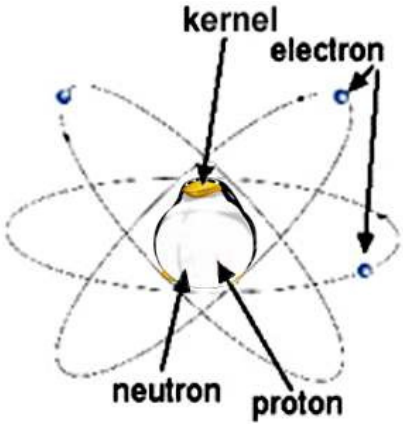
The Kernel

Patching and compiling



This chapter contains information about

Addidtional files needed
Installing a binary-kernel as rpm
Getting a fresh kernel tarball
Patching the kernel
Other patches

Configuring the kernel
--- Code maturity level options
--- Processor type and features
--- Enable loadable module support
--- General Setup
--- Block devices
--- Networking options
--- Network device support
--- Character devices
--- Character devices / mice
--- Filesystems
--- Filesystems / Network File System Support
--- Partition Types
--- Filesystems /Native Language support
--- Console Drivers

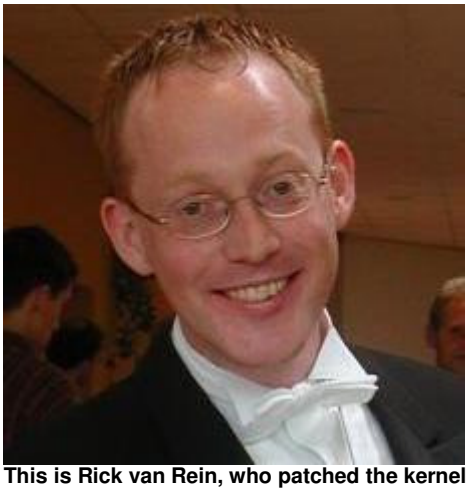
Kernel compilation
Module compilation
Copy the kernel to the boot-directory
Restart
Make bootdisk

If you're using **Debian** you should take a look at my **tips for installing** a packaged custom 2.4-kernel. It's much easier than going all the way from source. **;-)**

Why you need to patch and compile the kernel new on an Compaq Contura Aero Laptop-Computer

Code is written by real people:

This is Linus Torvalds, who wrote the Linux-kernel.



This is Rick van Rein, who patched the kernel so it is usable with the aero and 20 MB RAM.

As I already mentioned in the previous chapter **"The Bad RAM-problem"**, linux tends to crash after a while on the aero when you have 20MB of RAM - especially with pcmcia and the network enabled.

This can be solved with a kernel patch, the BadRAM-patch of Rick van Rein which can be found at:

http://rick.vanrein.org/linux/badram/

In this chapter I describe how to patch the linux-kernel with the BadRAM-patch.

Patching the kernel means to compile and install it new. This is also quite useful on the aero - I took the opportunity to eliminate unused drivers and gained more free ram which is essential on such a small machine.

Compiling a new kernel on a 486SX33-laptop with a maximum of 20 MB RAM needs time. All in all compiling alone takes five hours and there is more time needed for configuring and additional work. You may be faster if you install the aeros harddisk into a desktop-machine and compile the kernel there. Compiling on a pentium-200 with 128 MB RAM took only est. half an hour for compiling.

Addidtional files needed

For recompiling the kernel you need to have the compiler software installed. If you don't the compiling process will abort. The needed packages for Red Hat 6.1 are all listed in the chapter **"The Red Hat 6.1 Installation process"**.

With a Red Hat system you also should take a look at the red hat errata site, if there are any bugfixes or security updates for the compiler software.

For Red Hat 6.1 the ftp-site with the updates can be found at

ftp://updates.redhat.com/6.1/en/os/i386/

(For other Red Hat-versions simply replace the version-number in the ftp-address.)

Installing a binary-kernel as rpm

With Red Hat it is possible (but not suggested here) to install the kernel by simply downloading a *.rpm file and installing it. For instance: The official release Red Hat 6.1 came with kernel 2.2.12. For that Release Red Hat suggests to install the newer kernel 2.2.17.

The kernel files can be found at:

ftp://updates.redhat.com/6.1/en/os/i386/

So, if you want to install a rpm-kernel, downlad the needed files:

```
initscripts-4.70-1.i386.rpm
kernel-2.2.17-14.i386.rpm
kernel-pcmcia-cs-2.2.17-14.i386.rpm
kernel-source-2.2.17-14.i386.rpm
SysVinit-2.78-5.i386.rpm
```

The normally also needed file mkinitrd***.rpm is not necessary for the installation.

The installion of the kernel is excellently described at:

http://www.redhat.com/support/resources/howto/kernel-upgrade/

Follow these steps exactly. The important thing is: You don't upgrade your kernel - **you install a new (additional) one**.

Never use "rpm -Uvh" for installing a kernel-rpm!!!

If you accidently use the "update"-command (rpm -Uvh) with the new kernel file you will have much work (it ended up for me in acompletely new install of redhat6.1). So use the command "rpm -ivh".

All in all I don't think the installation of a kernel as rpm is useful on the aero. As you have to patch and recompile it anyway, you also could download a fresh tarball and use that one. If you use a fresh tarball together with the appropriate BadRAM-patch, you also won't receive any rej-errors while patching - the Red Hat rpm-kernel is already patched and extended for other purposes by Red Hat.

I can recommend kernel 2.2.19 because there is a version of rick van reins BadRAM-patch for this kernel which works quite well. I can't suggest to use a 2.4 kernel - there are the BadMem-patches, the successor of the BadRAM-project, but unfortunately BadMem didn't work.

If you wan't to use a 2.4 Red Hat - kernel you can install the kernel-source-rpm instead of a binary kernel. (For instance: kernel-source-2.4.20-20.9.i386.rpm") The kernel source rpm will install itself into /usr/src and you can treat it the same way as you would do it with a fresh kernel tarball as described further.

Also for red hat kernels there is a good source for adapted BadRAM-patches at:
BadRAM for RedHat Linux Kernels

Getting a fresh kernel tarball

If you decided to install the kernel as rpm-package like described above you already find the new kernel-source under /usr/src. It's called linux-2.2.17.

I instead decided to download a fresh kernel tarball and unpack it into this directory. The original kernels can be found at:

http://www.kernel.org/

In my first installation of linux on the aero, I took kernel 2.2.17, because Red Hat suggested an update to that version and there was also a BadRAM-patch for this version available. Now I have a RedHat 7.1 system on the aero, installed by plugging the aeros harddisk into my desktop computer. With that system I decided to take kernel 2.2.19, because there was also a BadRAM-patch available for that version, and also the ext3-patch which gives me the journaling ext3-filesystem.

Whatever kernel you take, download it as compressed *tar.gz-file for instance:

http://www.kernel.org/pub/linux/kernel/v2.2/linux-2.2.17.tar.gz

Of course in this stadium of the linux-installation on the aero the network is not up. So I downloaded the kernel with my desktop computer, booted the aero new and chose in lilo to start dos. Then I copied it from my desktop to the aero with winlink. Because winlink only knows 8.3-filenames I had to rename the file afterwards to give him back his original name.

I then restarted the aero with linux and copied the file from the dos-partition (/mnt/hda1) to the directory /usr/src. Afterwards I unpacked the kernel with the command:

```
tar xzpf linux-2.2.17.tar.gz
```

Now you have a clean source tree of the original kernel laying in /usr/src as a directory called linux. For a better overview I renamed that directory to linux-2.2.17 and made a symlink "linux" which points to that directory.

Patching the kernel

If you followed the steps so far you have kernel 2.2.17 installed in your system. The kernel files are in "/usr/src/linux-2.2.17".

In "/usr/src" there is also a symlink for the default linux directory "linux" which points to the booted version linux-2.2.17.

Now download the BadRam patch for this version from

http://rick.vanrein.org/linux/badram/software/BadRAM-2.2.17.1.patch

Copy the downloaded file to "/usr/src" (and if its still gzipped unpack it).

Look at the official linux-howto about patching the kernel, which can be found at:

http://www.linux.org/docs/ldp/howto/Kernel-HOWTO-6.html

```
cd /usr/src/linux
make clean
cd /usr/src
patch -p0 < BadRAM-2.2.17.1.patch
```

You will see things floating by, and - at least in my case - some error messages. So the patch wasn't able to change everything it had to change. That was because I worked with an kernel 2.2.17 which came from Red Hat and not from kernel.org. Red Hat already has changed some kernel-code, so a patch for the original code will produce errors. If you - what I suggest - take an original kernel from kernel.org, these errors won't occur. In all error cases, the patch creates files with the ending ".rej" (rejected).

Just for understanding: The source code that BadRAM has to change are only textfiles. The patch searches these files for lines that it knows and inserts additional text. But if those lines are on a different place (because there is already a patch installed or additional configuration has been made) it won't find them and reject the file.

So search for the ".rej" files with midnight commander and look at them. Each ".rej" file lists the original lines the patch wanted to change first and the lines with the new (patched) text inserted second. The lines it intended to change are marked with a "+".

Now all you have to do is to look up the rejected file and change it manually.

For instance if there is a *.rej file called

```
/usr/src/linux-2.2.17/Documentation/Configure.help.rej
```

open "/usr/src/linux-2.2.17/Documentation/Configure.help" look for the lines in the ".rej"-file and replace them per hand.

Other patches

If you are already patching your kernel, I would suggest to also take NOW a look on two additional patches:

- The ext3-patch that gives you the features of a journaling filesystem
- The openwall-security patch that is highly recommended if you intend to use the aero as server.

See the chapter **"Finished - What comes next"** for additional information.

Attention: Broken boot-penguin! If you decide to take kernel 2.2.19 you should now also solve the problem with the broken boot-logo of this kernel. Just copy the corrected file to your kernel-source at "/usr/src/linuc/include/linux" like it is described in the chapter **"Finished - What comes next"**.

Configuring the kernel

While compiling the kernel the first time it seemed to me, that this was in fact like working very hard on changing the os. Now after it was succesful I would say it is more like configuring "config.sys" and "autoexec.bat" in a msdos-system - only that the steps are more complicated and take much more time.

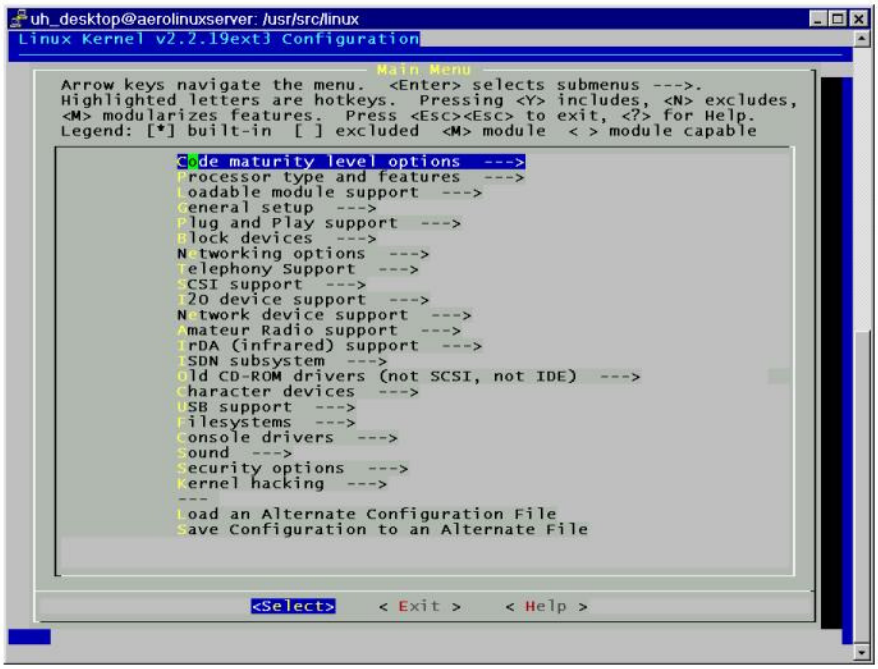
So here are those steps as I did them. Maybe some of them are not necessary - but they worked for me.

Note: In the meantime I changed to kernel 2.4.23. If you are interested in the kernel options I used with that version look **here**.

Change to the directory "/usr/src/linux"

Give in the commands:

```
make clean
make menuconfig
```



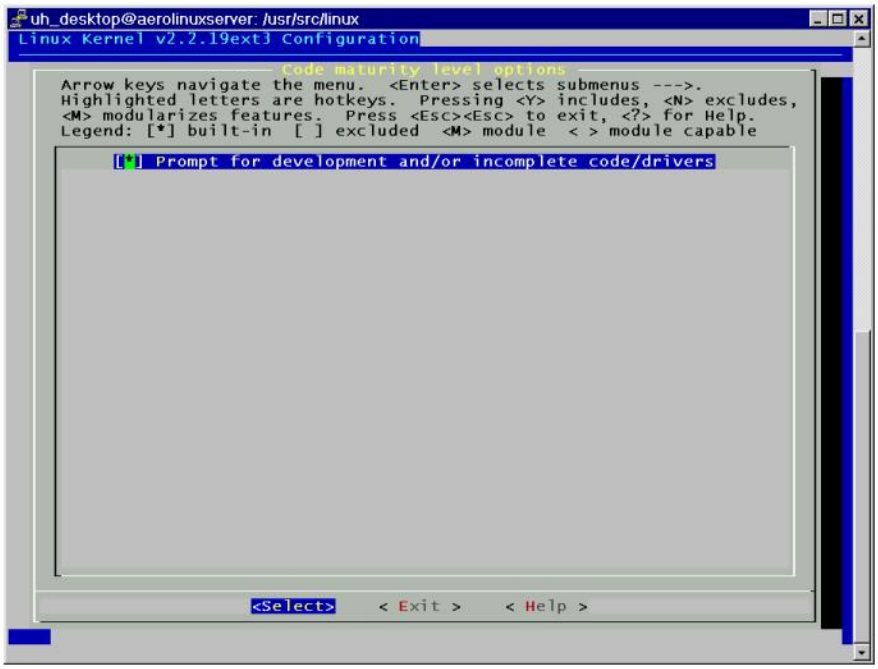
Now you can choose everything your next kernel should support. I will describe now the settings I made and that work with my aero. You have to find out yourself which settings are reasonable for your machine. I accepted many default settings. Also I may have forgotten something and of course my settings may not suit to others.

You already can get more information about every option in menuconfig by typing "?", so you should use think yourself about this and don't completely trust me, as I am a linux beginner too.

In the screenshot above there are several options I left disabled for my purpooses or because they don't make sense on the aero. The disabled optons are:

DISABLED: Plug and Play support
DISABLED: Telephony support
DISABLED: SCSI support
DISABLED: I20 device support
DISABLED: Amateur Radio support
DISABLED: IrDa Support
DISABLED: ISDN Subsystem
DISABLED: Old CD-ROM-drivers
DISABLED: USB support
DISABLED: Sound

Code maturity level options

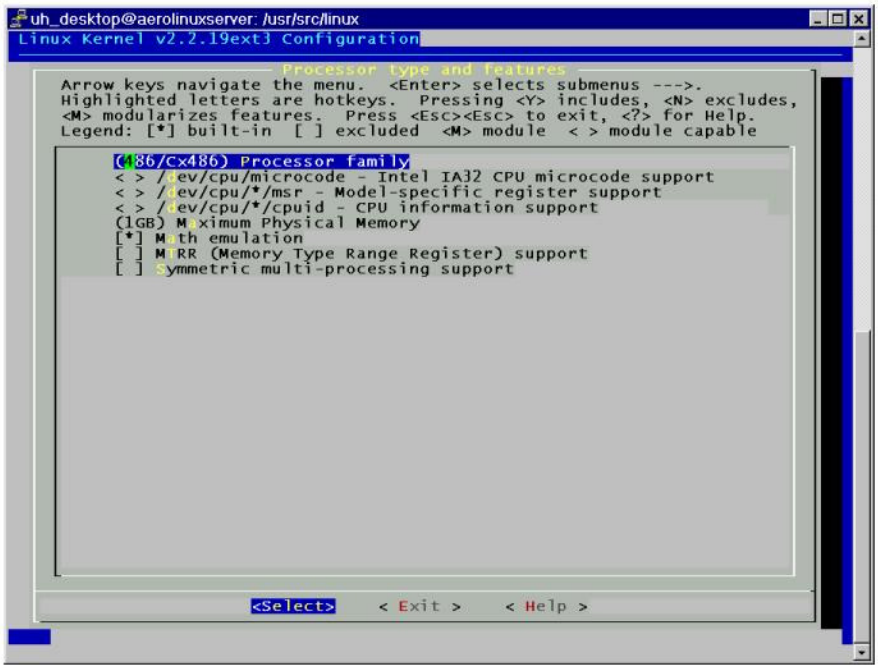


I enabled this option. The reason was to get access to the framebuffer-device options in the --> console drivers submenu. The framebuffer-device uses a kernel-driver for directly modifying the videomode. That means you have accelerated graphics and also a little penguin linux-logo appearing in the left corner of the screen while booting.

Enabling this option is also important if you want to use the aero as NFS-server in a linux-network. Only then there is an option in --> filesystems --> Network File Systems called "NFS Server support".

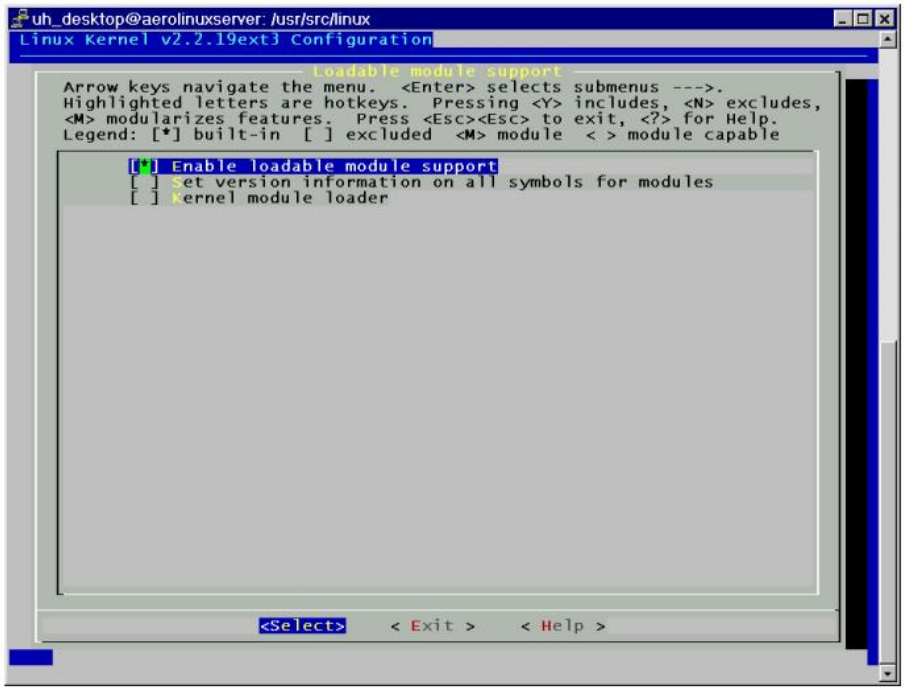
Of course you can switch it off if you are content with the console-graphics and want to use Samba for all your network.

Processor type and features



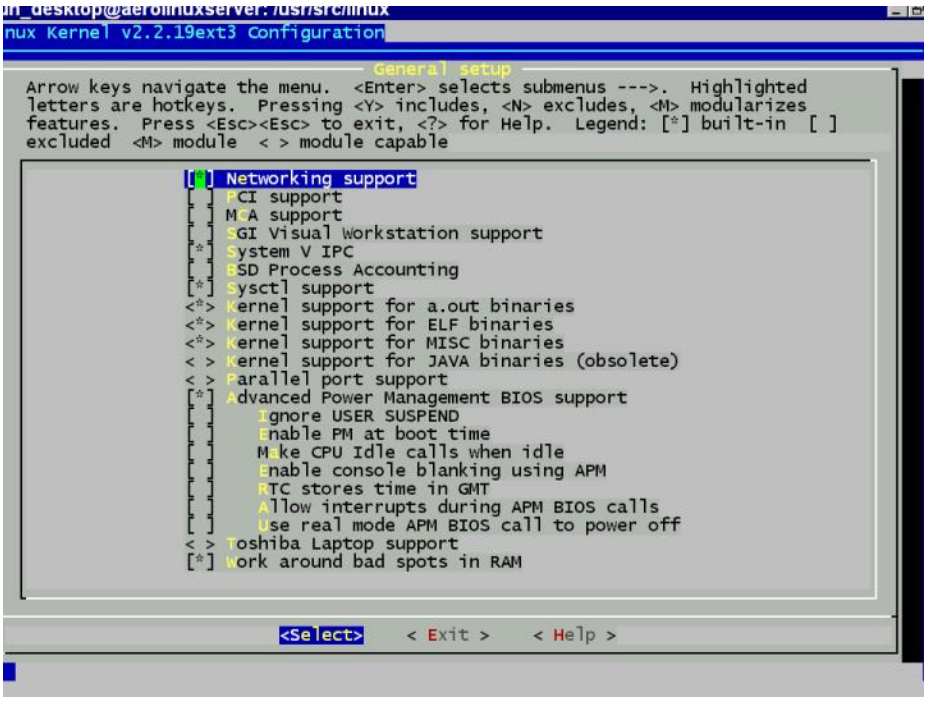
Think the aero is one of the few machines you have to set a 486/Cx486 - Processor family - and to enable the Math emulation. Setting the Maximun Physical Memory to 1 GB is required and of course worth a laugh with the aero ;-).

Enable loadable module support



I answered "yes" to this question, because of the pcmcia-cs-driver-package I wanted to install afterwards. This requires to be installed as module with a 2.2 kernel. I also configured some options as modules because I didn't know if I ever use them, but didn't want to do completely without.

General Setup



The most important option here for the aero is the last one:

"[*] Work around bad spots in RAM" is recommended to be checked if you want to use 20 MB RAM with the aero. This is the effect of the badRAM-patch I installed in the previous chapter.

The other options:

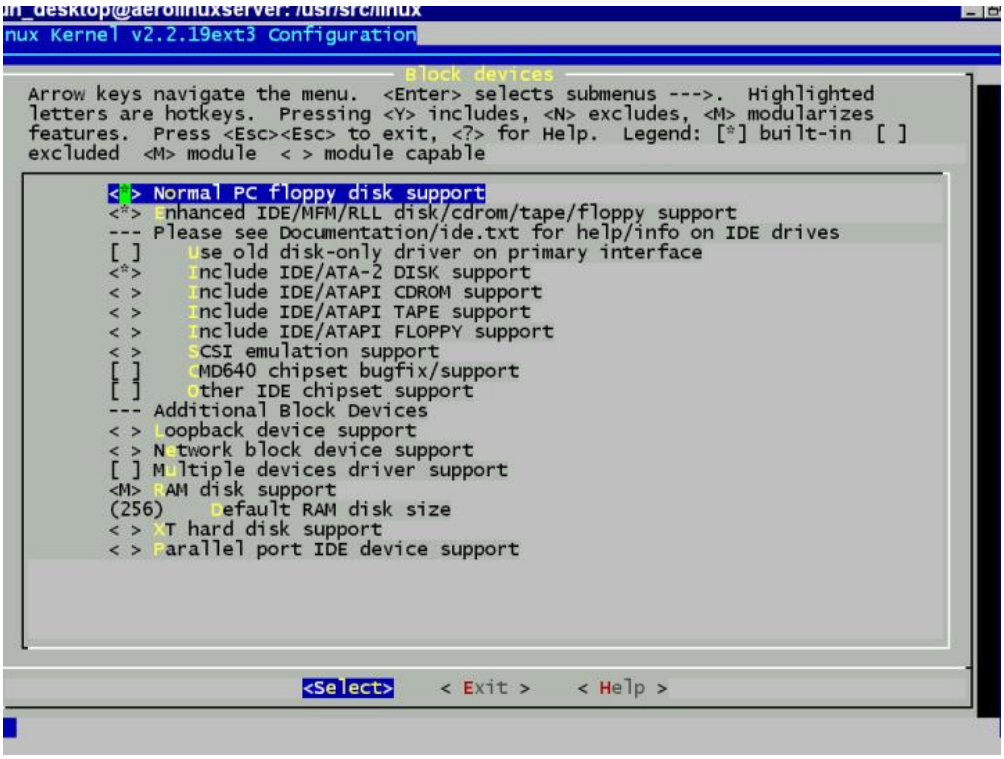
Networking support is necessary because I want to build up the machine to a webserver and therefore depend on networking. PCI support can be switched off if you intend to use the aero's harddisk only in the aero. The aero doesn't have PCI, so it won't need it.

Attention: You will need PCI-support, if you intend to take the aeros harddisk out from time to time and plug it into your desktop. I do this to compile software I want to use on the aero: For instance, compiling MySQL on the aero took more than 20 hours. In the desktop it was done after 45 Minutes. So take care to take one of the following steps: Either you compile PCI-support into the kernel. Or you compile two kernels - one with, one without PCI-support. Then you can modify lilo, so you can choose between both kernels at boot-up.

Advanced Power Management must be enabled with the aero. Otherwise you can't use PCMCIA-cards because the cardmanager uses apm-functions. Also without APM some features are lost: For instance your computer will not shut off after linux is stopped.

All other enabled functions are default and should be left as they are.

Block devices



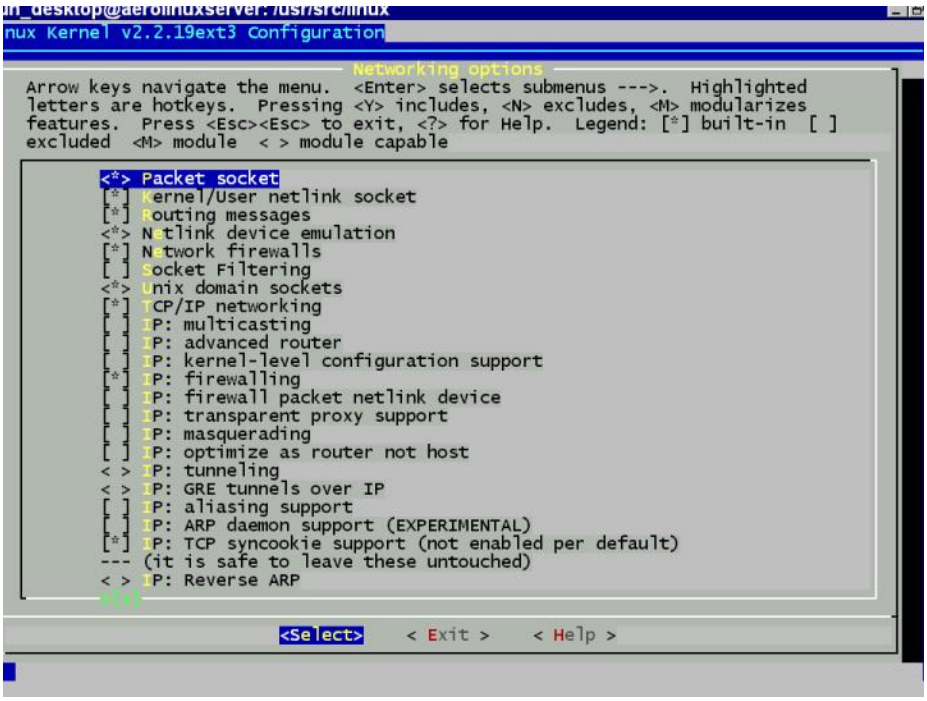
"Normal PC floppy disk support" should of course only be enabled if you own a compaq pcmcia-floppy-disk.

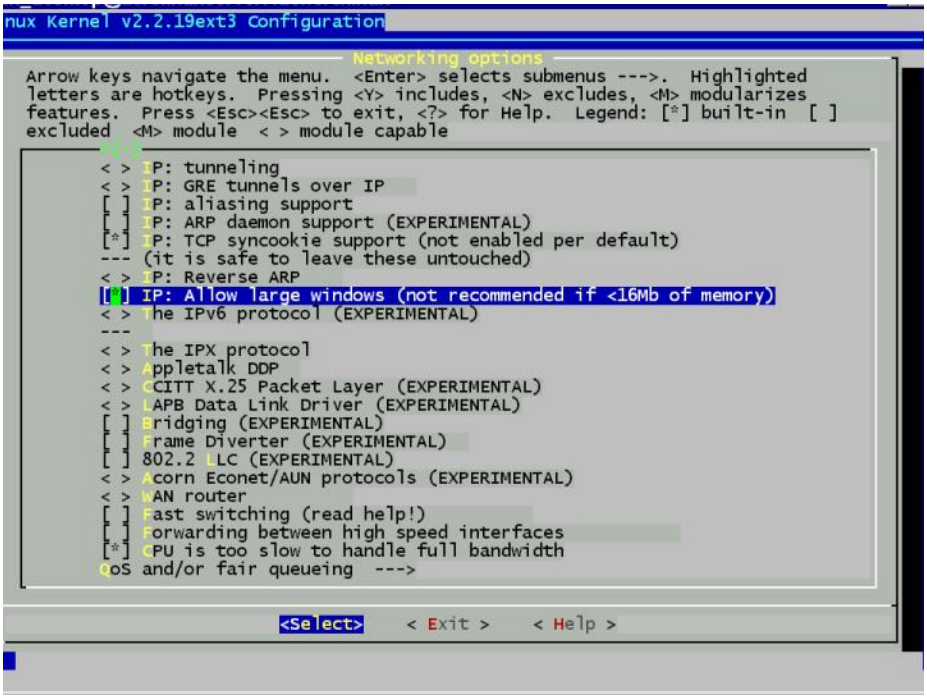
For the harddisk it was recommended for me to choose the "Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support". The "Old disk only driver on primary interface" didn't work for me. I also had to check the "Include IDE/ATA-2 DISK support" to make the kernel see my 6 GB harddisk - otherwise it saw only the first partitions.

I also compiled "RAM disk support" as module. As Ramdisk size I chose only 256kb, as the aero doesn't have much of it. The filesize you can later use with ramdisks doesn't depend on the size you choose here. For instance I am currently using two ramdisk of 200kb for the samba-fileserver (see chapter: "10.2 Putting the harddisk to sleep")

I disabled the parallel-port, because I don't intend to use a printer with that aero.

Networking options



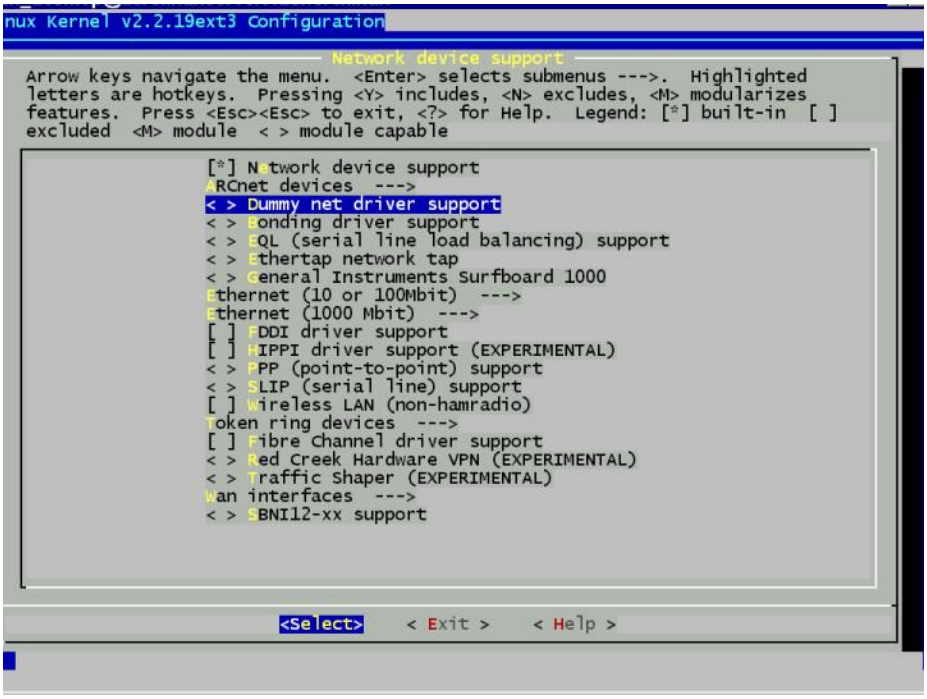


The "TCP syncookie support" was enabled for secutity reasons.

The option "--->QoS and/or fair queueing" was not enabled.

I also enabled the option "CPU is too slow to handle full bandwidth" because this is indeed the case.

Network device support



"Network device support" must be checked for networking. And that is the only option I enabled. As I don't intend to use SLIP, PPP or PLIP I said "No" to those options and also to the "Dummy net driver".

Also many submenus in this section were not enabled by me:

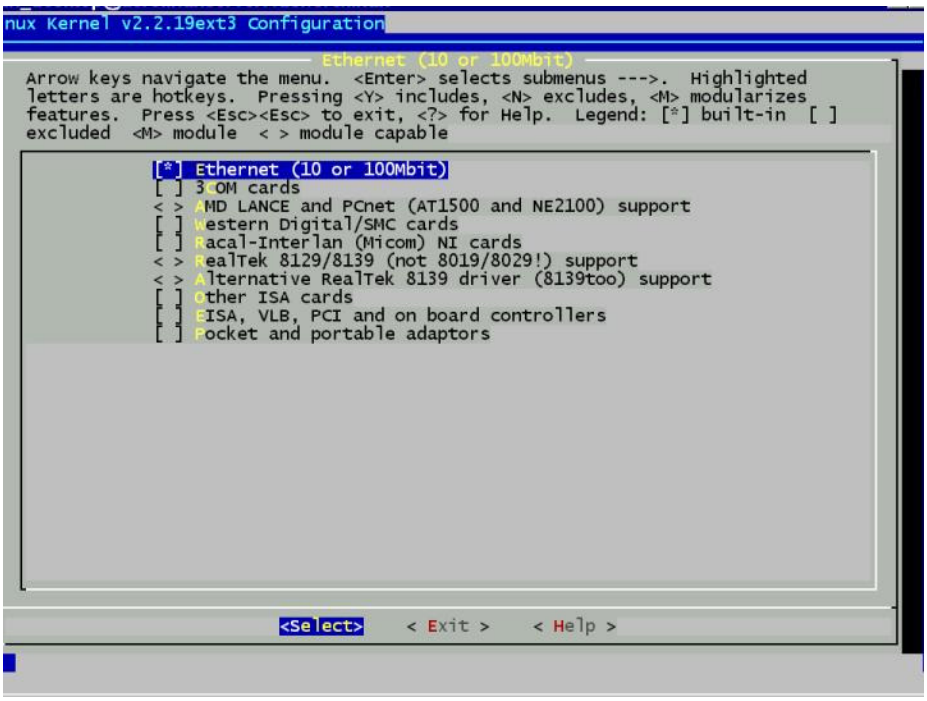
DISABLED: ---> ARCnet devices

DISABLED: ---> Ethernet 1000 Mbit

DISABLED: ---> Token ring devices

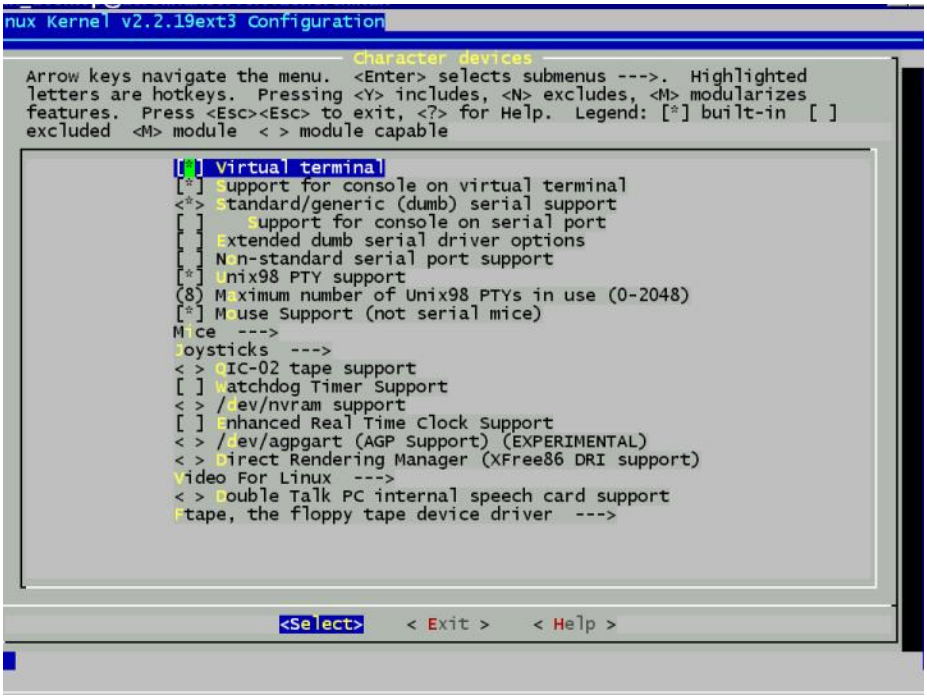
DISABLED: ---> Wan interfaces

But quite important is the submenu ---> Ethernet (10 or 100 Mbit):



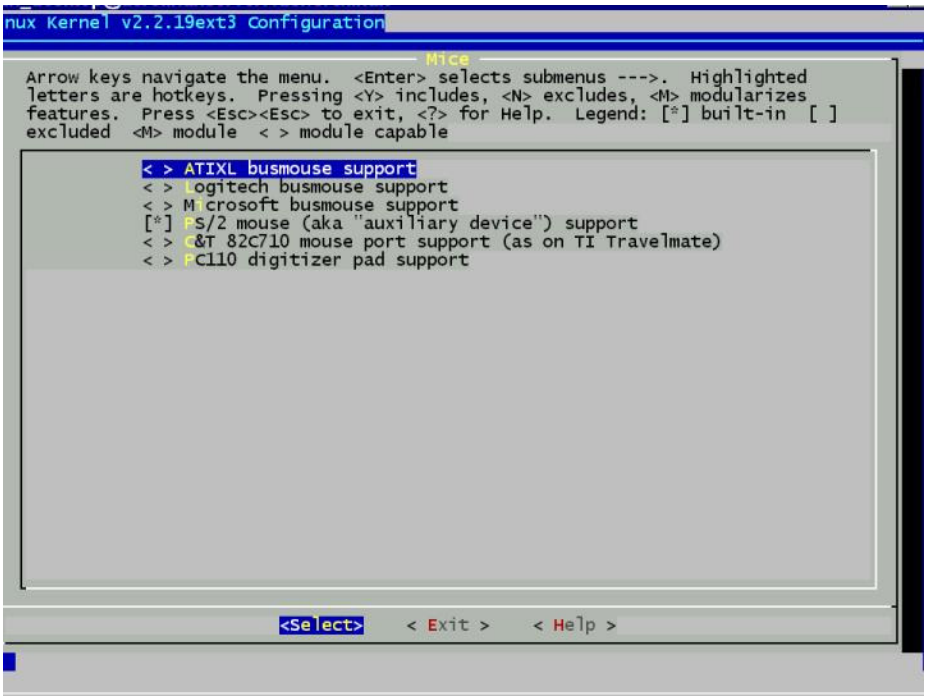
The only option you need to activate here is "Ethernet 10 or 100 Mbit" as the aero will work with pcmcia cards and this is something we'll realize as modules after the kernel-compilation.

Character devices



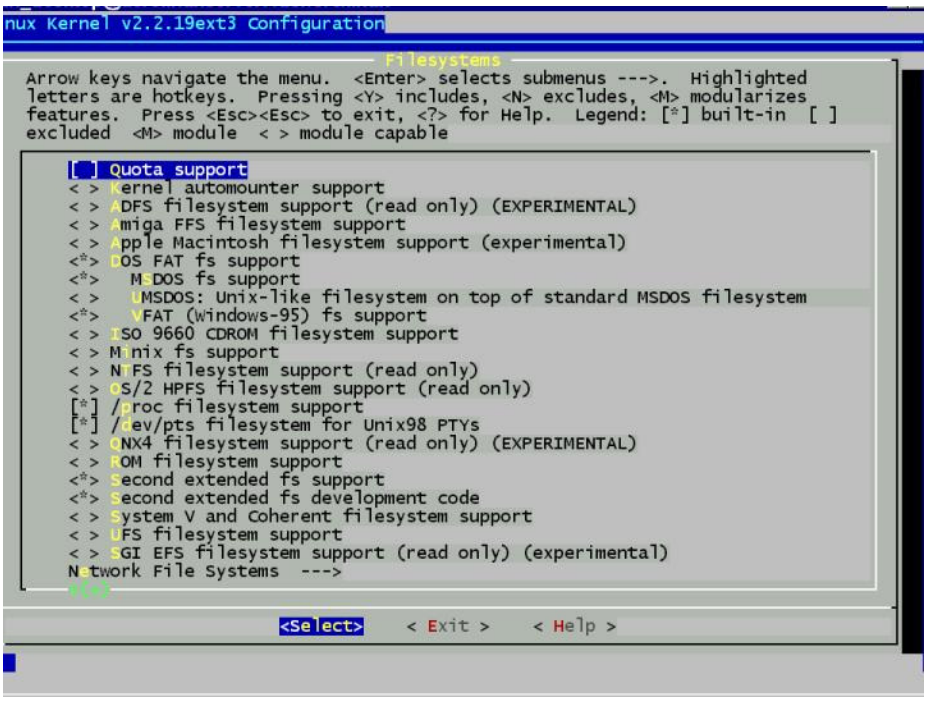
The options "virtual terminal", "Support for console on virtual terminal" and "Unix98 PTY support" are needed for remote control of the aero. As I want to run it as webserver which I intend to screw on my living-room-wall, I MUST login from a different machine - so I said "Yes". I reduced the "Maximum number of Unix98 PTYs in use" to eight.

Character devices / mice



In the subdirectory "mice" I said "yes" to "PS/2 mouse"

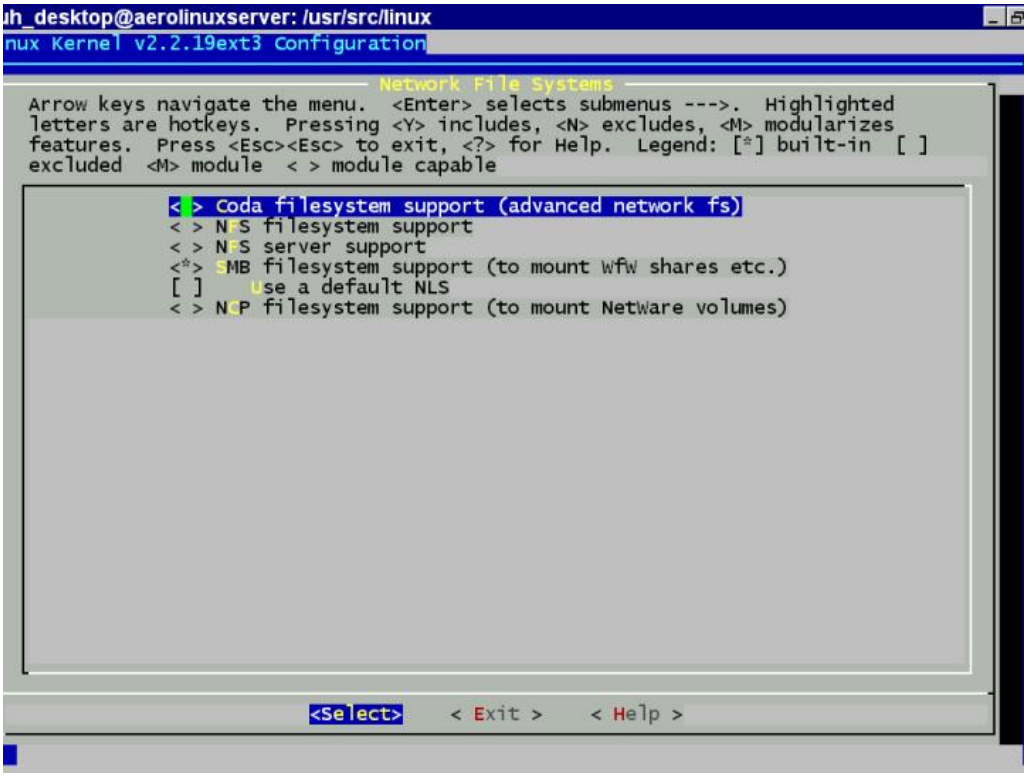
Filesystems



I said "yes" to "DOS Fat fs support" (DOS and VFAT). I don't need any other filesystems. The standard linux-filesystems "/proc", "/dev/pts" and "Second extended fs" must be also enabled.

Then there is another option: "Second extended fs development code". This is only possible if you have patched the kernel with the ext3-patch (see chapter "Additional Patches"). If enabled, the kernel supports ext3, which was the reason that I patched the kernel for it - so I enabled it.

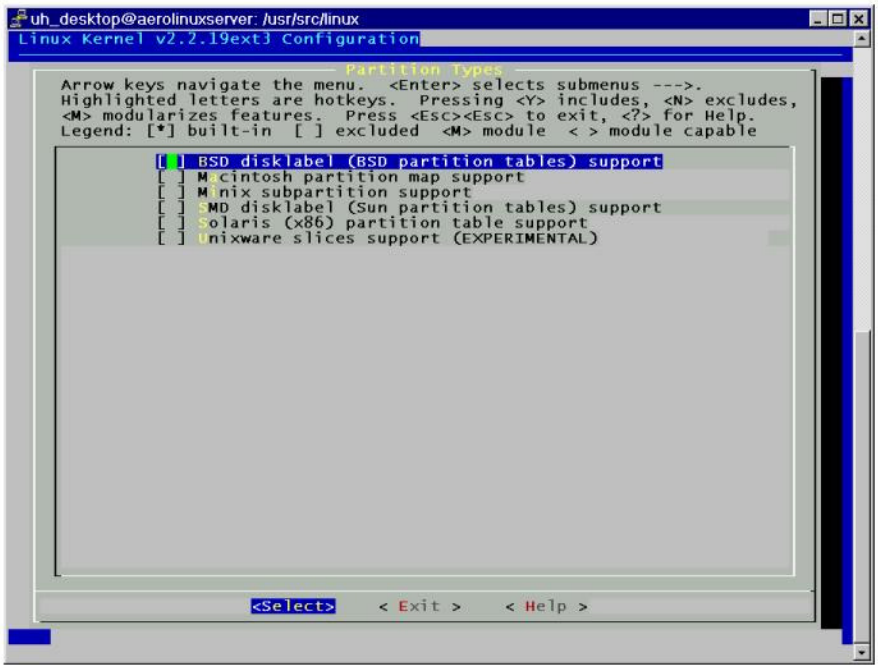
Filesystems / Network File System Support



I enabled only the "SMB File System support".

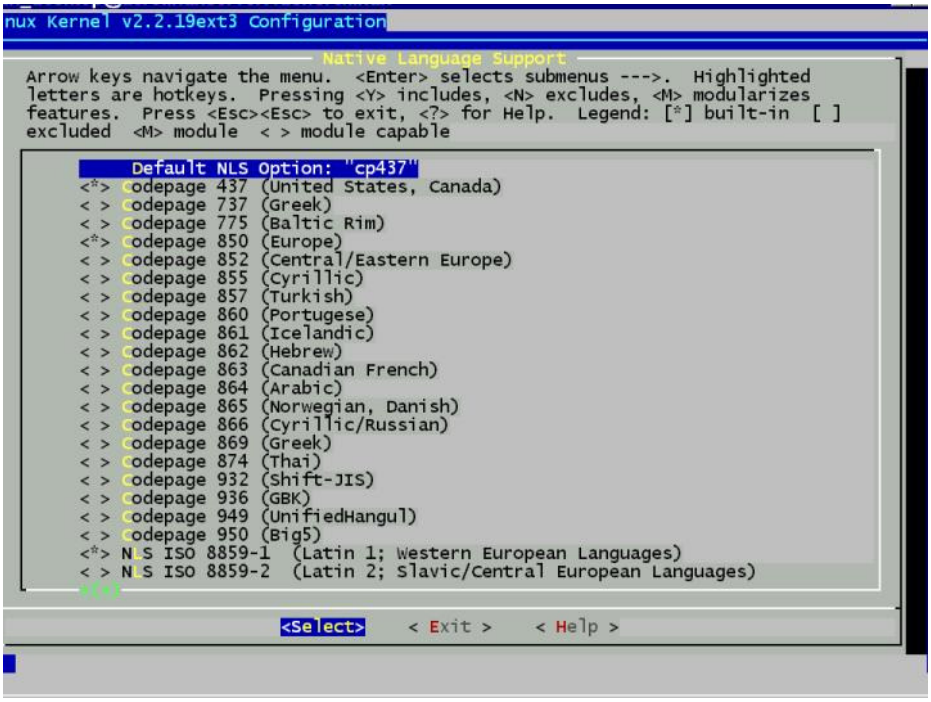
I didn't enable "NFS filesystem support" and also said "No" to the "NFS server support (NEW)" (which only shows up, because I enabled the option in "Code maturity level options"). Personally I don't intend to use NFS because I am doing well with samba and I very rarely connect the aero-server from another linux-system.

Partition Types



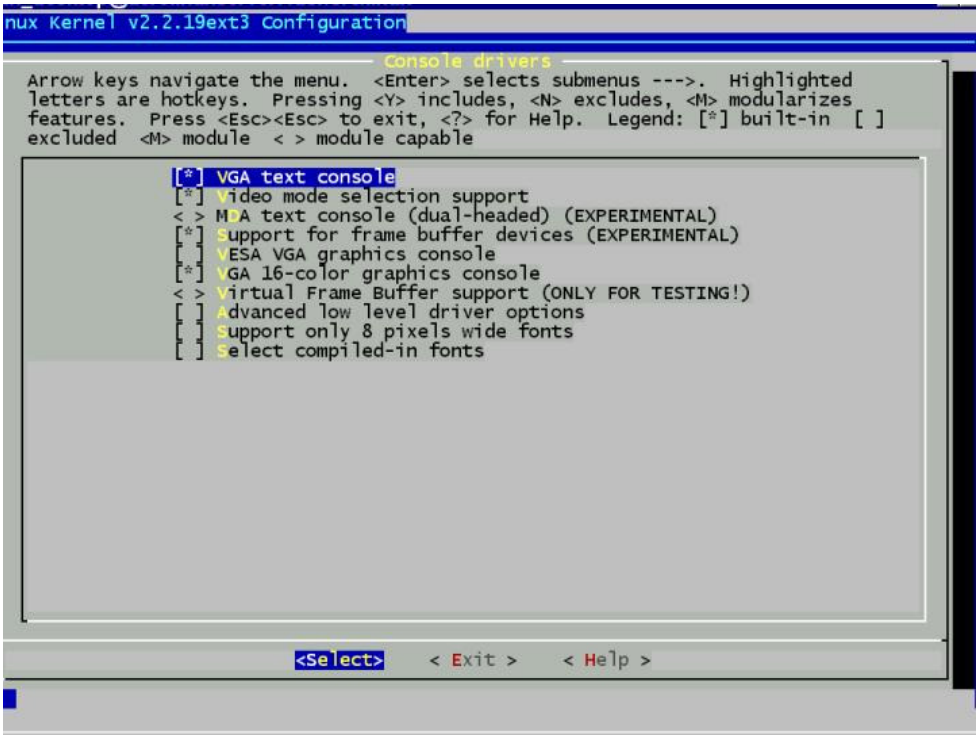
I did not enable any option here.

Filesystems /Native Language support



Beneath "Codepage 437" I said "yes" to "Codepage 850 (Europe)", "NLS ISO 8859-1 (Western Europe)".

Console Drivers



Beneath the default "VGA text console" and "Video mode selection support" (which brings higher console resolutions) I also enabled the "Support for framebuffer devices" and the apporpriate "VGA 16-color graphics console".

The framebuffer mode options are only available if you checked the option in "Code maturity level options". It is nice to have fast graphic-access under console and also incredibly motivating to see the little penguin at boot time.

If you use kernel-2.2.19 you should also look at chapter "10.13. Repair the little penguin at boot-time" for further information BEFORE you start compiling the kernel.

Kernel compilation

No you have to give in

```
make dep
```

this may take a while on the aero, so go drink a cup of coffee. (It took 20 minutes with my aero).

Now a very important step! Give in

```
make clean
```

And now you are ready to compile your kernel!

Give in:

```
make bzImage
```

I started the compilation on my aero at 1.10 am. At 5.57 am it was finished according to the timestamp of the new kernel. So it takes around 5 hours compiling a kernel on the aero.

Module compilation

Now we have to do the following steps:

Stay in "/usr/src/linux"

```
make modules
make modules_install
```

This may take a while again...

Copy the kernel to the boot-directory

The original kernel lays in "/boot". It is called "vmlinuz". In my case "vmlinuz" is symlink that points to "vmlinuz-2.2.17-14". So copy the old kernel from "/boot/vmlinuz-2.2.17-14" to "/boot/vmlinuz-2.2.17-14.old".

Now copy the just created kernel from "/usr/src/linux/arch/i386/boot/bzImage" to "/boot/vmlinuz-2.2.17-14".

Configure Lilo

Change the "Linux Loader Configuration" so it points to the new kernel:

```
---------My "lilo.conf"--------------------
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linuxnew
message=/boot/message

image=/boot/vmlinuz-2.2.12-32
          label=linux
          initrd=/boot/initrd-2.2.12-32.img
          read-only
          root=/dev/hda6

image=/boot/vmlinuz-2.2.17-14
          label=linuxnew
          read-only
          root=/dev/hda6


other=/dev/hda5
          label=setup

other=/dev/hda1
          label=dos

-------------------------------------------
```

If it works after a restart you can later use the new kernel as only one. Then you have to link the symlink vmlinuz against the new kernel vmlinuz-2.2.17-14. The lines in Lilo should afterwards look like this:

```
-------------------------------------------
image=/boot/vmlinuz
          label=linux
          read-only
          root=/dev/hda6
-------------------------------------------
```

Don't forget to command

```
lilo -v
```

as root-user in the shell after any change of lilo.conf to make it work.

Restart

I got many "unresolved symbols" errors while the first booting of the new kernel. I read somewhere that this is due to linux not refreshing the "/lib/modules" dependencies and kept old modules links in that directory.

So I copied the whole folder "/lib/modules/2.2.17-14" to "/lib/modules/2.2.17-14old".

Deleted everything in the folder "/lib/modules/2.2.17-14". And again went to "/usr/src/linux"

```
make modules_install
```

That created a new "~build" symlink in the directory "/lib/modules/2.2.17-14" which points to "/usr/src/2.2.17-14".

Next reboot was GREAT! Because I had nothing installed as modules the checking for dependencies was quick - all in all reboot was twice as much faster than before compiling the kernel. And it also brought me some more memory.

Make bootdisk

At the end you should make a new bootdisk for the new kernel. If you want to use the in Red Hat included software mkbootdisk, be sure you have installed mkbootdisk***.rpm in the install process. Mkbootdisk also won't work if pcmcia is enabled.

Alternatively you can do this:

- Boot the aero with the pcmcia floppy drive inserted (but no floppy in it)
- Insert a floppy
- To format the floppy type at the linux console:

```
mkfs -t ext2 -c /dev/fd0H1440
```

- To make the bootdisk type

```
dd if=/boot/vmlinuz of=/dev/fd0 bs=8192
```

The bootdisk is now created. Start the aero new with the floppy drive and the floppy inserted to check if it works.

Comments

Suggestions for this page? Ideas? Please drop a note!
Don't forget to add your email, if you appreciate a personal reply.
The comments are sorted from date.

 **previous** **index**  **next** 

Home · **manual & docs** · **drivers & updates** · **original software**
linux · **internal speaker** · **ads** · **the aero's wildest dream...** · **links**