:07 athlon kernel:  [<c011dad0>] autoremove_wake_function+0x0/0x50
Jan 19 05:13:07 athlon kernel:  [<c015130b>] swap_readpage+0x5b/0x80
Jan 19 05:13:07 athlon kernel:  [<c01513ea>] rw_swap_page_sync+0xba/0x110
Jan 19 05:13:07 athlon kernel:  [<c0137a7e>] mark_swapfiles+0x7e/0x1b0
Jan 19 05:13:07 athlon kernel:  [<c01386b9>] do_magic_resume_2+0x99/0xe0
Jan 19 05:13:07 athlon kernel:  [<c0339adf>] do_magic+0x11f/0x140
Jan 19 05:13:07 athlon kernel:  [<c01388bb>] do_software_suspend+0x6b/0x90
Jan 19 05:13:07 athlon kernel:  [<c025a4aa>] acpi_system_write_sleep+0xc3/0xe6
Jan 19 05:13:07 athlon kernel:  [<c015652d>] vfs_write+0xad/0x120
Jan 19 05:13:07 athlon kernel:  [<c015663f>] sys_write+0x3f/0x60
Jan 19 05:13:07 athlon kernel:  [<c010b45b>] syscall_call+0x7/0xb
Jan 19 05:13:07 athlon kernel: ok
Jan 19 05:13:07 athlon kernel: Restarting tasks...<3>bad: scheduling while atomic!

Jan 19 05:13:07 athlon kernel:  done
Jan 19 05:13:07 athlon kernel: bad: scheduling while atomic!
Jan 19 05:13:07 athlon kernel: Call Trace:
Jan 19 05:13:07 athlon kernel:  [<c011c1f6>] schedule+0x566/0x570
Jan 19 05:13:07 athlon kernel:  [<c015663f>] sys_write+0x3f/0x60
Jan 19 05:13:07 athlon kernel:  [<c010b482>] work_resched+0x5/0x16
Jan 19 05:13:07 athlon kernel:
Jan 19 05:13:07 athlon kernel: bad: scheduling while atomic!
Jan 19 05:13:07 athlon kernel: Call Trace:
Jan 19 05:13:07 athlon kernel:  [<c011c1f6>] schedule+0x566/0x570
Jan 19 05:13:07 athlon kernel:  [<c011a30d>] do_page_fault+0x11d/0x579
Jan 19 05:13:07 athlon kernel:  [<c01202f3>] release_console_sem+0xd3/0xe0
Jan 19 05:13:07 athlon kernel:  [<c011d127>] sys_sched_yield+0x87/0xd0
Jan 19 05:13:07 athlon kernel:  [<c0162393>] coredump_wait+0x33/0xa0
Jan 19 05:13:07 athlon kernel:  [<c0136a1a>] __print_symbol+0x12a/0x160
Jan 19 05:13:07 athlon kernel:  [<c016250f>] do_coredump+0x10f/0x1f5
Jan 19 05:13:07 athlon kernel:  [<c01293b8>] specific_send_sig_info+0xc8/0x140
Jan 19 05:13:07 athlon kernel:  [<c0128de5>] __dequeue_signal+0xf5/0x1a0
Jan 19 05:13:07 athlon kernel:  [<c0128ebd>] dequeue_signal+0x2d/0x90
Jan 19 05:13:07 athlon kernel:  [<c012b3a5>] get_signal_to_deliver+0x205/0x380
Jan 19 05:13:07 athlon kernel:  [<c010b234>] do_signal+0xb4/0xf0
Jan 19 05:13:07 athlon kernel:  [<c011b260>] recalc_task_prio+0x90/0x1a0
Jan 19 05:13:07 athlon kernel:  [<c011bf8e>] schedule+0x2fe/0x570
Jan 19 05:13:07 athlon kernel:  [<c011a1f0>] do_page_fault+0x0/0x579
Jan 19 05:13:07 athlon kernel:  [<c010b2c9>] do_notify_resume+0x59/0x5c
Jan 19 05:13:07 athlon kernel:  [<c010b4a6>] work_notifysig+0x13/0x15
```

Ура! Все работает. Ну или почти все.

Если вам интересно, то смотрите свой syslog.

Единственное замечание, которое возникает, – запись «while atomic». Ядро находится в режиме, когда нельзя обрабатывать прерывания – разделять работу. Также рекомендуется не иметь «активных» процессов в системе во время подготовки ко сну. Под «активными» процессами я подразумеваю те, которые на момент засыпания ведут активные операции по переработке данных (пусть, например, это будет операция select в mysql-базе).

Перед засыпанием мы запускали dmesg:

```
Resume Machine: resuming from /dev/hdc5
Resuming from device hdc5
Resume Machine: This is normal swap space
PM: Reading pmdisk image.
PM: Resume from disk failed.
ACPI: (supports S0 S1 S3 S4 S5)
```

Подсистема гибернации обнаружила, что в swap-разделе запись «SWAPSPACE2»:

```
dd if=/dev/hdc5 bs=1k count=4 2> /dev/null | strings | grep SWAP
```

поэтому возобновление не было произведено.

После просыпания опять запустим dmesg:

```
Fixing swap signatures... ok
Restarting tasks... done
```

Все прошло удачно. Поздравляю!

Теперь немного пошалим. После очередного засыпания укажем ядру, что возобновление из swap-раздела делать не надо.

Передаем параметры ядру:

```
"noresume init=/bin/sh"
```

Мы не в полноценном режиме, поэтому процессы из /etc/rc.d не запустились.

```
dd if=/dev/hdc5 bs=1k count=4 2> /dev/null | strings | grep SUSP
```

На выходе у нас строчка «S2SUSP», означающая, что swap-раздел работает в качестве хранилища для содержимого RAM.

Поэтому, когда вы увидите строку «Fixing swap signatures... ok», не удивляйтесь, так и должно быть.

С подменой сигнатур мы разобрались. Теперь идем дальше. А что если усыпить систему на одном железе, а разбудить на другом? Ну что ж, скажу сразу, мне пока этого не удалось. Опишу свои действия.

Я пересобрал ядро, чтобы оптимизация была под 586MMX-архитектуру. Это позволит загрузиться как под Athlon XP, так и под Intel MMX процессорами.

Перед сессией засыпания под Athlon XP 1700+ процессором я указал ядру, что памяти у меня немного, всего лишь 32 Мб (mem=32M). Это необходимо, так как на системе, где я собираюсь проснуться (точнее не я, а мой slackware-винт), всего-то 32 Мб памяти.

Смотрим в syslog после просыпания:

```
Jan 24 11:47:44 athlon kernel: Resume Machine: resuming from /dev/hdc5
Jan 24 11:47:44 athlon kernel: Resuming from device hdc5
Jan 24 11:47:44 athlon kernel: Resume Machine: Signature found, resuming
Jan 24 11:47:44 athlon kernel: Resume Machine: Incorrect machine type
Jan 24 11:47:44 athlon kernel: Resume Machine: Error -1 resuming
```

Очень жаль, попытка не удалась.

Попробуем теперь произвести на сходной системе, процессор, правда, не Athlon XP 1700+, а чуть пониже – 1600+.

Еще раз «усыпляю» на родном железе и просыпаемся на новой машине. Скажу сразу – сообщений от ядра я не получил, так как «Resume Machine» при восстановлении регистров процессора приказала долго жить. Так что в этом направлении можно еще работать. Теоретически никто не запрещает, вариант вполне зрелый.

Обращаю ваше внимание, что когда винт с установленной системой перемещаете по разным машинам, не забывайте указывать, сколько памяти у вас есть. Иначе получите следующие сообщения:

```
Jan 22 16:59:02 athlon kernel: Resume Machine: Signature found, resuming
Jan 22 16:59:02 athlon kernel: Resume Machine: Incorrect memory size
Jan 22 16:59:02 athlon kernel: Resume Machine: Error -1 resuming
```

Теперь о самом интересном – о возможных последствиях при возобновлении работы.

Первое, что бросилось в глаза, – невозможно прочитать smb-расшаренные ресурсы. Системы, где расшарены ресурсы, пингуются, а получить файлы нельзя (точнее зайти в точки монтирования). Приходится размонтировать и заново примонтировать эти ресурсы.

```
cat /etc/mtab | grep smbfs
```

Узнали точки монтирования и знаем, что необходимо перемонтировать заново.

Второе. Приложения, которые перед засыпанием использовали звуковую карту, при возобновлении работы хранят молчание. Не произошла инициализация контроллера?

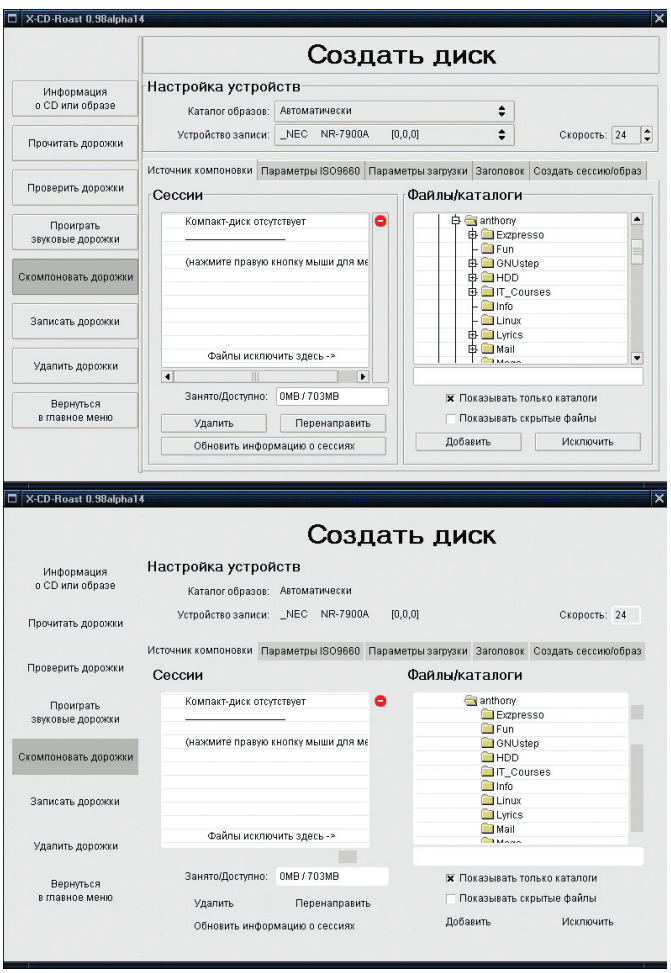
Укажу, что за система у меня.

```
bash-2.05b# lspci
00:00.0 Host bridge: VIA Technologies, Inc. VT8377 [KT400 AGP] Host Bridge
00:01.0 PCI bridge: VIA Technologies, Inc. VT8235 PCI Bridge
00:10.0 USB Controller: VIA Technologies, Inc. USB (rev 80)
00:10.1 USB Controller: VIA Technologies, Inc. USB (rev 80)
00:10.2 USB Controller: VIA Technologies, Inc. USB (rev 80)
00:10.3 USB Controller: VIA Technologies, Inc. USB 2.0 (rev 82)
00:11.0 ISA bridge: VIA Technologies, Inc. VT8235 ISA Bridge
00:11.1 IDE interface: VIA Technologies, Inc. VT82C586/B/686A/B PIPC Bus Master IDE
(rev 06)
00:11.5 Multimedia audio controller: VIA Technologies, Inc. VT8233 AC97 Audio Controller
(rev 50)
00:12.0 Ethernet controller: VIA Technologies, Inc. VT6102 [Rhine-II] (rev 74)
01:00.0 VGA compatible controller: nVidia Corporation NV18 [GeForce4 MX 440 AGP 8x]
(rev a2)
```

USB-свистка у меня нет, поэтому ничего сказать не могу про поведение USB-шины. Однако сообщения ниже меня смутили.

```
Jan 24 15:03:29 athlon kernel: drivers/usb/host/uhci-hcd.c: d800: host controller halted.
very bad
Jan 24 15:03:29 athlon kernel: drivers/usb/host/uhci-hcd.c: d400: host system error, PCI
problems?
Jan 24 15:03:29 athlon kernel: drivers/usb/host/uhci-hcd.c: d400: host controller halted.
very bad
Jan 24 15:03:29 athlon kernel: drivers/usb/host/uhci-hcd.c: d800: host controller halted.
very bad
Jan 24 15:03:29 athlon kernel: drivers/usb/host/uhci-hcd.c: d400: host controller halted.
very bad
Jan 24 15:03:29 athlon kernel: drivers/usb/host/uhci-hcd.c: d000: host system error, PCI
problems?
Jan 24 15:03:29 athlon kernel: drivers/usb/host/uhci-hcd.c: d000: host controller halted.
very bad
```

Под X-Window лично меня смущает, что библиотека GNOME неправильно отрисовывает некоторые элементы. Так как, некоторые приложения у меня на основе GNOME, то приведенные ниже картинки позволяют увидеть, что некоторые элементы отсутствуют. Видимо это ошибка в самой GNOME-библиотеке, хотя данный факт сейчас рассматривается на приложениях, использующих другие библиотеки.



Как видим, некоторые элементы отсутствуют. Рестарт X-сервера решает эти проблемы.

Итог

В целом, мне понравилась реализация «сна». При более детальном подходе, может быть, кто-то уже и решил указанные в статье недочеты, во всяком случае http://www.google.com – ваш надежный путеводитель.