Linux SMP HOWTO

Table of Contents

Table of Contents

Linux SMP HOWTO

David Mentré, `David.Mentre@irisa.fr`

v1.12.1, 25 october 2000

This HOWTO reviews main issues (and I hope solutions) related to SMP configuration under Linux.

---

8. Glossary

9. What's new ?

10. List of contributors

---

1. Introduction

Linux works on SMP (Symmetric Multi−Processors) machines. SMP support was introduced with kernel version 2.0, and has improved steadily ever since. The kernel locking granularity is much finer in 2.2.x than in 2.0.x, which enables better performance when processes are accessing the kernel!

HOWTO maintained by David Mentré (David.Mentre@irisa.fr). The latest edition of this HOWTO can be found at

- http://www.irisa.fr/prive/mentre/smp−howto/ (France)
- http://www.phy.duke.edu/brahma/smp−faq/ (USA)

If you want to contribute to this HOWTO, I would prefer a diff against the SGML version of this document, but any remarks (in plain text) will be greatly appreciated. If you send me an email about this HOWTO, please include a tag like `[Linux SMP HOWTO]` in the `Subject:` field of your e−mail. It helps me to automatically sort mails (and you will have a faster reply `;)`).

This HOWTO is an improvement of a first draft made by **Chris Pirih**.

All information contained in this HOWTO is provided "as is." All warranties, expressed, implied or statutory, concerning the accuracy of the information of the suitability for any particular use are hereby specifically disclaimed. While every effort has been taken to ensure the accuracy of the information contained in this HOWTO, the authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

---

2. Questions related to any architectures

2.1 Kernel Side

1. **Does Linux support multi−threading? If I start two or more processes, will they be distributed among the available CPUs?**

   Yes. Processes and kernel−threads are distributed among processors. User−space threads are not.

2. **What kind of architectures are supported in SMP?**

   From Alan Cox:

   > SMP is supported in 2.0 on the hypersparc (SS20, etc.) systems and Intel 486, Pentium or higher machines which are Intel MP1.1/1.4 compliant. **Richard Jelinek** adds: right now, systems have been tested up to 4 CPUs and the MP standard (and so Linux) theoretically allows up to 16 CPUs.
   >
   > SMP support for UltraSparc, SparcServer, Alpha and PowerPC machines is in available in 2.2.x.

   From Ralf Bächle:

   > MIPS, m68k and ARM does not support SMP; the latter two probly won't ever.
   >
   > That is, I'm going to hack on MIPS−SMP as soon as I get a SMP box ...

3. **How do I make a Linux SMP kernel?**

   Most Linux distributions don't provide a ready−made SMP−aware kernel, which means that you'll have to make one yourself. If you haven't made your own kernel yet, this is a great reason to learn how. Explaining how to make a new kernel is beyond the scope of this document; refer to the Linux Kernel Howto for more information. (**C. Polisher**)

   In kernel series 2.0 up to but not including 2.1.132, uncomment the `SMP=1` line in the main Makefile (`/usr/src/linux/Makefile`).

   In the 2.2 version, configure the kernel and answer "yes" to the question "Symmetric multi−processing support" (**Michael Elizabeth Chastain**).

   AND

   enable real time clock support by configuring the "RTC support" item (in "Character Devices" menu) (from **Robert G. Brown**). Note that inserting RTC support actually doesn't afaik prevent the known problem with SMP clock drift, but enabling this feature prevents lockup when the clock is read at boot time. A note from **Richard Jelinek** says also that activating the Enhanced RTC is necessary to get the second CPU working (identified) on some original Intel Mainboards.

   AND

   (x86 kernel) do NOT enable APM (advanced power management)! APM and SMP are not compatible, and your system will almost certainly (or at least probably `;)`) crash while booting if APM is enabled (**Jakob Oestergaard**). **Alan Cox** confirms this : 2.1.x turns APM off for SMP boxes. Basically APM is undefined in the presence of SMP systems, and anything could occur.

   AND

   (x86 kernel) enable "MTRR (Memory Type Range Register) support". Some BIOS are buggy as they do not activate cache memory for the second processor. The MTRR support contains code that solves such processor misconfiguration.

You must rebuild all your kernel and kernel modules when changing to and from SMP mode. Remember to `make modules` and `make modules_install` (from **Alan Cox**).

If you get module load errors, you probably did not rebuild and/or re−install your modules. Also with some 2.2.x kernels people have reported problems when changing the compile from SMP back to UP (uni−processor). To fix this, save your .config file, do *make mrproper*, restore your *.config* file, then remake your kernel (*make dep*, etc.) (**Wade Hampton**). Do not forget to run lilo after copying your new kernel.

Recap:

```
make config # or menuconfig or xconfig
make dep
make clean
make bzImage # or whatever you want
# copy the kernel image manually then RUN LILO
# or make lilo
make modules
make modules_install
```

4. **How do I make a Linux non−SMP kernel?**

   In the 2.0 series, **comment** the `SMP=1` line in the main Makefile (/usr/src/linux/Makefile).

   In the 2.2 series, configure the kernel and answer "no" to the question "Symmetric multi−processing support" (**Michael Elizabeth Chastain**).

   You must rebuild all your kernel and kernel modules when changing to and from SMP mode. Remember to `make modules` and `make modules_install` and remember to run lilo. See notes above about possible configuration problems.
5. **How can I tell if it worked?**

```
 cat /proc/cpuinfo
```

   Typical output (dual PentiumII):

```
processor       : 0
cpu             : 686
model           : 3
vendor_id       : GenuineIntel
[...]
bogomips        : 267.06

processor       : 1
cpu             : 686
model           : 3
vendor_id       : GenuineIntel
[...]
bogomips        : 267.06
```

6. **What is the status of converting the kernel toward finer grained locking and multithreading?**

   Linux kernel version 2.2 has signal handling, interrupts and some I/O stuff fine grain locked. The rest is gradually migrating. All the scheduling is SMP safe.

   Kernel version 2.3 (next 2.4) has really fine grained locking. In the 2.3 kernels the usage of the big kernel lock has basically disappeared, all major Linux kernel subsystems are fully threaded: networking, VFS, VM, IO, block/page caches, scheduling, interrupts, signals, etc. (**Ingo Molnar**)

7. **Does Linux SMP support processor affinity?**

   Standard kernel

   No and Yes. There is no way to force a process onto specific CPU's but the linux scheduler has a processor bias for each process, which tends to keep processes tied to a specific CPU.

   Patch

   Yes. Look at PSET − Processor Sets for the Linux kernel:

   > The goal of this project is to make a source compatible and functionally equivalent version of pset (as defined by SGI − partially removed from their IRIX 6.4 kernel) for Linux. This enables users to determine which processor or set of processors a process may run on. Possible uses include forcing threads to separate processors, timings, security (a `root' only CPU?) and probably more.

   It is focused around the syscall sysmp(). This function takes a number of parameters that determine which function is requested. Functions include:

   - binding a process/thread to a specific CPU
   - restricting a CPU's ability to execute some processes
   - restricting a CPU from running at all
   - forcing a cpu to run _only_ one process (and its children)
   - getting information about a CPU's state
   - creating/destroying sets of processors, to which processes may be bound

8. **Where should one report SMP bugs to?**

   Please report bugs to `linux-smp@vger.kernel.org`.

9. **What about SMP performance?**

   If you want to gauge the performance of your SMP system, you can run some tests made by Cameron MacKinnon and available at http://www.phy.duke.edu/brahma/benchmarks.smp.

   Also have a look at this article by Bryant, Hartner, Qi and Venkitachalam that compares 2.2 and 2.3/2.4 UP and SMP kernels : SMP Scalability Comparisons of Linux¨ Kernels 2.2.14 and 2.3.99 (**Ray Bryant**) (You'll find also a copy here)

2.2 User Side

1. **Do I really need SMP?**

   If you have to ask, you probably don't. **:**) Generally, multi−processor systems can provide better performance than uni−processor systems, but to realize any gains you need to consider many other factors besides the number of CPU's. For instance, on a given system, if the processor is generally idle much of the time due to a slow disk drive, then this system is "input/output bound", and probably won't benefit from additional processing power. If, on the other hand, a system has many simultaneously executing processes, and CPU utilization is very high, then you are likely to realize increased system performance. SCSI disk drives can be very effective when used with multiple processors, due to the way they can process multiple commands without tying up the CPU. (**C. Polisher**)

2. **Do I get the same performance from 2−300 MHz processors as from one 600 MHz processor?**

   This depends on the application, but most likely not. SMP adds some overhead that a faster uniprocessor box would not incur (**Wade Hampton**). **:**)

3. **How does one display mutiple cpu performance?**

   Thanks to **Samuel S. Chessman**, here are some useful utilities:

   Character based:

   > http://www.cs.inf.ethz.ch/~rauch/procps.html
   >
   > Basically, it's procps v1.12.2 (top, ps, et. al.) and some patches to support SMP.
   >
   > For 2.2.x, **Gregory R. Warnes** as made a patch available at http://queenbee.fhcrc.org/~warnes/procps

   Graphic:

   > xosview−1.5.1 supports SMP. And kernels above 2.1.85 (included) the cpuX entry in /proc/stat file.
   >
   > The official homepage for xosview is: http://lore.ece.utexas.edu/~bgrayson/xosview.html
   >
   > You'll find a version patched for 2.2.x kernels by **Kumsup Lee** : http://www.ima.umn.edu/~klee/linux/xosview−1.6.1−5a1.tgz

   By the way, you can't monitor processor scheduling precisely with xosview, as xosview itself causes a scheduling perturbation. (**H. Peter Anvin**)

   And **Rik van Riel** tell us why:

   > The answer is pretty simple. Basically there are 3 processes involved:
   >
   > 1. the cpu hog (low scheduling priority because it eats CPU)

2. xosview
3. X

The CPU hog is running on one CPU. Then xosview wakes up (on the other CPU) and starts sending commands to X, which wakes up as well.

Since both X and xosview have a much higher priority than the CPU hog, xosview will run on one CPU and X on the other.

Then xosview stops running and we have an idle CPU −−> Linux moves the CPU hog over to the newly idle CPU (X is still running on the CPU our hog was running on just before).

4. **How can I enable more than 1 process for my kernel compile?**

use:

```
# make [modules|zImage|bzImages] MAKE="make -jX"
where X=max number of processes.
WARNING: This won't work for "make dep".
```

With a 2.2 like kernel, see also the file `/usr/src/linux/Documentation/smp.txt` for specific instruction.

BTW, since running multiple compilers allows a machine with sufficient memory to use use the otherwise wasted CPU time during I/O caused delays, `make MAKE="make -j 2" -j 2` actually helps even on uniprocessor boxes (from **Ralf Bächle**).

5. **Why is the time given by the `time` command inaccurate?** (from **Joel Marchand**)

In the 2.0 series, the result given by the `time` command is false. The sum user+system is right *but* the spreading between user and system time is false.

More precisely: "The explanation is, that all time spent in processors other than the boot cpu is accounted as system time. If you time a program, add the user time and the system time, then you timing will be almost right, except for also including the system time that is correctly accounted for" (**Jakob Østergaard**).

This bug is corrected in 2.2 kernels.

2.3 SMP Programming

Section by **Jakob Østergaard**.

This section is intended to outline what works, and what doesn't when it comes to programming multi−threaded software for SMP Linux.

Parallelization methods

1. POSIX Threads
2. PVM / MPI Message Passing Libraries
3. fork() −− Multiple processes

Since both fork() and PVM/MPI processes usually do not share memory, but either communicate by means of IPC or a messaging API, they will not be described further in this section. They are not very specific to SMP, since they are used just as much − or more − on uniprocessor computers, and clusters thereof.

Only POSIX Threads provide us with multiple threads sharing ressources like − especially − memory. This is the thing that makes a SMP machine special, allowing many processors to share their memory. To use both (or more ;) processors of an SMP, use a kernel−thread library. A good library is the LinuxThreads, a pthread library made by Xavier Leroy which is now integrated with glibc2 (aka libc6). Newer Linux distributions include this library by default, hence you do not have to obtain a separate package to use kernel threads.

There are implementations of threads (and POSIX threads) that are application−level, and do not take advantage of the kernel−threading. These thread packages keep the threading in a single process, hence do not take advantage of SMP. However, they are good for many applications and tend to actually run faster than kernel−threads on single processor systems.

Multi−threading has never been really popular in the UN*X world though. For some reason, applications requiring multiple processes or threads, have mostly been written using fork(). Therefore, when using the thread approach, one runs into problems of incompatible (not thread−ready) libraries, compilers, and debuggers. GNU/Linux is no exception to this. Hopefully the next few sections will sched a little light over what is currently possible, and what is not.

The C Library

Older C libraries are not thread−safe. It is very important that you use GNU LibC (**glibc**), also known as **libc6**. Earlier versions are, of course possible to use, but it will cause you much more trouble than upgrading your system will, well probably :)

If you want to use GDB to debug your programs, see below.

Languages, Compilers and debuggers

There is a wealth of programming languages available for GNU/Linux, and many of them can be made to use threads one way or the other (some languages like Ada and Java even have threads as primitives in the language).

This section will, however, currently only describe C and C++. If you have experience in SMP Programming with other languages, please enlighten us.

GNU C and C++, as well as the EGCS C and C++ compilers work with the thread support from the standard C library (**glibc**). There are however a few issues:

1. When compiling C or C++, use the **−D_REENTRANT** define in the compiler command line. This is necessary to make certain error−handling functions work like the errno variable.
2. When using C++, If two threads throw exceptions concurrently, the program will segfault. The

compiler does not generate thread−safe exception code. The workaround is to put a pthread_mutex_lock(&global_exception_lock) in the constructor(s) of every class you throw(), and to put the corresponding pthread_mutex_unlock(...) in the destructor. It's ugly, but it works. This solution was given by **Markus Ferch**.

The GNU Debugger **GDB** as of version 4.18, should handle threads correctly. Most Linux distribution offer a patched, thread−aware gdb.

It is not necessary to patch **glibc** in any way just to make it work with threads. If you do not need to debug the software (this could be true for all machines that are not development workstations), there is no need to patch **glibc**.

Note that core−dumps are of no use when using multiple threads. Somehow, the core dump is attached to one of the currently running threads, and not to the program as a whole. Therefore, whenever you are debugging anything, run it from the debugger.

Hint: If you have a thread running haywire, like eating 100% CPU time, and you cannot seem to figure out why, here is a nice way to find out what's going on: Run the program straight from the shell, no GDB. Make the thread go haywire. Use **top** to get the PID of the process. Run GDB like **gdb program pid**. This will make GDB attach itself to the process with the PID you specified, and stop the thead. Now you have a GDB session with the offending thread, and can use **bt** and the like to see what is happening.

Other libraries

ElectricFence: This library is not thread safe. It should be possible, however, to make it work in SMP environments by inserting mutex locks in the ElectricFence code.

Other points about SMP Programming

1. **Where can I found more information about parallel programming?**

   Look at the Linux Parallel Processing HOWTO

   Lots of useful information can be found at Parallel Processing using Linux

   Look also at the Linux Threads FAQ

2. **Are there any threaded programs or libraries?**

   Yes. For programs, you should look at: Multithreaded programs on linux (I love hyperlinks, did you know that ? `;`))

   As far as library are concerned, there are:

   OpenGL Mesa library

   > Thanks to **David Buccarelli**, **Andreas Schiffler** and **Emil Briggs**, it exists in a multithreaded version (right now [1998−05−11], there is a working version that provides speedups of 5−30% on some OpenGL benchmarks). The multithreaded stuff is now included in the regular Mesa distribution as an experimental option. For more information, look at the

Mesa library

BLAS

Pentium Pro Optimized BLAS and FFTs for Intel Linux

Multithreaded BLAS routines are not available right now, but a dual proc library is planned for 1998−05−27, see Blas News for details.

The GIMP

Emil Briggs, the same guy who is involved in multithreaded Mesa, is also working on multithreaded The GIMP plugins. Look at http://nemo.physics.ncsu.edu/~briggs/gimp/index.html for more info.

---

3. x86 architecture specific questions

3.1 Why it doesn't work on my machine?

1. **Can I use my Cyrix/AMD/non−Intel CPU in SMP?**

   Short answer: no.

   Long answer: Intel claims ownership to the APIC SMP scheme, and unless a company licenses it from Intel they may not use it. There are currently no companies that have done so. (This of course can change in the future) FYI − Both Cyrix and AMD support the non−proprietary OpenPIC SMP standard but currently there are no motherboards that use it.

2. **Why doesn't my old Compaq work?**

   Put it into MP1.1/1.4 compliant mode.

   check "Configure Hardware" −> "View / Edit details" −> "Advanced mode" (F7 I think) for a configuration option "APIC mode" and set this to "full Table mode". This is an official Compaq recommandation. (**Daniel Roesen**)

   (**Adrian Portelli**)To do this:

   1. Press F10 when the server boots to enter the System Configuration Utility
   2. Press Enter to dismiss the splash screen
   3. Immediately press CTRL+A
   4. A message will appear informing you that you are now in "Advanced Mode"
   5. Then select "Configure Hardware" −> "View / Edit details"
   6. You will then see the advanced settings (intermixed with the ordinary ones)
   7. Stroll down to "APIC Mode" and then select "Fully Mapped"
   8. Save changes and reboot

3. **Why doesnt my ALR work?**

>From **Robert Hyatt** : ALR Revolution quad−6 seems quite safe, while some older revolution quad machines without P6 processors seem "iffy"...

4. **Why does SMP go so slowly?** or **Why does one CPU show a very low bogomips value while the first one is normal?**

   >From **Alan Cox**: If one of your CPU's is reporting a very low bogomips value the cache is not enabled on it. Your vendor probably provides a buggy BIOS. Get the patch to work around this or better yet send it back and buy a board from a competent supplier.

   A 2.0 kernel (> 2.0.36) contains the MTRR patch which should solve this problem (select option "Handle buggy SMP BIOSes with bad MTRR setup" in the "General setup" menu).

   I think buggy SMP BIOS handling is automatic in latest 2.2 kernels.
5. **I've heard IBM machines have problems**

   Some IBM machines have the MP1.4 bios block in the EBDA, allowed but not supported below 2.2 kernels.

   There is an old 486SLC based IBM SMP box. Linux/SMP requires hardware FPU support.
6. **Is there any advantage of Intel MP 1.4 over 1.1 specification?**

   Nope (according to Alan **:)**), 1.4 is just a stricker specs of 1.1.
7. **Why does the clock drift so rapidly when I run linux SMP?**

   This is known problem with IRQ handling and long kernel locks in the 2.0 series kernels. Consider upgrading to a later 2.2 kernel.

   >From **Jakob Oestergaard**: Or, consider running xntpd. That should keep your clock right on time. (I think that I've heard that enabling RTC in the kernel also fixes the clock drift. It works for me! but I'm not sure whether that's general or I'm just being lucky)

   There are some kernel fixes in the later 2.2.x series that may fix this.
8. **Why are my CPU's numbered 0 and 2 instead of 0 and 1 (or some other odd numbering)?**

   The CPU number is assigned by the MB manufacturer and doesn't mean anything. Ignore it.
9. **My quad−Xeon system hangs as soon as it has decompressed the kernel**

   (**Doug Ledford**) Try recompiling LILO with LARGE_EBDA support and then making sure to always use make bzImage when compiling the kernel. That appears to have fixed the SMP boot hangs here on Intel multi−Xeon boards. However, please note that this also appears to break LILO in that the root= option no longer works, so make sure you rdev your kernel image at the same time you run lilo to make sure that the kernel loads the correct root filesystem at boot.

   (**Robert M. Hyatt**) With 3 cpus, do you have a terminator in the 4th slot?

10. **During boot machine hang signaling an IOAPIC problem**

    Try boot options "noapic" (**John Aldrich**) and/or "reboot=bios" (**Terry Shull**).

11. **My system locks up during heavy NFS traffic**

    Try the later 2.2.x kernels and the knfsd patches. This is currently under investigation. (**Wade Hampton**)

12. **My system locks up with no oops messages**

    If you are using kernels 2.2.11 or 2.2.12, get the latest kernel. For example 2.2.13 has a number of SMP fixes. Several people have reported these kernels to be unstable for SMP. These same kernels may have NFS problems that can cause lockups. Also, use a serial console to capture your oops messages. (**Wade Hampton**)

    If the problem remains (and the other suggestions on this list didn't help either), then you could try the latest 2.3 kernels. They have more verbose (and more robust) SMP/APIC code, and automatic hard−lockup−prevention code which will produce meaningful oopses instead of a silent hang. (**Ingo Molnar**)

    (**Osamu Aoki**) You MUST also *disable* all BIOS related power save features. Example of good configuration (Dual Celeron 466 Abit BP6):

    ```
    POWER MANAGEMENT SETUP.
      ACPI:             Disabled
      POWER MANAGEMENT: Disabled
      PM CONTROL by APM: No
    ```

    If power management features are activated, some random freeze can occur.

13. **Debugging lockups**

    (item by **Wade Hampton**)

    A good means of debugging lockups is to get the ikd patch from Andrea Arcangeli: ftp://ftp.suse.com/pub/people/andrea/kernel−patches

    There are several of debug options, but do NOT use the soft lockup option! For newer SMP boxes, turn kernel debugging then turn on the NMI oopser. To verify that the NMI oopser is working, after booting the new kernel, `/cat /proc/interrupts` and verify that you are getting NMIs. When the box locks up, you should get an OOPS.

    You may also try the %eip option. This allows the kernel to print on the console the %eip address every time a kernel function is called. When the box locks up, write down the first column ordered by the second column then lookup the addresses in the System.map file. This works only in console mode.

    Also note that the use of a serial console can greatly facilitate debugging kernel lockups, not just

SMP kernel lockups!

14. **"APIC error interrupt on CPU#n, should never happen" messages in logs**

    A message like:

    ```
    APIC error interrupt on CPU#0, should never happen.
    ... APIC ESR0: 00000002
    ... APIC ESR1: 00000000
    ```

    indicates a 'receive checksum error'. This cannot be caused by Linux as the APIC message checksumming part is completely in hardware. It might be marginal hardware. As long as you dont see any instability, they are not a problem − APIC messages are retried until delivered. (**Ingo Molnar**)

3.2 Possible causes of crash

In this section you'll find some **possible** reasons for a crash of an SMP machine (credits are due to **Jakob Østergaard** for this part). As far as I (David) know, theses problems are Intel specific.

- **Cooling problems**

  >From **Ralf Bächle**: [Related to case size and fans] It's important that the air is flowing. It of course can't where cables etc. are preventing this like in too small cases. On the other side I've seen oversized cases causing big problems. There are some tower cases on the market that actually are worse for cooling than desktops. In short, the right thing is thinking about aerodynamics in the case. Extra cases for hot peripherals are usefull as well.

  Of course you can always go to Radio Shack (or similar) and get another fan. You can use the lm_sensors to monitor the CPU temperature of newer PII and PIII processors. This might help you to determine if heat is a problem. (**Wade Hampton**)

- **Bad memory**

  Don't buy cheap RAM and don't use mixed RAM modules on a motherboard that is picky about it.

  Especially Tyan motherboards are known to be picky about RAM speed.

  There have been some report of 10ns PC100 RAM being sold with motherboards where the CPU really needs 8ns RAM. (**Wade Hampton**)

- **Bad combination of different stepping CPUs**

  Check `/proc/cpuinfo` to see that your CPUs are same stepping.

- **If your system is unstable, then DON'T overclock it!**

  ...and even if it is stable, DON'T overclock.

>From **Ralf Bächle**: Overclocking causes very subtle problems. I have a nice example, one of my overclocked old machines misscomputes a couple of pixels of a 640 x 400 fractal. The problem is only visible when comparing them using tools. So better say *never, nuncas, jamais, niemals* overclock.

- **2.0.x kernel and fast ethernet** (from **Robert G. Brown**)

  2.0.x kernels on high performance fast ethernet systems have significant (and known) problems with a race/deadlock condition in the networking interrupt handler.

  The solution is to get the latest 100BT development drivers from CESDIS Linux Ethernet device drivers site (ones that define SMPCHECK).

- **A bug in the 440FX chipset** (from **Emil Briggs**)

  If you had a system using the 440FX chipset then your problem with the lockups was possibly due to a documented errata in the chipset. Here is a reference

  References: Intel 440FX PCIset 82441FX (PMC) and 82442FX (DBX) Specification Update. pg. 13

  http://www.intel.com/design/pcisets/specupdt/297654.htm

  The problem can be fixed with a BIOS workaround (Or a kernel patch) and in fact David Wragg wrote a patch that's included with Richard Gooch's MTTR patch. For more information and a fix look here:

  http://nemo.physics.ncsu.edu/~briggs/vfix.html

- **DONT run emm386.exe before booting linux SMP**

  >From **Mark Duguid**, dumb rule #1 with W6LI motherboards. `;)`

- **If the machine reboots/freezes after a while, there can be two good BIOS + memory related reasons** (from **Jakob Østergaard**)
  - If the BIOS has settings like "memory hole at 16M" and/or "OS/2 memory > 64MB", try disabling them both. Linux does not always react well with theese options.
  - If you have more than 64 MB of memory in the machine, and you specified the exact number manually in the LILO configuration, you should specify one MB less than you actually have in the machine. If you have 128 MB, you lilo.conf line looks like: append="mem=127M"
- **Be aware of IRQ related problems**

  Sometime, some cards are not recognized or can trigger IRQ conflicts. Try shuffling cards on slots in different ways and possibly moving them to different IRQs.

  Contributed by **hASCII** : removing an " append="hisax=9,2,3"" line in lilo.conf allowed using a kernel from the 2.1.xx series with activated ISDN + Hisax support. Kernels from the 2.0.xx series doesn't make problems like this.

  Try also to set BIOS setup option like "MP 1.4 mode" or "route PCI interrupts through IOAPIC", or "OS Type" not set to DOS neither Novell (**Ingo Molnar**).

- **Floppy access while sound is active**

  If you lockup when trying to access the floppy (for example while sound is playing) you may have to edit drivers/pci/quirks.c and set `/int isa_dma_bridge_buggy = 1;` This is a problem with my Dell WS400 dual PII/300, 2.2.x, SMP (**Wade Hampton**).

3.3 Motherboard specific information

Please note: Some more specific information can be found with the list of Motherboards rumored to run Linux SMP

Motherboards with known problems

- none right now

3.4 Low cost SMP Linux box (dual Celeron box)

(**Stéphane Écolivet**)

The lowest cost SMP Linux boxes with nowadays buyable processors are dual Celeron systems. Such a system is not officially possible according to Intel. Better think about the second generation of Celeron, those with 128 Kb L2 cache.

Is it possible to run a dual Intel Celeron box ?

Official answer from Intel: no, Celeron cannot work in SMP mode.

Practical answer: it is possible, but requires hardware alteration for Slot 1 processors. Alteration is described by Tomohiro Kawada on his Dual Celeron System page. Of course, this kind of modification removes warranties... Some versions of Celeron processor are also available in Socket 370 format. In that case, alteration may just be done on the Socket 370 to Slot 1 adapter or may even be sold pre−wired for SMP use. (**Andy Poling**, **Hans − Erik Skyttberg**, **James Beard**)

There is also a motherboard (ABIT BP6) allowing two Celerons in Socket 370 format to be inserted (**Martijn Kruithof**, **Ryan McCue**). ABIT Computer BP6 verified tested and native to linux with dual ppga socket 370 (**Andre Hedrick**).

How does Linux behave on a dual Celeron system ?

Fine, thank you.

Celeron processors are known to be easily overclockable. And dualCeleron system ?

It **may** work. However, overclocking this kind of system is not as easy as overclocking a mono−processor one. It is definitly not a good idea for a production system. For personal use, dual Celeron 300A systems running rock−solid at 450 MHz have been reported. (**numerous people**)

And making a quad Celeron system ?

It is impossible. Celeron processors have nearly the same features as basic Pentium II chips. If you want more than 2 processors in your system, you'll have to look at Pentium Pro, Pentium II Xeon or Pentium III (?) boxes.

What about mixing Celeron and Pentium II processor ?

A system using a "re−enable" Celeron processor and a Pentium II processor with the same steppings **may theorically** work.

Alexandre Charbey as made such a system:

- Asus P2B−D motherboard, proc 1: Celeron 366, proc 2: Pentium II 400@266
- 66Mhz and 75Mhz bus frenquencies where functionnal
- the fastest processor (in this case the Celeron) should be put on the second slot. Swapping processors (fatest first) leads to quick failure.

---

4. Sparc architecture specific questions

4.1 Which Sparc machines are supported ?

Quoting the UltraLinux web page (only SMP systems):

- UltraSPARC PCI based workstations: Ultra60, Ultra450
- UltraSPARC SBUS based servers: Enterprise 1, 2, 150
- UltraSPARC SBUS based large servers: Enterprise 3000, 4000, 5000, 6000, 10000
- UltraSPARC PCI based servers: Enterprise 250, 450
- SPARC sun4m SMP machines (**Anton Blanchard**)
- Starfire E10000

UltraLinux has ran on a 14 CPUs machine (see the dmesg output) and on a Starfire E10000 with 24 CPUs (see the dmesg output).

4.2 Specific problem related to Sparc SMP support

(**David Miller**) There should not be any worries.

The only known problem, and one we don't intend to fix, is that if you build an SMP kernel for 32−bit (ie. non−ultrasparc) systems, this kernel will not work on sun4c systems.

4.3 SMP specific limit with current kernel (2.2)

(**David Miller**) There is a bug in the include/linux/tasks.h header file, it needs to define NR_CPUS to 64 on UltraSparc as this is the upper limit for the hardware we support :−)

---

5. PowerPC architecture specific questions

5.1 Which PPC machines are supported ?

* PowerSurge boards (including UMAX s900)
* PowerMac
* Motorola MTX: support under developement. Patches not yet integrated into the main kernel (**Troy Benjegerdes**)

(**Cort Dougan**) Not supported: PPC RS/6000 systems

5.2 Specific problem related to PPC SMP support

Nothing. Usual SMP compiling (see above). As usual, be aware, modules are specific either for UP or SMP. Recompile them. (**Paul Mackerras**)

---

6. Alpha architecture specific questions

6.1 Which Alpha machines are supported ?

(**Geerten Kuiper**) SMP works for most, if not all, AXP servers.

(**Jay A Estabrook**) SMP does seem to work on most of our [Compaq] boxes with 2 or more CPUs. That includes :

* AS2000/2100 (SABLE)
* AS4000/4100 (RAWHIDE)
* DS20 (DP264)
* GS320 (see the bootlog for a 31 CPUs machine)

It does not include :

* AS2100A (LYNX)
* TurboLaser bigboys (8200/8400)

(**Alpha Processor Inc**) SMP support has been qualified for all API SMP systems starting from later 2.2−series kernels (approximately kernel 2.2.7). At the time of writing, that is :

* DP264
* UP2000

See API's support website for more info.

6.2 Specific problem related to Alpha SMP support

None (really ? :−)

---

7. Useful pointers

7.1 Various

- Parallel Processing using Linux
- Linux Parallel Processing HOWTO
- **(outdated)** Linux SMP home page
- linux−smp mailing list

  To **subscribe**, send `subscribe linux-smp` in the message body at majordomo@vger.kernel.org

  To **unsubscribe**, send `unsubscribe linux-smp` in the message body at majordomo@vger.kernel.org

  Linux SMP archives

  Linux SMP archives at progressive−comp.com
- pthread library made by Xavier Leroy
- Motherboards rumored to run Linux SMP
- procps
- procps patch for 2.2.x
- xosview
- xosview for 2.2.x
- SMP Performance of Linux
- CESDIS Linux Ethernet device drivers site
- Dual Celeron System

7.2 Multithreaded programs and library

- Linux Threads FAQ
- Multithreaded programs on linux
- Pentium Pro Optimized BLAS and FFTs for Intel Linux (not available right now, but a dual proc library is planned for 5/27/98, see Blas News for details)
- Mesa library (with experimental multi−threading)
- Parallel plugins for The GIMP

7.3 SMP specific patches

- Patch for a bug in the 440FX chipset
- PSET − Processor Sets for the Linux kernel
- Ingo Molnar SMP patches (for experts only, please read linux−smp@vger.kernel.org)

7.4 Parallelizing/Optimizing Compilers for 586/686 machines (Sumit Roy)

- Pentium Compiler Group creators of pgcc

- Absoft , Fortran 90 and Fortran 77 compilers
- The Portland Group, Inc., supports the OpenMP standard for Fortran parallelization on Linux
- Pacific−Sierra Research Corporation, has a free F90 compiler for Linux, as well as parallelizing compilers for SMP Linux
- Applied Parallel Research, currently have parallelizing compilers for WinNT
- KAI has a C++−Compiler for Linux, that understands OpenMPI. It is called Guide_OpenMP. Info under http://www.kai.com/parallel/kappro/guide. (**Gero Wedemann**)

---

8. Glossary

8.1 Definitions

- **SMP** Symmetric Multi−Processors.
- **UP** Uni−Processor: system with one processor.
- **APIC** Advanced Programmable Interrupt Controler.
- **thread** A thread is a processor activity in a process. The same process can have multiple threads. Those threads share the process address space and can therefore share data.
- **pthread** Posix thread, threads defined by the Posix standard.
- **process** Program in execution, with its environment.
- **MTRR** Memory Type Range Register
- **APM** Advanced Power Management.
- **FPU** Floating Point Unit. Also called arithmetic co−processor.
- **IRQ** Interrupt ReQuest.
- **EBDA** ??
- **oops** Internal kernel error.
- **Cluster** Group of computers that achieve a common computation (also known as Beowulf within the Linux community).

8.2 Concepts

- **Data Races**

  A data race happens when to processes want to modify a shared variable concurrently without protecting themselves from the effect of the other process.

  Let A a shared variable. Let P1 and P2 two processes that access this variable. Those two processes are making the same following operation: "read A in tmp variable (local to the precess); do tmp = tmp + 1 ; write tmp in A". If the A variable is not protected by a lock, resulting executions could not correspond to what is espected. For example, here is two examples if one do not lock A:

```
case #1:
A=0
P1: read A -> tmp1 (so tmp1 is 0)
P2: read A -> tmp2 (so tmp2 is 0)
P1: tmp1 = tmp1 + 1 (so tmp1 is 1)
P2: tmp2 = tmp2 + 1 (so tmp2 is 1)
P1: tmp1 -> write A (so A is 1)
P2: tmp2 -> write A (so A is 1)

case #2:
```

```
A=0
P1: read A -> tmp1 (so tmp1 is 0)
P1: tmp1 = tmp1 + 1 (so tmp1 is 1)
P1: tmp1 -> write A (so A is 1)
P2: read A -> tmp2 (so tmp2 is 1)
P2: tmp2 = tmp2 + 1 (so tmp2 is 2)
P2: tmp2 -> write A (so A is 2)
```

To avoid this kind of problem, one uses a lock:

```
A=0:
P1: lock A
P1: read A -> tmp1 (so tmp1 is 0)
P2: lock A (so P2 is blocked)
P1: tmp1 = tmp1 + 1 (so tmp1 is 1)
P1: tmp1 -> write A (so A is 1)
P1: unlock A (so P2 is unblocked)
P2: read A -> tmp2 (so tmp2 is 1)
P2: tmp2 = tmp2 + 1 (so tmp2 is 2)
P2: tmp2 -> write A (so A is 2)
P2: unlock A
```

- **Deadlock**

  This is an inter−blocking that occurs when two processes want to access at shared variables mutually locked. For example, let A and B two locks and P1 and P2 two processes:

```
P1: lock A
P2: lock B
P1: lock B (so P1 is blocked by P2)
P2: lock A (so P2 is blocked by P1)
```

  Process P1 is blocked because it is waiting for the unlocking of B variable by P2. However P2 also needs the A variable to finish its computation and free B. So we have a deadlock.

  In this example, the problem is very simple. But imagine what can happen in a 2 millions of lines of code (like the linux kernel) with hundreds of locks. :−)

9. What's new ?

v1.12.1, 25 october 2000

- Put all authors in Bryant, Hartner, Qi and Venkitachalam paper

v1.12, 22 october 2000

- Explanation on why not trust Xosview on scheduling (**Rik van Riel**)
- A pointer to an article that compares 2.2 and 2.4 kernels (**Ray Bryant**)

v1.11, 8 october 2000

- Linux boots on a Sun E1000 with 24 CPUs
- Linux boots on a AlphaServer with 31 CPUs

v1.10, 5 october 2000

- New linux−smp mailing−list adress : linux−smp@vger.kernel.org (me)
- Tell where to find RTC setting in kernel config (**Patrick Doyle**)
- glossary updated and concepts added (from a french version made by **Ludovic Danigo**)
- Fixed an inconsistency (**Matthias Schniedermeyer**)
- Deleted wrong links (**Johan Ekenberg**)

v1.9.1, 28 September 2000

- updated with a submission from **Stig Telfer** detailing SMP support on API Alpha systems

v1.9, 13 january 2000

- Remember to disable all BIOS power−save features (**Osamu Aoki**)
- Explain how to access to Compaq server into advanced configuration mode (**Adrian Portelli**)

v1.8, 8 november 1999

- quad−celeron motherboard was a hoax, restored old paragraph (**Simen Timian Thoresen**)

v1.7, 6 november 1999

- new introduction (**C. Polisher** aka cp)
- numerous typo and grammatical fixes (cp)
- introductory paragraph on kernel compilation (cp)
- introductory paragraph on SMP need (cp)
- reference on KAI optimizing compiler (**Gero Wedemann**)
- quad−celeron motherboard exists (**Jeffrey H. Ingber**)

v1.6, 21 october 1999

- added information on xosview scheduling perturbation
- added "APIC error interrupt on CPU#n" message information
- added information on hard lockup
- deleted section "How to optain maximum performance" (was obsolete)
- added info on dual systems with different x86 procs (a Celeron and a P−II)

v1.5, 4 october 1999

- more precision in PSET description

v1.4, 30 september 1999

- precize to enable MTRR support for an x86 SMP kernel (me)

v1.3, 29 september 1999

- many many grammar and typographical fixes (**Wade Hampton** aka hww)
- added info in short introduction related to 2.2/2.4/2.0 diffs (hww)
- added step by step things to do to recompile a kernel (hww and me)

- added info related to SMP/UP modules problems (hww)
- added precision in Posix Threads section related to user (hww) vs. kernel threads (hww)
- new item about NFS and kernel lock (hww)
- new item about kernel lock without message (hww)
- new item about debugging lockup problems (hww)
- added info about heating problems (hww)
- miscellaneous updates I've forget about (hww)
- new item about floppy access and sound (hww)

v1.2, 27 septembre 1999

- name change: this document is now a HOWTO. TWD, and fast! (**Guylhem Aznar**)

v1.1, 26 septembre 1999

- added a link to first Chris Pirih FAQ draft
- expanted an IRQ related problems

v1.00, 25 septembre 1999

- first upgrade in a long long time!
- reprocessed the whole FAQ: 2.2 is here and 2.4 soon
- added kernel locking information from Ingo Molnar
- deleted item "How will my application perform under SMP?": outdated
- deleted item "My SMP system is locking up all the time.": outdated
- deleted item "You are running 2.0.35 aren't you ?": outdated
- deleted item "Some hardware is also known to cause problems.": outdated
- blanked section "Motherboards with known problems". We should restart from scratch
- deleted section "Motherboards with NO known problems": outdated
- updated dual celeron section (numerous people)
- added "SPARC sun4m SMP machines" to supported SMP sparc machines (**Anton Blanchard**)
- added a "During boot machine hang signaling an IOAPIC problem" item in "Why it doesn't work on my machine?" section
- added a "What about SMP performances?" item
- updated "Why doesn't my old Compaq work?" item
- fixed an outdated pointer
- added a pointer to Ingo test SMP patches

v0.54, 13 march 1999

- Added a section about SMP Alpha systems

v0.53, 08 march 1999

- Added a section about SMP PowerPC systems

v0.52, 07 march 1999

- Added a section about SMP Sparc systems

v0.51, 06 march 1999

- Added a dual−celeron section
- Deleted Adaptec section
- Updated procps link
- Updated xosview link
- Added an answer for quad Xeon boot hang
- Updated item about glibc patch for gd: should be included in RH 5.2

v0.50, 03 february 1999

- Updated "Multithreaded programs on linux" link

v0.49, 13 january 1999

- Update about CONFIG_SMP. Added .txt to Documentation/smp. (**Michael Elizabeth Chastain**)

v0.48, 10 december 1998

- Mispelled corrected. Email address corrected.

v0.47, 20 november 1998

- Added that 2.0.36 as the MTRR patch (related to the BogoMips problem)

v0.46, 10 november 1998

- Update about Epox KP6−LS motherboards

v0.45, 25 october 1998

- Corrected an error regarding /proc/stat file
- Added a pointer to CESDIS Ethernet Linux Drivers site

v0.44, 14 october 1998

- Updated the link to the web page: *Motherboards rumored to run Linux SMP*
- Added Jakob explanation how to time SMP systems with 2.0 kernels

v0.43, 9 september 1998

- Updated first question in section 3.1
- Updated mt−Mesa link: multi−threaded is now included as experimental in the Mesa distribution

v0.42, 2 september 1998

- Minor cosmetic update in sect 3.3
- Two links (multithreaded Mesa and SMP performance) marked outdated
- Updated the item about threads and exceptions in C++ (sect 3.3)

v0.41, 1 september 1998

- Added a major section: "3.3 SMP Programming" written by Jakob Østergaard
- moved some item of section "3.2 User side" in sect 3.3

v0.40, 27 august 1998

- Updated section 3.1, item 7: processor affinity

v0.39, 27 august 1998

- Updated needed Award BIOS version for Tyan motherboards (**hASCII**)
- Added an item on IRQ in the crash section (me and **hASCII**)
- Added good support of Asus P2B−DS (**Ulf Rompe**)
- Added another smp−list archive in pointer section (**Hank Leininger**)

v0.38, 8 august 1998

- Added a pointer to the Linux Threads FAQ

v0.37, 30 July 1998

- **Emil Briggs** is working on parallel plugins for Gimp (see "Is there any threaded programs or library?", sect. "User side")

v0.36, 26 July 1998

- Thanks to **Jakob Østergaard**, two changes in "Possible causes of Crash"
  - Changed 2.0.33 to 2.0.35 (latest stable)
  - Added a "BIOS related causes of failure"

v0.35, 14 July 1998

- Added N440BX Server Board in Motherboards with NO problems
- Added a succes story for GigaByte motherboard with BIOS upgrade
- Added a "How to obtain maximum performance ?" section (waiting for your contributions ;)

v0.34, 10 june 1998

- Added a "Parallelizing/Optimizing Compilers for 586/686 machines" section in section "Useful Pointers", thanks to **Sumit Roy**
- Corrected a mispelling, "Asus P/I−UP5" is in fact "Asus P/I−P65UP5"

v0.33, 3 june 1998

- Yet another success story for a GigaByte DLX Motherboard.
- A tip for Tyan motherboards, disable the "DRAM Fast Leadoff" BIOS option

v0.32, 27 may 1998

- Asus P/I−UP5 added in the motherboard−with−NO−problem section

v0.31, 18 may 1998

- Elitegroup P6LX2−A works with 2.1.100 and 101
- Bugs should be reported to `linux-smp@vger.rutgers.edu`

v0.30, 12 may 1998

- SuperMicro is now in the motherboard−with−NO−problem section

v0.29, 11 may 1998

- A success story for a GigaByte 686 motherboard with 2.1.101
- Added a new item in the "User Side" section: "Is there any threaded programs or library?"
- OpenGL Mesa library is beeing multithreaded. Cool! See the new section for details.

v0.28, 09 may 1998

- A US mirror of this FAQ is now available (see Introduction)
- Merge of the two confusing Gigabyte 686 entries

v0.27, 05 may 1998

- New info for the Adaptec and TekRam drivers
- Micronics W6−LI motherboard works under SMP

---

10. List of contributors

Many thanks to those who help me to maintain this HOWTO:

1. Tigran A. Aivazian
2. John Aldrich
3. Niels Ammerlaan
4. H. Peter Anvin
5. Osamu Aoki
6. Guylhem Aznar
7. Ralf Bächle
8. James Beard
9. Troy Benjegerdes
10. Anton Blanchard
11. Emil Briggs
12. Robert G. Brown
13. Ray Bryant
14. Alexandre Charbey
15. Michael Elizabeth Chastain
16. Samuel S. Chessman
17. Alan Cox
18. Andrew Crane
19. Cort Dougan
20. Patrick Doyle

21. Mark Duguid
22. Stéphane Écolivet
23. Johan Ekenberg
24. Jocelyne Erhel
25. Jay A Estabrook
26. Byron Faber
27. Mark Garlanger
28. hASCII
29. Wade Hampton
30. Andre Hedrick
31. Claus−Justus Heine
32. Benedikt Heinen
33. Florian Hinzmann
34. Moni Hollmann
35. Robert M. Hyatt
36. Jeffrey H. Ingber
37. Richard Jelinek
38. Tony Kocurko
39. Geerten Kuiper
40. Martijn Kruithof
41. Doug Ledford
42. Kumsup Lee
43. Hank Leininger
44. Ryan McCue
45. Paul Mackerras
46. Cameron MacKinnon
47. Joel Marchand
48. David Maslen
49. Chris Mauritz
50. Jean−Francois Micouleau
51. David Miller
52. Ingo Molnar
53. Ulf Nielsen
54. Jakob Oestergaard
55. C Polisher
56. Adrian Portelli
57. Matt Ranney
58. Daniel Roesen
59. Ulf Rompe
60. Jean−Michel Rouet
61. Volker Reichelt
62. Sean Reifschneider
63. Rik van Riel
64. Sumit Roy
65. Thomas Schenk
66. Matthias Schniedermeyer
67. Terry Shull
68. Chris K. Skinner
69. Hans − Erik Skyttberg
70. Szakacsits Szabolcs
71. Jukka Tainio

72. Stig Telfer
73. Simen Timian Thoresen
74. El Warren
75. Gregory R. Warnes
76. Gero Wedemann
77. Christopher Allen Wing
78. Leonard N. Zubkoff