Some warnings, first.

```
 * BIG FAT WARNING *********************************************************
 *
 * If you touch anything on disk between suspend and resume...
 *                               ...kiss your data goodbye.
 *
 * If you do resume from initrd after your filesystems are mounted...
 *                               ...bye bye root partition.
 *                       [this is actually same case as above]
 *
 * If you have unsupported (*) devices using DMA, you may have some
 * problems. If your disk driver does not support suspend... (IDE does),
 * it may cause some problems, too. If you change kernel command line
 * between suspend and resume, it may do something wrong. If you change
 * your hardware while system is suspended... well, it was not good idea;
 * but it will probably only crash.
 *
 * (*) suspend/resume support is needed to make it safe.
 *
 * If you have any filesystems on USB devices mounted before software suspend,
 * they won't be accessible after resume and you may lose data, as though
 * you have unplugged the USB devices with mounted filesystems on them;
 * see the FAQ below for details.  (This is not true for more traditional
 * power states like "standby", which normally don't turn USB off.)
```

You need to append resume=/dev/your_swap_partition to kernel command line. Then you suspend by

```
echo shutdown > /sys/power/disk; echo disk > /sys/power/state
```

. If you feel ACPI works pretty well on your system, you might try

```
echo platform > /sys/power/disk; echo disk > /sys/power/state
```

. If you have SATA disks, you'll need recent kernels with SATA suspend support. For suspend and resume to work, make sure your disk drivers are built into kernel -- not modules. [There's way to make suspend/resume with modular disk drivers, see FAQ, but you probably should not do that.]

If you want to limit the suspend image size to N bytes, do

```
echo N > /sys/power/image_size
```

before suspend (it is limited to 500 MB by default).

Article about goals and implementation of Software Suspend for Linux

Author: Gábor Kuti
Last revised: 2003-10-20 by Pavel Machek

Idea and goals to achieve

Nowadays it is common in several laptops that they have a suspend button. It

saves the state of the machine to a filesystem or to a partition and switches to standby mode. Later resuming the machine the saved state is loaded back to ram and the machine can continue its work. It has two real benefits. First we save ourselves the time machine goes down and later boots up, energy costs are real high when running from batteries. The other gain is that we don't have to
interrupt our programs so processes that are calculating something for a long time shouldn't need to be written interruptible.

swsusp saves the state of the machine into active swaps and then reboots or powerdowns. You must explicitly specify the swap partition to resume from with ``resume='' kernel option. If signature is found it loads and restores saved state. If the option ``noresume'' is specified as a boot parameter, it skips the resuming.

In the meantime while the system is suspended you should not add/remove any of the hardware, write to the filesystems, etc.

Sleep states summary
====================

There are three different interfaces you can use, /proc/acpi should work like this:

In a really perfect world:
```
echo 1 > /proc/acpi/sleep       # for standby
echo 2 > /proc/acpi/sleep       # for suspend to ram
echo 3 > /proc/acpi/sleep       # for suspend to ram, but with more power
conservative
echo 4 > /proc/acpi/sleep       # for suspend to disk
echo 5 > /proc/acpi/sleep       # for shutdown unfriendly the system
```

and perhaps
```
echo 4b > /proc/acpi/sleep      # for suspend to disk via s4bios
```

Frequently Asked Questions
==========================

Q: well, suspending a server is IMHO a really stupid thing,
but... (Diego Zuccato):

A: You bought new UPS for your server. How do you install it without bringing machine down? Suspend to disk, rearrange power cables, resume.

You have your server on UPS. Power died, and UPS is indicating 30 seconds to failure. What do you do? Suspend to disk.

Q: Maybe I'm missing something, but why don't the regular I/O paths work?

A: We do use the regular I/O paths. However we cannot restore the data to its original location as we load it. That would create an inconsistent kernel state which would certainly result in an oops. Instead, we load the image into unused memory and then atomically copy it back to it original location. This implies, of course, a maximum

image size of half the amount of memory.

There are two solutions to this:

* require half of memory to be free during suspend. That way you can read "new" data onto free spots, then cli and copy

* assume we had special "polling" ide driver that only uses memory between 0-640KB. That way, I'd have to make sure that 0-640KB is free during suspending, but otherwise it would work...

suspend2 shares this fundamental limitation, but does not include user data and disk caches into "used memory" by saving them in advance. That means that the limitation goes away in practice.

Q: Does linux support ACPI S4?

A: Yes. That's what echo platform > /sys/power/disk does.

Q: What is 'suspend2'?

A: suspend2 is 'Software Suspend 2', a forked implementation of suspend-to-disk which is available as separate patches for 2.4 and 2.6 kernels from swsusp.sourceforge.net. It includes support for SMP, 4GB highmem and preemption. It also has a extensible architecture that allows for arbitrary transformations on the image (compression, encryption) and arbitrary backends for writing the image (eg to swap or an NFS share[Work In Progress]). Questions regarding suspend2 should be sent to the mailing list available through the suspend2 website, and not to the Linux Kernel Mailing List. We are working toward merging suspend2 into the mainline kernel.

Q: What is the freezing of tasks and why are we using it?

A: The freezing of tasks is a mechanism by which user space processes and some kernel threads are controlled during hibernation or system-wide suspend (on some architectures). See freezing-of-tasks.txt for details.

Q: What is the difference between "platform" and "shutdown"?

A:

shutdown: save state in linux, then tell bios to powerdown

platform: save state in linux, then tell bios to powerdown and blink "suspended led"

"platform" is actually right thing to do where supported, but "shutdown" is most reliable (except on ACPI systems).

Q: I do not understand why you have such strong objections to idea of selective suspend.

A: Do selective suspend during runtime power management, that's okay. But it's useless for suspend-to-disk. (And I do not see how you could use it for suspend-to-ram, I hope you do not want that).

Lets see, so you suggest to

* SUSPEND all but swap device and parents
* Snapshot
* Write image to disk
* SUSPEND swap device and parents
* Powerdown

Oh no, that does not work, if swap device or its parents uses DMA,
you've corrupted data. You'd have to do

* SUSPEND all but swap device and parents
* FREEZE swap device and parents
* Snapshot
* UNFREEZE swap device and parents
* Write
* SUSPEND swap device and parents

Which means that you still need that FREEZE state, and you get more
complicated code. (And I have not yet introduce details like system
devices).

Q: There don't seem to be any generally useful behavioral
distinctions between SUSPEND and FREEZE.

A: Doing SUSPEND when you are asked to do FREEZE is always correct,
but it may be unneccessarily slow. If you want your driver to stay simple,
slowness may not matter to you. It can always be fixed later.

For devices like disk it does matter, you do not want to spindown for
FREEZE.

Q: After resuming, system is paging heavily, leading to very bad interactivity.

A: Try running

```
cat `cat /proc/[0-9]*/maps | grep / | sed 's:.* /:/:' | sort -u` > /dev/null
```

after resume. swapoff -a; swapon -a may also be useful.

Q: What happens to devices during swsusp? They seem to be resumed
during system suspend?

A: That's correct. We need to resume them if we want to write image to
disk. Whole sequence goes like

```
      Suspend part

      running system, user asks for suspend-to-disk

      user processes are stopped

      suspend(PMSG_FREEZE): devices are frozen so that they don't interfere
                            with state snapshot
```

state snapshot: copy of whole used memory is taken with interrupts disabled

resume(): devices are woken up so that we can write image to swap

write image to swap

suspend(PMSG_SUSPEND): suspend devices so that we can power off

turn the power off

Resume part

(is actually pretty similar)

running system, user asks for suspend-to-disk

user processes are stopped (in common case there are none, but with resume-from-initrd, noone knows)

read image from disk

suspend(PMSG_FREEZE): devices are frozen so that they don't interfere with image restoration

image restoration: rewrite memory with image

resume(): devices are woken up so that system can continue

thaw all user processes

Q: What is this 'Encrypt suspend image' for?

A: First of all: it is not a replacement for dm-crypt encrypted swap. It cannot protect your computer while it is suspended. Instead it does protect from leaking sensitive data after resume from suspend.

Think of the following: you suspend while an application is running that keeps sensitive data in memory. The application itself prevents the data from being swapped out. Suspend, however, must write these data to swap to be able to resume later on. Without suspend encryption your sensitive data are then stored in plaintext on disk. This means that after resume your sensitive data are accessible to all applications having direct access to the swap device which was used for suspend. If you don't need swap after resume these data can remain on disk virtually forever. Thus it can happen that your system gets broken in weeks later and sensitive data which you thought were encrypted and protected are retrieved and stolen from the swap device. To prevent this situation you should use 'Encrypt suspend image'.

During suspend a temporary key is created and this key is used to encrypt the data written to disk. When, during resume, the data was read back into memory the temporary key is destroyed which simply means that all data written to disk during suspend are then inaccessible so they can't be stolen later on. The only thing that you must then take care of is that you call 'mkswap' for the swap

partition used for suspend as early as possible during regular boot. This asserts that any temporary key from an oopsed suspend or from a failed or aborted resume is erased from the swap device.

As a rule of thumb use encrypted swap to protect your data while your system is shut down or suspended. Additionally use the encrypted suspend image to prevent sensitive data from being stolen after resume.

Q: Can I suspend to a swap file?

A: Generally, yes, you can. However, it requires you to use the "resume=" and "resume_offset=" kernel command line parameters, so the resume from a swap file cannot be initiated from an initrd or initramfs image. See swsusp-and-swap-files.txt for details.

Q: Is there a maximum system RAM size that is supported by swsusp?

A: It should work okay with highmem.

Q: Does swsusp (to disk) use only one swap partition or can it use multiple swap partitions (aggregate them into one logical space)?

A: Only one swap partition, sorry.

Q: If my application(s) causes lots of memory & swap space to be used (over half of the total system RAM), is it correct that it is likely to be useless to try to suspend to disk while that app is running?

A: No, it should work okay, as long as your app does not mlock() it. Just prepare big enough swap partition.

Q: What information is useful for debugging suspend-to-disk problems?

A: Well, last messages on the screen are always useful. If something is broken, it is usually some kernel driver, therefore trying with as little as possible modules loaded helps a lot. I also prefer people to suspend from console, preferably without X running. Booting with init=/bin/bash, then swapon and starting suspend sequence manually usually does the trick. Then it is good idea to try with latest vanilla kernel.

Q: How can distributions ship a swsusp-supporting kernel with modular disk drivers (especially SATA)?

A: Well, it can be done, load the drivers, then do echo into /sys/power/disk/resume file from initrd. Be sure not to mount anything, not even read-only mount, or you are going to lose your data.

Q: How do I make suspend more verbose?

A: If you want to see any non-error kernel messages on the virtual terminal the kernel switches to during suspend, you have to set the kernel console loglevel to at least 4 (KERN_WARNING), for example by doing

```
        # save the old loglevel
        read LOGLEVEL DUMMY < /proc/sys/kernel/printk
        # set the loglevel so we see the progress bar.
        # if the level is higher than needed, we leave it alone.
        if [ $LOGLEVEL -lt 5 ]; then
                echo 5 > /proc/sys/kernel/printk
                fi

        IMG_SZ=0
        read IMG_SZ < /sys/power/image_size
        echo -n disk > /sys/power/state
        RET=$?
        #
        # the logic here is:
        # if image_size > 0 (without kernel support, IMG_SZ will be zero),
        # then try again with image_size set to zero.
        if [ $RET -ne 0 -a $IMG_SZ -ne 0 ]; then # try again with minimal image
size
                echo 0 > /sys/power/image_size
                echo -n disk > /sys/power/state
                RET=$?
        fi

        # restore previous loglevel
        echo $LOGLEVEL > /proc/sys/kernel/printk
        exit $RET
```

Q: Is this true that if I have a mounted filesystem on a USB device and I suspend to disk, I can lose data unless the filesystem has been mounted with "sync"?

A: That's right ... if you disconnect that device, you may lose data. In fact, even with "-o sync" you can lose data if your programs have information in buffers they haven't written out to a disk you disconnect, or if you disconnect before the device finished saving data you wrote.

Software suspend normally powers down USB controllers, which is equivalent to disconnecting all USB devices attached to your system.

Your system might well support low-power modes for its USB controllers while the system is asleep, maintaining the connection, using true sleep modes like "suspend-to-RAM" or "standby". (Don't write "disk" to the /sys/power/state file; write "standby" or "mem".) We've not seen any hardware that can use these modes through software suspend, although in theory some systems might support "platform" modes that won't break the USB connections.

Remember that it's always a bad idea to unplug a disk drive containing a mounted filesystem. That's true even when your system is asleep! The safest thing is to unmount all filesystems on removable media (such USB, Firewire, CompactFlash, MMC, external SATA, or even IDE hotplug bays) before suspending; then remount them after resuming.

There is a work-around for this problem. For more information, see Documentation/usb/persist.txt.

Q: Can I suspend-to-disk using a swap partition under LVM?

A: No. You can suspend successfully, but you'll not be able to resume. uswsusp should be able to work with LVM. See suspend.sf.net.

Q: I upgraded the kernel from 2.6.15 to 2.6.16. Both kernels were compiled with the similar configuration files. Anyway I found that suspend to disk (and resume) is much slower on 2.6.16 compared to 2.6.15. Any idea for why that might happen or how can I speed it up?

A: This is because the size of the suspend image is now greater than for 2.6.15 (by saving more data we can get more responsive system after resume).

There's the /sys/power/image_size knob that controls the size of the image. If you set it to 0 (eg. by echo 0 > /sys/power/image_size as root), the 2.6.15 behavior should be restored. If it is still too slow, take a look at suspend.sf.net -- userland suspend is faster and supports LZF compression to speed it up further.