System

The Linux kernel runs on a wide range of SuperH systems, this does not attempt to be a complete list, but merely a list of the platforms most actively worked on and supported.Distributions

Sega

- Dreamcast - limited to a model of machines that can start by MIL-CD) and suggest with Broad Band Adapter (The contents are LAN adapter).

Hitachi ULSI Systems

- MS7206SE01 (SH72060 Solution Engine)
- MS7750SE01 (SH7750(sh4) Solution Engine)
- MS7709SE01 (SH7709(sh3) Solution Engine)

SuperH, Inc.

- MicroDev

HP Jornada

- 525 (SH7709 (sh3))
- 548 (SH7709A (sh3))
- 620LX (SH7709 (sh3))
- 660LX (SH7709 (sh3)
- 680 (SH7709A (sh3))
- 690 (SH7709A (sh3))

Renesas Technology Corp.

- RTS7751R2D CE Linux Forum (CELF) Compliant Evaluation Board

Renesas Europe/MPC Data Limited

- EDOSK7705 (SH7705 sh3))
- EDOSK7760 (SH7760 sh4)
- EDOSK7751R (SH7751R sh4)

CQ Publishing Co., Ltd.

http://www.cqpub.co.jp/eda/CqREEK/SH4PCI.HTM

- CQ RISC Evaluation Kit (CqREEK)/SH4-PCI with Linux

Kyoto Microcomputer Co., Ltd. (KMC or KuC)

http://www.kmckk.co.jp/eng/

- Solution Platform KZP-01 (KZP-01 Mainboard + KZ-SH4RPCI-01 SH4 CPU Board)

Silicon Linux Co., Ltd.

- CAT760 (SH7760)
  - Product Information Released 2006/02/10.
  - CPU: SH7760(SH-4) 200MHz Bus clock: 66MHz
  - MEMORY:
    - Boot & Filesystem FlashROM 16Mbytes
    - MAIN MEMORY SDRAM 64MBytes（Onboard）
    - BATTERY Backup SRAM 512KBytes（Onboard from Power Supplied by Mainboard）
  - External Interfaces:
    - Serial Interface R?S232Level x2 3.3V LVTTL x1
    - Ethernet 100BaseT(AX88796) x1ch
  - Power Supply: +3.3V/GND Single Power Supply
  - Power Consumption: stable 200mA Max 600mA
  - Board Size: 73x60mm + Baseboard 105x148mm (A6 Paper Size)
  - Operating Systems:
    - Linux Kernel 2.6.15 Busybox Based Mini System

- CAT709 (SH7709S)
- CAT68701 (SH7780R for A-one CATBUS (designed for 68000 board) compliance)

Daisen Electronic Industrial Co., Ltd.

- SH2000 (SH7709A 118MHz)
- SH2002 (SH7709S 200MHz)
- SH-500 (SH7709S 118MHz)
- SH-1000 (SH7709S 133MHz)
- SH-2004 (SH7750R 240MHz)

IO-DATA Device, Inc. (NAS Series)

http://www.iodata.jp/product/hdd/lanhdd/

- LAN-iCN (NAS Adapter for IODATA HDD with "i-connect" Interface)
- LAN-iCN2 (NAS Adapter for IODATA HDD with "i-connect" Interface)
- LANDISK (SH4-266MHz[FSB133MHz] RAM64MB UDMA133 USB x2 10/100Base-T)
  - HDL-xxxU (LANDISK Series NAS Standard Model)
  - HDL-xxxUR(LANDISK with RICOH IPSiO G series print monitor for Windows support)
  - HDL-WxxxU(LANDISK with wide body & twin drive support for Heavy storage or RAID1)
  - HDL-AV250(LANDISK with Home Network DLNA guideline support)
  - LANTank(LANDISK kit SuperTank(CHALLENGER) Series)
    - HDL-WxxxU based twin drive bulk NAS kit. LANTank have a special feature witch supported network media server(cf. iTunes etc..).

TOWA MECCS Corporation

- TMM1000 (SH7709)
- TMM1100 (SH7727)
- TMM1200 (SH7727)

Sophia Systems

http://www.ss-technologies.co.jp/index.html

- Sophia SH7709A Evaluation Board
- Sophia SH7750 Evaluation Board
- Sophia SH7751 Evaluation Board

Moving Eye, Inc.

- A3pci7003 (Using SH7750/ART-Linux (Linux with Realtime Extension))

AlphaProject Co., Ltd.

http://www.apnet.co.jp/product/ms104/ms104-sh4.html

- MS104-SH4 (SH7750R/PC104(Embedded ISA Bus) with apLinux)

Interface Corporation

MPC-SH02 (SH7750S: ATX Motherboard Style)

PCI-SH02xx (SH7750S: PCI-CARD Style)

TAC Inc.

http://www.tacinc.jp/

- T-SH7706LAN another name "Mitsuiwa SH3 board" SH-MIN (SH7706A/128MHz Flash512KB SDRAM 8MB 10BASE-T)

SecureComputing/Snapgear (older products, check ebay etc, all can netboot and have a debug header)

- SG530 (SH7751@166MHz RAM16MB FLASH4MB 2x10/100 1xSerial)
- SG550 (SH7751@166MHz RAM16MB FLASH8MB 2x10/100 1xSerial)
- SG570 (SH7751R@240MHz RAM16MB FLASH8MB 3x10/100 1xSerial)
- SG575 (SH7751R@240MHz RAM64MB FLASH16MB 3x10/100 1xSerial)
- SG630 (SH7751@166MHz PCI NIC card RAM16MB FLASH4MB 1x10/100 1xSerial-header)
- SG635 (SH7751R@240MHz PCI NIC card RAM16MB FLASH16MB 1x10/100 1xSerial-header)

Sega Dreamcast

The Dreamcast is Sega's SH-4 based console gaming system, it uses the SH7091 CPU subtype.

HOWTO Boot The Dreamcast

Please note, the method described immediately below describes building a kernel for an NFS root mounted system for which you need some sort of LAN connection between your NFS server/build machine and the Dreamcast. It is also possible to build a working kernel with only a serial connection - see the bottom of the page for the differences between the two methods.

Overview (NFS root)

- Install cross compiler
- Cross compile kernel
- Burn CD-R which will load a small daemon
- Setup DHCP/NFS server
- Transmit kernel to running daemon
- Have fun !

Install cross compiler

First you need a cross-compiler! Visit Dan Kegel's website and build a SH cross-compiler. For Gentoo users, you can simply do this:

http://kegel.com/crosstool/

```
# emerge crossdev && crossdev -t sh4
```

See the FrontPage for other methods.

Cross compile kernel

See the FrontPage for how to get the source and use it.

For an NFS root, I normally use this simple kernel append:

```
$ grep CONFIG_CMDLINE .config

CONFIG_CMDLINE="console=ttySC1,115200 panic=3 root=/dev/nfs
nfsroot=192.168.1.4:/mnt/dreamcast,v3,tcp init=/bin/bash"
```

The panic will automatically reboot the dreamcast when something goes bad so you don't have to manually cycle the power. 192.168.1.4 is my NFS server and I connect to it using NFSv3 over TCP since it's a *lot* more reliable than NFSv2 or UDP.

Burn Bootable Image

Prepare the image you want to burn

You will need an `IP.BIN` file (this is the bootstrap code that gets everything going). Then you will need a `1ST_READ.BIN` file. This is the binary that is hardcoded to be executed by `IP.BIN`. So, when creating the ISO image, make sure you have `1ST_READ.BIN` in the / of it.

- Some pre-made files may be found at [http://dev.gentoo.org/~vapier/dreamcast/]

Figure out how to use cdrecord

Basically, you need to know what to pass to the `dev` parameter. Replace all `dev=0,0,0` with whatever you have on your system. I use an IDE drive, so I just have to use `dev=/dev/cdr`.

Creating the image

First we need to create the audio session on the disc:

```
# dd if=/dev/zero bs=2352 count=300 of=audio.raw
# cdrecord dev=0,0,0 -multi -audio audio.raw
```

Now we make the ISO from the `cdimage` directory. Remember, the file `cdimage/1ST_READ.BIN` must exist for this to work.

```
# mkisofs -l -r -C 0,11702 -G IP.BIN -o tmp.iso cdimage/
```

Burning the image

Now burn it :P.

```
# cdrecord dev=0,0,0 -multi -xa tmp.iso
```

Now put the disc in your Dreamcast and it should work, hooray !

Setup DHCP/NFS server

Install a dhcp server such as ISC's dhcp. Here's a basic config:

http://www.isc.org/downloads/DHCP/

```
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.200 192.168.1.250;
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.1.255;
}

host dreamcast {
  hardware ethernet 00:d0:f1:02:b7:61;
  fixed-address 192.168.1.6;
}
```

You will have to change the ethernet MAC to match the MAC of your BBA. 192.168.1.6 will be the IP given to the dreamcast.

Setup /etc/exports for the nfsroot:

```
/mnt/nfsroot  dreamcast(async,rw,no_root_squash)
```

Then unpack a SH distro into /mnt/nfsroot

Transmit kernel to running daemon

Connect the serial port of your PC to the dreamcast via a Coder Cable. You can find directions online on how to make your own.

Run a terminal program such as minicom and connect it to your local serial port (normally /dev/ttyS0 or /dev/ttyS1 ... this value has nothing to do with the command line given to the kernel earlier). Make sure you set it to run at the same buad as what you built into the kernel (normally 115200).

Once you've compiled your kernel, install dc-tool-ip and use it to send the kernel to the dreamcast:

```
$ dc-tool-ip -n -x arch/sh/boot/zImage -t x.x.x.x
```

Early User Space method

(Use this if you haven't got some sort of NIC - you'll still need a serial cable though and it also works with Dreamcasts with BBAs/LAs)

Since 2.5.xx, the linux kernel build has included a replacement method, called Early User Space or INITRAMFS (for inital RAM filesystem), for the initrd - the initial ram disk. Previously initrds were the standard way to boot - without NFS Root - diskless systems like the Dreamcast (they are also used in the big ix86 distros to load up drivers before the 'real' root filesystem is mounted).

INITRAMFS offers an opportunity to bundle a root file system inside the zImage (built as above). The kernel will then (depending on the presence of the INITRAMFS archive and the kernel command line) boot the bundled filesystem (or rather mount the filesystem and then run the specified init).

The steps required to do this are very similar to those specified above for the NFS root kernel.

Specifically you need to use the cross compilation tools to build both the kernel and an appropriate filesystem for the Dreamcast.

When building the kernel you need to ensure CONFIG_INITRAMFS_SOURCE is specified (see the INITRAMFS documentation). I have this:

```
CONFIG_INITRAMFS_SOURCE="/home/adrian/linux-sh/initrd"
```

Where /home/adrian/linux-sh/initrd is a busybox installation with the addition of an init in the root directory (created simply by ln -s ./bin/busybox init)

The kernel build process turns the specified directory into an appropriate .cpio.gz which is bundled into the kernel zImage which is then passed to the Dreamcast using either serial or network tools as described above.

(When building the kernel avoid specifying kernel autoconfiguration and root filsystem on NFS for obvious reasons and similarly ensure that the kernel command line does not specify a root filesystem).

Additional Information

- [http://linuxdc.sourceforge.net] Linux on the Dreamcast, the original project page, still a good resource for various device support.
- [http://curmudgeongamer.com/article.php?story=20040105232756356]
- [http://mc.pp.se/dc/cdr.html]
- [http://adk.napalm-x.com/dc/dcload-ip/]

MS7206SE01

MS7206SE01 (SH72060 Solution Engine)

This is Hitachi ULSI Systems' reference platform for the [SH7206] processor.

Loading RedBoot

The first step for working with this board should be getting a better loader on it. Porting of this was done by Yoshinori Sato. For anyone used to working with Solution Engine boards, this process should be pretty familiar. But nevertheless, Sato-san has more detailed instructions here. Likewise, S-records and other things can be found here, and here.

Page TopLinux Support anchor.png

Porting of the 2.6 kernel to this platform has been started and is currently in progress. Work for this is being done in the following branch:

http://uclinux-h8.sourceforge.jp/ms7206se01/

Linux Support

Porting of the 2.6 kernel to this platform has been started and is currently in progress. Work for this is being done in the following branch:

```
lethal@linux-sh.org--linux/shnommu--sh72060-devel--2.6
```

Toolchain Support

This board is wired as big endian out of the box, and while this is configurable, it is generally intended to remain in this configuration due to how the MRSHPC is configured. Most of the current readily available toolchains are aimed at little endian use, and also don't factor in running without an MMU. As a result of this, the uClibc buildroot is the only sensible way to produce a toolchain for this platform.

http://www.uclibc.org/toolchains.html

Buildroot now also has a target device configuration for this board, though this is only of use to those building gcc 4.x and later toolchains (specifically targetted at sh2a_nofpueb).

- Pre gcc 4.x toolchains

Ideally an sh2eb-linux-uclibc toolchain should be created, though for those interested in hacking on the kernel for this platform, an sh4eb-linux-uclibc toolchain with a recent binutils (at least 2.16.90.0.1) will suffice. The newer versions of binutils support proper ISA tuning for SH-2A, which is a significant benefit for hunting down bogus instructions in the current kernel tree that need to be handled.

- Newer toolchains

For gcc 4.x and up, it's recommended to use an sh2a_nofpueb-linux-uclibc toolchain. While the naming convention leaves quite a lot to be desired, it ensures that the ISA tuning is taken care of transparently (though it is still possible to manually tune the ISA via -Wa,-isa= if for some reason some deviation is necessary). Transparent tuning will give us the best results when building anything other than the kernel where we don't want to bother with explicit ISA tuning.