Linux/MIPS HOWTO

Ralf Bächle, ralf@gnu.org v0.1, 31 March 1999

This FAQ describes the MIPS port of the Linux operating system, common problems and their solutions, availability and more. It also tries to be a little helpful to other people who might read this FAQ in an attempt to find information that actually should be covered elsewhere.

Contents

1 What is Linux/MIPS? 3

2 What hardware does Linux/MIPS support? 3

2.1 Hardware platforms . . . 3

2.1.1 Acer PICA . . . 3

2.1.2 Baget/MIPS series . . . 4

2.1.3 Cobalt Qube and Raq . . . 4

2.1.4 Netpower 100 . . . 4

2.1.5 Nintendo 64 . . . 4

2.1.6 Silicon Graphics Indy . . . 4

2.1.7 Silicon Graphics Challenge S . . . 5

2.1.8 Silicon Graphics Indigo . . . 5

2.1.9 Serial console on SGI machines . . . 6

2.1.10 Motorola 68k based machines like the Iris 3000 . . . 6

2.1.11 SGI VisPC . . . 6

2.1.12 Other Silicon Graphics machines . . . 6

2.1.13 Sony Playstation . . . 6

2.1.14 SNI RM200C . . . 7

2.1.15 SNI RM200 . . . 7

2.1.16 SNI RM300C . . . 7

2.1.17 SNI RM400 . . . 7

2.1.18 Algorithmics P4032 . . . 7

2.1.19 Algorithmics P5064 . . . 7

2.1.20 DECstation series . . . 7

2.1.21 Mips Magnum 4000 / Olivetti M700-10 . . . 8

2.1.22 MIPS Magnum 4000SC . . . 8

2.1.23 VaxStation . . . 9

2.2 Processor types . . . 9

2.2.1 R2000, R3000 family . . . 9
2.2.2 R6000 . . . 9
2.2.3 R4000 and R5000 family . . . 9
2.2.4 R8000 . . . 9
2.2.5 R10000 . . . 10

3 Linux distributions. 10
3.1 RedHat . . . 10
3.2 Debian . . . 10

4 Linux/MIPS net resources. 10
4.1 Anonymous FTP servers. . . . 10
4.2 Anonymous CVS servers. . . . 11
4.3 Web servers. . . . 11
4.4 Mailing lists. . . . 11

5 Installation of Linux/MIPS and common problems. 12
5.1 NFS booting fails. . . . 12
5.2 Self compiled kernels crash when booting. . . . 12
5.3 Booting the kernel on the Indy fails with PROM error messages . . . 12
5.4 Where can I get the little endian firmware for my SNI? . . . 13
5.5 ld dies with signal 6 . . . 13

6 Milo 13
6.1 Building Milo . . . 13
6.2 Pandora . . . 13

7 Loadable Modules 14

8 How do I setup a crosscompiler? 14
8.1 Diskspace requirements . . . 14
8.2 Byte order . . . 14
8.3 Configuration names . . . 15
8.4 Installation of GNU Binutils. . . . 15
8.5 Assert.h . . . 15
8.6 First installation of egcs . . . 15
8.7 float.h . . . 16
8.8 Installing the kernel sources . . . 16
8.9 Installing GNU libc . . . 16

8.10 Building egcs again . . . 17
8.11 Should I build the C++, Objective C or F77 compilers? . . . 18
8.12 GDB . . . 18

9 Related Literature **18**
9.1 See MIPS Run . . . 18
9.2 The MIPS Programmer's Handbook . . . 19
9.3 Computer Architecture - A Quantitative Approach . . . 19

10 Linux/MIPS news **19**

1 What is Linux/MIPS?

Linux/MIPS is a port of the widespread UNIX clone Linux to the MIPS architecture. Linux/MIPS is running on a large number of technically very different systems ranging from little embedded systems and servers to large desktop machines and servers that, at least at the time when they were introduced into the market, were the best of their class.

Linux/MIPS advantages over other operating systems at this time are

- The entire Linux system consists only of Free Software.
- Excellent Price/Performance ratio.
- Availability of large amounts of software of which a large part again is Free Software.
- Binary compatibility across a growing number of platforms.
- Small footprint making Linux/MIPS suitable for many embedded systems.

In short, Linux has been designed and ships with Fahrvergnügen. However as usual your mileage may vary and you should examine Linux's suitability for your purpose which purpose this document tries to serve.

2 What hardware does Linux/MIPS support?

2.1 Hardware platforms

Many machines are available with a number of different CPU options of which not all are currently supported. Please check section 2.2 (Processor Types) to make sure your CPU type is supported. This is a listing of machines that are running Linux/MIPS, systems to which Linux/MIPS could be ported or systems that people have an interest in running Linux/MIPS.

2.1.1 Acer PICA

The *Acer PICA* is derived from the *Mips Magnum 4000* design. It has a R4400PC CPU running at 133Mhz or optionally 150Mhz plus a 512kb (optionally 2mb) second level cache; the Magnum's G364 gfx card was replaced with a S3 968 based one. The system is supported with the exception of the X server.

2.1.2 Baget/MIPS series

The Baget series includes several boxes which have R3000 processors: Baget 23, Baget 63, and Baget 83. Baget 23 and 63 have BT23-201 or BT23-202 motherboards with R3500A (which is basically a R3000A chip) at 25 MHz and R3081E at 50 MHz respectively. The BT23-201 board has VME bus and VIC068, VAC068 chips as system controllers. The BT23-202 board has PCI as internal bus and VME as internal. Support for BT23-201 board has been done by *Gleb Raiko (rajko@mech.math.msu.su)* and *Vladimir Roganov (vroganov@msiu.ru)* with a bit help from *Serguei Zimin (zimin@msiu.ru)*. Support for BT23-202 is under development along with Baget 23B which consists of 3 BT23-201 boards with shared VME bus.

Baget 83 is mentioned here for completeness only. It has only 2mb RAM and it's too small to run Linux. The Baget/MIPS code has been merged with the DECstation port; source for both is available at `<http://decstation.unix-ag.org/>`.

2.1.3 Cobalt Qube and Raq

The Cobalt Qube product series are low cost headless server systems based on a IDT R5230. Cobalt has developed its own Linux/MIPS variant to fit the special requirements of the Qube as well as possible. Basically the Qube kernel has been derived from Linux/MIPS 2.1.56, then backported to 2.0.30 for stability's sake, then optimized. Cobalt kernels are available from Cobalt's ftp site `<http://www.cobaltnet.com>`. The Cobalt Qube support has never been integrated into the official Linux/MIPS 2.1.x kernels.

2.1.4 Netpower 100

The *Netpower 100* is apparently an *Acer PICA* in disguise. It should therefore be supported but this is untested. If there is a problem then it is probably the machine detection.

2.1.5 Nintendo 64

The *Nintendo 64* is R4300 based game console with 4mb RAM. Its graphics chips were developed by Silicon Graphics for Nintendo. Right now this port has pipe dream status and will continue to be in that state until Nintendo decides to publish the necessary technical information. The question remains as to whether this is a good idea.

2.1.6 Silicon Graphics Indy

The Indy is currently the only (mostly) supported Silicon Graphics machine. The only supported graphics card is the Newport card aka "XL" graphics. The Indy is available with a large number of CPU options at various clock rates all of which are supported. There is currently no X server available for the Indy; *Alan Cox (alan@lxorguk.ukuu.org.uk)* is working on one.

Strange numbers of available memory On bootup the kernel on the Indy will report available memory with a message like

```
Memory: 27976k/163372k available (1220k kernel code, 2324k data)
```

The large difference between the first pair of numbers is caused by a 128mb area in the Indy's memory address space which mirrors up to the first 128mb of memory. The difference between the two numbers will always be about 128mb and does not indicate a problem of any kind.

Indy PROM related problems Several people have reported these problems with their machines after upgrading them typically from surplus parts. There are several PROM versions for the Indy available. Machines with old PROM versions which have been upgraded to newer CPU variants like a R4600SC or R5000SC module can crash during the self test with an error message like

```
Exception: <vector=Normal>
Status register: 0x30004803<CU1,CU0,IM7,IM4,IPL=???,MODE=KERNEL,EXL,IE>
Cause register: 0x4000<CE=0,IP7,EXC=INT>
Exception PC: 0xbfc0b598
Interrupt exception
CPU Parity Error Interrupt
Local I/O interrupt register 1: 0x80 <VR/GIO2>
CPU parity error register: 0x80b<B0,B1,B3,SYSAD_PAR>
CPU parity error: address: 0x1fc0b598
NESTED EXCEPTION #1 at EPC: 9fc3df00; first exception at PC: bfc0b598
```

In that case you'll have to upgrade your machine's PROMs to a newer version or go back to an older CPU version. Usually R4000SC or R4400SC modules should work in that case. Just to be clear, this is a problem which is unrelated to Linux. It's only mentioned here because several Linux users have asked about it.

ELF support in old PROM versions Old PROM versions don't know about the ELF binary format which the Linux kernel uses, that is can't boot Linux directly. The preferable solution for this is of course a PROM upgrade. Alternatively you can use Sash of IRIX 5 or newer to boot the kernel. Sash knows how to load ELF binaries and doesn't care if it's an IRIX or Linux kernel. Simply type "Sash" to the prom monitor. You should get another shell prompt, this time from Sash. Now launch Linux as usual.

Sash can read EFS or XFS filesystems or read the kernel from bootp / tftp. That means if you intend to use Sash for booting the kernel from local disk you'll still have to have a minimal IRIX installation on your system.

Why is so much memory reserved on my Indy? On bootup the 'Memory: ...' message on an Indy says that there is 128mb of RAM reserved. That is ok; just like the PC architecture has a gap in its memory address space between 640kb and 1024kb, the Indy has a 128mb-sized area in its memory map where the first 128mb of its memory is mirrored. Linux knows about it and just ignores that memory, thus this message.

2.1.7 Silicon Graphics Challenge S

This machine is very similar to the Indy; the difference is that it doesn't have a keyboard and a GFX card but has an additional SCSI WD33C95 based adapter. This WD33C95 hostadapter is currently not supported.

2.1.8 Silicon Graphics Indigo

This machine is only being mentioned here because occasionally people have confused it with Indys. The Indigo series is a different architecture however and therefore yet unsupported. *Andrew R. Baker (andrewb@uab.edu)* announced a university project to port Linux to the Indigo on January 2, 1999.

2.1.9 Serial console on SGI machines

Make sure the kernel you're using includes the appropriate driver for a serial interface and serial console. Set the *console* ARC environment variable to either the value *d1* or *d2* for Indy and Challenge S depending on which serial interface you're going to use as console.

If you have the problem that all kernel messages appear on the serial console on bootup but everything is missing from the point when init starts, then you probably have the wrong setup for your /dev/console. You can find more information about this in the Linux kernel source documentation; it's in /usr/src/linux/Documentation/serial-console.txt if you have the kernel source installed.

2.1.10 Motorola 68k based machines like the Iris 3000

These are *very* old machines, probably more than ten years old by now. As these machines are not based on MIPS processors this document is the wrong place to search for information. However, in order to make things easy, these machines are currently not supported.

2.1.11 SGI VisPC

This is actually an x86 based system, therefore not covered by this FAQ. But to make your search for answers simple, here it is. *Ken Klingman (kck@mailbox.esd.sgi.com)* posted on January 17, 1999 to SGI's Linux mailing list:

```
We are working on it.  We're actually close to getting
the base level system support into the 2.2 release.
Software-only X and OpenGL should follow relatively
shortly, but hardware-accelerated OpenGL is still
some time off.  See www.precisioninsight.com for
news about hardware-accelerated OpenGL.
```

For more information see the Documentation/ of Linux kernel versions from 2.2.0 and newer. There is additional information available on the web on <http://www.linux.sgi.com/intel/>. Note that the SGI/MIPS and SGI/Intel people are working independently of each other, therefore the sources in the anonymous CVS on linus.linux.sgi.com may or may not work for Intel machines; we don't test this.

2.1.12 Other Silicon Graphics machines

At this time no other Silicon Graphics machine is supported. This also applies to the *very* old Motorola 68k based systems.

2.1.13 Sony Playstation

The Sony Playstation is based on an R3000 derivative and uses a set of graphics chips developed by Sony themselves. While the machine in theory would be capable of running Linux, a port is difficult, since Sony so far hasn't provided the necessary technical information. This still leaves the question of whether the port would be worthwhile. So in short, nothing has happend yet even though many people have shown their interest in trying Linux on a Playstation so far.

2.1.14 SNI RM200C

In contrast to the RM200 (see below) this machine has EISA and PCI slots. The RM200 is supported with the exception of the availability of the onboard NCR53c810A SCSI controller.

2.1.15 SNI RM200

If your machine has both EISA and PCI slots, then it is an RM200C; please see above. Due to the slight architectural differences of the RM200 and the RM200C this machine isn't currently supported in the official sources. *Michael Engel (engel@numerik.math.uni-siegen.de)* has managed to get his RM200 working partially but the patches haven't yet been included in the official Linux/MIPS sources.

2.1.16 SNI RM300C

The RM300 is technically very similar to the RM200C. It should be supported by the current Linux kernel, but we haven't yet received any reports.

2.1.17 SNI RM400

The RM400 isn't supported.

2.1.18 Algorithmics P4032

The Algorithmics P4032 port is at the time of this writing still running Linux 2.1.36.

2.1.19 Algorithmics P5064

The P5064 is basically an R5000-based 64bit variant of the P4032. It's not yet supported but a Linux port will be quite easy.

2.1.20 DECstation series

During the late 80's and the early 90's Digital (now Compaq) built MIPS based Workstations named DECstation resp. DECsystem. Other x86 and Alpha based machines were sold under the name DECstation, but these are obviously not subject of this FAQ. Support for DECstations is still under development, started by Paul M. Antoine. These days most of the work is done by *Harald Koerfgen (Harald.Koerfgen@home.ivm.de)* and others. On the Internet, DECstation-related information can be found at `<http://decstation.unix-ag.org/>`.

The DECstation family ranges from the DECstation 2100 with an R2000/R2010 chipset at 12 Mhz to the DECstation 5000/260 with a 60 MHz R4400SC.

The following DECstation models are actively supported:

- 2100, codename PMAX
- 5000/xx (Personal DECstation), codename MAXine
- 5000/1xx, codename 3MIN
- 5000/2x0, codename 3MAX+

- 5900/2x0 (identical to the 3MAX+).

These DECstation models are orphaned because nobody is working on them, but support for these should be relatively easy to achieve.

- 3100, identical to the 2100 except the R2000A/R2010A @ 16 MHz
- 5100, codename MIPSMATE, almost identical to the 2100 but with an R3000/R3010 chipset.
- 5000/200, codename 3MAX

The other members of the DECstation family, besides the x86 based ones, should be considered as VAXen with the CPU replaced by a MIPS CPU. There is absolutely no information available about these machines and support for them is unlikely to happen ever unless the VAXLinux port comes to a new life. These are:

- 5400, codename MIPSFAIR
- 5500, codename MIPSFAIR2
- 5800, codename ISIS

The R2000/R3000 support in the Linux/MIPS kernel is a merge of the DECstation and Baget/MIPS code and isn't yet integrated into the official Linux/MIPS source tree.

2.1.21 Mips Magnum 4000 / Olivetti M700-10

These two machines are almost completely identical. Back during the ACE initiative Olivetti licensed the Jazz design and marketed the machine with Windows NT as OS. MIPS Computer Systems, Inc. itself bought the Jazz design and marketed it as the MIPS Magnum 4000 series of machines. Magnum 4000 systems were marketed with Windows NT and RISC/os as operating systems.

The firmware on the machine depended on the operating system which was installed. Linux/MIPS supports only the little endian firmware on these two types of machines. Since the M700-10 was only marketed as an NT machine all M700-10 machines have this firmware installed. The MIPS Magnum case is somewhat more complex. If your machine has been configured big endian for RISC/os then you need to reload the little endian firmware. This firmware was originally included on a floppy with the delivery of every Magnum. If you don't have the floppy anymore you can download it via anonymous ftp from `<ftp://ftp.fnet.fr>`.

It is possible to reconfigure the M700 for headless operation by setting the firmware environment variables ConsoleIn and ConsoleOut to multi()serial(0)term(). Also try the command *listdev* which will show the available ARC devices.

In some cases, like where the G364 graphics card is missing but the console is still configured to use normal graphics it will be necessary to set the configuration jumper JP2 on the board. After the next reset the machine will reboot with the console on COM2.

2.1.22 MIPS Magnum 4000SC

The Mips Magnum 4000SC is the same as a Magnum 4000 (see above) with the exception that it uses an R4000SC CPU.

2.1.23 VaxStation

As the name already implies this machine is a member of Digital Equipment's VAX family. It's mentioned here because people often confuse it with Digital's MIPS based DECstation family due to the similar type numbers. These two families of architectures share little technical similarities. Unfortunately the VaxStation, like the entire VAX family, is currently unsupported.

2.2 Processor types

2.2.1 R2000, R3000 family

The R2000 is the original MIPS processor. It's a 32 bit processor which was clocked at 8MHz back in '85 when the first MIPS processors came to the market. Later versions were clocked faster: for instance, the R3000 is a 100% compatible redesign of the R2000, just clocked faster. Because of their high compatibility, where this document mentions the R3000, in most cases the same facts also apply to the R2000.

The R3000A is basically an R2000 plus an R3010 FPU and 64k cache running at up to 40Mhz and integrated into the same chip. Support for the R3000 processor is currently in the works by various people. *Harald Koerfgen (Harald.Koerfgen@home.ivm.de)* and *Gleb O. Raiko (raiko@niisi.msk.ru)* have both independently worked on patches which haven't yet been integrated into the official Linux/MIPS sources.

2.2.2 R6000

Sometimes people confuse the R6000, a MIPS processor, with RS6000, a series of workstations made by IBM. So if you're reading this in hope of finding out more about Linux on IBM machines you're reading the wrong document.

The R6000 is currently not supported. It is a 32-bit MIPS ISA 2 processor and a pretty interesting and weird piece of silicon. It was developed and produced by a company named *BIT Technology*. Later NEC took over the semiconductor production. It was built in ECL technology, the same technology that was and still is being used to build extremely fast chips like those used in some Cray computers. The processor had its TLB implemented as part of the last couple of lines of the external primary cache, a technology called *TLB slice*. That means its MMU is substantially different from those of the R3000 or R4000 series, which is also one of the reasons why the processor isn't supported.

2.2.3 R4000 and R5000 family

Linux supports many of the members of the R4000 family. Currently these are R4000PC, R4400PC, R4300, R4600, R4700, R5000, R5230, R5260. Many others are probably working as well.

Not supported are R4000MC and R4400MC CPUs (that is multiprocessor systems) as well as R5000 systems with a CPU controlled second level cache. This means where the cache is controlled by the R5000 itself in contrast to some external external cache controller. The difference is important because, unlike other systems, especially PCs, on MIPS the cache is architecturally visible and needs to be controlled by software.

Special credit goes to *Ulf Carlsson (grim@zigzegv.ml.org)* who provided the CPU module for debugging the R4000SC / R4400SC support.

2.2.4 R8000

The R8000 is currently unsupported partly because this processor is relatively rare and has only been used in a few SGI machines, partly because the Linux/MIPS developers don't have such a machine.

The R8000 is a pretty interesting piece of silicon. Unlike the other members of the MIPS family it is a set of seven chips. Its cache and TLB architecture is pretty different from the other members of the MIPS family. It was born as a hack to get the floating point crown back to Silicon Graphics before the R10000 is finished.

2.2.5 R10000

The R10000 is currently unsupported because the Linux/MIPS developers don't have an R10000 machine.

3 Linux distributions.

3.1 RedHat

For MIPSeb, there's Rough Cuts Linux, previously known as Hard Hat Linux, which is most of Red Hat Linux 5.1 ported for MIPSeb. You can get this at `<ftp://ftp.linux.sgi.com/pub/hardhat>`.

It is also bundled along with M68k, UltraSparc and PowerPC in a package called "Rough Cuts" pressed by Red Hat, and available wherever Red Hat products are sold. This is a very convenient way to get it without having to download 280MB. You can order Rough Cuts directly from Red Hat at `<http://www.redhat.com/product.phtml/RC1000>`.

As well, there's a distribution based on Red Hat 5.2 that's targetting the Cobalt Qubes; those binaries will work perfectly on other MIPSel architectures available at `<ftp://intel.cleveland.lug.net/pub/Mipsel>`.

3.2 Debian

A Debian port is underway. Current efforts are being bootstrapped using SGI/Linux as a base, and dpkg compiles natively with few changes. In addition to the SGI version, some interest has been shown in little endian platforms. Keep an eye on the Debian-MIPS Port page, `<http://www.debian.org/ports/mips/>` for developments.

4 Linux/MIPS net resources.

4.1 Anonymous FTP servers.

The two primary anonymous FTP servers for Linux/MIPS are

ftp.linux.sgi.com

This server should satisfy almost all your Linux/MIPS related ftp desires. Really.

ftp.fnet.fr

This server is currently pretty outdated; it's included here mostly for completeness and for people with interest in prehistoric software.

On all these ftp servers there is a list of mirror sites you may want to use for faster access.

4.2 Anonymous CVS servers.

For those who always want to stay on the bleeding edge and want to avoid having to download patch files or full tarballs we also have an anonymous CVS server. Using CVS you can checkout the Linux/MIPS source tree with the following commands:

```
cvs -d :pserver:cvs@linus.linux.sgi.com:/cvs login
(Only needed the first time you use anonymous CVS, the password is "cvs")
cvs -d :pserver:cvs@linus.linux.sgi.com:/cvs co <repository>
```

where you insert linux, libc, or gdb for <repository>.

The other important CVS archive of the Linux community is vger.rutgers.edu where a lot of code is being collected before being sent to Linus for distribution. Although vger itself no longer offers anonymous access, there are mirror sites which do provide anonymous access. For details how to access them see `<http://cvs.on.openprojects.net/>`. The modules which are of interest are "linux", "modutils", "pciutils", "netutils".

4.3 Web servers.

The two primary anonymous web servers for Linux/MIPS are

www.linux.sgi.com

This server covers most of Linux/MIPS; it's somewhat SGI centric but since Linux/MIPS tries to be the same on every platform most of its information is of interest to all users.

lena.fnet.fr

This server is currently pretty outdated; it's included here mostly for completeness.

All these servers have mirrors scattered all over the world; you may want to use one for best performance.

4.4 Mailing lists.

There are three Linux/MIPS oriented mailing lists:

linux-mips@fnet.fr

This mailing list is used for most non-SGI related communication of all kinds. Subscription is handled by a human; send your subscription requests to *linux-mips-mips@fnet.fr*. You can unsubscribe from this mailing list by sending *unsubscribe <your-email-address>* to the same address.

linux@engr.sgi.com

This mailing list currently has the most traffic. It's somewhat SGI-centric but is nevertheless of interest especially to developers as a good number of SGI engineers are subscribed to this list. Subscription to this list is handled via *Majordomo (majordomo@engr.sgi.com)*; just send an email with the words *subscribe linux-mips*. In order to unsubscribe send *unsubscribe linux-mips*. Note that you have to be subscribed if you want to post; the growth of spam forced us into that policy.

linux-mips@vger.rutgers.edu

This mailing list has only very low traffic as most people tend to use one of the above mailing lists. Subscription is handled via *Majordomo (majordomo@vger.rutgers.edu)*; just send an email with the words *subscribe linux-mips*. In order to unsubscribe send *unsubscribe linux-mips*.

5 Installation of Linux/MIPS and common problems.

5.1 NFS booting fails.

Usually the reason for this is that people have unpacked the tar archive under IRIX, not Linux. Since the representation of device files over NFS is not standardized between various Unices, this fails. The symptom is that the system dies with the error message "Warning: unable to open an initial console." right after mounting the NFS filesystem.

For now the workaround is to use a Linux system (doesn't need to be MIPS) to unpack the installation archive onto the NFS server. The NFS server itself may be any type of UNIX.

5.2 Self compiled kernels crash when booting.

When I build my own kernel, it crashes. On an Indy the crash message looks like the following; the same problem hits other machines as well but may look completely different.

```
Exception: <vector=UTLB Miss>
Status register: 0x300004803<CU1,CU0,IM4,IPL=???,MODE=KERNEL,EXL,IE>
Cause register: 0x8008<CE=0,IP8,EXC=RMISS>
Exception PC: 0x881385cc, Exception RA: 0x88002614
exception, bad address: 0x47c4
Local I/O interrupt register 1: 0x80 <VR/GIO2>
Saved user regs in hex (&gpda 0xa8740e48, &_regs 0xa8741048):
  arg: 7 8bfff938 8bfffc4d 880025dc
  tmp: 8818c14c 8818c14c 10 881510c4 14 8bfad9e0 0 48
  sve: 8bfdf3e8 8bfffc40 8bfb2720 8bfff938 a8747420 9fc56394 0 9fc56394
  t8 48 t9 8bfffee66 at 1 v0 0 v1 8bfff890 k1 bad11bad
  gp 881dfd90 fp 9fc4be88 sp 8bfff8b8 ra 88002614

PANIC: Unexpected exception
```

This problem is caused by a still unfixed bug in Binutils newer than version 2.7. As a workaround, change the following line in arch/mips/Makefile from:

```
LINKFLAGS       = -static -N
```

to:

```
LINKFLAGS       = -static
```

5.3 Booting the kernel on the Indy fails with PROM error messages

```
>> boot bootp()/vmlinux
73264+592+11520+331680+27848d+3628+5792 entry: 0x8df9a960
Setting $netaddres to 192.168.1.5 (from server deadmoon)
Obtaining /vmlinux from server deadmoon
```

```
Cannot load bootp()/vmlinux
Illegal f_magic number 0x7f45, expected MIPSELMAGIC or MIPSEBMAGIC.
```

This problem only happens for Indys with very old PROM versions which cannot handle the ELF binary format which Linux uses. A solution for this problem is in the works.

5.4 Where can I get the little endian firmware for my SNI?

SNI's system can be operated in both big and little endian modes. At this time Linux/MIPS only supports the little endian firmware. This is somewhat unlucky since SNI hasn't shipped that firmware for quite some time, since they dropped NT.

When running in big endian mode the firmware looks similar to an SGI Indy which is already supported, therefore fixing the SNI support will be relativly easy. Interested hackers should contact *Ralf Bächle (ralf@gnu.org)*.

5.5 ld dies with signal 6

```
collect2: ld terminated with signal 6 [Aborted]
```

This is a known bug in older binutils versions. You will have to upgrade to binutils 2.8.1 plus very current patches.

6 Milo

Milo is the boot loader used to boot the little endian MIPS systems with ARC firmware, currently the Jazz family and the SNI RM 200. While Milo uses the same name and has a similar purpose to the Alpha version of Milo, these two Milos have nothing else in common. They were developed by different people, don't share any code, and work on different hardware platforms. The fact that both have the same name is just a kind of historic "accident".

Plans are to remove the need for Milo in the near future.

6.1 Building Milo

The building procedure of Milo is described in detail in the README files in the Milo package. Since Milo has some dependencies to kernel header files which have changed over time Milo often cannot be built easily; however the Milo distribution includes binaries for both Milo and Pandora.

6.2 Pandora

Pandora is a simple debugger. It has been primarily developed in order to analyze undocumented systems. Pandora includes a dissassembler, memory dump functions and more. If you only want to use Linux there is no need to install Pandora. It's small though.

7 Loadable Modules

Using modules on Linux/MIPS is quite easy; it should work as expected for people who have used it on other Linux systems. If you want to run a module-based system then you should have at least kernel version 980919 and modutils newer than version 2.1.121 installed. Older versions won't work.

8 How do I setup a crosscompiler?

First of all go and download the following source packages:

- binutils-2.8.1.tar.gz
- egcs-1.0.2.tar.gz
- glibc-2.0.6.tar.gz
- glibc-crypt-2.0.6.tar.gz
- glibc-localedata-2.0.6.tar.gz
- glibc-linuxthreads-2.0.6.tar.gz

These are the currently recommended versions. Older versions may or may not be working. If you're trying to use older versions please don't send bug reports; we don't care. When installing please install things in the order binutils, egcs, then glibc. Unless you have older versions already installed, changing the order *will* fail. The installation description below mentions a number of patches which you can get from the respective SRPM packages on *ftp.linux.sgi.com*. However since these SRPM packages are intended to be compiled natively it's not possible to just rebuild them.

8.1 Diskspace requirements

For the installation you'll have to choose a directory for installation. I'll refer to that directory below with <prefix>. To avoid a certain problem it's best to use the same value for <prefix> as your native gcc. For example if your gcc is installed in /usr/bin/gcc then choose /usr for <prefix>. You must use the same <prefix> value for all the packages that you're going to install.

During compilation you'll need about 31mb diskspace for binutils; for installation you'll need 7mb diskspace for on <prefix>'s partition. Building egcs requires 71mb and installation 14mb. GNU libc requires 149mb diskspace during compilation and 33mb for installation. Note these numbers are just a guideline and may differ significantly for different processor and operating system architectures.

8.2 Byte order

One of the special features of the MIPS architecture is that all processors except the R8000 can be configured to run either in big or in little endian mode. Byte order means the way the processor stores multibyte numbers in memory. Big endian machines store the byte with the highest value digits at the lowest address while little endian machines store it at the highest address. Think of it as writing multi-digit numbers from left to right or vice versa.

In order to setup your crosscompiler correctly you have to know the byte order of the crosscompiler target. If you don't already know, check the section 2.1 (Hardware Platforms) for your machine's byteorder.

8.3 Configuration names

Many of the packages based on autoconf support many different architectures and operating systems. In order to differentiate between these many configurations, names are constructed with <cpu>-<company>-<os> or even <cpu>-<company>-<kernel>-<os>. Expressed this way the configuration names of Linux/MIPS are mips-unknown-linux-gnu for big endian targets or mipsel-unknown-linux-gnu for little endian targets. These names are a bit long and are allowed to be abbreviated to mips-linux or mipsel-linux. You *must* use the same configuration name for all packages that comprise your crosscompilation environment. Also, while other names like mips-sni-linux or mipsel-sni-linux are legal configuration names, use mips-linux or mipsel-linux instead; these are the configuration names known to other packages like the Linux kernel sources and they'd otherwise have to be changed for crosscompilation.

I'll refer to the target configuration name below with <target>.

8.4 Installation of GNU Binutils.

This is the first and simplest part - at least as long as you're trying to install on any halfway-sane UNIX flavour. Just cd into a directory with enough free space and do the following:

```
gzip -cd binutils-<version>.tar.gz | tar xf -
cd binutils-<version>
patch -p1 < ../binutils-<version>-mips.patch
./configure --prefix=<prefix> --target=<target>
make CFLAGS=-O2
make install
```

This usually works very easily. On certain machines using GCC 2.7.x as compiler is known to dump core. This is a known bug in GCC and can be fixed by upgrading to GCC 2.8.1 or egcs.

8.5 Assert.h

Some people have an old assert.h headerfile installed, probably a leftover from an old crosscompiler installation. This file may cause autoconf scripts to fail silently; it was never necessary and was only installed because of a bug in older GCC versions. Check to see if the file <prefix>/<target>/include/assert.h exists in your installation. If so, just delete the it: it should never have been installed.

8.6 First installation of egcs

Now the not-so-funny part begins: there is a so-called bootstrap problem. In our case that means the installation process of egcs needs an already- installed glibc, but we cannot compile glibc because we don't have a working crosscompiler yet. Luckily you'll only have to go through this once when you install a crosscompiler for the first time. Later when you already have glibc installed things will be much smoother. So now do:

```
gzip -cd egcs-<version>.tar.gz | tar xf -
cd egcs-<version>
for i in egcs-1.0.2-libio.patch egcs-1.0.2-hjl.patch \
      egcs-1.0.2-rth1.patch egcs-1.0.2-rth2.patch egcs-1.0.2-rth3.patch \
      egcs-1.0.2-rth4.patch egcs-1.0.2-hjl2.patch egcs-1.0.2-jim.patch \
```

```
      egcs-1.0.2-haifa.patch egcs-1.0.1-objcbackend.patch \
      egcs-1.0.2-mips.patch; do patch -p1 -d < ../$i; done
./configure --prefix=<prefix> --with-newlib --target=<target>
cd gcc
make LANGUAGES="c"
```

Note that we deliberately don't build gcov, protoize, unprotoize and the libraries. Gcov doesn't make sense in a crosscompiler environment and protoize and unprotoize might even overwrite your native programs - this is a bug in the gcc makefiles. Finally we cannot build the libraries because we don't have glibc installed yet. If everything went successfully, install with:

```
make LANGUAGES="c" install
```

8.7 float.h

Another bootstrap problem is that building GCC requires running programs on the machine for which GCC will generate code, but since a crosscompiler is running on a different type of machine this cannot work. When building GCC this happens for the header file float.h. Luckily there is a simple solution: download the header file from one of the Linux/MIPS ftp servers or rip it from one of the native Linux/MIPS binary packages. Later when recompiling or upgrading egcs usually the already-installed float.h file will do because float.h changes rarely. Install it with:

```
cp float.h <prefix>/<target/<version>/include/float.h
```

where <version> is the internal version number of the egcs version you're using. For egcs 1.0.2 for example you would use egcs-2.90.27 for <version>. If not sure - ls is your friend.

8.8 Installing the kernel sources

Installing the kernel sources is simple. Just place them into some directory of your choice and configure them such that some files which are generated by the procedure will be installed. This works the same as you're used to when configuring the kernel sources for native compilation. The only problem you may run into is that you may need to install some required GNU programs like bash or have to override the manufacturer-provided versions of programs by placing the GNU versions earlier in the PATH variable. When configuring you should answer the question "Are you using a crosscompiler", that is the option CONFIG_CROSSCOMPILE, with "yes". When you're done with configuring type make clean; make depend; make. The last make command will generate the header file <linux/version.h> which compiling some programs depends on. This file is generated right at the beginning of the make command, so if you're not interested in actually building a kernel you may interrupt the compilation after this file has been built. It may be a good idea, however, to compile the kernel as a test for your newly-built crosscompiler.

If you only want the crosscompiler for building the kernel, you're done. Crosscompiling libc is only required to be able to compile user applications.

8.9 Installing GNU libc

Do:

```
gzip -cd glibc-2.0.6.tar.gz | tar xf -
cd glibc-2.0.6
gzip -cd glibc-crypt-2.0.6.tar.gz | tar xf -
gzip -cd glibc-localedata-2.0.6.tar.gz | tar xf -
gzip -cd glibc-linuxthreads-2.0.6.tar.gz | tar xf -
patch -p1 < ../glibc-2.0.6-mips.patch
mkdir build
cd build
CC=<target>-gcc BUILD_CC=gcc AR=<target>-ar RANLIB=<target>-ranlib \
      ../configure --prefix=/usr --host=<target> \
      --enable-add-ons=crypt,linuxthreads,localedata --enable-profile
make
```

You now have a compiled GNU libc which still needs to be installed. Do *not* just type make install. That would overwrite your host system's files with Linux/MIPS-specific files with disastrous effects. Instead install GNU libc into some other arbitrary directory <somedir> from which we'll move the parts we need for crosscompilation into the actual target directory:

```
make install_root=<somedir> install
```

Now cd into <somedir> and finally install GNU libc manually:

```
cd usr/include
find . -print | cpio -pumd <prefix>/<target>/include
cd ../../lib
find . -print | cpio -pumd <prefix>/<target>/lib
cd ../usr/lib
find . -print | cpio -pumd <prefix>/<target>/lib
```

GNU libc also contains extensive online documentation. Your systems might already have a version of this documentation installed, so if you don't want to install the info pages, which will save you a less than a megabyte, or already have them installed, skip the next step:

```
cd ../info
gzip -9 *.info*
find . -name \*.info\* -print | cpio -pumd <prefix>/info
```

If you're not bootstrapping your installation is now finished.

8.10 Building egcs again

The first attempt of building egcs was stopped by lack of a GNU libc. Since we now have libc installed we can rebuild egcs but this time as complete as a crosscompiler installation can be:

```
gzip -cd egcs-<version>.tar.gz | tar xf -
cd egcs-<version>
for i in egcs-1.0.2-libio.patch egcs-1.0.2-hjl.patch \
```

```
        egcs-1.0.2-rth1.patch egcs-1.0.2-rth2.patch egcs-1.0.2-rth3.patch \
        egcs-1.0.2-rth4.patch egcs-1.0.2-hjl2.patch egcs-1.0.2-jim.patch \
        egcs-1.0.2-haifa.patch egcs-1.0.1-objcbackend.patch \
        egcs-1.0.2-mips.patch; do patch -p1 < ../$i; done
./configure --prefix=<prefix> --target=<target>
make LANGUAGES="c c++ objective-c f77"
```

As you can see the procedure is the same as the first time with the exception that we dropped the –with-newlib option. This option was necessary to avoid the libgcc build breaking due to the lack of libc. Now install with:

```
make LANGUAGES="c c++ objective-c f77" install
```

You're almost finished. All you have left to do now is to reinstall float.h, which has been overwritten by the last make install command. You'll have to do this every time you reinstall egcs as a crosscompiler. If you think you don't need the Objective C or F77 compilers you can omit them from above commands; each will save you about 3mb. Do not build gcov, protoize or unprotoize.

8.11 Should I build the C++, Objective C or F77 compilers?

The answer to this question largely depends on your use of your crosscompiler environment. If you only intend to rebuild the Linux kernel then you have no need for the full blown setup and can safely omit the Objective C and F77 compilers. You must, however, build the C++ compiler, because building the libraries included with the egcs distribution requires C++.

8.12 GDB

Building GDB as crossdebugger is only of interest to kernel developers; for them GDB may be a life saver. Such a remote debugging setup always consists of two parts: the remote debugger GDB running on one machine and the target machine running the Linux/MIPS kernel being debugged. The machines are typically interconnected with a serial line. The target machine's kernel needs to be equipped with a "debugging stub" which communicates with the GDB host machine using the remote serial protocol.

Depending on the target's architecture you may have to implement the debugging stub yourself. In general you'll only have to write very simple routines for serial. The task is further simplified by the fact that most machines are using similar serial hardware typically based on the 8250, 16450 or derivatives.

9 Related Literature

9.1 See MIPS Run

author Dominic Sweetman, published Morgan Kaufmann, ISBN 1-55860-410-3.

This is intended as a pretty comprehensive guide to programming MIPS, wherever it's different from programming any other 32-bit CPU. It's the first time anyone tried to write a readable and comprehensive explanation and account of the wide range of MIPS CPUs available, and should be very helpful for anyone programming MIPS who isn't insulated by someone else's operating system. And the author is a free-unix enthusiast who subscribes to the Linux/MIPS mailing list!

John Hennessey, father of the MIPS architecture, was kind enough to write in the foreword: " ... this book is the best combination of completeness and readability of any book on the MIPS architecture ..."

It includes some context about RISC CPUs, a description of the architecture and instruction set including the "co-processor 0" instructions used for CPU control; sections on caches, exceptions, memory management and floating point. There's a detailed assembly language guide, some stuff about porting, and some fairly heavy-duty software examples.

Available from:

- <http://www.algor.co.uk/algor/info/seemipsrun.html> (europe)
- <http://www.mkp.com/books_catalog/1-55860-410-3.asp> (US)

and from good bookshops anywhere. It's 512 pages and costs around $50 in the US, £39.95 in the UK.

I'd be inclined to list two other books too, both from Morgan Kaufmann and available from www.mkp.com or any good bookshop:

9.2 The MIPS Programmer's Handbook

authors Farquhar and Bunce, published by Morgan Kaufmann, ISBN 1-55860-297-6.

A readable introduction to the practice of programming MIPS at the low level, by the author of PMON. Strengths: lots of examples; weakness: leaves out some big pieces of the architecture (such as memory management, floating point and advanced caches) because they didn't feature in the LSI "embedded" products this book was meant to partner.

9.3 Computer Architecture - A Quantitative Approach

authors Hennessy & Patterson, published Morgan Kaufmann, ISBN 1-58860-329-8.

The bible of modern computer architecture and a must-read if you want to understand what makes programs run slow or fast. Is it about MIPS? Well, it's mostly about something very *like* MIPS... Its sole defect is its size and weight - but unlike most big books it's worth every page.

10 Linux/MIPS news

Some of this chapter is pretty historic ...

04-Dec-98

Ariel Faigon announces that SGI has joined Linux International.

13-Oct-98

Ralf Bächle fixes the support for R4000SC / R4400SC CPUs.

12-Oct-98

Vladimir Roganov reports that his R3000 system is now stable enough to compile GDB.

03-Oct-98

Harald Körfgen reports that his DECstation 5000/133 is now running single user. Congratulations!

29-Sep-98

Ralf starts rewriting this FAQ to fit with reality.

10-Jun-98

ftp.linux.sgi.com now offers anonymous CVS access.

01-Feb-98

First commercial Linux/MIPS based product accounced.

26-Jan-98

One more timewarp in this list because the maintainer is lazy^H^H^H busy coding. The driver for the NCR53c8xx has been modified and has been successfully tested with several machines, most notably the SNI RM200. Even better, the initial version seems to be reliable.

Already some time ago Thomas Bogendörfer implemented the necessary changes to the NCR53C9x driver aka ESP driver, so there is now SCSI support for the builtin hostadapters in the Mips Magnum 4000, Olivetti M700-10 and Acer PICA.

28-Nov-97

First public release of X11 client binaries.

30-Aug-97

Duh, time warp in this page once again. A lot has happend in the meantime and the maintainer of this pages is a lazy person that rather prefers to code and hack than write docs...

SGI now has its own Linux/MIPS server reachable as http://www.linux.sgi.com, with lots of SGI specific information and many links. The server is also reachable under ftp.linux.sgi.com. In addition to binaries, sources and docs specific to Silicon Graphic machines this server also has all the other Linux/MIPS stuff in stock. Only available on this server is the developers' cvs archive for download. Sorry, no anonymous CVS yet.

Silicon Graphics has supported some of the Linux key developers' work on Linux/MIPS with hardware. As a result the work is now advancing more quickly and Ralf is no longer the lone workhorse ...

Already available for some time the Indy port is now in the standard kernel source tree.

Long missing, but finally there: Thomas Bogendoerfer contributed patches to the NCR53C9x driver for Mips Magnum 4000, Olivetti M700 and Acer PICA.

Many more packages of a RedHat port to MIPS are now available for ftp download. Installing is still more a thing for experts ... but we're working on it!

Eeecmacs lovers will be pleased to hear that this FAQ has been edited by Emacs running on a Linux/MIPS machine.

6-May-97

David Monro releases version 1.01 of bfsd. bfsd is a daemon that can be used to boot the machines built by Mips Computersystems, Inc. over a network.

10-Jun-96

Release of Linux/MIPS kernel 2.0.4. This release features a partially rewritten signal handler that should match POSIX.1.

3-Jun-96

First release of shared libraries for Linux/MIPS based on GNU libc snapshot 960619.

Release of Linux/MIPS kernel 2.0.1.

25-May-96

David S. Miller starts working on SGI support at Silicon Graphics.

20-May-96

Release 1.3.98 of the kernel adds support for the SNI RM200 PCI.

27-Mar-96

Linux/MIPS works as NFS server.

The IDE CD driver now also supports Linux/MIPS.

24-Mar-96

Added reference to literature available online form SGI to the FAQ.

23-Mar-96

New chapter in the FAQ about the ARC standard.

27-Jan-96

Release of Milo 0.26 and a kernel patch to use it. This release passes parameters to the kernel in a completly different way that makes porting Linux/MIPS to another architecture a lot easier.

24-Jan-96

Release of crosscompiler binaries based on the FSF's Binutils version 2.6. This release brings lots of new features and many bugfixes.

21-Jan-96

Warner Losh started working on a port of Linux/MIPS to Deskstation rPC44.

20-Jan-96

Linux/MIPS kernel updated to version 1.3.58.

Patch gcc-2.7.2-1.diffs.gz has been released.

Patch binutils-2.6-1.diffs.gz has been released. This patch contains lots of bugfixes. The Linux kernel Makefiles will automatically detect whether Binutils 2.6 or an older version is installed and use the new features resulting in a much smaller kernel executable which is especially useful for bootdisks.

15-Jan-96

Release of a complete root and /usr filesystem that can be NFS mounted to use a Linux/MIPS system as a diskless client. A native development kit based on GCC 2.7.2, Binutils 2.6 and GNU libc snapshot 951218 is included as well as many of the standard utilities.

25-Dec-95

Linux/MIPS boots off an NFS filesystem as a diskless client. This also means that the rest of Linux/MIPS networking is operational now.

7-Jan-95

Soft-N-Hard GMBH and SNI sign a contract. SNI will loan an RM200 to Soft-N-Hard for porting Linux/MIPS to it.

22-Sep-95

The Linux/MIPS FTP archive and mailing list have been moved to fnet.fr. (There is much more news I currently have no time to document)

18-Jul-95

New crossdevelopment tools released. GCC-2.6.3-2 and Binutils-2.5.2-2 for Linux/i386 need kernels with ELF support and libc-5.0.9 installed. The new crossdev tools are required for Linux/MIPS kernels above 1.2.9. A.out versions of the crossdev tools will follow soon.

14-Jul-95

We have a working shell!

12-Jul-95

Patches 2.6.3-2 for Linux/MIPS GCC released. This compiler better complies with the MIPS standard of symbol names.

10-Jul-95

Linux/MIPS kernel 1.2.9 released.

9-Jun-95

Milo 0.24 released. This version features improved machine type detection and many cleanups and bugfixes.

24-May-95

Linux/MIPS kernel 1.2.8 released. This version features many bugfixes and has the Magnum 4000 specific changes from Linux-1.2.7 integrated.

Milo 0.23 released. This version features built-in support for Olivetti M700 machines. Milo is now split into two binaries: A simple bootloader and a standalone debugger/monitor with boot capability.

23-5-95

Linux/MIPS kernel 1.2.7 on Olivetti M700 mounts root file system.

22-May-95

Linux/MIPS kernel 1.2.7 on Mips Magnum 4000 mounts root file system.

Added NEC RiscStation and RiscServer to target list.

Milo 0.22 successfully tested on NEC RiscStation and RiscServer.

18-May-95

Linux/MIPS kernel 1.2.7 released. This release features initial Magnum 4000 support and tons of bugfixes.

12-May-95

Milo 0.22 released. This version contains some cleanups and several bugfixes.

5-May-95

The Linux/MIPS archive is now also available from ftp://ftp.mcc.ac.uk/pub/linux/MIPS.

3-May-95

Milo 0.21 released. This version features more built-in debugger/monitor commands and contains some important bug fixes.

30-Apr-95

Milo 0.20 released. This version features a built-in debugger/monitor and a lot of new library functions.

Port to Olivetti M700 started.

26-Apr-95

Linux/MIPS kernel 1.2.6 released.

13-Apr-95

Milo 0.19 released. This version includes some minor fixes plus initial support for kernels in ELF format.

13-Apr-95

Milo 0.18b released. This version includes support for Mips Magnum 4000. Port to Mips Magnum 4000 started.

27-Mar-95

Linux/MIPS kernel 1.2.2 released. Kernel now mounts its root file system.

22-Mar-95

Milo 0.18 released. This version includes support for Deskstation rPC44 systems.

Port to DeskStation rPC44 started.