ARM Linux 2.6 and upper

Please check <ftp://ftp.arm.linux.org.uk/pub/armlinux> for updates.

Compilation of kernel

In order to compile ARM Linux, you will need a compiler capable of generating ARM ELF code with GNU extensions. GCC 3.3 is known to be a good compiler. Fortunately, you needn't guess. The kernel will report an error if your compiler is a recognized offender.

To build ARM Linux natively, you shouldn't have to alter the ARCH = line in the top level Makefile. However, if you don't have the ARM Linux ELF tools installed as default, then you should change the CROSS_COMPILE line as detailed below.

If you wish to cross-compile, then alter the following lines in the top level make file:

```
ARCH = <whatever>
```

with:

```
ARCH = arm
```

and:

```
CROSS_COMPILE=
```

to:

```
CROSS_COMPILE=<your-path-to-your-compiler-without-gcc>
```

eg.:

```
CROSS_COMPILE=arm-linux-
```

Do a 'make config', followed by 'make Image' to build the kernel (arch/arm/boot/Image). A compressed image can be built by doing a 'make zImage' instead of 'make Image'.

Bug reports etc

Please send patches to the patch system. For more information, see http://www.arm.linux.org.uk/developer/patches/info.php Always include some explanation as to what the patch does and why it is needed.

Bug reports should be sent to linux-arm-kernel@lists.arm.linux.org.uk, or submitted through the web form at http://www.arm.linux.org.uk/developer/

When sending bug reports, please ensure that they contain all relevant information, eg. the kernel messages that were printed before/during the problem, what you were doing, etc.

Include files

Several new include directories have been created under include/asm-arm, which are there to reduce the clutter in the top-level directory. These directories, and their purpose is listed below:

*arch-**	machine/platform specific header files
hardware	driver-internal ARM specific data structures/definitions
mach	descriptions of generic ARM to specific machine interfaces
*proc-**	processor dependent header files (currently only two categories)

Machine/Platform support

The ARM tree contains support for a lot of different machine types. To continue supporting these differences, it has become necessary to split machine-specific parts by directory. For this, the machine category is used to select which directories and files get included (we will use $(MACHINE) to refer to the category)

To this end, we now have arch/arm/mach-$(MACHINE) directories which are designed to house the non-driver files for a particular machine (eg, PCI, memory management, architecture definitions etc). For all future machines, there should be a corresponding arch/arm/mach-$(MACHINE)/include/mach directory.

Modules

Although modularisation is supported (and required for the FP emulator), each module on an ARM2/ARM250/ARM3 machine when is loaded will take memory up to the next 32k boundary due to the size of the pages. Therefore, is modularisation on these machines really worth it?

However, ARM6 and up machines allow modules to take multiples of 4k, and as such Acorn RiscPCs and other architectures using these processors can make good use of modularisation.

ADFS Image files

You can access image files on your ADFS partitions by mounting the ADFS partition, and then using the loopback device driver. You must have losetup installed.

Please note that the PCEmulator DOS partitions have a partition table at the start, and as such, you will have to give '-o offset' to losetup.

Request to developers

When writing device drivers which include a separate assembler file, please include it in with the C file, and not the arch/arm/lib directory. This allows the driver to be compiled as a loadable module without requiring half the code to be compiled into the kernel image.

In general, try to avoid using assembler unless it is really necessary. It makes drivers far less easy to port to other hardware.

ST506 hard drives

The ST506 hard drive controllers seem to be working fine (if a little slowly). At the moment they will only work off the controllers on an A4x0's motherboard, but for it to work off a Podule just requires someone with a podule to add the addresses for the IRQ mask and the HDC base to the source.

As of 31/3/96 it works with two drives (you should get the ADFS *configure* harddrive set to 2). I've got an internal 20MB and a great big external 5.25" FH 64MB drive (who could ever want more :-)).

I've just got 240K/s off it (a dd with bs=128k); thats about half of what RiscOS gets; but it's a heck of a lot better than the 50K/s I was getting last week :-)

Known bug: Drive data errors can cause a hang; including cases where the controller has fixed the error using ECC. (Possibly ONLY in that case...hmm).

1772 Floppy

This also seems to work OK, but hasn't been stressed much lately. It hasn't got any code for disc change detection in there at the moment which could be a bit of a problem! Suggestions on the correct way to do this are welcome.

*CONFIG_MACH_* and *CONFIG_ARCH_*

A change was made in 2003 to the macro names for new machines. Historically, *CONFIG_ARCH_* was used for the bonafide architecture, e.g. SA1100, as well as implementations of the architecture, e.g. Assabet. It was decided to change the implementation macros to read *CONFIG_MACH_* for clarity. Moreover, a retroactive fixup has not been made because it would complicate patching.

Previous registrations may be found online.

> <http://www.arm.linux.org.uk/developer/machines/>

Kernel entry (head.S)

The initial entry into the kernel is via head.S, which uses machine independent code. The machine is selected by the value of 'r1' on entry, which must be kept unique.

Due to the large number of machines which the ARM port of Linux provides for, we have a method to manage this which ensures that we don't end up duplicating large amounts of code.

We group machine (or platform) support code into machine classes. A class typically based around one or more system on a chip devices, and acts as a natural container around the actual implementations. These classes are given directories - arch/arm/mach-<class> - which contain the source files and include/mach/ to support the machine class.

For example, the SA1100 class is based upon the SA1100 and SA1110 SoC devices, and contains the code to support the way the on-board and off- board devices are used, or the device is setup, and provides that machine specific "personality."

For platforms that support device tree (DT), the machine selection is controlled at runtime by passing the device tree blob to the kernel. At compile-time, support for the machine type must be selected. This allows for a single multiplatform kernel build to be used for several machine types.

For platforms that do not use device tree, this machine selection is controlled by the machine type ID, which acts both as a run-time and a compile-time code selection method. You can register a new machine via the web site at:

> <http://www.arm.linux.org.uk/developer/machines/>

Note: Please do not register a machine type for DT-only platforms. If your platform is DT-only, you do not need a registered machine type.

Russell King (15/03/2004)