Linux Ethernet−Howto

Table of Contents

Linux Ethernet−Howto......1
by Paul Gortmaker......1
1. Introduction......1
2. Frequently Asked Questions......1
3. Performance Tips......1
4. Vendor/Manufacturer/Model Specific Information......1
5. Cables, Coax, Twisted Pair......2
6. Software Configuration and Card Diagnostics......2
7. Technical Information......3
8. Networking with a Laptop/Notebook Computer......3
9. Miscellaneous......3
1. Introduction......3
1.1 New Versions of this Document......4
1.2 Using the Ethernet−Howto......4
1.3 HELP − It doesn't work!......5
1.4 Type of cable that your card should support......6
2. Frequently Asked Questions......7
2.1 What card should I buy for Linux?......7
2.2 Alpha Drivers −− Getting and Using them......7
2.3 Using More than one Ethernet Card per Machine......8
2.4 The ether= thing didn't do anything for me. Why?......10
2.5 Problems with NE1000 / NE2000 cards (and clones)......10
2.6 Problems with SMC Ultra/EtherEZ and WD80*3 cards......14
2.7 Problems with 3Com cards......15
2.8 FAQs Not Specific to Any Card......16
Linux and ISA Plug and Play Ethernet Cards......16
Ethercard is Not Detected at Boot......16
ifconfig reports the wrong I/O address for the card......17
PCI machine detects card but driver fails probe......17
Shared Memory ISA cards in PCI Machine do not work (0xffff)......17
Card seems to send data but never receives anything......18
Asynchronous Transfer Mode (ATM) Support......18
Gigabyte Ethernet Support......18
FDDI Support......18
Full Duplex Support......18
Ethernet Cards for Linux on SMP Machines......19
Ethernet Cards for Linux on Alpha/AXP PCI Boards......20
Ethernet for Linux on SUN/Sparc Hardware......21
Ethernet for Linux on Other Hardware......21
Linking 10 or 100 BaseT without a Hub......21
SIOCSIFxxx: No such device......21
SIOCSFFLAGS: Try again......21
Using `ifconfig' and Link UNSPEC with HW−addr of 00:00:00:00:00:00......22
Huge Number of RX and TX Errors......22
Entries in /dev/ for Ethercards......22
Linux and ``trailers''......22
Access to the raw Ethernet Device......22

Table of Contents

3. Performance Tips....................23
3.1 General Concepts....................23
3.2 ISA Cards and ISA Bus Speed....................23
3.3 Setting the TCP Rx Window....................24
3.4 Increasing NFS performance....................24
4. Vendor/Manufacturer/Model Specific Information....................25
4.1 3Com....................25
3c501....................26
EtherLink II, 3c503, 3c503/16....................26
Etherlink Plus 3c505....................27
Etherlink−16 3c507....................27
Etherlink III, 3c509 / 3c509B....................27
3c515....................28
3c523....................28
3c527....................28
3c529....................28
3c562....................29
3c575....................29
3c579....................29
3c589 / 3c589B....................29
3c590 / 3c595....................29
3c592 / 3c597....................30
3c900 / 3c905 / 3c905B / 3c905C....................30
3c985....................30
4.2 Accton....................30
Accton MPX....................30
Accton EN1203, EN1207, EtherDuo−PCI....................31
Accton EN2209 Parallel Port Adaptor (EtherPocket)....................31
Accton EN2212 PCMCIA Card....................31
4.3 Allied Telesyn/Telesis....................31
AT1500....................31
AT1700....................31
AT2400....................32
AT2450....................32
AT2500....................32
AT2540FX....................32
4.4 AMD / Advanced Micro Devices....................32
AMD LANCE (7990, 79C960/961/961A, PCnet−ISA)....................32
AMD 79C965 (PCnet−32)....................33
AMD 79C970/970A (PCnet−PCI)....................33
AMD 79C971 (PCnet−FAST)....................34
AMD 79C972 (PCnet−FAST+)....................34
AMD 79C974 (PCnet−SCSI)....................34
4.5 Ansel Communications....................34
AC3200 EISA....................34
4.6 Apricot....................35
Apricot Xen−II On Board Ethernet....................35

Table of Contents

4.7 Arcnet...35
4.8 AT&T...35
AT&T T7231 (LanPACER+)...35
4.9 Boca Research...35
Boca BEN400...36
Boca BEN (ISA, VLB, PCI)...36
4.10 Cabletron...36
E10**, E10**−x, E20**, E20**−x...36
E2100...36
E22**...37
4.11 Cogent...37
EM100−ISA/EISA...37
Cogent eMASTER+, EM100−PCI, EM400, EM960, EM964...38
4.12 Compaq...38
Compaq Deskpro / Compaq XL (Embedded AMD Chip)...38
Compaq Nettelligent/NetFlex (Embedded ThunderLAN Chip)...38
Compaq PCI card...38
4.13 Danpex...39
Danpex EN9400...39
4.14 D−Link...39
DE−100, DE−200, DE−220−T, DE−250...39
DE−520...39
DE−528...39
DE−530...39
DE−600...40
DE−620...40
DE−650...40
DFE−530TX...40
DFE−538TX...40
4.15 DFI...40
DFINET−300 and DFINET−400...40
4.16 Digital / DEC...41
DEPCA, DE100/1, DE200/1/2, DE210, DE422...41
Digital EtherWorks 3 (DE203, DE204, DE205)...41
DE425 EISA, DE434, DE435, DE500...41
DEC 21040, 21041, 2114x, Tulip...42
4.17 Farallon...42
Farallon Etherwave...42
Farallon PCI 593...43
4.18 Fujitsu...43
Fujitsu FMV−181/182/183/184...43
4.19 Hewlett Packard...43
HP Night Director+ 10/100...43
27245A...43
HP EtherTwist, PC Lan+ (27247, 27252A)...44
HP−J2405A...44
HP−Vectra On Board Ethernet...44

Table of Contents

HP 10/100 VG Any Lan Cards (27248B, J2573, J2577, J2585, J970, J973)....44
HP NetServer 10/100TX PCI (D5013A)....44
4.20 IBM / International Business Machines....45
IBM Thinkpad 300....45
IBM Credit Card Adaptor for Ethernet....45
IBM 10/100 EtherJet PCI....45
IBM Token Ring....45
4.21 ICL Ethernet Cards....45
ICL EtherTeam 16i/32....45
4.22 Intel Ethernet Cards....46
Ether Express....46
Ether Express PRO/10 (PRO/10+)....46
Ether Express PRO/10 PCI (EISA)....46
Ether Express PRO 10/100B....46
4.23 Kingston....47
4.24 LinkSys....47
LinkSys Etherfast 10/100 Cards....47
LinkSys Pocket Ethernet Adapter Plus (PEAEPP)....47
LinkSys PCMCIA Adaptor....47
4.25 Microdyne (Eagle)....48
Microdyne Exos 205T....48
4.26 Mylex....48
Mylex LNE390A, LNE390B....48
Mylex LNP101....48
Mylex LNP104....48
4.27 Novell Ethernet, NExxxx and associated clones....49
NE1000, NE2000....49
NE2000−PCI (RealTek/Winbond/Compex)....49
NE−10/100....50
NE1500, NE2100....50
NE/2 MCA....50
NE3200....50
NE3210....51
NE5500....51
4.28 Proteon....51
Proteon P1370−EA....51
Proteon P1670−EA....51
4.29 Pure Data....51
PDUC8028, PDI8023....51
4.30 Racal−Interlan....51
ES3210....52
NI5010....52
NI5210....52
NI6510 (not EB)....52
EtherBlaster (aka NI6510EB)....52
4.31 RealTek....53
RealTek RTL8002/8012 (AT−Lan−Tec) Pocket adaptor....53

Table of Contents

RealTek 8009.....53
RealTek 8019.....53
RealTek 8029.....53
RealTek 8129/8139.....53
4.32 Sager.....54
Sager NP943.....54
4.33 Schneider & Koch.....54
SK G16.....54
4.34 SEEQ.....54
SEEQ 8005.....54
4.35 SMC (Standard Microsystems Corp.).....54
WD8003, SMC Elite.....55
WD8013, SMC Elite16.....55
SMC Elite Ultra.....55
SMC Elite Ultra32 EISA.....56
SMC EtherEZ (8416).....56
SMC EtherPower PCI (8432).....56
SMC EtherPower II PCI (9432).....57
SMC 1211TX 10/100.....57
SMC 3008.....57
SMC 3016.....57
SMC−9000 / SMC 91c92/4.....58
SMC 91c100.....58
4.36 Texas Instruments.....58
ThunderLAN.....58
4.37 Thomas Conrad.....58
Thomas Conrad TC−5048.....58
4.38 VIA.....58
VIA 86C926 Amazon.....58
VIA 86C100A Rhine II (and 3043 Rhine I).....59
4.39 Western Digital.....59
4.40 Winbond.....59
Winbond 89c840.....59
Winbond 89c904, 89c905, 89c906.....59
Winbond 89c940.....59
4.41 Xircom.....60
Xircom PE1, PE2, PE3−10B*.....60
Xircom PCMCIA Cards.....60
4.42 Zenith.....60
Z−Note.....60
4.43 Znyx.....60
Znyx ZX342 (DEC 21040 based).....60
4.44 Identifying an Unknown Card.....61
Identifying the Network Interface Controller.....61
Identifying the Ethernet Address.....61
Identifying the Card by the FCC ID Number.....62
Tips on Trying to Use an Unknown Card.....62

Table of Contents

4.45 Drivers for Non−Ethernet Devices....62
5. Cables, Coax, Twisted Pair....63
5.1 Thin Ethernet (thinnet)....63
5.2 Twisted Pair....64
5.3 Thick Ethernet....65
6. Software Configuration and Card Diagnostics....65
6.1 Configuration Programs for Ethernet Cards....65
WD80x3 Cards....65
Digital / DEC Cards....66
NE2000+ or AT/LANTIC Cards....66
3Com Cards....66
6.2 Diagnostic Programs for Ethernet Cards....67
7. Technical Information....67
7.1 Programmed I/O vs. Shared Memory vs. DMA....67
Programmed I/O (e.g. NE2000, 3c509)....67
Shared memory (e.g. WD80x3, SMC−Ultra, 3c503)....67
Slave (normal) Direct Memory Access (e.g. none for Linux!)....68
Bus Master Direct Memory Access (e.g. LANCE, DEC 21040)....68
7.2 Performance Implications of Bus Width....68
ISA Eight bit vs ISA 16 bit Cards....68
7.3 32 Bit (VLB/EISA/PCI) Ethernet Cards....69
7.4 Writing a Driver....69
7.5 Driver interface to the kernel....69
Probe....70
Interrupt handler....70
Transmit function....70
Receive function....70
Open function....70
Close function (optional)....70
Miscellaneous functions....70
7.6 Technical information from 3Com....71
7.7 Notes on AMD PCnet / LANCE Based cards....71
7.8 Multicast and Promiscuous Mode....72
7.9 The Berkeley Packet Filter (BPF)....72
8. Networking with a Laptop/Notebook Computer....73
8.1 Using SLIP....73
8.2 PCMCIA Support....73
8.3 ISA Ethercard in the Docking Station....74
8.4 Pocket / parallel port adaptors....74
9. Miscellaneous....74
9.1 Passing Ethernet Arguments to the Kernel....74
The ether command....74
The reserve command....75
9.2 Using the Ethernet Drivers as Modules....76
9.3 Related Documentation....77
9.4 Disclaimer and Copyright....77
9.5 Closing....78

Linux Ethernet−Howto

by Paul Gortmaker

v2.8, Oct 29, 2000

This is the Ethernet−Howto, which is a compilation of information about which ethernet devices can be used for Linux, and how to set them up. Note that this Howto is focused on the hardware and low level driver aspect of the ethernet cards, and does not cover the software end of things like `ifconfig` and `route`. See the Network Howto for that stuff.

---

1. Introduction

- 1.1 New Versions of this Document
- 1.2 Using the Ethernet−Howto
- 1.3 HELP − It doesn't work!
- 1.4 Type of cable that your card should support

2. Frequently Asked Questions

- 2.1 What card should I buy for Linux?
- 2.2 Alpha Drivers −− Getting and Using them
- 2.3 Using More than one Ethernet Card per Machine
- 2.4 The `ether=` thing didn't do anything for me. Why?
- 2.5 Problems with NE1000 / NE2000 cards (and clones)
- 2.6 Problems with SMC Ultra/EtherEZ and WD80*3 cards
- 2.7 Problems with 3Com cards
- 2.8 FAQs Not Specific to Any Card.

3. Performance Tips

- 3.1 General Concepts
- 3.2 ISA Cards and ISA Bus Speed
- 3.3 Setting the TCP Rx Window
- 3.4 Increasing NFS performance

4. Vendor/Manufacturer/Model Specific Information

- 4.1 3Com
- 4.2 Accton
- 4.3 Allied Telesyn/Telesis
- 4.4 AMD / Advanced Micro Devices
- 4.5 Ansel Communications
- 4.6 Apricot
- 4.7 Arcnet

- 4.8 AT&T
- 4.9 Boca Research
- 4.10 Cabletron
- 4.11 Cogent
- 4.12 Compaq
- 4.13 Danpex
- 4.14 D−Link
- 4.15 DFI
- 4.16 Digital / DEC
- 4.17 Farallon
- 4.18 Fujitsu
- 4.19 Hewlett Packard
- 4.20 IBM / International Business Machines
- 4.21 ICL Ethernet Cards
- 4.22 Intel Ethernet Cards
- 4.23 Kingston
- 4.24 LinkSys
- 4.25 Microdyne (Eagle)
- 4.26 Mylex
- 4.27 Novell Ethernet, NExxxx and associated clones.
- 4.28 Proteon
- 4.29 Pure Data
- 4.30 Racal−Interlan
- 4.31 RealTek
- 4.32 Sager
- 4.33 Schneider & Koch
- 4.34 SEEQ
- 4.35 SMC (Standard Microsystems Corp.)
- 4.36 Texas Instruments
- 4.37 Thomas Conrad
- 4.38 VIA
- 4.39 Western Digital
- 4.40 Winbond
- 4.41 Xircom
- 4.42 Zenith
- 4.43 Znyx
- 4.44 Identifying an Unknown Card
- 4.45 Drivers for Non−Ethernet Devices

5. Cables, Coax, Twisted Pair

- 5.1 Thin Ethernet (thinnet)
- 5.2 Twisted Pair
- 5.3 Thick Ethernet

6. Software Configuration and Card Diagnostics

- 6.1 Configuration Programs for Ethernet Cards
- 6.2 Diagnostic Programs for Ethernet Cards

7. Technical Information

- 7.1 Programmed I/O vs. Shared Memory vs. DMA
- 7.2 Performance Implications of Bus Width
- 7.3 32 Bit (VLB/EISA/PCI) Ethernet Cards
- 7.4 Writing a Driver
- 7.5 Driver interface to the kernel
- 7.6 Technical information from 3Com
- 7.7 Notes on AMD PCnet / LANCE Based cards
- 7.8 Multicast and Promiscuous Mode
- 7.9 The Berkeley Packet Filter (BPF)

8. Networking with a Laptop/Notebook Computer

- 8.1 Using SLIP
- 8.2 PCMCIA Support
- 8.3 ISA Ethercard in the Docking Station.
- 8.4 Pocket / parallel port adaptors.

9. Miscellaneous.

- 9.1 Passing Ethernet Arguments to the Kernel
- 9.2 Using the Ethernet Drivers as Modules
- 9.3 Related Documentation
- 9.4 Disclaimer and Copyright
- 9.5 Closing

---

1. Introduction

The Ethernet−Howto covers what cards you should and shouldn't buy; how to set them up, how to run more than one, and other common problems and questions. It contains detailed information on the current level of support for all of the most common ethernet cards available.

It does *not* cover the software end of things, as that is covered in the NET−3 Howto. Also note that general non−Linux specific questions about Ethernet are not (or at least they should not be) answered here. For those types of questions, see the excellent amount of information in the *comp.dcom.lans.ethernet* FAQ. You can FTP it from `rtfm.mit.edu` just like all the other newsgroup FAQs.

This present revision covers distribution kernels up to and including 2.2.17.

The Ethernet−Howto is by:

> Paul Gortmaker, `p_gortmaker@yahoo.com`

The primary source of information for the initial ASCII−only version of the Ethernet−Howto was:

> Donald J. Becker, `becker@scyld.com`

who we should thank for writing a lot of the ethernet card drivers that are presently available for Linux. He also is the author of the original NFS server too. Thanks Donald!

This document is Copyright (c) 1993−2000 by Paul Gortmaker. Please see the Disclaimer and Copying information at the end of this document (copyright) for information about redistribution of this document and the usual `we are not responsible for what you manage to break...' type legal stuff.

1.1 New Versions of this Document

New versions of this document can be retrieved from:

Ethernet−HOWTO

or for those wishing to use FTP and/or get non−HTML formats:

Sunsite HOWTO Archive

This is the `official' location − it can also be found on various Linux WWW/ftp mirror sites. Updates will be made as new information and/or drivers becomes available. If this copy that you are reading is more than 6 months old, then you should check to see if an updated copy is available.

This document is available in various formats (postscript, dvi, ASCII, HTML, etc.). I would recommend viewing it in HTML (via a WWW browser) or the Postscript/dvi format. Both of these contain cross−references that are not included in the plain text ASCII format.

1.2 Using the Ethernet−Howto

As this guide is getting bigger and bigger, you probably don't want to spend the rest of your afternoon reading the whole thing. And the good news is that you don't *have* to read it all. The HTML and Postscript/dvi versions have a table of contents which will really help you find what you need a lot faster.

Chances are you are reading this document beacuse you can't get things to work and you don't know what to do or check. The next section (HELP − It doesn't work!) is aimed at newcomers to linux and will point you in the right direction.

Typically the same problems and questions are asked *over and over* again by different people. Chances are your specific problem or question is one of these Frequently Asked Questions, and is answered in the FAQ portion of this document . (The FAQ section). Everybody should have a look through this section before posting for help.

If you haven't got an ethernet card, then you will want to start with deciding on a card. (What card should I buy...)

If you have already got an ethernet card, but are not sure if you can use it with Linux, then you will want to read the section which contains specific information on each manufacturer, and their cards. (Vendor Specific...)

If you are interested in some of the technical aspects of the Linux device drivers, then you can have a browse of the section with this type of information. (Technical Information)

1.3 HELP − It doesn't work!

Okay, don't panic. This will lead you through the process of getting things working, even if you have no prior background in linux or ethernet hardware.

First thing you need to do is figure out what model your card is so you can determine if Linux has a driver for that particular card. Different cards typically have different ways of being controlled by the host computer, and the linux driver (if there is one) contains this control information in a format that allows linux to use the card. If you don't have any manuals or anything of the sort that tell you anything about the card model, then you can try the section on helping with mystery cards (reference section: Identifying an Unknown Card).

Now that you know what type of card you have, read through the details of your particular card in the card specific section (reference section: Vendor Specific...) which lists in alphabetical order, card manufacturers, individual model numbers and whether it has a linux driver or not. If it lists it as `Not Supported' you can pretty much give up here. If you can't find your card in that list, then check to see if your card manual lists it as being `compatible' with another known card type. For example there are hundreds, if not thousands of different cards made to be compatible with the original Novell NE2000 design.

Assuming you have found out that a linux driver exists for your card, you now have to find it and make use of it. Just because linux has a driver for your card does *not* mean that it is built into every kernel. (The kernel is the core operating system that is first loaded at boot, and contains drivers for various pieces of hardware, among other things.) Depending on who made the particular linux distribution you are using, there may be only a few pre−built kernels, and a whole bunch of drivers as smaller separate modules, or there may be a whole lot of kernels, covering a vast combination of built−in driver combinations.

Most linux distributions now ship with a bunch of small modules that are the various drivers. The required modules are typically loaded late in the boot process, or on−demand as a driver is needed to access a particualr device. You will need to attach this module to the kernel after it has booted up. See the information that came with your distribution on installing and using modules, along with the module section in this document. (Using the Ethernet Drivers as Modules)

If you didn't find either a pre−built kernel with your driver, or a module form of the driver, chances are you have a typically uncommon card, and you will have to build your own kernel with that driver included. Once you have linux installed, building a custom kernel is not difficult at all. You essentially answer yes or no to what you want the kernel to contain, and then tell it to build it. There is a Kernel−HowTo that will help you along.

At this point you should have somehow managed to be booting a kernel with your driver built in, or be loading it as a module. About half of the problems people have are related to not having driver loaded one way or another, so you may find things work now.

If it still doesn't work, then you need to verify that the kernel is indeed detecting the card. To do this, you need to type `dmesg | more` when logged in after the system has booted and all modules have been loaded. This will allow you to review the boot messages that the kernel scrolled up the screen during the boot process. If the card has been detected, you should see somewhere in that list a message from your card's driver that starts with `eth0`, mentions the driver name and the hardware parameters (interrupt setting, input/output port address, etc) that the card is set for. (Note: At boot, linux lists all the PCI cards installed in the system, regardless of what drivers are available − do not mistake this for the driver detection which comes later!)

If you don't see a driver indentification message like this, then the driver didn't detect your card, and that is why things aren't working. See the FAQ (The FAQ Section) for what to do if your card is not detected. If you have a NE2000 compatible, there is also some NE2000 specific tips on getting a card detected in the FAQ section as well.

If the card is detected, but the detection message reports some sort of error, like a resource conflict, then the driver probably won't have initialized properly and the card still wont be useable. Most common error messages of this sort are also listed in the FAQ section, along with a solution.

If the detection message seems okay, then double check the card resources reported by the driver against those that the card is physically set for (either by little black jumpers on the card, or by a software utility supplied by the card manufacturer.) These must match exactly. For example, if you have the card jumpered or configured to IRQ 15 and the driver reports IRQ 10 in the boot messages, things will not work. The FAQ section discusses the most common cases of drivers incorrectly detecting the configuration information of various cards.

At this point, you have managed to get you card detected with all the correct parameters, and hopefully everything is working. If not, then you either have a software configuration error, or a hardware configuration error. A software configuration error is not setting up the right network addresses for the `ifconfig` and `route` commands, and details of how to do that are fully described in the Network HowTo and the `Network Administrator's Guide' which both probably came on the CD−ROM you installed from.

A hardware configuration error is when some sort of resource conflict or mis−configuration (that the driver didn't detect at boot) stops the card from working properly. This typically can be observed in several different ways. (1) You get an error message when `ifconfig` tries to open the device for use, such as ``SIOCSFFLAGS: Try again''. (2) The driver reports `eth0` error messages (viewed by `dmesg | more`) or strange inconsistencies for each time it tries to send or receive data. (3) Typing `cat /proc/net/dev` shows non−zero numbers in one of the errs, drop, fifo, frame or carrier columns for `eth0`. (4) Typing `cat /proc/interrupts` shows a zero interrupt count for the card. Most of the typical hardware configuration errors are also discussed in the FAQ section.

Well, if you have got to this point and things still aren't working, read the FAQ section of this document, read the vendor specific section detailing your particular card, *and if it still doesn't work* then you may have to resort to posting to an appropriate newsgroup for help. If you do post, please detail all relevant information in that post, such as the card brand, the kernel version, the driver boot messages, the output from `cat /proc/net/dev`, a clear description of the problem, and of course what you have already tried to do in an effort to get things to work.

You would be surprised at how many people post useless things like ``Can someone help me? My ethernet doesn't work.'' and nothing else. Readers of the newsgroups tend to ignore such silly posts, whereas a detailed and informational problem description may allow a `linux−guru' to spot your problem right away. Of course the same holds true when e−mailing a problem report − always provide as much information as possible.

1.4 Type of cable that your card should support

The twisted pair cables, with the RJ−45 (giant phone jack) connectors is technically called 10BaseT. You may also hear it called UTP (Unsheilded Twisted Pair).

The thinnet, or thin ethernet cabling, (RG−58 coaxial cable) with the BNC (metal push and turn−to−lock) connectors is technically called 10Base2.

The older thick ethernet (10mm coaxial cable) which is only found in older installations is called 10Base5. The 15 pin D−shaped plug found on some ethernet cards (the AUI connector) is used to connect to thick ethernet and external transcievers.

Most ethercards also come in a `Combo' version for only $10−$20 more. These have both twisted pair and thinnet transceiver built−in, allowing you to change your mind later.

Most installations will use 10BaseT/100BaseT 10Base2 does not offer any upgrade path to 100Base−whatever. 10Base2 is fine for hobbyists setting up a home network when purchasing a hub is not desireable for some reason or another.

See Cables, Coax... for other concerns with different types of ethernet cable.

---

2. Frequently Asked Questions

Here are some of the more frequently asked questions about using Linux with an Ethernet connection. Some of the more specific questions are sorted on a `per manufacturer basis'. Chances are the question you want an answer for has already been asked (and answered!) by someone else, so even if you don't find your answer here, you probably can find what you want from a news archive such as Dejanews.

2.1 What card should I buy for Linux?

The answer to this question depends heavily on exactly what you intend on doing with your net connection, and how much traffic it will see.

If you only expect a single user to be doing the occasional ftp session or WWW connection, then even an old 8 bit ISA card will probably keep you happy.

If you intend to set up a server, and you require the CPU overhead of moving data over the network to be kept to a minimum, you probably want to look at one of the PCI cards that uses a chip with bus−mastering capapbility, such as the DEC tulip (21xxx) chip, or the AMD PCnet−PCI chip.

If you fall somewhere in the middle of the above, then any one of the low cost PCI or 16 bit ISA cards with stable drivers will do the job for you.

2.2 Alpha Drivers −− Getting and Using them

I heard that there is an updated or preliminary/alpha driver available for my card. Where can I get it?

The newest of the `new' drivers can be found on Donald's ftp site: `www.scyld.com` in the `/pub/linux/` area. Things change here quite frequently, so just look around for it. Alternatively, it may be easier to use a WWW browser on:

Don's Linux Home Page

to locate the driver that you are looking for. (Watch out for WWW browsers that silently munge the source by replacing TABs with spaces and so on − use ftp, or at least an FTP URL for downloading if unsure.)

Now, if it really is an alpha, or pre−alpha driver, then please treat it as such. In other words, don't complain because you can't figure out what to do with it. If you can't figure out how to install it, then you probably shouldn't be testing it. Also, if it brings your machine down, don't complain. Instead, send us a well documented bug report, or even better, a patch!

Note that some of the `useable' experimental/alpha drivers have been included in the standard kernel source tree. When running `make config` one of the first things you will be asked is whether to ``Prompt for development and/or incomplete code/drivers''. You will have to answer `Y' here to get asked about including any alpha/experiemntal drivers.

2.3 Using More than one Ethernet Card per Machine

What needs to be done so that Linux can run two ethernet cards?

The answer to this question depends on whether the driver(s) is/are being used as a loadable module or are compiled directly into the kernel. Most linux distributions use modular drivers now. This saves distributing lots of kernels, each with a different driver set built in. Instead a single basic kernel is used and the individual drivers that are need for a particular user's system are loaded once the system has booted far enough to access the driver module files (usually stored in `/lib/modules/`).

With the Driver as a Module: In the case of PCI drivers, the module will typically detect all of the installed cards of that brand model automatically. However, for ISA cards, probing for a card is not a safe operation, and hence you typically need to supply the I/O base address of the card so the module knows where to look. This information is stored in the file `/etc/conf.modules`.

As an example, consider a user that has two ISA NE2000 cards, one at `0x300` and one at `0x240` and what lines they would have in their `/etc/conf.modules` file:

```
alias eth0 ne
alias eth1 ne
options ne io=0x240,0x300
```

What this does: This says that if the administrator (or the kernel) does a `modprobe eth0` or a `modprobe eth1` then the `ne.o` driver should be loaded for either `eth0` or `eth1`. Furthermore, when the `ne.o` module is loaded, it should be loaded with the options `io=0x240,0x300` so that the driver knows where to look for the cards. Note that the `0x` is important − things like `300h` as commonly used in the DOS world won't work. Switching the order of the `0x240` and the `0x300` will switch which physical card ends up as `eth0` and `eth1`.

Most of the ISA module drivers can take multiple comma separated I/O values like this example to handle multiple cards. However, some (older?) drivers, such as the 3c501.o module are currently only able to handle one card per module load. In this case you can load the module twice to get both cards detected. The `/etc/conf.modules` file in this case would look like:

```
alias eth0 3c501
alias eth1 3c501
options eth0 −o 3c501−0 io=0x280 irq=5
options eth1 −o 3c501−1 io=0x300 irq=7
```

In this example the `−o` option has been used to give each instance of the module a unique name, since you can't have two modules loaded with the same name. The `irq=` option has also been used to to specify the hardware IRQ setting of the card. (This method can also be used with modules that accept comma separated

I/O values, but it is less efficient since the module ends up being loaded twice when it doesn't really need to be.)

As a final example, consider a user with one 3c503 card at `0x350` and one SMC Elite16 (wd8013) card at `0x280`. They would have:

```
alias eth0 wd
alias eth1 3c503
options wd io=0x280
options 3c503 io=0x350
```

For PCI cards, you typically only need the `alias` lines to correlate the `ethN` interfaces with the appropriate driver name, since the I/O base of a PCI card can be safely detected.

The available modules are typically stored in `/lib/modules/`uname −r`/net` where the `uname −r` command gives the kernel version (e.g. 2.0.34). You can look in there to see which one matches your card. Once you have the correct settings in your `conf.modules` file, you can test things out with:

```
modprobe ethN
dmesg | tail
```

where `N' is the number of the ethernet interface you are testing.

With the Driver Compiled into the Kernel: If you have the driver compiled into the kernel, then the hooks for multiple ethercards are all there. However, note that at the moment only *one* ethercard is auto−probed for by default. This helps to avoid possible boot time hangs caused by probing sensitive cards.

(Note: As of late 2.1.x kernels, the boot probes have been sorted into safe and unsafe, so that all safe (e.g. PCI and EISA) probes will find all related cards automatically. Systems with more than one ethernet card with at least one of them being an ISA card will still need to do one of the following.)

There are two ways that you can enable auto−probing for the second (and third, and...) card. The easiest method is to pass boot−time arguments to the kernel, which is usually done by LILO. Probing for the second card can be achieved by using a boot−time argument as simple as `ether=0,0,eth1`. In this case `eth0` and `eth1` will be assigned in the order that the cards are found at boot. Say if you want the card at `0x300` to be `eth0` and the card at `0x280` to be `eth1` then you could use

```
LILO: linux ether=5,0x300,eth0 ether=15,0x280,eth1
```

The `ether=` command accepts more than the IRQ + I/O + name shown above. Please have a look at Passing Ethernet Arguments... for the full syntax, card specific parameters, and LILO tips.

These boot time arguments can be made permanent so that you don't have to re−enter them every time. See the LILO configuration option ``append`' in the LILO manual.

The second way (not recommended) is to edit the file `Space.c` and replace the `0xffe0` entry for the I/O address with a zero. The `0xffe0` entry tells it not to probe for that device −− replacing it with a zero will enable autoprobing for that device.

2.4 The `ether=` thing didn't do anything for me. Why?

As described above, the `ether=` command *only* works for drivers that are compiled into the kernel. Now most distributions use the drivers in a modular form, and so the `ether=` command is rarely used anymore. (Some older documentation has yet to be updated to reflect this change.) If you want to apply options for a modular ethernet driver you *must* make changes to the `/etc/conf.modules` file.

If you *are* using a compiled in driver, and have added an `ether=` to your LILO configuration file, note that it won't take effect until you re−run `lilo` to process the updated configuration file.

2.5 Problems with NE1000 / NE2000 cards (and clones)

Problem: PCI NE2000 clone card is not detected at boot with v2.0.x.

Reason: The `ne.c` driver up to v2.0.30 only knows about the PCI ID number of RealTek 8029 based clone cards. Since then, several others have also released PCI NE2000 clone cards, with different PCI ID numbers, and hence the driver doesn't detect them.

Solution: The easiest solution is to upgrade to a v2.0.31 (or newer) version of the linux kernel. It knows the ID numbers of about five different NE2000−PCI chips, and will detect them automatically at boot or at module loading time. If you upgrade to 2.0.34 (or newer) there is a PCI−only specific NE2000 driver that is slightly smaller and more efficient than the original ISA/PCI driver.

Problem: PCI NE2000 clone card is reported as an ne1000 (8 bit card!) at boot or when I load the ne.o module for v2.0.x, and hence doesn't work.

Reason: Some PCI clones don't implement byte wide access (and hence are not truly 100% NE2000 compatible). This causes the probe to think they are NE1000 cards.

Solution: You need to upgrade to v2.0.31 (or newer) as described above. The driver(s) now check for this hardware bug.

Problem: PCI NE2000 card gets terrible performance, even when reducing the window size as described in the Performance Tips section.

Reason: The spec sheets for the original 8390 chip, desgined and sold over ten years ago, noted that a dummy read from the chip was required before each write operation for maximum reliablity. The driver has the facility to do this but it has been disabled by default since the v1.2 kernel days. One user has reported that re−enabling this `mis−feature' helped their performance with a cheap PCI NE2000 clone card.

Solution: Since it has only been reported as a solution by one person, don't get your hopes up. Re−enabling the read before write fix is done by simply editing the driver file in `linux/drivers/net/`, uncommenting the line containing `NE_RW_BUGFIX` and then rebuilding the kernel or module as appropriate. Please send an e−mail describing the performance difference and type of card/chip you have if this helps you. (The same can be done for the `ne2k-pci.c` driver as well).

Problem: The `ne2k-pci.c` driver reports error messages like `timeout waiting for Tx RDC` with a PCI NE2000 card and doesn't work right.

Reason: Your card and/or the card to PCI bus link can't handle the long word I/O optimization used in this driver.

Solution: Firstly, check the settings available in the BIOS/CMOS setup to see if any related to PCI bus timing are too aggressive for reliable operation. Otherwise using the ISA/PCI `ne.c` driver (or removing the `#define USE_LONGIO` from `ne2k-pci.c`) should let you use the card.

Probem: ISA Plug and Play NE2000 (such as RealTek 8019) is not detected.

Reason: The original NE2000 specification (and hence the linux NE2000 driver) does not have support for Plug and Play.

Solution: Use the DOS configuration disk that came with the card to disable PnP, and to set the card to a specified I/O address and IRQ. Add a line to `/etc/conf.modules` like `options ne io=0xNNN` where `0xNNN` is the hex I/O address you set the card to. (This assumes you are using a modular driver; if not then use an `ether=0,0xNNN,eth0` argument at boot). You may also have to enter the BIOS/CMOS setup and mark the IRQ as Legacy−ISA instead of PnP. Alternatively, if you need to leave PnP enabled for compatibility with some other operating system, then look into the *isapnptools* package. Try `man isapnp` to see if it is already installed on your system. If not, then have a look at the following URL:

ISA PNP Tools

Problem: NE*000 driver reports `not found (no reset ack)' during boot probe.

Reason: This is related to the above change. After the initial verification that an 8390 is at the probed I/O address, the reset is performed. When the card has completed the reset, it is supposed to acknowedge that the reset has completed. Your card doesn't, and so the driver assumes that no NE card is present.

Solution: You can tell the driver that you have a bad card by using an otherwise unused `mem_end` hexidecimal value of `0xbad` at boot time. You *have* to also supply a non−zero I/O base for the card when using the `0xbad` override. For example, a card that is at `0x340` that doesn't ack the reset would use something like:

```
LILO: linux ether=0,0x340,0,0xbad,eth0
```

This will allow the card detection to continue, even if your card doesn't ACK the reset. If you are using the driver as a module, then you can supply the option `bad=0xbad` just like you supply the I/O address.

Problem: NE*000 card hangs machine at first network access.

Reason: This problem has been reported for kernels as old as 1.1.57 to the present. It appears confined to a few software configurable clone cards. It appears that they expect to be initialized in some special way.

Solution: Several people have reported that running the supplied DOS software config program and/or the supplied DOS driver prior to warm booting (i.e. loadlin or the `three−finger−salute') into linux allowed the card to work. This would indicate that these cards need to be initialized in a particular fashion, slightly different than what the present Linux driver does.

Problem: NE*000 ethercard at `0x360` doesn't get detected.

Reason: Your NE2000 card is `0x20` wide in I/O space, which makes it hit the parallel port at `0x378`. Other devices that could be there are the second floppy controller (if equipped) at `0x370` and the secondary IDE controller at `0x376--0x377`. If the port(s) are already registered by another driver, the kernel will not let the probe happen.

Solution: Either move your card to an address like `0x280, 0x340, 0x320` or compile without parallel printer support.

Problem: Network `goes away' every time I print something (NE2000)

Reason: Same problem as above, but you have an older kernel that doesn't check for overlapping I/O regions. Use the same fix as above, and get a new kernel while you are at it.

Problem: NE*000 ethercard probe at 0xNNN: 00 00 C5 ... not found. (invalid signature yy zz)

Reason: First off, do you have a NE1000 or NE2000 card at the addr. 0xNNN? And if so, does the hardware address reported look like a valid one? If so, then you have a poor NE*000 clone. All NE*000 clones are supposed to have the value `0x57` in bytes 14 and 15 of the SA PROM on the card. Yours doesn't −− it has `yy zz' instead.

Solution: There are two ways to get around this. The easiest is to use an `0xbad` mem_end value as described above for the `no reset ack' problem. This will bypass the signature check, as long as a non−zero I/O base is also given. This way no recompilation of the kernel is required.

The second method (for hackers) involves changing the driver itself, and then recompiling your kernel (or module). The driver (/usr/src/linux/drivers/net/ne.c) has a "Hall of Shame" list at about line 42. This list is used to detect poor clones. For example, the DFI cards use `DFI' in the first 3 bytes of the PROM, instead of using `0x57` in bytes 14 and 15, like they are supposed to.

Problem: The machine hangs during boot right after the `8390...' or `WD....' message. Removing the NE2000 fixes the problem.

Solution: Change your NE2000 base address to something like `0x340`. Alternatively, you can use the ``reserve='' boot argument in conjunction with the ``ether='' argument to protect the card from other device driver probes.

Reason: Your NE2000 clone isn't a good enough clone. An active NE2000 is a bottomless pit that will trap any driver autoprobing in its space. Changing the NE2000 to a less−popular address will move it out of the way of other autoprobes, allowing your machine to boot.

Problem: The machine hangs during the SCSI probe at boot.

Reason: It's the same problem as above, change the ethercard's address, or use the reserve/ether boot arguments.

Problem: The machine hangs during the soundcard probe at boot.

Reason: No, that's really during the silent SCSI probe, and it's the same problem as above.

Problem: NE2000 not detected at boot – no boot messages at all

Solution: There is no `magic solution' as there can be a number of reasons why it wasn't detected. The following list should help you walk through the possible problems.

1) Build a new kernel with only the device drivers that you need. Verify that you are indeed booting the fresh kernel. Forgetting to run lilo, etc. can result in booting the old one. (Look closely at the build time/date reported at boot.) Sounds obvious, but we have all done it before. Make sure the driver is in fact included in the new kernel, by checking the `System.map` file for names like `ne_probe`.

2) Look at the boot messages carefully. Does it ever even mention doing a ne2k probe such as `NE*000 probe at 0xNNN: not found (blah blah)' or does it just fail silently. There is a big difference. Use `dmesg|more` to review the boot messages after logging in, or hit Shift−PgUp to scroll the screen up after the boot has completed and the login prompt appears.

3) After booting, do a `cat /proc/ioports` and verify that the full iospace that the card will require is vacant. If you are at `0x300` then the ne2k driver will ask for `0x300−0x31f`. If any other device driver has registered even one port anywhere in that range, the probe will not take place at that address and will silently continue to the next of the probed addresses. A common case is having the lp driver reserve `0x378` or the second IDE channel reserve `0x376` which stops the ne driver from probing `0x360−0x380`.

4) Same as above for `cat /proc/interrupts`. Make sure no other device has registered the interrupt that you set the ethercard for. In this case, the probe will happen, and the ether driver will complain loudly at boot about not being able to get the desired IRQ line.

5) If you are still stumped by the silent failure of the driver, then edit it and add some printk() to the probe. For example, with the ne2k you could add/remove lines (marked with a `+' or `−') in `linux/drivers/net/ne.c` like:

```
    int reg0 = inb_p(ioaddr);

+    printk("NE2k probe - now checking %x\n",ioaddr);
-    if (reg0 == 0xFF)
+    if (reg0 == 0xFF) {
+       printk("NE2k probe - got 0xFF (vacant I/O port)\n");
        return ENODEV;
+    }
```

Then it will output messages for each port address that it checks, and you will see if your card's address is being probed or not.

6) You can also get the ne2k diagnostic from Don's ftp site (mentioned in the howto as well) and see if it is able to detect your card after you have booted into linux. Use the `` `-p 0xNNN' `` option to tell it where to look for the card. (The default is `0x300` and it doesn't go looking elsewhere, unlike the boot−time probe.) The output from when it finds a card will look something like:

```
Checking the ethercard at 0x300.
  Register 0x0d (0x30d) is 00
  Passed initial NE2000 probe, value 00.
8390 registers: 0a 00 00 00 63 00 00 00 01 00 30 01 00 00 00 00
SA PROM  0: 00 00 00 00 c0 c0 b0 b0 05 05 65 65 05 05 20 20
SA PROM 0x10: 00 00 07 07 0d 0d 01 01 14 14 02 02 57 57 57 57

        NE2000 found at 0x300, using start page 0x40 and end page 0x80.
```

Your register values and PROM values will probably be different. Note that all the PROM values are doubled for a 16 bit card, and that the ethernet address (00:00:c0:b0:05:65) appears in the first row, and the double `0x57` signature appears at the end of the PROM.

The output from when there is no card installed at `0x300` will look something like this:

```
Checking the ethercard at 0x300.
  Register 0x0d (0x30d) is ff
  Failed initial NE2000 probe, value ff.
8390 registers: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
SA PROM       0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
SA PROM 0x10: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

 Invalid signature found, wordlength 2.
```

The `0xff` values arise because that is the value that is returned when one reads a vacant I/O port. If you happen to have some other hardware in the region that is probed, you may see some non `0xff` values as well.

7) Try warm booting into linux from a DOS boot floppy (via loadlin) after running the supplied DOS driver or config program. It may be doing some extra (i.e. non−standard) "magic" to initialize the card.

8) Try Russ Nelson's ne2000.com packet driver to see if even it can see your card −− if not, then things do not look good. Example:

```
A:> ne2000 0x60 10 0x300
```

The arguments are software interrupt vector, hardware IRQ, and I/O base. You can get it from any msdos archive in pktdrv11.zip −− The current version may be newer than 11.

2.6 Problems with SMC Ultra/EtherEZ and WD80*3 cards

Problem: You get messages such as the following:

```
eth0: bogus packet size: 65531, status=0xff, nxpg=0xff
```

Reason: There is a shared memory problem.

Solution: The most common reason for this is PCI machines that are not configured to map in ISA memory devices. Hence you end up reading the PC's RAM (all `0xff` values) instead of the RAM on the card that contains the data from the received packet.

Other typical problems that are easy to fix are board conflicts, having cache or `shadow ROM' enabled for that region, or running your ISA bus faster than 8Mhz. There are also a surprising number of memory failures on ethernet cards, so run a diagnostic program if you have one for your ethercard.

Problem: SMC EtherEZ doesn't work in non−shared memory (PIO) mode.

Reason: Older versions of the Ultra driver only supported the card in the shared memory mode of operation.

Solution: The driver in kernel version 2.0 and above also supports the programmed I/O mode of operation. Upgrade to v2.0 or newer.

Problem: Old wd8003 and/or jumper−settable wd8013 always get the IRQ wrong.

Reason: The old wd8003 cards and jumper−settable wd8013 clones don't have the EEPROM that the driver can read the IRQ setting from. If the driver can't read the IRQ, then it tries to auto−IRQ to find out what it is. And if auto−IRQ returns zero, then the driver just assigns IRQ 5 for an 8 bit card or IRQ 10 for a 16 bit card.

Solution: Avoid the auto−IRQ code, and tell the kernel what the IRQ that you have jumpered the card to in your module configuration file (or via a boot time argument for in−kernel drivers).

Problem: SMC Ultra card is detected as wd8013, but the IRQ and shared memory base is wrong.

Reason: The Ultra card looks a lot like a wd8013, and if the Ultra driver is not present in the kernel, the wd driver may mistake the ultra as a wd8013. The ultra probe comes before the wd probe, so this usually shouldn't happen. The ultra stores the IRQ and mem base in the EEPROM differently than a wd8013, hence the bogus values reported.

Solution: Recompile with only the drivers you need in the kernel. If you have a mix of wd and ultra cards in one machine, and are using modules, then load the ultra module first.

2.7 Problems with 3Com cards

Problem: The 3c503 picks IRQ N, but this is needed for some other device which needs IRQ N. (eg. CD ROM driver, modem, etc.) Can this be fixed without compiling this into the kernel?

Solution: The 3c503 driver probes for a free IRQ line in the order {5, 9/2, 3, 4}, and it should pick a line which isn't being used. The driver chooses when the card is `ifconfig`'ed into operation.

If you are using a modular driver, you can use module parameters to set various things, including the IRQ value.

The following selects IRQ9, base location `0x300`, <ignored value>, and if_port #1 (the external transceiver).

```
io=0x300 irq=9 xcvr=1
```

Alternately, if the driver is compiled into the kernel, you can set the same values at boot by passing parameters via LILO.

```
LILO: linux ether=9,0x300,0,1,eth0
```

The following selects IRQ3, probes for the base location, <ignored value>, and the default if_port #0 (the internal transceiver)

```
LILO: linux ether=3,0,0,0,eth0
```

Problem: 3c503: configured interrupt X invalid, will use autoIRQ.

Reason: The 3c503 card can only use one of IRQ{5, 2/9, 3, 4} (These are the only lines that are connected to the card.) If you pass in an IRQ value that is not in the above set, you will get the above message. Usually, specifying an interrupt value for the 3c503 is not necessary. The 3c503 will autoIRQ when it gets ifconfig'ed, and pick one of IRQ{5, 2/9, 3, 4}.

Solution: Use one of the valid IRQs listed above, or enable autoIRQ by not specifying the IRQ line at all.

Problem: The supplied 3c503 drivers don't use the AUI (thicknet) port. How does one choose it over the default thinnet port?

Solution: The 3c503 AUI port can be selected at boot−time for in−kernel drivers, and at module insertion for modular drivers. The selection is overloaded onto the low bit of the currently−unused dev−>rmem_start variable, so a boot−time parameter of:

```
LILO: linux ether=0,0,0,1,eth0
```

should work for in−kernel drivers.

To specify the AUI port when loading as a module, just append `xcvr=1` to the module options line along with your I/O and IRQ values.

2.8 FAQs Not Specific to Any Card.

Linux and ISA Plug and Play Ethernet Cards

For best results (and minimum aggravation) it is recommended that you use the (usually DOS) program that came with your card to disable the PnP mechanism and set it to a fixed I/O address and IRQ. Make sure the I/O address you use is probed by the driver at boot, or if using modules then supply the address as an `io=` option in `/etc/conf.modules`. You may also have to enter the BIOS/CMOS setup and mark the IRQ as Legacy−ISA instead of PnP (if your computer has this option).

Note that you typically don't need DOS installed to run a DOS based configuration program. You can usually just boot a DOS floppy disk and run them from the supplied floppy disk. You can also download OpenDOS and FreeDOS for free.

If you require PnP enabled for compatibility with some other operating system then you will have to use the isapnptools package with linux to configure the card(s) each time at boot. You will still have to make sure the I/O address chosen for the card is probed by the driver or supplied as an `io=` option.

Some systems have an `enable PnP OS' (or similar named) option in the BIOS/CMOS setup menu which will need to be disabled in nearly all cases or the cards won't work properly, or even be detected . Best described by one user who said `I don't know what it does behind the scenes, but it seems to be evil.'

Ethercard is Not Detected at Boot.

The usual reason for this is that people are using a kernel that does not have support for their particular card built in. For a modular kernel, it usually means that the required module has not been requested for loading, or that an I/O address needs to be specified as a module option.

If you are using a modular based kernel, such as those installed by most of the linux distributions, then try and use the configuration utility for the distribution to select the module for your card. For ISA cards, it is a good idea to determine the I/O address of the card and add it as an option (e.g. `io=0x340`) if the configuration utility asks for any options. If there is no configuration utility, then you will have to add the correct module name (and options) to `/etc/conf.modules` −− see `man modprobe` for more details.

If you are using a pre−compiled kernel that is part of a distribution set, then check the documentation to see which kernel you installed, and if it was built with support for your particular card. If it wasn't, then your options are to try and get one that has support for your card, or build your own.

It is usually wise to build your own kernel with only the drivers you need, as this cuts down on the kernel size (saving your precious RAM for applications!) and reduces the number of device probes that can upset sensitive hardware. Building a kernel is not as complicated as it sounds. You just have to answer yes or no to a bunch of questions about what drivers you want, and it does the rest.

The next main cause is having another device using part of the I/O space that your card needs. Most cards are 16 or 32 bytes wide in I/O space. If your card is set at `0x300` and 32 bytes wide, then the driver will ask for `0x300-0x31f`. If any other device driver has registered even one port anywhere in that range, the probe will not take place at that address and the driver will silently continue to the next of the probed addresses. So, after booting, do a `cat /proc/ioports` and verify that the full I/O space that the card will require is vacant.

Another problem is having your card jumpered to an I/O address that isn't probed by default. The list of probed addresses for each driver is easily found just after the text comments in the driver source. Even if the I/O setting of your card is not in the list of probed addresses, you can supply it at boot (for in−kernel drivers) with the `ether=` command as described in Passing Ethernet Arguments... Modular drivers can make use of the `io=` option in `/etc/conf.modules` to specify an address that isn't probed by default.

`ifconfig` reports the wrong I/O address for the card.

No it doesn't. You are just interpreting it incorrectly. This is *not* a bug, and the numbers reported are correct. It just happens that some 8390 based cards (wd80x3, smc−ultra, etc) have the actual 8390 chip living at an offset from the first assigned I/O port. This is the value stored in `dev->base_addr`, and is what `ifconfig` reports. If you want to see the full range of ports that your card uses, then try `cat /proc/ioports` which will give the numbers you expect.

PCI machine detects card but driver fails probe.

Some PCI BIOSes may not enable all PCI cards at power−up, especially if the BIOS option `PNP OS' is enabled. This mis−feature is to support the current release of Windows which still uses some real−mode drivers. Either disable this option, or try and upgrade to a newer driver which has the code to enable a disabled card.

Shared Memory ISA cards in PCI Machine do not work (`0xffff`)

This will usually show up as reads of lots of `0xffff` values. No shared memory cards of any type will work in a PCI machine unless you have the PCI ROM BIOS/CMOS SETUP configuration set properly. You have to set it to allow shared memory access from the ISA bus for the memory region that your card is trying to use. If you can't figure out which settings are applicable then ask your supplier or local computer guru. For AMI BIOS, there is usually a "Plug and Play" section where there will be an ``ISA Shared Memory Size'' and ``ISA Shared Memory Base'' settings. For cards like the wd8013 and SMC Ultra, change the size from the default of `Disabled' to 16kB, and change the base to the shared memory address of your card.

Card seems to send data but never receives anything.

Do a `cat /proc/interrupts`. A running total of the number of interrupt events your card generates will be in the list given from the above. If it is zero and/or doesn't increase when you try to use the card then there is probably a physical interrupt conflict with another device installed in the computer (regardless of whether or not the other device has a driver installed/available). Change the IRQ of one of the two devices to a free IRQ.

Asynchronous Transfer Mode (ATM) Support

Werner Almesberger has been working on ATM support for linux. He has been working with the Efficient Networks ENI155p board (Efficient Networks) and the Zeitnet ZN1221 board (Zeitnet).

Werner says that the driver for the ENI155p is rather stable, while the driver for the ZN1221 is presently unfinished.

Check the latest/updated status at the following URL:

Linux ATM Support

Gigabyte Ethernet Support

Is there any gigabyte ethernet support for Linux?

Yes, there are currently at least two. A driver for the Packet Engines G−NIC PCI Gigabit Ethernet adapter is available in the v2.0 and v2.2 kernels For more details, support, and driver updates, see:

`http://www.scyld.com/linux/drivers/yellowfin.html`

The `acenic.c` driver available in the v2.2 kernels can be used for the Alteon AceNIC Gigabit Ethernet card and other Tigon based cards such as the 3Com 3c985. The driver should also work on the NetGear GA620, however this has yet to be verified.

FDDI Support

Is there FDDI support for Linux?

Yes. Larry Stefani has written a driver for v2.0 with Digital's DEFEA (FDDI EISA) and DEFPA (FDDI PCI) cards. This was included into the v2.0.24 kernel. Currently no other cards are supported though.

Full Duplex Support

Will Full Duplex give me 20MBps? Does Linux support it?

Cameron Spitzer writes the following about full duplex 10Base−T cards: ``If you connect it to a full duplex switched hub, and your system is fast enough and not doing much else, it can keep the link busy in both directions. There is no such thing as full duplex 10BASE−2 or 10BASE−5 (thin and thick coax). Full Duplex works by disabling collision detection in the adapter. That's why you can't do it with coax; the LAN won't run that way. 10BASE−T (RJ45 interface) uses separate wires for send and receive, so it's possible to run both ways at the same time. The switching hub takes care of the collision problem. The signalling rate is 10

Mbps."

So as you can see, you still will only be able to receive or transmit at 10Mbps, and hence don't expect a 2x performance increase. As to whether it is supported or not, that depends on the card and possibly the driver. Some cards may do auto−negotiation, some may need driver support, and some may need the user to select an option in a card's EEPROM configuration. Only the serious/heavy user would notice the difference between the two modes anyway.

Ethernet Cards for Linux on SMP Machines

If you spent the extra money on a multi processor (MP) computer then buy a good ethernet card as well. For v2.0 kernels it wasn't really an issue, but it definitely is for v2.2. Most of the older non−intelligent (e.g. ISA bus PIO and shared memory design) cards were never designed with any consideration for use on a MP machine. The executive summary is to buy an intelligent modern design card and make sure the driver has been written (or updated) to handle MP operation. (The key words here are `modern design' − the PCI−NE2000's are just a 10+ year old design on a modern bus.) Looking for the text `spin_lock` in the driver source is a good indication that the driver has been written to deal with MP operation. The full details of why you should buy a good card for MP use (and what happens if you dont) follow.

In v2.0 kernels, only one processor was allowed `in kernel' (i.e. changing kernel data and/or running device drivers) at any given time. So from the point of view of the card (and the associated driver) nothing was different from uni processor (UP) operation and things just continued to work. (This was the easiest way to get a working MP version of Linux − one big lock around the whole kernel only allows one processor in at a time. This way you know that you won't have two processors trying to change the same thing at the same time!)

The downside to only allowing one processor in the kernel at a time was that you only got MP performance if the running programs were self contained and calculation intensive. If the programs did a lot of input/output (I/O) such as reading or writing data to disk or over a network, then all but one of the processors would be stalled waiting on their I/O requests to be completed while the one processor running in kernel frantically tries to run all the device drivers to fill the I/O requests. The kernel becomes the bottleneck and since there is only one processor running in the kernel, the performance of a MP machine in the heavy I/O, single−lock case quickly degrades close to that of a single processor machine.

Since this is clearly less than ideal (esp. for file/WWW servers, routers, etc.) the v2.2 kernels have finer grained locking − meaning that more than one processor can be in the kernel at a time. Instead of one big lock around the whole kernel, there are a lot of smaller locks protecting critical data from being manipulated by more than one processor at a time − e.g. one processor can be running the driver for the network card, while another processor is running the driver for the disk drive at the same time.

Okay, with that all in mind here are the snags: The finer locking means that you can have one processor trying to send data out through an ethernet driver while another processor tries to access the same driver/card to do something else (such as get the card statistics for `cat /proc/net/dev`). Oops − your card stats just got sent out over the wire, while you got data for your stats instead. Yes, the card got confused by being asked to do two (or more!) things at once, and chances are it crashed your machine in the process.

So, the driver that worked for UP is no longer good enough − it needs to be updated with locks that control access to the underlying card so that the three tasks of receive, transmit and manipulation of configuration data are serialized to the degree required by the card for stable operation. The scary part here is that a driver not yet updated with locks for stable MP operation will probably appear to be working in a MP machine under light network load, but will crash the machine or at least exhibit strange behaviour when two (or more!)

processors try to do more than one of these three tasks at the same time.

The updated MP aware ethernet driver will (at a minimum) require a lock around the driver that limits access at the entry points from the kernel into the driver to `one at a time please'. With this in place, things will be serialized so that the underlying hardware should be treated just as if it was being used in a UP machine, and so it should be stable. The downside is that the one lock around the whole ethernet driver has the same negative performance implications that having one big lock around the whole kernel had (but on a smaller scale) − i.e. you can only have one processor dealing with the card at a time. [Technical Note: The performance impact may also include increased interrupt latencies if the locks that need to be added are of the `irqsave` type and they are held for a long time.]

Possible improvements on this situation can be made in two ways. You can try to minimize the time between when the lock is taken and when it is released, and/or you can implement finer grained locking within the driver (e.g. a lock around the whole driver would be overkill if a lock or two protecting against simultaneous access to a couple of sensitive registers/settings on the card would suffice).

However, for older non−intelligent cards that were never designed with MP use in mind, neither of these improvements may be feasible. Worse yet is that the non−intelligent cards typically require the processor to move the data between the card and the computer memory, so in a worst case scenario the lock will be held the whole time that it takes to move each 1.5kB data packet over an ISA bus.

The more modern intelligent cards typically move network data directly to and from the computer memory without any help from a processor. This is a big win, since the lock is then only held for the short time it takes the processor to tell the card where in memory to get/store the next network data packet. More modern card designs are less apt to require a single big lock around the whole driver as well.

Ethernet Cards for Linux on Alpha/AXP PCI Boards

As of v2.0, only the 3c509, depca, de4x5, pcnet32, and all the 8390 drivers (wd, smc−ultra, ne, 3c503, etc.) have been made `architecture independent' so as to work on the DEC Alpha CPU based systems. Other updated PCI drivers from Donald's WWW page may also work as these have been written with architecture independence in mind.

Note that the changes that are required to make a driver architecture independent aren't that complicated. You only need to do the following:

−multiply all `jiffies` related values by HZ/100 to account for the different HZ value that the Alpha uses. (i.e `timeout=2;` becomes `timeout=2*HZ/100;`)

−replace any I/O memory (640k to 1MB) pointer dereferences with the appropriate readb() writeb() readl() writel() calls, as shown in this example.

```
-       int *mem_base = (int *)dev->mem_start;
-       mem_base[0] = 0xba5eba5e;
+       unsigned long mem_base = dev->mem_start;
+       writel(0xba5eba5e, mem_base);
```

−replace all memcpy() calls that have I/O memory as source or target destinations with the appropriate one of `memcpy_fromio()` or `memcpy_toio()`.

Details on handling memory accesses in an architecture independent fashion are documented in the file `linux/Documentation/IO−mapping.txt` that comes with recent kernels.

Ethernet for Linux on SUN/Sparc Hardware.

For the most up to date information on Sparc stuff, try the following URL:

Linux Sparc

Note that some Sparc ethernet hardware gets its MAC address from the host computer, and hence you can end up with multiple interfaces all with the same MAC address. If you need to put more than one interface on the same net then use the `hw` option to `ifconfig` to assign unique MAC address.

Issues regarding porting PCI drivers to the Sparc platform are similar to those mentioned above for the AXP platform. In addition there may be some endian issues, as the Sparc is big endian, and the AXP and ix86 are little endian.

Ethernet for Linux on Other Hardware.

There are several other hardware platforms that Linux can run on, such as Atari/Amiga (m68k). As in the Sparc case it is best to check with the home site of each Linux port to that platform to see what is currently supported. (Links to such sites are welcome here – send them in!)

Linking 10 or 100 BaseT without a Hub

Can I link 10/100BaseT (RJ45) based systems together without a hub?

You can link 2 machines easily, but no more than that, without extra devices/gizmos. See Twisted Pair −− it explains how to do it. And no, you can't hack together a hub just by crossing a few wires and stuff. It's pretty much impossible to do the collision signal right without duplicating a hub.

SIOCSIFxxx: No such device

I get a bunch of `SIOCSIFxxx: No such device' messages at boot, followed by a `SIOCADDRT: Network is unreachable' What is wrong?

Your ethernet device was not detected at boot/module insertion time, and when `ifconfig` and `route` are run, they have no device to work with. Use `dmesg | more` to review the boot messages and see if there are any messages about detecting an ethernet card.

SIOCSFFLAGS: Try again

I get `SIOCSFFLAGS: Try again' when I run `ifconfig' −− Huh?

Some other device has taken the IRQ that your ethercard is trying to use, and so the ethercard can't use the IRQ. You don't necessairly need to reboot to resolve this, as some devices only grab the IRQs when they need them and then release them when they are done. Examples are some sound cards, serial ports, floppy disk driver, etc. You can type `cat /proc/interrupts` to see which interrupts are presently *in use*. Most of the Linux ethercard drivers only grab the IRQ when they are opened for use via `ifconfig'. If you can get the other device to `let go' of the required IRQ line, then you should be able to `Try again' with ifconfig.

Using `ifconfig' and Link UNSPEC with HW−addr of 00:00:00:00:00:00

When I run ifconfig with no arguments, it reports that LINK is UNSPEC (instead of 10Mbs Ethernet) and it also says that my hardware address is all zeros.

This is because people are running a newer version of the `ifconfig' program than their kernel version. This new version of ifconfig is not able to report these properties when used in conjunction with an older kernel. You can either upgrade your kernel, `downgrade' ifconfig, or simply ignore it. The kernel knows your hardware address, so it really doesn't matter if ifconfig can't read it.

You may also get strange information if the `ifconfig` program you are using is a lot older than the kernel you are using.

Huge Number of RX and TX Errors

When I run ifconfig with no arguments, it reports that I have a huge error count in both rec'd and transmitted packets. It all seems to work ok −− What is wrong?

Look again. It says `RX packets` *big number* **PAUSE** `errors 0` **PAUSE** `dropped 0` **PAUSE** `overrun 0`. And the same for the `TX` column. Hence the big numbers you are seeing are the total number of packets that your machine has rec'd and transmitted. If you still find it confusing, try typing `cat /proc/net/dev` instead.

Entries in `/dev/` for Ethercards

I have /dev/eth0 as a link to /dev/xxx. Is this right?

Contrary to what you have heard, the files in /dev/* are not used. You can delete any `/dev/wd0`, `/dev/ne0` and similar entries.

Linux and ``trailers''

Should I disable trailers when I `ifconfig' my ethercard?

You can't disable trailers, and you shouldn't want to. `Trailers' are a hack to avoid data copying in the networking layers. The idea was to use a trivial fixed−size header of size `H', put the variable−size header info at the end of the packet, and allocate all packets `H' bytes before the start of a page. While it was a good idea, it turned out to not work well in practice. If someone suggests the use of `−trailers', note that it is the equivalent of sacrificial goats blood. It won't do anything to solve the problem, but if problem fixes itself then someone can claim deep magical knowledge.

Access to the raw Ethernet Device

How do I get access to the raw ethernet device in linux, without going through TCP/IP and friends?

```
        int s=socket(AF_INET,SOCK_PACKET,htons(ETH_P_ALL));
```

This gives you a socket receiving every protocol type. Do `recvfrom()` calls to it and it will fill the sockaddr with device type in sa_family and the device name in the sa_data array. I don't know who originally invented SOCK_PACKET for Linux (its been in for ages) but its superb stuff. You can use it to send stuff raw too via `sendto()` calls. You have to have root access to do either of course.

3. Performance Tips

Here are some tips that you can use if you are suffering from low ethernet throughput, or to gain a bit more speed on those ftp transfers.

The `ttcp.c` program is a good test for measuring raw throughput speed. Another common trick is to do a `ftp> get large_file /dev/null` where `large_file` is > 1MB and residing in the buffer cache on the Tx'ing machine. (Do the `get' at least twice, as the first time will be priming the buffer cache on the Tx'ing machine.) You want the file in the buffer cache because you are not interested in combining the file access speed from the disk into your measurement. Which is also why you send the incoming data to `/dev/null` instead of onto the disk.

3.1 General Concepts

Even an 8 bit card is able to receive back−to−back packets without any problems. The difficulty arises when the computer doesn't get the Rx'd packets off the card quick enough to make room for more incoming packets. If the computer does not quickly clear the card's memory of the packets already received, the card will have no place to put the new packet.

In this case the card either drops the new packet, or writes over top of a previously received packet. Either one seriously interrupts the smooth flow of traffic by causing/requesting re−transmissions and can seriously degrade performance by up to a factor of 5!

Cards with more onboard memory are able to ``buffer'' more packets, and thus can handle larger bursts of back−to−back packets without dropping packets. This in turn means that the card does not require as low a latency from the the host computer with respect to pulling the packets out of the buffer to avoid dropping packets.

Most 8 bit cards have an 8kB buffer, and most 16 bit cards have a 16kB buffer. Most Linux drivers will reserve 3kB of that buffer (for two Tx buffers), leaving only 5kB of receive space for an 8 bit card. This is room enough for only three full sized (1500 bytes) ethernet packets.

3.2 ISA Cards and ISA Bus Speed

As mentioned above, if the packets are removed from the card fast enough, then a drop/overrun condition won't occur even when the amount of Rx packet buffer memory is small. The factor that sets the rate at which packets are removed from the card to the computer's memory is the speed of the data path that joins the two −− that being the ISA bus speed. (If the CPU is a dog−slow 386sx−16, then this will also play a role.)

The recommended ISA bus clock is about 8MHz, but many motherboards and peripheral devices can be run at higher frequencies. The clock frequency for the ISA bus can usually be set in the CMOS setup, by selecting a divisor of the mainboard/CPU clock frequency. Some ISA and PCI/ISA mainboards may not have this option, and so you are stuck with the factory default.

For example, here are some receive speeds as measured by the TTCP program on a 40MHz 486, with an 8 bit WD8003EP card, for different ISA bus speeds.

```
        ISA Bus Speed (MHz)      Rx TTCP (kB/s)
        -------------------      --------------
        6.7                      740
        13.4                     970
        20.0                     1030
        26.7                     1075
```

You would be hard pressed to do better than 1075kB/s with *any* 10Mb/s ethernet card, using TCP/IP. However, don't expect every system to work at fast ISA bus speeds. Most systems will not function properly at speeds above 13MHz. (Also, some PCI systems have the ISA bus speed fixed at 8MHz, so that the end user does not have the option of increasing it.)

In addition to faster transfer speeds, one will usually also benefit from a reduction in CPU usage due to the shorter duration memory and I/O cycles. (Note that hard disks and video cards located on the ISA bus will also usually experience a performance increase from an increased ISA bus speed.)

Be sure to back up your data prior to experimenting with ISA bus speeds in excess of 8MHz, and thouroughly test that all ISA peripherals are operating properly after making any speed increases.

3.3 Setting the TCP Rx Window

Once again, cards with small amounts of onboard RAM and relatively slow data paths between the card and the computer's memory run into trouble. The default TCP Rx window setting is 32kB, which means that a fast computer on the same subnet as you can dump 32k of data on you without stopping to see if you received any of it okay.

Recent versions of the `route` command have the ability to set the size of this window on the fly. Usually it is only for the local net that this window must be reduced, as computers that are behind a couple of routers or gateways are `buffered' enough to not pose a problem. An example usage would be:

```
        route add <whatever> ... window <win_size>
```

where `win_size` is the size of the window you wish to use (in bytes). An 8 bit 3c503 card on an ISA bus operating at a speed of 8MHz or less would work well with a window size of about 4kB. Too large a window will cause overruns and dropped packets, and a drastic reduction in ethernet throughput. You can check the operating status by doing a `cat /proc/net/dev` which will display any dropped or overrun conditions that occurred.

3.4 Increasing NFS performance

Some people have found that using 8 bit cards in NFS clients causes poorer than expected performance, when using 8kB (native Sun) NFS packet size.

The possible reason for this could be due to the difference in on board buffer size between the 8 bit and the 16 bit cards. The maximum ethernet packet size is about 1500 bytes. Now that 8kB NFS packet will arrive as about 6 back to back maximum size ethernet packets. Both the 8 and 16 bit cards have no problem Rx'ing back to back packets. The problem arises when the machine doesn't remove the packets from the cards buffer in time, and the buffer overflows. The fact that 8 bit cards take an extra ISA bus cycle per transfer doesn't help either. What you *can* do if you have an 8 bit card is either set the NFS transfer size to 2kB (or even 1kB), or try increasing the ISA bus speed in order to get the card's buffer cleared out faster. I have found that an old WD8003E card at 8MHz (with no other system load) can keep up with a large receive at 2kB NFS size, but not at 4kB, where performance was degraded by a factor of three.

On the other hand, if the default mount option is to use 1kB size and you have at least a 16 bit ISA card, you may find a significant increase in going to 4kB (or even 8kB).

---

4. Vendor/Manufacturer/Model Specific Information

The following lists many cards in alphabetical order by vendor name and then product identifier. Beside each product ID, you will see either `Supported', `Semi−Supported' or `Not Supported'.

Supported means that a driver for that card exists, and many people are happily using it and it seems quite reliable.

Semi−Supported means that a driver exists, but at least one of the following descriptions is true: (1) The driver and/or hardware are buggy, which may cause poor performance, failing connections or even crashes. (2) The driver is new or the card is fairly uncommon, and hence the driver has seen very little use/testing and the driver author has had very little feedback. Obviously (2) is preferable to (1), and the individual description of the card/driver should make it clear which one holds true. In either case, you will probably have to answer `Y' when asked ``Prompt for development and/or incomplete code/drivers?'' when running `make config`.

Not Supported means there is not a driver currently available for that card. This could be due to a lack of interest in hardware that is rare/uncommon, or because the vendors won't release the hardware documentation required to write a driver.

Note that the difference between `Supported' and `Semi−Supported' is rather subjective, and is based on user feedback observed in newsgroup postings and mailing list messages. (After all, it is impossible for one person to test all drivers with all cards for each kernel version!!!) So be warned that you may find a card listed as semi−supported works perfectly for you (which is great), or that a card listed as supported gives you no end of troubles and problems (which is not so great).

After the status, the name of the driver given in the linux kernel is listed. This will also be the name of the driver module that would be used in the `alias eth0 driver_name` line that is found in the `/etc/conf.modules` module configuration file.

4.1 3Com

If you are not sure what your card is, but you think it is a 3Com card, you can probably figure it out from the assembly number. 3Com has a document `Identifying 3Com Adapters By Assembly Number' (ref 24500002) that would most likely clear things up. See Technical Information from 3Com for info on how to get documents from 3Com.

Also note that 3Com has a WWW/FTP site with various goodies: `www.3Com.com` that you may want to check out. They even have linux drivers for some of their cards there you may wish to test out. It has been reported that their drivers are not stable and/or unuseable on SMP and non ix86 based machines, so you may want to keep that in mind.

3c501

Status: Semi−Supported, Driver Name: 3c501

This obsolete stone−age 8 bit card is really too brain−damaged to use. Avoid it like the plague. Do not purchase this card, even as a joke. It's performance is horrible, and it breaks in many ways.

For those not yet convinced, the 3c501 can only do one thing at a time −− while you are removing one packet from the single−packet buffer it cannot receive another packet, nor can it receive a packet while loading a transmit packet. This was fine for a network between two 8088−based computers where processing each packet and replying took 10's of msecs, but modern networks send back−to−back packets for almost every transaction.

AutoIRQ works, DMA isn't used, the autoprobe only looks at `0x280` and `0x300`, and the debug level is set with the third boot−time argument.

Once again, the use of a 3c501 is *strongly discouraged*! Even more so with a IP multicast kernel, as you will grind to a halt while listening to *all* multicast packets. See the comments at the top of the source code for more details.

EtherLink II, 3c503, 3c503/16

Status: Supported, Driver Name: 3c503 (+8390)

The 3c503 does not have ``EEPROM setup'', so a diagnostic/setup program isn't needed before running the card with Linux. The shared memory address of the 3c503 is set using jumpers that are shared with the boot PROM address. This is confusing to people familiar with other ISA cards, where you always leave the jumper set to ``disable'' unless you have a boot PROM.

These cards should be about the same speed as the same bus width WD80x3, but turn out to be actually a bit slower. These shared−memory ethercards also have a programmed I/O mode that doesn't use the 8390 facilities (their engineers found too many bugs!) The Linux 3c503 driver can also work with the 3c503 in programmed−I/O mode, but this is slower and less reliable than shared memory mode. Also, programmed−I/O mode is not as well tested when updating the drivers. You shouldn't use the programmed−I/O mode unless you need it for MS−DOS compatibility.

The 3c503's IRQ line is set in software, with no hints from an EEPROM. Unlike the MS−DOS drivers, the Linux driver has capability to autoIRQ: it uses the first available IRQ line in {5,2/9,3,4}, selected each time the card is ifconfig'ed. (Older driver versions selected the IRQ at boot time.) The ioctl() call in `ifconfig' will return EAGAIN if no IRQ line is available at that time.

Some common problems that people have with the 503 are discussed in Problems with....

If you intend on using this driver as a loadable module you should probably see Using the Ethernet Drivers as Modules for module specific information.

Note that some old diskless 386 workstations have an on board 3c503 (made by 3Com and sold under different names, like `Bull') but the vendor ID is not a 3Com ID and so it won't be detected. More details can be found in the Etherboot package, which you will need anyways to boot these diskless boxes.

Etherlink Plus 3c505

Status: Semi−Supported, Driver Name: 3c505

These cards use the i82586 chip but are not that many of them about. It is included in the standard kernel, but it is classed as an alpha driver. See Alpha Drivers for important information on using alpha−test ethernet drivers with Linux.

There is also the file `/usr/src/linux/drivers/net/README.3c505` that you should read if you are going to use one of these cards. It contains various options that you can enable/disable.

Etherlink−16 3c507

Status: Semi−Supported, Driver Name: 3c507

This card uses one of the Intel chips, and the development of the driver is closely related to the development of the Intel Ether Express driver. The driver is included in the standard kernel release, but as an alpha driver. See Alpha Drivers for important information on using alpha−test ethernet drivers with Linux.

Etherlink III, 3c509 / 3c509B

Status: Supported, Driver Name: 3c509

This card is fairly inexpensive and has good performance for an ISA non−bus−master design. The drawbacks are that the original 3c509 requires very low interrupt latency. The 3c509B shouldn't suffer from the same problem, due to having a larger buffer. (See below.) These cards use PIO transfers, similar to a ne2000 card, and so a shared memory card such as a wd8013 will be more efficient in comparison.

The original 3c509 has a small packet buffer (4kB total, 2kB Rx, 2kB Tx), causing the driver to occasionally drop a packet if interrupts are masked for too long. To minimize this problem, you can try unmasking interrupts during IDE disk transfers (see `man hdparm`) and/or increasing your ISA bus speed so IDE transfers finish sooner.

The newer model 3c509B has 8kB on board, and the buffer can be split 4/4, 5/3 or 6/2 for Rx/Tx. This setting is changed with the DOS configuration utility, and is stored on the EEPROM. This should alleviate the above problem with the original 3c509.

3c509B users should use either the supplied DOS utility to disable the *plug and play* support, *and* to set the output media to what they require. The linux driver currently does *not* support the Autodetect media setting, so you *have* to select 10Base−T or 10Base−2 or AUI. Note that to turn off PnP entirely, you should do a `3C5X9CFG /PNP:DISABLE` and then follow that with a hard reset to ensure that it takes effect.

Some people ask about the ``Server or Workstation'' and ``Highest Modem Speed'' settings presented in the DOS configuration utility. Donald writes ``These are only hints to the drivers, and the Linux driver does not use these parameters: it always optimizes for high throughput rather than low latency (`Server'). Low latency was critically important for old, non−windowed, IPX throughput. To reduce the latency the MS−DOS driver for the 3c509 disables interrupts for some operations, blocking serial port interrupts. Thus the need for the `modem speed' setting. The Linux driver avoids the need to disable interrupts for long periods by operating only on whole packets e.g. by not starting to transmit a packet until it is completely transferred to the card."

Note that the ISA card detection uses a different method than most cards. Basically, you ask the cards to respond by sending data to an ID_PORT (port `0x100` to `0x1ff` on intervals of `0x10`). This detection method means that a particular card will *always* get detected first in a multiple ISA 3c509 configuration. The card with the lowest hardware ethernet address will *always* end up being `eth0`. This shouldn't matter to anyone, except for those people who want to assign a 6 byte hardware address to a particular interface. If you have multiple 3c509 cards, it is best to append `ether=0,0,ethN` commands without the I/O port specified (i.e. use I/O=zero) and allow the probe to sort out which card is first. Using a non−zero I/O value will ensure that it does not detect all your cards, so don't do it.

If this really bothers you, have a look at Donald's latest driver, as you may be able to use a `0x3c509` value in the unused mem address fields to order the detection to suit your needs.

3c515

Status: Supported, Driver Name: 3c515

This is 3Com's ISA 100Mbps offering, codenamed ``CorkScrew''. A relatively new driver from Donald for these cards is included in the v2.2 kernels. For the most up to date information, you should probably look on the Vortex page:

Vortex

3c523

Status: Semi−Supported, Driver Name: 3c523

This MCA bus card uses the i82586, and Chris Beauregard has modified the ni52 driver to work with these cards. The driver for it can be found in the v2.2 kernel source tree.

More details can be found on the MCA−Linux page at `http://glycerine.cetmm.uni.edu/mca/`

3c527

Status: Not Supported.

Yes, another MCA card. No, not too much interest in it. Better chances with the 3c529 if you are stuck with MCA.

3c529

Status: Supported, Driver Name: 3c509

This card actually uses the same chipset as the 3c509. Donald actually put hooks into the 3c509 driver to check for MCA cards after probing for EISA cards, and before probing for ISA cards, long before MCA support was added to the kernel. The required MCA probe code is included in the driver shipped with v2.2 kernels. More details can be found on the MCA−Linux page at:

http://glycerine.cetmm.uni.edu/mca/

3c562

Status: Supported, Driver Name: 3c589 (distributed separately)

This PCMCIA card is the combination of a 3c589B ethernet card with a modem. The modem appears as a standard modem to the end user. The only difficulty is getting the two separate linux drivers to share one interrupt. There are a couple of new registers and some hardware interrupt sharing support. You need to use a v2.0 or newer kernel that has the support for interrupt sharing.

Thanks again to Cameron for getting a sample unit and documentation sent off to David Hinds. Look for support in David's PCMCIA package release.

See PCMCIA Support for more info on PCMCIA chipsets, socket enablers, etc.

3c575

Status: Unknown.

A driver for this PCMCIA card is under development and hopefully will be included in David's PCMCIA package in the future. Best to check the PCMCIA package to get the current status.

3c579

Status: Supported, Driver Name: 3c509

The EISA version of the 509. The current EISA version uses the same 16 bit wide chip rather than a 32 bit interface, so the performance increase isn't stunning. Make sure the card is configured for EISA addressing mode. Read the above 3c509 section for info on the driver.

3c589 / 3c589B

Status: Semi−Supported, Driver Name: 3c589

Many people have been using this PCMCIA card for quite some time now. Note that support for it is not (at present) included in the default kernel source tree. The "B" in the name means the same here as it does for the 3c509 case.

There are drivers available on Donald's ftp site and in David Hinds PCMCIA package. You will also need a supported PCMCIA controller chipset. See PCMCIA Support for more info on PCMCIA drivers, chipsets, socket enablers, etc.

3c590 / 3c595

Status: Supported, Driver Name: 3c59x

These ``Vortex" cards are for PCI bus machines, with the '590 being 10Mbps and the '595 being 3Com's 100Mbs offering. Also note that you can run the '595 as a '590 (i.e. in a 10Mbps mode). The driver is included in the v2.0 kernel source, but is also continually being updated. If you have problems with the driver in the v2.0 kernel, you can get an updated driver from the following URL:

Vortex

Note that there are two different 3c590 cards out there, early models that had 32kB of on−board memory, and later models that only have 8kB of memory. Chances are you won't be able to buy a new 3c59x for much longer, as it is being replaced with the 3c90x card. If you are buying a used one off somebody, try and get the 32kB version. The 3c595 cards have 64kB, as you can't get away with only 8kB RAM at 100Mbps!

A thanks to Cameron Spitzer and Terry Murphy of 3Com for sending cards and documentation to Donald so he could write the driver.

3c592 / 3c597

Status: Supported, Driver Name: 3c59x

These are the EISA versions of the 3c59x series of cards. The 3c592/3c597 (aka Demon) should work with the vortex driver discussed above.

3c900 / 3c905 / 3c905B / 3c905C

Status: Supported, Driver Name: 3c59x

These cards (aka `Boomerang', aka EtherLink III XL) have been released to take over the place of the 3c590/3c595 cards.

The support for the Cyclone `B' revision was only recently added. To use this card with older v2.0 kernels, you must obtain the updated `3c59x.c` driver from Donald's site at:

Vortex−Page

3c985

Status: Supported, Driver Name: acenic

This driver, by Jes Sorensen, is available in v2.2 kernels It supports several other Gigabit cards in addition to the 3Com model.

4.2 Accton

Accton MPX

Status: Supported, Driver Name: ne (+8390)

Don't let the name fool you. This is still supposed to be a NE2000 compatible card, and should work with the ne2000 driver.

Accton EN1203, EN1207, EtherDuo−PCI

Status: Supported, Driver Name: de4x5, tulip, OR rtl8139

Apparently there have been several revisions of the EN1207 (A through D) with A, B, and C being tulip based and the D revision being RealTek 8139 based (different driver). So as with all purchases, you should try and make sure you can return it if it doesn't work for you.

Accton EN2209 Parallel Port Adaptor (EtherPocket)

Status: Semi−Supported, Driver Name: ?

A driver for these parallel port adapters is available but not yet part of the 2.0 or 2.1 kernel source. You have to get the driver from:

`http://www.unix-ag.uni-siegen.de/~nils/accton_linux.html`

Accton EN2212 PCMCIA Card

Status: Semi−Supported, Driver Name: ?

David Hinds has been working on a driver for this card, and you are best to check the latest release of his PCMCIA package to see what the present status is.

4.3 Allied Telesyn/Telesis

AT1500

Status: Supported, Driver Name: lance

These are a series of low−cost ethercards using the 79C960 version of the AMD LANCE. These are bus−master cards, and hence one of the faster ISA bus ethercards available.

DMA selection and chip numbering information can be found in AMD LANCE.

More technical information on AMD LANCE based Ethernet cards can be found in Notes on AMD....

AT1700

Status: Supported, Driver Name: at1700

Note that to access this driver during `make config` you still have to answer `Y' when asked ``Prompt for development and/or incomplete code/drivers?'' at the first. This is simply due to lack of feedback on the driver stability due to it being a relatively rare card. If you have problems with the driver that ships with the kernel then you may be interested in the alternative driver available at: `http://www.cc.hit-u.ac.jp/nagoya/at1700/`

The Allied Telesis AT1700 series ethercards are based on the Fujitsu MB86965. This chip uses a programmed I/O interface, and a pair of fixed−size transmit buffers. This allows small groups of packets to be sent back−to−back, with a short pause while switching buffers.

A unique feature is the ability to drive 150ohm STP (Shielded Twisted Pair) cable commonly installed for Token Ring, in addition to 10baseT 100ohm UTP (unshielded twisted pair). A fibre optic version of the card (AT1700FT) exists as well.

The Fujitsu chip used on the AT1700 has a design flaw: it can only be fully reset by doing a power cycle of the machine. Pressing the reset button doesn't reset the bus interface. This wouldn't be so bad, except that it can only be reliably detected when it has been freshly reset. The solution/work−around is to power−cycle the machine if the kernel has a problem detecting the AT1700.

AT2400

Status: Supported, Driver Name: ne, ne2k−pci (+8390)

Yet another PCI NE2000 clone card. This one is based on the RealTek 8029 chip.

AT2450

Status: Supported, Driver Name: pcnet32

This is the PCI version of the AT1500, and it doesn't suffer from the problems that the Boca 79c970 PCI card does. DMA selection and chip numbering information can be found in AMD LANCE.

More technical information on AMD LANCE based Ethernet cards can be found in Notes on AMD....

AT2500

Status: Semi−Supported, Driver Name: rtl8139

This card uses the RealTek 8139 chip − see the section RealTek 8139.

AT2540FX

Status: Semi−Supported, Driver Name: eepro100

This card uses the i82557 chip, and hence may/should work with the eepro100 driver. If you try this please send in a report so this information can be updated.

4.4 AMD / Advanced Micro Devices

Carl Ching of AMD was kind enough to provide a very detailed description of all the relevant AMD ethernet products which helped clear up this section.

AMD LANCE (7990, 79C960/961/961A, PCnet−ISA)

Status: Supported, Driver Name: lance

There really is no AMD ethernet card. You are probably reading this because the only markings you could find on your card said AMD and the above number. The 7990 is the original `LANCE' chip, but most stuff (including this document) refer to all these similar chips as `LANCE' chips. (...incorrectly, I might add.)

These above numbers refer to chips from AMD that are the heart of many ethernet cards. For example, the Allied Telesis AT1500 (see AT1500) and the NE1500/2100 (see NE1500) use these chips.

The 7990/79c90 have long been replaced by newer versions. The 79C960 (a.k.a. PCnet−ISA) essentially contains the 79c90 core, along with all the other hardware support required, which allows a single−chip ethernet solution. The 79c961 (PCnet−ISA+) is a jumperless Plug and Play version of the '960. The final chip in the ISA series is the 79c961A (PCnet−ISA II), which adds full duplex capabilities. All cards with one of these chips should work with the lance.c driver, with the exception of very old cards that used the original 7990 in a shared memory configuration. These old cards can be spotted by the lack of jumpers for a DMA channel.

One common problem people have is the `busmaster arbitration failure' message. This is printed out when the LANCE driver can't get access to the bus after a reasonable amount of time has elapsed (50us). This usually indicates that the motherboard implementation of bus−mastering DMA is broken, or some other device is hogging the bus, or there is a DMA channel conflict. If your BIOS setup has the `GAT option' (for Guaranteed Access Time) then try toggling/altering that setting to see if it helps.

Also note that the driver only looks at the addresses: `0x300, 0x320, 0x340, 0x360` for a valid card, and any address supplied by an `ether=` boot argument is silently ignored (this will be fixed) so make sure your card is configured for one of the above I/O addresses for now.

The driver will still work fine, even if more than 16MB of memory is installed, since low−memory `bounce−buffers' are used when needed (i.e. any data from above 16MB is copied into a buffer below 16MB before being given to the card to transmit.)

The DMA channel can be set with the low bits of the otherwise−unused dev−>mem_start value (a.k.a. PARAM_1). (see PARAM_1) If unset it is probed for by enabling each free DMA channel in turn and checking if initialization succeeds.

The HP−J2405A board is an exception: with this board it's easy to read the EEPROM−set values for the IRQ, and DMA.

See Notes on AMD... for more info on these chips.

AMD 79C965 (PCnet−32)

Status: Supported, Driver Name: pcnet32

This is the PCnet−32 −− a 32 bit bus−master version of the original LANCE chip for VL−bus and local bus systems. chip. While these chips can be operated with the standard `lance.c` driver, a 32 bit version (`pcnet32.c`) is also available that does not have to concern itself with any 16MB limitations associated with the ISA bus.

AMD 79C970/970A (PCnet−PCI)

Status: Supported, Driver Name: pcnet32

This is the PCnet−PCI −− similar to the PCnet−32, but designed for PCI bus based systems. Please see the above PCnet−32 information. This means that you need to build a kernel with PCI BIOS support enabled. The '970A adds full duplex support along with some other features to the original '970 design.

Note that the Boca implementation of the 79C970 fails on fast Pentium machines. This is a hardware problem, as it affects DOS users as well. See the Boca section for more details.

AMD 79C971 (PCnet−FAST)

Status: Supported, Driver Name: pcnet32

This is AMD's 100Mbit chip for PCI systems, which also supports full duplex operation. It was introduced in June 1996.

AMD 79C972 (PCnet−FAST+)

Status: Supported, Driver Name: pcnet32

This has been confirmed to work just like the '971.

AMD 79C974 (PCnet−SCSI)

Status: Supported, Driver Name: pcnet32

This is the PCnet−SCSI −− which is basically treated like a '970 from an Ethernet point of view. Also see the above information. Don't ask if the SCSI half of the chip is supported −− this is the *Ethernet−HowTo*, not the SCSI−HowTo.

4.5 Ansel Communications

AC3200 EISA

Status: Semi−Supported, Driver Name: ac3200

Note that to access this driver during `make config` you still have to answer `Y' when asked ``Prompt for development and/or incomplete code/drivers?'' at the first. This is simply due to lack of feedback on the driver stability due to it being a relatively rare card.

This driver is included in the present kernel as an alpha test driver. It is based on the common NS8390 chip used in the ne2000 and wd80x3 cards. Please see Alpha Drivers in this document for important information regarding alpha drivers.

If you use it, let one of us know how things work out, as feedback has been low, even though the driver has been in the kernel since v1.1.25.

If you intend on using this driver as a loadable module you should probably see Using the Ethernet Drivers as Modules for module specific information.

4.6 Apricot

Apricot Xen−II On Board Ethernet

Status: Semi−Supported, Driver Name: apricot

This on board ethernet uses an i82596 bus−master chip. It can only be at I/O address `0x300`. By looking at the driver source, it appears that the IRQ is also hardwired to 10.

Earlier versions of the driver had a tendency to think that anything living at `0x300` was an apricot NIC. Since then the hardware address is checked to avoid these false detections.

4.7 Arcnet

Status: Supported, Driver Name: arcnet (arc−rimi, com90xx, com20020)

With the very low cost and better performance of ethernet, chances are that most places will be giving away their Arcnet hardware for free, resulting in a lot of home systems with Arcnet.

An advantage of Arcnet is that all of the cards have identical interfaces, so one driver will work for everyone. It also has built in error handling so that it supposedly never loses a packet. (Great for UDP traffic!) Note that the arcnet driver uses `arc0' as its name instead of the usual `eth0' for ethernet devices.

There are information files contained in the standard kernel for setting jumpers, general hints and where to mail bug reports.

Supposedly the driver also works with the 100Mbs ARCnet cards as well!

4.8 AT&T

Note that AT&T's StarLAN is an orphaned technology, like SynOptics LattisNet, and can't be used in a standard 10Base−T environment, without a hub that `speaks' both.

AT&T T7231 (LanPACER+)

Status: Not Supported.

These StarLAN cards use an interface similar to the i82586 chip. At one point, Matthijs Melchior (`matthijs.n.melchior@att.com`) was playing with the 3c507 driver, and almost had something useable working. Haven't heard much since that.

4.9 Boca Research

Yes, they make more than just multi−port serial cards.

Boca BEN400

Status: Supported, Driver Name: ne (+8390)

Apparently this is a NE2000 clone, using a VIA VT86C916 chip.

Boca BEN (ISA, VLB, PCI)

Status: Supported, Driver Name: lance, pcnet32

These cards are based on AMD's PCnet chips. Perspective buyers should be warned that many users have had endless problems with these VLB/PCI cards. Owners of fast Pentium systems have been especially hit. Note that this is not a driver problem, as it hits DOS/Win/NT users as well. Boca's technical support number is (407) 241−8088, and you can also reach them at `75300.2672@compuserve.com`. The older ISA cards don't appear to suffer the same problems.

Boca was offering a `warranty repair' for affected owners, which involved adding one of the missing capacitors, but it appears that this fix didn't work 100 percent for most people, although it helps some.

More general information on the AMD chips can be found in AMD LANCE.

More technical information on AMD LANCE based Ethernet cards can be found in Notes on AMD....

4.10 Cabletron

Donald writes: `Yes, another one of these companies that won't release its programming information. They waited for months before actually confirming that all their information was proprietary, deliberately wasting my time. Avoid their cards like the plague if you can. Also note that some people have phoned Cabletron, and have been told things like `a D. Becker is working on a driver for linux' −− making it sound like I work for them. This is NOT the case.'

Apparently Cabletron has changed their policy with respect to programming information (like Xircom) since Donald made the above comment several years ago −− send e−mail to `support@ctron.com` if you want to verify this or ask for programming information. However, at this point in time, there is little demand for modified/updated drivers for the older E20xx and E21xx cards.

E10**, E10**−x, E20**, E20**−x

Status: Semi−Supported, Driver Name: ne (+8390)

These are NEx000 almost−clones that are reported to work with the standard NEx000 drivers, thanks to a ctron−specific check during the probe. If there are any problems, they are unlikely to be fixed, as the programming information is unavailable.

E2100

Status: Semi−Supported, Driver Name: e2100 (+8390)

Again, there is not much one can do when the programming information is proprietary. The E2100 is a poor design. Whenever it maps its shared memory in during a packet transfer, it maps it into the *whole 128K region!* That means you **can't** safely use another interrupt−driven shared memory device in that region, including another E2100. It will work most of the time, but every once in a while it will bite you. (Yes, this problem can be avoided by turning off interrupts while transferring packets, but that will almost certainly lose clock ticks.) Also, if you mis−program the board, or halt the machine at just the wrong moment, even the reset button won't bring it back. You will *have* to turn it off and *leave* it off for about 30 seconds.

Media selection is automatic, but you can override this with the low bits of the dev−>mem_end parameter. See PARAM_2. Module users can specify an `xcvr=N` value as an `option` in the `/etc/conf.modules` file.

Also, don't confuse the E2100 for a NE2100 clone. The E2100 is a shared memory NatSemi DP8390 design, roughly similar to a brain−damaged WD8013, whereas the NE2100 (and NE1500) use a bus−mastering AMD LANCE design.

There is an E2100 driver included in the standard kernel. However, seeing as programming info isn't available, don't expect bug−fixes. Don't use one unless you are already stuck with the card.

If you intend on using this driver as a loadable module you should probably see Using the Ethernet Drivers as Modules for module specific information.

E22**

Status: Semi−Supported, Driver Name: lance

According to information in a Cabletron Tech Bulletin, these cards use the standard AMD PC−Net chipset (see AMD PC−Net) and should work with the generic lance driver.

4.11 Cogent

Here is where and how to reach them:

```
Cogent Data Technologies, Inc.
175 West Street, P.O. Box 926
Friday Harbour, WA 98250, USA.

Cogent Sales
15375 S.E. 30th Place, Suite 310
Bellevue, WA 98007, USA.

Technical Support:
Phone (360) 378-2929 between 8am and 5pm PST
Fax (360) 378-2882
Compuserve GO COGENT
Bulletin Board Service (360) 378-5405
Internet: support@cogentdata.com
```

EM100−ISA/EISA

Status: Semi−Supported, Driver Name: smc9194

These cards use the SMC 91c100 chip and may work with the SMC 91c92 driver, but this has yet to be verified.

Cogent eMASTER+, EM100−PCI, EM400, EM960, EM964

Status: Supported, Driver Name: de4x5, tulip

These are yet another DEC 21040 implementation that should hopefully work fine with the standard 21040 driver.

The EM400 and the EM964 are four port cards using a DEC 21050 bridge and 4 21040 chips.

See DEC 21040 for more information on these cards, and the present driver situation.

4.12 Compaq

Compaq aren't really in the business of making ethernet cards, but a lot of their systems have embedded ethernet controllers on the motherboard.

Compaq Deskpro / Compaq XL (Embedded AMD Chip)

Status: Supported, Driver Name: pcnet32

Machines such as the XL series have an AMD 79c97x PCI chip on the mainboard that can be used with the standard LANCE driver. But before you can use it, you have to do some trickery to get the PCI BIOS to a place where Linux can see it. Frank Maas was kind enough to provide the details:

`` The problem with this Compaq machine however is that the PCI directory is loaded in high memory, at a spot where the Linux kernel can't (won't) reach. Result: the card is never detected nor is it usable (sideline: the mouse won't work either) The workaround (as described thoroughly in http://www−c724.uibk.ac.at/XL/) is to load MS−DOS, launch a little driver Compaq wrote and then load the Linux kernel using LOADLIN. Ok, I'll give you time to say `yuck, yuck', but for now this is the only working solution I know of. The little driver simply moves the PCI directory to a place where it is normally stored (and where Linux can find it).''

More general information on the AMD chips can be found in AMD LANCE.

Compaq Nettelligent/NetFlex (Embedded ThunderLAN Chip)

Status: Supported, Driver Name: tlan

These systems use a Texas Instruments ThunderLAN chip Information on the ThunderLAN driver can be found in ThunderLAN.

Compaq PCI card

Status: Supported, Driver Name: eepro100

Check your card − if it has part number 323551−821 and/or an intel 82558 chip on it then it is another Intel EEPro100 based card.

4.13 Danpex

Danpex EN9400

Status: Supported, Driver Name: de4x5, tulip

Yet another card based on the DEC 21040 chip, reported to work fine, and at a relatively cheap price.

See DEC 21040 for more information on these cards, and the present driver situation.

4.14 D−Link

DE−100, DE−200, DE−220−T, DE−250

Status: Supported, Driver Name: ne (+8390)

Some of the early D−Link cards didn't have the `0x57` PROM signature, but the ne2000 driver knows about them. For the software configurable cards, you can get the config program from `www.dlink.com`. The DE2** cards were the most widely reported as having the spurious transfer address mismatch errors with early versions of linux. Note that there are also cards from Digital (DEC) that are also named DE100 and DE200, but the similarity stops there.

DE−520

Status: Supported, Driver Name: pcnet32

This is a PCI card using the PCI version of AMD's LANCE chip. DMA selection and chip numbering information can be found in AMD LANCE.

More technical information on AMD LANCE based Ethernet cards can be found in Notes on AMD....

DE−528

Status: Supported, Driver Name: ne, ne2k−pci (+8390)

Apparently D−Link have also started making PCI NE2000 clones.

DE−530

Status: Supported, Driver Name: de4x5, tulip

This is a generic DEC 21040 PCI chip implementation, and is reported to work with the generic 21040 tulip driver. Note that this is NOT the DFE−530.

See DEC 21040 for more information on these cards, and the present driver situation.

DE−600

Status: Supported, Driver Name: de600

Laptop users and other folk who might want a quick way to put their computer onto the ethernet may want to use this. The driver is included with the default kernel source tree. Expect about 180kb/s transfer speed from this via the parallel port. You should read the README.DLINK file in the kernel source tree.

Note that the device name that you pass to `ifconfig` is *now* `eth0` and not the previously used `dl0`.

If your parallel port is *not* at the standard `0x378` then you will have to recompile, as the address is compiled directly into the driver. Also note that some laptops implement the on−board parallel port at `0x3bc` which is where the parallel ports on monochrome cards were/are.

DE−620

Status: Supported, Driver Name: de620

Same as the DE−600, only with two output formats. Bjorn has written a driver for this model, for kernel versions 1.1 and above. See the above information on the DE−600.

DE−650

Status: Semi−Supported, Driver Name: de650 (?)

Some people have been using this PCMCIA card for some time now with their notebooks. It is a basic 8390 design, much like a NE2000. The LinkSys PCMCIA card and the IC−Card Ethernet are supposedly DE−650 clones as well. Note that at present, this driver is *not* part of the standard kernel, and so you will have to do some patching. See PCMCIA Support in this document.

DFE−530TX

Status Supported, Driver Name: via−rhine

Another card using the VIA Rhine chipset. (see VIA Rhine) Don't confuse this with the DE−530 which is a tulip based card.

DFE−538TX

Status Supported, Driver Name: rtl8139, 8139too

This card uses the RealTek 8139 chip − see the section RealTek 8139.

4.15 DFI

DFINET−300 and DFINET−400

Status: Supported, Driver Name: ne (+8390)

Yet another poor NE clone card − these use `DFI' in the first 3 bytes of the prom, instead of using `0x57` in bytes 14 and 15, which is what all the NE1000 and NE2000 cards should use. (The 300 is an 8 bit pseudo NE1000 clone, and the 400 is a pseudo NE2000 clone.)

4.16 Digital / DEC

DEPCA, DE100/1, DE200/1/2, DE210, DE422

Status: Supported, Driver Name: depca

There is documentation included in the source file `depca.c', which includes info on how to use more than one of these cards in a machine. Note that the DE422 is an EISA card. These cards are all based on the AMD LANCE chip. See AMD LANCE for more info. A maximum of two of the ISA cards can be used, because they can only be set for `0x300` and `0x200` base I/O address. If you are intending to do this, please read the notes in the driver source file `depca.c` in the standard kernel source tree.

This driver will also work on Alpha CPU based machines, and there are various ioctl()s that the user can play with.

Digital EtherWorks 3 (DE203, DE204, DE205)

Status: Supported, Driver Name: ewrk3

These cards use a proprietary chip from DEC, as opposed to the LANCE chip used in the earlier cards like the DE200. These cards support both shared memory or programmed I/O, although you take about a 50%performance hit if you use PIO mode. The shared memory size can be set to 2kB, 32kB or 64kB, but only 2 and 32 have been tested with this driver. David says that the performance is virtually identical between the 2kB and 32kB mode. There is more information (including using the driver as a loadable module) at the top of the driver file `ewrk3.c` and also in `README.ewrk3`. Both of these files come with the standard kernel distribution. This driver has Alpha CPU support like depca.c does.

The standard driver has a number of interesting ioctl() calls that can be used to get or clear packet statistics, read/write the EEPROM, change the hardware address, and the like. Hackers can see the source code for more info on that one.

David has also written a configuration utility for this card (along the lines of the DOS program `NICSETUP.EXE`) along with other tools. These can be found on most Linux FTP sites in the directory `/pub/Linux/system/Network/management` −− look for the file `ewrk3tools−X.XX.tar.gz`.

DE425 EISA, DE434, DE435, DE500

Status: Supported, Driver Name: de4x5, tulip

These cards are based on the 21040 chip mentioned below. The DE500 uses the 21140 chip to provide 10/100Mbs ethernet connections. Have a read of the 21040 section below for extra info. There are also some compile−time options available for non−DEC cards using this driver. Have a look at `README.de4x5` for details.

All the Digital cards will autoprobe for their media (except, temporarily, the DE500 due to a patent issue).

This driver is also Alpha CPU ready and supports being loaded as a module. Users can access the driver internals through ioctl() calls − see the 'ewrk3' tools and the de4x5.c sources for information about how to do this.

DEC 21040, 21041, 2114x, Tulip

Status: Supported, Driver Name: de4x5, tulip

The DEC 21040 is a bus−mastering single chip ethernet solution from Digital, similar to AMD's PCnet chip. The 21040 is specifically designed for the PCI bus architecture. Apparently these chips are no longer being produced, as Intel has bought the semiconductor portion of DEC and is favouring their own ethernet chip(s).

You have a choice of *two* drivers for cards based on this chip. There is the DE425 driver discussed above, and the generic 21040 `tulip' driver.

Warning: Even though your card may be based upon this chip, *the drivers may not work for you*. David C. Davies writes:

``There are no guarantees that either `tulip.c' OR `de4x5.c' will run any DC2114x based card other than those they've been written to support. WHY?? You ask. Because there is a register, the General Purpose Register (CSR12) that (1) in the DC21140A is programmable by each vendor and they all do it differently (2) in the DC21142/3 this is now an SIA control register (a la DC21041). The only small ray of hope is that we can decode the SROM to help set up the driver. However, this is not a guaranteed solution since some vendors (e.g. SMC 9332 card) don't follow the Digital Semiconductor recommended SROM programming format.''

In non−technical terms, this means that if you aren't sure that an unknown card with a DC2114x chip will work with the linux driver(s), then make sure you can return the card to the place of purchase *before* you pay for it.

The 21041 chip is also found in place of the 21040 on most of the later SMC EtherPower cards. The 21140 is for supporting 100Base−T and works with the Linux drivers for the 21040 chip. To use David's `de4x5` driver with non−DEC cards, have a look at `README.de4x5` for details.

If you are having trouble with the tulip driver, you can try the newest version from Donald's ftp/WWW site.

Tulip Driver

There is also a (non−exhaustive) list of various cards/vendors that use the 21040 chip.

4.17 Farallon

Farallon sells EtherWave adaptors and transceivers. This device allows multiple 10baseT devices to be daisy−chained.

Farallon Etherwave

Status: Supported, Driver Name: 3c509

This is reported to be a 3c509 clone that includes the EtherWave transceiver. People have used these successfully with Linux and the present 3c509 driver. They are too expensive for general use, but are a great option for special cases. Hublet prices start at $125, and Etherwave adds $75−$100 to the price of the board −− worth it if you have pulled one wire too few, but not if you are two network drops short.

Farallon PCI 593

Status: Supported, Driver Name: de4x5, tulip

It has been reported that this card was detected with the `de4x5` driver.

4.18 Fujitsu

Unlike many network chip manufacturers, Fujitsu have also made and sold some network cards based upon their chip.

Fujitsu FMV−181/182/183/184

Status: Supported, Driver Name: fmv18x

According to the driver, these cards are a straight forward Fujitsu MB86965 implementation, which would make them very similar to the Allied Telesis AT1700 cards.

4.19 Hewlett Packard

The 272** cards use programmed I/O, similar to the NE*000 boards, but the data transfer port can be `turned off' when you aren't accessing it, avoiding problems with autoprobing drivers.

Thanks to Glenn Talbott for helping clean up the confusion in this section regarding the version numbers of the HP hardware.

HP Night Director+ 10/100

Status: Supported, Driver Name: pcnet32

Apparently these cards use the AMD 79C972 chip.

27245A

Status: Supported, Driver Name: hp (+8390)

8 bit 8390 based 10BaseT, not recommended for all the 8 bit reasons. It was re−designed a couple years ago to be highly integrated which caused some changes in initialization timing which only affected testing programs, not LAN drivers. (The new card is not `ready' as soon after switching into and out of loopback mode.)

If you intend on using this driver as a loadable module you should probably see Using the Ethernet Drivers as Modules for module specific information.

HP EtherTwist, PC Lan+ (27247, 27252A)

Status: Supported, Driver Name: hp+ (+8390)

The HP PC Lan+ is different to the standard HP PC Lan card. This driver was added to the list of drivers in the standard kernel during the v1.1.x development cycle. It can be operated in either a PIO mode like a ne2000, or a shared memory mode like a wd8013.

The 47B is a 16 Bit 8390 based 10BaseT w/AUI, and the 52A is a 16 Bit 8390 based ThinLAN w/AUI. These cards have 32K onboard RAM for Tx/Rx packet buffering instead of the usual 16KB, and they both offer LAN connector autosense.

If you intend on using this driver as a loadable module you should probably see Using the Ethernet Drivers as Modules for module specific information.

HP−J2405A

Status: Supported, Driver Name: lance

These are lower priced, and slightly faster than the 27247/27252A, but are missing some features, such as AUI, ThinLAN connectivity, and boot PROM socket. This is a fairly generic LANCE design, but a minor design decision makes it incompatible with a generic `NE2100' driver. Special support for it (including reading the DMA channel from the board) is included thanks to information provided by HP's Glenn Talbott.

More technical information on LANCE based cards can be found in Notes on AMD...

HP−Vectra On Board Ethernet

Status: Supported, Driver Name: lance

The HP−Vectra has an AMD PCnet chip on the motherboard. DMA selection and chip numbering information can be found in AMD LANCE.

More technical information on LANCE based cards can be found in Notes on AMD...

HP 10/100 VG Any Lan Cards (27248B, J2573, J2577, J2585, J970, J973)

Status: Supported, Driver Name: hp100

This driver also supports some of the Compex VG products. Since the driver supports ISA, EISA and PCI cards, it is found under ISA cards when running `make config` on a kernel source.

HP NetServer 10/100TX PCI (D5013A)

Status: Supported, Driver Name: eepro100

Apparently these are just a rebadged Intel EtherExpress Pro 10/100B card. See the Intel section for more information.

4.20 IBM / International Business Machines

IBM Thinkpad 300

Status: Supported, Driver Name: znet

This is compatible with the Intel based Zenith Z−note. See Z−note for more info.

Supposedly this site has a comprehensive database of useful stuff for newer versions of the Thinkpad. I haven't checked it out myself yet.

Thinkpad−info

For those without a WWW browser handy, try `peipa.essex.ac.uk:/pub/tp750/`

IBM Credit Card Adaptor for Ethernet

Status: Semi−Supported, Driver Name: ? (distributed separately)

People have been using this PCMCIA card with Linux as well. Similar points apply, those being that you need a supported PCMCIA chipset on your notebook, and that you will have to patch the PCMCIA support into the standard kernel. See PCMCIA Support in this document.

IBM 10/100 EtherJet PCI

Status: Supported, Driver Name: eepro100

This card is reported to be compatible with the Intel EtherExpress Pro 100 driver.

IBM Token Ring

Status: Semi−Supported, Driver Name: ibmtr

To support token ring requires more than only writing a device driver, it also requires writing the source routing routines for token ring. It is the source routing that would be the most time comsuming to write.

Initial driver development was done with IBM ISA and MCA token ring cards, and tested on an MCA 16/4 Megabit Token Ring board, but it should work with other Tropic based boards.

4.21 ICL Ethernet Cards

ICL EtherTeam 16i/32

Status: Supported, Driver Name: eth16i

This driver supports both the ISA (16i) and EISA (32) versions of the card. It uses the Fujitsu MB86965 chip that is also used on the at1700 cards.

4.22 Intel Ethernet Cards

Note that the naming of the various Intel cards is ambiguous and confusing at best. If in doubt, then check the `i8xxxx` number on the main chip on the card or for PCI cards, use the PCI information in the `/proc` directory and then compare that to the numbers listed here.

Ether Express

Status: Supported, Driver Name: eexpress

This card uses the intel i82586. Earlier versions of this driver (in v1.2 kernels) were classed as alpha−test, as it didn't work well for most people. The driver in the v2.0 kernel seems to work much better for those who have tried it, although the driver source still lists it as experimental and more problematic on faster machines.

The comments at the top of the driver source list some of the problems (and fixes!) associated with these cards. The slowdown hack of replacing all the `outb` with `outb_p` in the driver has been reported to avoid lockups for at least one user. Also check that the size of the RAM buffer reported by the driver matches what the Intel configuration utility reports.

Ether Express PRO/10 (PRO/10+)

Status: Supported, Driver Name: eepro

Bao Chau Ha has written a driver for these cards that has been included into early 1.3.x kernels. It may also work with some of the Compaq built−in ethernet systems that are based on the i82595 chip. You may have to use the configuration utility that came with the card to disable PnP support where applicable.

Ether Express PRO/10 PCI (EISA)

Status: Semi−Supported, Driver Name: ? (distributed separately)

There is a driver for the PCI version that is distributed separately from the default kernel. These cards use the PLX9036 PCI interface chip with the Intel i82596 LAN controller chip. If your card has the i82557 chip, then you *don't* have this card, but rather the version discussed next, and hence want the EEPro100 driver instead.

You can get the alpha driver for the PRO/10 PCI card, along with instructions on how to use it at:

EEPro10 Driver

If you have the EISA card, you will probably have to hack the driver a bit to account for the different (PCI vs. EISA) detection mechanisms that are used in each case.

Ether Express PRO 10/100B

Status: Supported, Driver Name: eepro100

Note that this driver will *not* work with the older 100A cards. The chip numbers listed in the driver are i82557/i82558. For driver updates and/or driver support, have a look at:

EEPro−100B Page

4.23 Kingston

Kingston make various cards, including NE2000+, AMD PCnet based cards, and DEC tulip based cards. Most of these cards should work fine with their respective driver. See Kingston Web Page

4.24 LinkSys

LinkSys make a handful of different NE2000 clones, some straight ISA cards, some ISA plug and play and some even ne2000−PCI clones based on one of the supported ne2000−PCI chipsets. There are just too many models to list here.

LinkSys are linux−friendly, with a linux specific WWW support page, and even have Linux printed on the boxes of some of their products. Have a look at:

`http://www.linksys.com/support/solution/nos/linux.htm`

LinkSys Etherfast 10/100 Cards.

Status: Supported, Driver Name: tulip

Note that with these cards there have been several `revisions' (i.e. different chipset used) all with the same card name. The 1st used the DEC chipset. The 2nd revision used the Lite−On PNIC 82c168 PCI Network Interface Controller, and the 3rd revision of the card uses a LinkSys 82c169 NIC chip. Support for the latter two has been merged into the standard tulip driver −− you may need a driver upgrade to get support for them depending on how old your current driver version is.

More PNIC information is available at:

`http://www.scyld.com/linux/drivers/pnic.html`

More information on the various versions of these cards can be found at the LinkSys WWW site mentioned above.

LinkSys Pocket Ethernet Adapter Plus (PEAEPP)

Status: Supported, Driver Name: de620

This is supposedly a DE−620 clone, and is reported to work well with that driver. See DE−620 for more information.

LinkSys PCMCIA Adaptor

Status: Supported, Driver Name: de650 (?)

This is supposed to be a re−badged DE−650. See DE−650 for more information.

4.25 Microdyne (Eagle)

Eagle Technology (aka Novell cards) was sold to Microdyne. If you can't find your card listed here, check the Novell section of this document. While Microdyne are not actively selling network cards anymore, there is still some stuff relating to their products on their site at `ftp.mcdy.com`

Microdyne Exos 205T

Status: Semi−Supported, Driver Name: ?

Another i82586 based card. Dirk Niggemann `dirk−n@dircon.co.uk` has written a driver that he classes as ``pre−alpha'' that he would like people to test. Mail him for more details.

4.26 Mylex

Mylex can be reached at the following numbers, in case anyone wants to ask them anything.

```
        MYLEX CORPORATION, Fremont
        Sales:  800-77-MYLEX, (510) 796-6100
        FAX:    (510) 745-8016.
```

They also have a web site: Mylex WWW Site

Mylex LNE390A, LNE390B

Status: Supported, Driver Name: lne390 (+8390)

These are fairly old EISA cards that make use of a shared memory implementation similar to the wd80x3. A driver for these cards is available in the current 2.1.x series of kernels. Ensure you set the shared memory address below 1MB or above the highest address of the physical RAM installed in the machine.

Mylex LNP101

Status: Supported, Driver Name: de4x5, tulip

This is a PCI card that is based on DEC's 21040 chip. It is selectable between 10BaseT, 10Base2 and 10Base5 output. The LNP101 card has been verified to work with the generic 21040 driver.

See the section on the 21040 chip (DEC 21040) for more information.

Mylex LNP104

Status: Semi−Supported, Driver Name: de4x5, tulip

The LNP104 uses the DEC 21050 chip to deliver *four* independent 10BaseT ports. It should work with recent 21040 drivers that know how to share IRQs, but nobody has reported trying it yet (that I am aware of).

4.27 Novell Ethernet, NExxxx and associated clones.

The prefix `NE' came from Novell Ethernet. Novell followed the cheapest NatSemi databook design and sold the manufacturing rights (spun off?) Eagle, just to get reasonably−priced ethercards into the market. (The now ubiquitous NE2000 card.)

NE1000, NE2000

Status: Supported, Driver Name: ne (+8390)

The ne2000 is now a generic name for a bare−bones design around the NatSemi 8390 chip. They use programmed I/O rather than shared memory, leading to easier installation but slightly lower performance and a few problems. Some of the more common problems that arise with NE2000 cards are listed in Problems with...

Some NE2000 clones use the National Semiconductor `AT/LANTic' 83905 chip, which offers a shared memory mode similar to the wd8013 and EEPROM software configuration. The shared memory mode will offer less CPU usage (i.e. more efficient) than the programmed I/O mode.

In general it is not a good idea to put a NE2000 clone at I/O address `0x300` because nearly *every* device driver probes there at boot. Some poor NE2000 clones don't take kindly to being prodded in the wrong areas, and will respond by locking your machine. Also `0x320` is bad because SCSI drivers probe into `0x330`.

Donald has written a NE2000 diagnostic program (ne2k.c) for all ne2000 cards. See Diagnostic Programs for more information.

If you intend on using this driver as a loadable module you should probably see Using the Ethernet Drivers as Modules for module specific information.

NE2000−PCI (RealTek/Winbond/Compex)

Status: Supported, Driver Name: ne, ne2k−pci (+8390)

Yes, believe it or not, people are making PCI cards based on the more than ten year old interface design of the ne2000. At the moment nearly all of these cards are based on the RealTek 8029 chip, or the Winbond 89c940 chip. The Compex, KTI, VIA and Netvin cards apparently also use these chips, but have a different PCI ID.

The latest v2.0 kernel has support to automatically detect all these cards and use them. (If you are using a kernel v2.0.34 or older, you should upgrade to ensure your card will be detected.) There are now two drivers to choose from; the original ISA/PCI `ne.c` driver, and a relatively new PCI−only `ne2k−pci.c` driver.

To use the original ISA/PCI driver you have to say `Y' to the `Other ISA cards' option when running `make config` as you are actually using the same NE2000 driver as the ISA cards use. (That should also give you a hint that these cards aren't anywhere as intelligent as say a PCNet−PCI or DEC 21040 card...)

The newer PCI−only driver differs from the ISA/PCI driver in that all the support for old NE1000 8 bit cards has been removed and that data is moved to/from the card in bigger blocks, without any intervening pauses that the older ISA−NE2000's required for reliable operation. The result is a driver that is slightly smaller and slightly more efficient, but don't get too excited as the difference will not be obvious under normal use. (If you really wanted maximum efficiency/low CPU use, then a PCI−NE2000 is simply a very poor choice.) Driver updates and more information can be found at:

```
http://www.scyld.com/linux/drivers/ne2k-pci.html
```

If you have a NE2000 PCI card that is *not* detected by the most current version of the driver, please contact the maintainer of the NE2000 driver as listed in `/usr/src/linux/MAINTAINERS` along with the output from a `cat /proc/pci` and `dmesg` so that support for your card can also be added to the driver.

Also note that various card makers have been known to put `NE2000 Compatible' stickers on their product boxes even when it is completely different (e.g. PCNet−PCI or RealTek 8139). If in doubt check the main chip number against this document.

NE−10/100

Status: Not Supported.

These are ISA 100Mbps cards based on the National Semiconductor DP83800 and DP83840 chips. There is currently no driver support, nor has anyone reported that they are working on a driver. Apparently documentation on the chip is unavailable with the exception of a single PDF file that doesn't give enough details for a driver.

NE1500, NE2100

Status: Supported, Driver Name: lance

These cards use the original 7990 LANCE chip from AMD and are supported using the Linux lance driver. Newer NE2100 clones use the updated PCnet/ISA chip from AMD.

Some earlier versions of the lance driver had problems with getting the IRQ line via autoIRQ from the original Novell/Eagle 7990 cards. Hopefully this is now fixed. If not, then specify the IRQ via LILO, and let us know that it still has problems.

DMA selection and chip numbering information can be found in AMD LANCE.

More technical information on LANCE based cards can be found in Notes on AMD...

NE/2 MCA

Status: Semi−Supported, Driver Name: ne2

There were a few NE2000 microchannel cards made by various companies. This driver, available in v2.2 kernels, will detect the following MCA cards: Novell Ethernet Adapter NE/2, Compex ENET−16 MC/P, and the Arco Ethernet Adapter AE/2.

NE3200

Status: Not Supported.

This old EISA card uses a 8MHz 80186 in conjunction with an i82586. Nobody is working on a driver for it, as there is no information available on the card, and no real demand for a driver either.

NE3210

Status: Supported, Driver Name: ne3210 (+8390)

This EISA card is completely different from the NE3200, as it uses a Nat Semi 8390 chip. The driver can be found in the v2.2 kernel source tree. Ensure you set the shared memory address below 1MB or above the highest address of the physical RAM installed in the machine.

NE5500

Status: Supported, Driver Name: pcnet32

These are just AMD PCnet−PCI cards ('970A) chips. More information on LANCE/PCnet based cards can be found in AMD LANCE.

4.28 Proteon

Proteon P1370−EA

Status: Supported, Driver Name: ne (+8390)

Apparently this is a NE2000 clone, and works fine with Linux.

Proteon P1670−EA

Status: Supported, Driver Name: de4x5, tulip

This is yet another PCI card that is based on DEC's Tulip chip. It has been reported to work fine with Linux.

See the section on the 21040 chip (DEC 21040) for more driver information.

4.29 Pure Data

PDUC8028, PDI8023

Status: Supported, Driver Name: wd (+8390)

The PureData PDUC8028 and PDI8023 series of cards are `almost clones' of the wd80x3 cards − there is special code in the `wd.c` driver to probe for these cards.

4.30 Racal−Interlan

Racal Interlan can be reached via WWW at `www.interlan.com`. I believe they were also known as MiCom−Interlan at one point in the past.

ES3210

Status: Semi−Supported, Driver Name: es3210

This is an EISA 8390 based shared memory card. An experimetal driver is shipped with v2.2 kernels and it is reported to work fine, but the EISA IRQ and shared memory address detection appears not to work with (at least) the early revision cards. (This problem is not unique to the Linux world either...) In that case, you have to supply them to the driver. For example, card at IRQ 5 and shared memory `0xd0000`, with a modular driver, add `options es3210 irq=5 mem=0xd0000` to `/etc/conf.modules`. Or with the driver compiled into the kernel, supply at boot `ether=5,0,0xd0000,eth0` The I/O base is automatically detected and hence a value of zero should be used.

NI5010

Status: Semi−Supported, Driver Name: ni5010

You used to have to go get the driver for these old 8 bit MiCom−Interlan cards separately, but now it is shipped with the v2.2 kernels as an experimental driver.

NI5210

Status: Semi−Supported, Driver Name: ni52

This card also uses one of the Intel chips. Michael Hipp has written a driver for this card. It is included in the standard kernel as an `alpha' driver. Michael would like to hear feedback from users that have this card. See Alpha Drivers for important information on using alpha−test ethernet drivers with Linux.

NI6510 (not EB)

Status: Semi−Supported, Driver Name: ni65

There is also a driver for the LANCE based NI6510, and it is also written by Michael Hipp. Again, it is also an `alpha' driver. For some reason, this card is not compatible with the generic LANCE driver. See Alpha Drivers for important information on using alpha−test ethernet drivers with Linux.

EtherBlaster (aka NI6510EB)

Status: Supported, Driver Name: lance

As of kernel 1.3.23, the generic LANCE driver had a check added to it for the `0x52, 0x44` NI6510EB specific signature. Others have reported that this signature is not the same for all NI6510EB cards however, which will cause the lance driver to not detect your card. If this happens to you, you can change the probe (at about line 322 in lance.c) to printk() out what the values are for your card and then use them instead of the `0x52, 0x44` defaults.

The cards should probably be run in `high−performance' mode and not in the NI6510 compatible mode when using the lance driver.

4.31 RealTek

RealTek RTL8002/8012 (AT−Lan−Tec) Pocket adaptor

Status: Supported, Driver Name: atp

This is a generic, low−cost OEM pocket adaptor being sold by AT−Lan−Tec, and (likely) a number of other suppliers. A driver for it is included in the standard kernel. Note that there is substantial information contained in the driver source file `atp.c'.

Note that the device name that you pass to `ifconfig` was *not* `eth0` but `atp0` for earlier versions of this driver.

RealTek 8009

Status: Supported, Driver Name: ne (+8390)

This is an ISA NE2000 clone, and is reported to work fine with the linux NE2000 driver. The `rset8009.exe` program can be obtained from RealTek's WWW site at `http://www.realtek.com.tw` − or via ftp from the same site.

RealTek 8019

Status: Supported, Driver Name: ne (+8390)

This is a Plug and Pray version of the above. Use the DOS software to disable PnP and enable jumperless configuration; set the card to a sensible I/O address and IRQ and you should be ready to go. (If using the driver as a module, don't forget to add an `io=0xNNN` option to `/etc/conf.modules`). The `rset8019.exe` program can be obtained from RealTek's WWW site at `http://www.realtek.com.tw` − or via ftp from the same site.

RealTek 8029

Status: Supported, Driver Name: ne, ne2k−pci (+8390)

This is a PCI single chip implementation of a NE2000 clone. Various vendors are now selling cards with this chip. See NE2000−PCI for information on using any of these cards. Note that this is still a 10+ year old design just glued onto a PCI bus. Performance won't be staggeringly better than the equivalent ISA model.

RealTek 8129/8139

Status: Semi−Supported, Driver Name: rtl8139, 8139too

Another PCI single chip ethernet solution from RealTek. A driver for cards based upon this chip was included in the v2.0.34 release of linux. You currently have to answer `Y' when asked if you want experimental drivers for v2.2 kernels to get access to this driver.

Donald says that cards based on this chip are around the same price (or even cheaper in places − 13 bucks!) as a PCI NE2000 clone card, and while the 8139 design is not the best 10/100 board, it is better than a PCI

NE2000 clone card.

The 2.4.x kernels have another driver called `8139too` which was based on rtl8139 but tries to adress some of the more common problems people were reporting, so you may wish to try that if using a 2.4 kernel.

4.32 Sager

Sager NP943

Status: Semi−Supported, Driver Name: 3c501

This is just a 3c501 clone, with a different S.A. PROM prefix. I assume it is equally as brain dead as the original 3c501 as well. The driver checks for the NP943 I.D. and then just treats it as a 3c501 after that. See 3Com 3c501 for all the reasons as to why you really don't want to use one of these cards.

4.33 Schneider & Koch

SK G16

Status: Supported, Driver Name: sk_g16

This driver was included into the v1.1 kernels, and it was written by PJD Weichmann and SWS Bern. It appears that the SK G16 is similar to the NI6510, in that it is based on the first edition LANCE chip (the 7990). Once again, it appears as though this card won't work with the generic LANCE driver.

4.34 SEEQ

SEEQ 8005

Status: Supported, Driver Name: seeq8005

There is little information about the card included in the driver, and hence little information to be put here. If you have a question, you are probably best trying to e−mail the driver author as listed in the source.

4.35 SMC (Standard Microsystems Corp.)

The ethernet part of Western Digital was bought out by SMC many years ago when the wd8003 and wd8013 were the main product. Since then SMC has continued making 8390 based ISA cards (Elite16, Ultra, EtherEZ) and also added several PCI products to their range.

Contact information for SMC:

SMC / Standard Microsystems Corp., 80 Arkay Drive, Hauppage, New York, 11788, USA. Technical Support via phone: 800−992−4762 (USA) or 800−433−5345 (Canada) or 516−435−6250 (Other Countries). Literature requests: 800−SMC−4−YOU (USA) or 800−833−4−SMC (Canada) or 516−435−6255 (Other Countries). Technical Support via E−mail: `techsupt@ccmail.west.smc.com`. FTP Site: `ftp.smc.com`. WWW Site: SMC.

WD8003, SMC Elite

Status: Supported, Driver Name: wd (+8390)

These are the 8−bit versions of the card. The 8 bit 8003 is slightly less expensive, but only worth the savings for light use. Note that some of the non−EEPROM cards (clones with jumpers, or old *old* old wd8003 cards) have no way of reporting the IRQ line used. In this case, auto−irq is used, and if that fails, the driver silently assings IRQ 5. You can get the SMC setup/driver disks from SMC's ftp site. Note that some of the newer SMC `SuperDisk' programs will fail to detect the real old EEPROM−less cards. The file `SMCDSK46.EXE` seems to be a good all−round choice. Also the jumper settings for all their cards are in an ASCII text file in the aforementioned archive. The latest (greatest?) version can be obtained from `ftp.smc.com`.

As these are basically the same as their 16 bit counterparts (WD8013 / SMC Elite16), you should see the next section for more information.

WD8013, SMC Elite16

Status: Supported, Driver Name: wd (+8390)

Over the years the design has added more registers and an EEPROM. (The first wd8003 cards appeared about ten years ago!) Clones usually go by the `8013' name, and usually use a non−EEPROM (jumpered) design. Late model SMC cards will have the SMC 83c690 chip instead of the original Nat Semi DP8390 found on earlier cards. The shared memory design makes the cards a bit faster than PIO cards, especially with larger packets. More importantly, from the driver's point of view, it avoids a few bugs in the programmed−I/O mode of the 8390, allows safe multi−threaded access to the packet buffer, and it doesn't have a programmed−I/O data register that hangs your machine during warm−boot probes.

Non−EEPROM cards that can't just read the selected IRQ will attempt auto−irq, and if that fails, they will silently assign IRQ 10. (8 bit versions will assign IRQ 5)

Cards with a non standard amount of memory on board can have the memory size specified at boot (or as an option in `/etc/conf.modules` if using modules). The standard memory size is 8kB for an 8bit card and 16kB for a 16bit card. For example, the older WD8003EBT cards could be jumpered for 32kB memory. To make full use of that RAM, you would use something like (for I/O=0x280 and IRQ 9):

```
        LILO: linux ether=9,0x280,0xd0000,0xd8000,eth0
```

Also see 8013 problems for some of the more common problems and frequently asked questions that pop up often.

If you intend on using this driver as a loadable module you should probably see Using the Ethernet Drivers as Modules for module specific information.

SMC Elite Ultra

Status: Supported, Driver Name: smc−ultra (+8390)

This ethercard is based on the 83c790 chip from SMC, which has a few new features over the 83c690. While it has a mode that is similar to the older SMC ethercards, it's not entirely compatible with the old WD80*3 drivers. However, in this mode it shares most of its code with the other 8390 drivers, while operating slightly faster than a WD8013 clone.

Since part of the Ultra *looks like* an 8013, the Ultra probe is supposed to find an Ultra before the wd8013 probe has a chance to mistakenly identify it.

Donald mentioned that it is possible to write a separate driver for the Ultra's `Altego' mode which allows chaining transmits at the cost of inefficient use of receive buffers, but that will probably not happen.

Bus−Master SCSI host adaptor users take note: In the manual that ships with Interactive UNIX, it mentions that a bug in the SMC Ultra will cause data corruption with SCSI disks being run from an aha−154X host adaptor. This will probably bite aha−154X compatible cards, such as the BusLogic boards, and the AMI−FastDisk SCSI host adaptors as well.

SMC has acknowledged the problem occurs with Interactive, and older Windows NT drivers. It is a hardware conflict with early revisions of the card that can be worked around in the driver design. The current Ultra driver protects against this by only enabling the shared memory during data transfers with the card. Make sure your kernel version is at least 1.1.84, or that the driver version reported at boot is at least `smc−ultra.c:v1.12` otherwise you are vulnerable.

If you intend on using this driver as a loadable module you should probably see Using the Ethernet Drivers as Modules for module specific information.

SMC Elite Ultra32 EISA

Status: Supported, Driver Name: smc−ultra32 (+8390)

This EISA card shares a lot in common with its ISA counterpart. A working (and stable) driver is included in both v2.0 and v2.2 kernels. Thanks go to Leonard Zubkoff for purchasing some of these cards so that linux support could be added for them.

SMC EtherEZ (8416)

Status: Supported, Driver Name: smc−ultra (+8390)

This card uses SMC's 83c795 chip and supports the Plug 'n Play specification. It also has an *SMC Ultra* compatible mode, which allows it to be used with the Linux Ultra driver. For best results, use the SMC supplied program (avail. from their www/ftp site) to disable PnP and configure it for shared memory mode. See the above information for notes on the Ultra driver.

For v1.2 kernels, the card had to be configured for shared memory operation. However v2.0 kernels can use the card in shared memory or programmed I/O mode. Shared memory mode will be slightly faster, and use less CPU resources as well.

SMC EtherPower PCI (8432)

Status: Supported, Driver Name: de4x5, tulip

NB: The EtherPower II is an entirely different card. See below! These cards are a basic DEC 21040 implementation, i.e. one big chip and a couple of transceivers. Donald has used one of these cards for his development of the generic 21040 driver (aka `tulip.c`). Thanks to Duke Kamstra, once again, for supplying a card to do development on.

Some of the later revisons of this card use the newer DEC 21041 chip, which may cause problems with older versions of the tulip driver. If you have problems, make sure you are using the latest driver release, which may not yet be included in the current kernel source tree.

See DEC 21040 for more details on using one of these cards, and the current status of the driver.

Apparently, the latest revision of the card, the EtherPower−II uses the 9432 chip. It is unclear at the moment if this one will work with the present driver. As always, if unsure, check that you can return the card if it doesn't work with the linux driver *before* paying for the card.

SMC EtherPower II PCI (9432)

Status: Semi−Supported, Driver Name: epic100

These cards, based upon the SMC 83c170 chip, are entirely different than the Tulip based cards. A new driver has been included in kernels v2.0 and v2.2 to support these cards. For more details, see:

`http://www.scyld.com/linux/drivers/epic100.html`

SMC 1211TX 10/100

Status: Semi−Supported, Driver Name: rtl8139

Apparently SMC is no longer the same company that brought you cards like the Ultra and the EPIC. The chip design part is now called SMSC and you will see the SMC name stuck on low end OEM boards like this one − a RealTek 8139 with a modified EEPROM.

SMC 3008

Status: Not Supported.

These 8 bit cards are based on the Fujitsu MB86950, which is an ancient version of the MB86965 used in the Linux at1700 driver. Russ says that you could probably hack up a driver by looking at the at1700.c code and his DOS packet driver for the Tiara card (tiara.asm). They are not very common.

SMC 3016

Status: Not Supported.

These are 16bit I/O mapped 8390 cards, much similar to a generic NE2000 card. If you can get the specifications from SMC, then porting the NE2000 driver would probably be quite easy. They are not very common.

SMC−9000 / SMC 91c92/4

Status: Supported, Driver Name: smc9194

The SMC9000 is a VLB card based on the 91c92 chip. The 91c92 appears on a few other brand cards as well, but is fairly uncommon.

SMC 91c100

Status: Semi−Supported, Driver Name: smc9194

The SMC 91c92 driver is supposed to work for cards based on this 100Base−T chip, but at the moment this is unverified.

4.36 Texas Instruments

ThunderLAN

Status: Supported, Driver Name: tlan

This driver covers many Compaq built−in ethernet devices, including the NetFlex and Netelligent groups. It also supports the Olicom 2183, 2185, 2325 and 2326 products.

4.37 Thomas Conrad

Thomas Conrad TC−5048

This is yet another PCI card that is based on DEC's 21040 chip.

See the section on the 21040 chip (DEC 21040) for more information.

4.38 VIA

You probably won't see a VIA networking card, as VIA make several networking chips that are then used by others in the construction of an ethernet card. They have a WWW site at:

`http://www.via.com.tw/`

VIA 86C926 Amazon

Status: Supported, Driver Name: ne, ne2k−pci (+8390)

This controller chip is VIA's PCI−NE2000 offering. You can choose between the ISA/PCI `ne.c` driver or the PCI−only `ne2k-pci.c` driver. See the PCI−NE2000 section for more details.

VIA 86C100A Rhine II (and 3043 Rhine I)

Status Supported, Driver Name: via−rhine

This relatively new driver can be found in current 2.0 and 2.1 kernels. It is an improvement over the 86C926 NE2000 chip in that it supports bus master transfers, but strict 32 bit buffer alignment requirements limit the benefit gained from this. For more details and driver updates, see:

`http://www.scyld.com/linux/drivers/via-rhine.html`

4.39 Western Digital

Please see SMC for information on SMC cards. (SMC bought out Western Digital's network card section many years ago.)

4.40 Winbond

Winbond don't really make and sell complete cards to the general public −− instead they make single chip ethernet solutions that other companies buy, stick onto a PCI board with their own name and then sell through retail stores. Some setup programs and tech support is available at:

`http://www.winbond.com.tw`

Winbond 89c840

Status: Semi−Supported, Driver Name: winbond−840

This chip has been described as `the mutant spawn of a NE2000 and a Tulip clone' −− see the driver notes for more details. The driver isn't currently shipped with the kernel, as it is in the testing phase (as of Sept 1998). It is available at:

`http://www.scyld.com/linux/drivers/test/winbond-840.c`

Winbond 89c904, 89c905, 89c906

Status: Supported, Driver Name: ne (+8390)

These are Winbond's ISA 10Mbps ne2000 compatible ethernet chips. Setup programs are available at the Winbond site.

Winbond 89c940

Status: Supported, Driver Name: ne, ne2k−pci (+8390)

This chip is one of the two commonly found on the low price PCI ne2000 cards sold by lots of manufacturers. Note that this is still a 10+ year old design just glued onto a PCI bus. Performance won't be staggeringly better than the equivalent ISA model.

4.41 Xircom

For the longest time, Xircom wouldn't release the programming information required to write a driver, unless you signed your life away. Apparently enough linux users have pestered them for driver support (they claim to support all popular networking operating systems...) so that they have changed their policy to allow documentation to be released without having to sign a non−disclosure agreement. Some people have said they they will release the source code to the SCO driver, while others have been told that they are no longer providing information on `obsolete' products like the earlier PE models. If you are interested and want to check into this yourself, you can reach Xircom at 1−800−874−7875, 1−800−438−4526 or +1−818−878−7600.

Xircom PE1, PE2, PE3−10B*

Status: Not Supported.

Not to get your hopes up, but if you have one of these parallel port adaptors, you may be able to use it in the DOS emulator with the Xircom−supplied DOS drivers. You will have to allow DOSEMU access to your parallel port, and will probably have to play with SIG (DOSEMU's Silly Interrupt Generator).

Xircom PCMCIA Cards

Status: Semi−Supported, Driver Name: ????

Some of the Xircom PCMCIA card(s) have drivers that are available with David Hinds PCMCIA package. Check there for the most up to date indformation

4.42 Zenith

Z−Note

Status: Supported, Driver Name: znet

The built−in Z−Note network adaptor is based on the Intel i82593 using *two* DMA channels. There is an (alpha?) driver available in the present kernel version. As with all notebook and pocket adaptors, it is under the `Pocket and portable adaptors' section when running `make config`. Also note that the IBM ThinkPad 300 is compatible with the Z−Note.

4.43 Znyx

Znyx ZX342 (DEC 21040 based)

Status: Supported, Driver Name: de4x5, tulip

You have a choice of *two* drivers for cards based on this chip. There is the DE425 driver written by David, and the generic 21040 driver that Donald has written.

Note that as of 1.1.91, David has added a compile time option that may allow non−DEC cards (such as the Znyx cards) to work with this driver. Have a look at `README.de4x5` for details.

See DEC 21040 for more information on these cards, and the present driver situation.

4.44 Identifying an Unknown Card

Okay, so your uncle's cousin's neighbour's friend had a brother who found an old ISA ethernet card in the AT case he was using as a cage for his son's pet hampster. Somehow you ended up with the card and want to try and use it with linux, but nobody has a clue what the card is and there isn't any documentation.

First of all, look for any obvious model numbers that might give a clue. Any model number that contains 2000 will most likely be a NE2000 clone. Any cards with 8003 or 8013 on them somewhere will be Western/Digital WD80x3 cards or SMC Elite cards or clones of them.

Identifying the Network Interface Controller

Look for the biggest chip on the card. This will be the network controller (NIC) itself, and most can be identified by the part number. If you know which NIC is on the card, the following might be able to help you figure out what card it is.

Probably still the most common NIC is the National Semiconductor DP8390 aka NS32490 aka DP83901 aka DP83902 aka DP83905 aka DP83907. And those are just the ones made by National! Other companies such as Winbond and UMC make DP8390 and DP83905 clone parts, such as the Winbond 89c904 (DP83905 clone) and the UMC 9090. If the card has some form of 8390 on it, then chances are it is a ne1000 or ne2000 clone card. The second most common 8390 based card are wd80x3 cards and clones. Cards with a DP83905 can be configured to be an ne2000 *or* a wd8013. Never versions of the genuine wd80x3 and SMC Elite cards have an 83c690 in place of the original DP8390. The SMC Ultra cards have an 83c790, and use a slightly different driver than the wd80x3 cards. The SMC EtherEZ cards have an 83c795, and use the same driver as the SMC Ultra. All BNC cards based on some sort of 8390 or 8390 clone will usually have an 8392 (or 83c692, or ???392) 16 pin DIP chip very close to the BNC connector.

Another common NIC found on older cards is the Intel i82586. Cards having this NIC include the 3c505, 3c507, 3c523, Intel EtherExpress−ISA, Microdyne Exos−205T, and the Racal−Interlan NI5210.

The original AMD LANCE NIC was numbered AM7990, and newer revisions include the 79c960, 79c961, 79c965, 79c970, and 79c974. Most cards with one of the above will work with the Linux LANCE driver, with the exception of the old Racal−Interlan NI6510 cards that have their own driver.

Newer PCI cards having a DEC 21040, 21041, 21140, or similar number on the NIC should be able to use the linux tulip or de4x5 driver.

Other PCI cards having a big chip marked RTL8029 or 89C940 or 86C926 are ne2000 clone cards, and the ne driver in linux version v2.0 and up should automatically detect these cards at boot.

Identifying the Ethernet Address

Each ethernet card has its own six byte address that is unique to that card. The first three bytes of that address are the same for each card made by that particular manufacturer. For example all SMC cards start with `00:00:c0`. The last three are assigned by the manufacturer uniquely to each individual card as they are produced.

If your card has a sticker on it giving all six bits of its address, you can look up the vendor from the first three. However it is more common to see only the last three bytes printed onto a sticker attached to a socketed PROM, which tells you nothing.

You can determine which vendors have which assigned addresses from RFC−1340. Apparently there is a more up to date listing available in various places as well. Try a WWW or FTP search for `EtherNet-codes` or `Ethernet-codes` and you will find something.

Identifying the Card by the FCC ID Number

As part of the certification process a card typically has to pass before being sold to the user, it gets tested by the FCC, and from this gets a FCC ID which is supposed to be printed on the card somewhere. For example, a card has on it `FCC ID: J158013EWC` − and this card happens to be a SMC/WD8013−EWC. Some web sites like `www.driverguide.com` and `drdriver.com` make use of listings of FCC IDs that may help with less obvious ID numbers.

Tips on Trying to Use an Unknown Card

If you are still not sure what the card is, but have at least narrowed it down some, then you can build a kernel with a whole bunch of drivers included, and see if any of them autodetect the card at boot.

If the kernel doesn't detect the card, it may be that the card is not configured to one of the addresses that the driver probes when looking for a card. In this case, you might want to try getting `scanport.tar.gz` from your local linux ftp site, and see if that can locate where your card is jumpered for. It scans ISA I/O space from `0x100` to `0x3ff` looking for devices that aren't registered in `/proc/ioports`. If it finds an unknown device starting at some particular address, you can then explicity point the ethernet probes at that address with an `ether=` boot argument.

If you manage to get the card detected, you can then usually figure out the unknown jumpers by changing them one at a time and seeing at what I/O base and IRQ that the card is detected at. The IRQ settings can also usually be determined by following the traces on the back of the card to where the jumpers are soldered through. Counting the `gold fingers' on the backside, from the end of the card with the metal bracket, you have IRQ 9, 7, 6, 5, 4, 3, 10, 11, 12, 15, 14 at fingers 4, 21, 22, 23, 24, 25, 34, 35, 36, 37, 38 respectively. Eight bit cards only have up to finger 31.

Jumpers that appear to do nothing usually are for selecting the memory address of an optional boot ROM. Other jumpers that are located near the BNC or RJ−45 or AUI connectors are usually to select the output media. These are also typically near the `black box' voltage converters marked YCL, Valor, or Fil−Mag.

A nice collection of jumper settings for various cards can be found at the following URL:

Ethercard Settings

4.45 Drivers for Non−Ethernet Devices

There are a few other drivers that are in the linux source that present an *ethernet−like* device to network programs, while not really being ethernet. These are briefly listed here for completeness.

`dummy.c` − The purpose of this driver is to provide a device to point a route through, but not to actually transmit packets.

`eql.c` – Load Equalizer, enslaves multiple devices (usually modems) and balances the Tx load across them while presenting a single device to the network programs.

`ibmtr.c` – IBM Token Ring, which is not really ethernet. Broken−Ring requires source routing and other uglies.

`loopback.c` – Loopback device, for which all packets from your machine and destined for your own machine go. It essentially just moves the packet off the Tx queue and onto the Rx queue.

`pi2.c` – Ottawa Amateur Radio Club PI and PI2 interface.

`plip.c` – Parallel Line Internet Protocol, allows two computers to send packets to each other over two joined parallel ports in a point−to−point fashion.

`ppp.c` – Point−to−Point Protocol (RFC1331), for the Transmission of Multi−protocol Datagrams over a Point−to−Point Link (again usually modems).

`slip.c` – Serial Line Internet Protocol, allows two computers to send packets to each other over two joined serial ports (usually via modems) in a point−to−point fashion.

`tunnel.c` – Provides an IP tunnel through which you can tunnel network traffic transparently across subnets

`wavelan.c` – An Ethernet−like radio transceiver controlled by the Intel 82586 coprocessor which is used on other ethercards such as the Intel EtherExpress.

---

5. Cables, Coax, Twisted Pair

If you are starting a network from scratch, you will have to decide whether to use thin ethernet (RG58 co−ax cable with BNC connectors) or 10baseT (twisted pair telco−style cables with RJ−45 eight wire `phone' connectors). The old−fashioned thick ethernet, RG−5 cable with N connectors is obsolete and rarely seen anymore.

See Type of cable... for an introductory look at cables. Also note that the FAQ from *comp.dcom.lans.ethernet* has a lot of useful information on cables and such. FTP to rtfm.mit.edu and look in `/pub/usenet-by-hierarchy/` for the FAQ for that newsgroup.

5.1 Thin Ethernet (thinnet)

Thin ethernet cable is pretty inexpensive. If you are making your own cables solid−core RG58A is $0.27/m. and stranded RG58AU is $0.45/m. Twist−on BNC connectors are < $2 ea., and other misc. pieces are similarly inexpensive. It is essential that you properly terminate each end of the cable with 50 ohm terminators, so budget $2 ea. for a pair. It's also vital that your cable have no `stubs' −− the `T' connectors must be attached directly to the ethercards.

There are two main drawbacks to using thinnet. The first is that it is limited to 10Mb/sec – 100Mb/sec requires twisted pair. The second drawback is that if you have a big loop of machines connected together, and some bonehead breaks the loop by taking one cable off the side of his tee, the whole network goes down because it sees an infinite impedance (open circuit) instead of the required 50 ohm termination. Note that you can remove the tee piece from the card itself without killing the whole subnet, as long as you don't remove the cables from the tee itself. Of course this will disturb the machine that you pull the actual tee off of. 8−) And if you are doing a small network of two machines, you *still* need the tees and the 50 ohm terminators −− you *can't* just cable them together!

There are also some fancy cable systems which *look like* a single lead going to the card, but the lead is actually two runs of cable laying side−by−side covered by an outer sheath, giving the lead an oval shaped cross−section. At the turnaround point of the loop, a BNC connector is spliced in which connects to your card. So you have the equivalent of two runs of cable and a BNC T, but in this case, it is impossible for the user to remove a cable from one side of the T and disturb the network.

5.2 Twisted Pair

Twisted pair networks require active hubs, which start around $50, and the raw cable cost can actually be higher than thinnet. You can pretty much ignore claims that you can use your existing telephone wiring as it is a rare installation where that turns out to be the case.

On the other hand, all 100Mb/sec ethernet proposals use twisted pair, and most new business installations use twisted pair. Also, Russ Nelson adds that `New installations should use Category 5 wiring. Anything else is a waste of your installer's time, as 100Base−whatever is going to require Cat 5.'

If you are only connecting two machines, it is possible to avoid using a hub, by swapping the Rx and Tx pairs (1−2 and 3−6).

If you hold the RJ−45 connector facing you (as if you were going to plug it into your mouth) with the lock tab on the top, then the pins are numbered 1 to 8 from left to right. The pin usage is as follows:

```
Pin Number              Assignment
----------              ----------
1                       Output Data (+)
2                       Output Data (-)
3                       Input Data (+)
4                       Reserved for Telephone use
5                       Reserved for Telephone use
6                       Input Data (-)
7                       Reserved for Telephone use
8                       Reserved for Telephone use
```

If you want to make a cable, the following should spell it out for you. Differential signal pairs must be on the same twisted pair to get the required minimal impedance/loss of a UTP cable. If you look at the above table, you will see that 1+2 and 3+6 are the two sets of differential signal pairs. Not 1+3 and 2+6 !!!!!! At 10MHz, with short lengths, you *may* get away with such errors, if it is only over a short length. Don't even think about it at 100MHz.

For a normal patch cord, with ends `A' and `B', you want straight through pin−to−pin mapping, with the input and output each using a pair of twisted wires (for impedance issues). That means 1A goes to 1B, 2A goes to 2B, 3A goes to 3B and 6A goes to 6B. The wires joining 1A−1B and 2A−2B must be a twisted pair. Also the wires joining 3A−3B and 6A−6B must be another twisted pair.

Now if you don't have a hub, and want to make a `null cable', what you want to do is make the input of `A' be the output of `B' and the output of `A' be the input of `B', without changing the polarity. Tha means connecting 1A to 3B (out+ A to in+ B) and 2A to 6B (out− A to in− B). These two wires must be a twisted pair. They carry what card/plug `A' considers output, and what is seen as input for card/plug `B'. Then connect 3A to 1B (in+ A to out+ B) and also connect 6A to 2B (in− A to out− B). These second two must also be a twisted pair. They carry what card/plug `A' considers input, and what card/plug `B' considers output.

So, if you consider a normal patch cord, chop one end off of it, swap the places of the Rx and Tx twisted pairs into the new plug, and crimp it down, you then have a `null' cable. Nothing complicated. You just want to feed the Tx signal of one card into the Rx of the second and vice versa.

Note that before 10BaseT was ratified as a standard, there existed other network formats using RJ−45 connectors, and the same wiring scheme as above. Examples are SynOptics's LattisNet, and AT&T's StarLAN. In some cases, (as with early 3C503 cards) you could set jumpers to get the card to talk to hubs of different types, but in most cases cards designed for these older types of networks will not work with standard 10BaseT networks/hubs. (Note that if the cards also have an AUI port, then there is no reason as to why you can't use that, combined with an AUI to 10BaseT transceiver.)

5.3 Thick Ethernet

Thick ethernet is mostly obsolete, and is usually used only to remain compatible with an existing implementation. You can stretch the rules and connect short spans of thick and thin ethernet together with a passive $3 N−to−BNC connector, and that's often the best solution to expanding an existing thicknet. A correct (but expensive) solution is to use a repeater in this case.

6. Software Configuration and Card Diagnostics

In most cases, if the configuration is done by software, and stored in an EEPROM, you will usually have to boot DOS, and use the vendor supplied DOS program to set the cards IRQ, I/O, mem_addr and whatnot. Besides, hopefully it is something you will only be setting once. If you don't have the DOS software for your card, try looking on the WWW site of your card manufacturer. If you don't know the site name, take a guess at it, i.e. `www.my_vendor.com' where `my_vendor' is the name of your card manufacturer. This works for SMC, 3Com, and many *many* other manufacturers.

There are some cards for which Linux versions of the config utils exist, and they are listed here. Donald has written a few small card diagnostic programs that run under Linux. Most of these are a result of debugging tools that he has created while writing the various drivers. Don't expect fancy menu−driven interfaces. You will have to read the source code to use most of these. Even if your particular card doesn't have a corresponding diagnostic, you can still get some information just by typing `cat /proc/net/dev` −− assuming that your card was at least detected at boot.

In either case, you will have to run most of these programs as root (to allow I/O to the ports) and you probably want to shut down the ethercard before doing so by typing `ifconfig eth0 down` first.

6.1 Configuration Programs for Ethernet Cards

WD80x3 Cards

For people with wd80x3 cards, there is the program `wdsetup` which can be found in `wdsetup-0.6a.tar.gz` on Linux ftp sites. It is not being actively maintained, and has not been updated for quite a while. If it works fine for you then great, if not, use the DOS version that you should have got with your card. If you don't have the DOS version, you will be glad to know that the SMC setup/driver disks are available at SMC's ftp site. Of course, you *have* to have an EEPROM card to use this utility. Old, *old* wd8003 cards, and some wd8013 clones use jumpers to set up the card instead.

Digital / DEC Cards

The Digital EtherWorks 3 card can be configured in a similar fashion to the DOS program `NICSETUP.EXE`. David C. Davies wrote this and other tools for the EtherWorks 3 in conjunction with the driver. Look on you local linux FTP site in the directory `/pub/linux/system/Network/management` for the file that is named `ewrk3tools−X.XX.tar.gz`.

NE2000+ or AT/LANTIC Cards

Some Nat Semi DP83905 implementations (such as the AT/LANTIC and the NE2000+) are software configurable. (Note that these cards can also emulate a wd8013 card!) You can get the file `/pub/linux/setup/atlantic.c` from Donald's ftp server, `www.scyld.com` to configure this card. In addition, the configuration programs for the Kingston DP83905 cards seem to work with all cards, as they don't check for a vendor specific address before allowing you to use them. Follow the following URL: Kingston Software and get `20XX12.EXE` and `INFOSET.EXE`.

Be careful when configuring NE2000+ cards, as you can give them bad setting values which can cause problems. A typical example is accidentally enabling the boot ROM in the EEPROM (even if no ROM is installed) to a setting that conflicts with the VGA card. The result is a computer that just beeps at you when you turn it on and nothing appears on the screen.

You can typically recover from this by doing the following: Remove the card from the machine, and then boot and enter the CMOS setup. Change the `Display Adapter' to `Not Installed' and change the default boot drive to `A:' (your floppy drive). Also change the `Wait for F1 if any Error' to `Disabled'. This way, the computer should boot without user intervention. Now create a bootable DOS floppy (`format a: /s /u') and copy the program `default.exe` from the `20XX12.EXE` archive above onto that floppy. Then type `echo default > a:autoexec.bat` so that the program to set the card back to sane defaults will be run automatically when you boot from this floppy. Shut the machine off, re−install the ne2000+ card, insert your new boot floppy, and power it back up. It will still probably beep at you, but eventually you should see the floppy light come on as it boots from the floppy. Wait a minute or two for the floppy to stop, indicating that it has finished running the `default.exe` program, and then power down your computer. When you then turn it on again, you should hopefully have a working display again, allowing you to change your CMOS settings back, and to change the card's EEPROM settings back to the values you want.

Note that if you don't have DOS handy, you can do the whole method above with a linux boot disk that automatically runs Donald's `atlantic` program (with the right command line switches) instead of a DOS boot disk that automatically runs the `default.exe` program.

3Com Cards

The 3Com Etherlink III family of cards (i.e. 3c5x9) can be configured by using another config utility from Donald. You can get the file `3c5x9setup.c` from Donald's ftp server, `www.scyld.com` to configure these cards. (Note that the DOS 3c5x9B config utility may have more options pertaining to the new ``B'' series of the Etherlink III family.)

6.2 Diagnostic Programs for Ethernet Cards

Any of the diagnostic programs that Donald has written can be obtained from his website.

Ethercard Diagnostics

Allied Telesis AT1700 −− `at1700.c`

Cabletron E21XX −− `e21.c`

HP PCLAN+ −− `hp+.c`

Intel EtherExpress −− `eexpress.c`

PCI NE2000 cards −− `ne2k−pci−diag.c`

ISA NE2000 cards −− `ne2k.c`

RealTek (ATP) Pocket adaptor `atp−diag.c`

All Other Cards −− try typing `cat /proc/net/dev` and `dmesg` to see what useful info the kernel has on the card in question.

---

7. Technical Information

For those who want to understand a bit more about how the card works, or play with the present drivers, or even try to make up their own driver for a card that is presently unsupported, this information should be useful. If you do not fall into this category, then perhaps you will want to skip this section.

7.1 Programmed I/O vs. Shared Memory vs. DMA

If you can already send and receive back−to−back packets, you just can't put more bits over the wire. Every modern ethercard can receive back−to−back packets. The Linux DP8390 drivers (wd80x3, SMC−Ultra, 3c503, ne2000, etc) come pretty close to sending back−to−back packets (depending on the current interrupt latency) and the 3c509 and AT1500 hardware have no problem at all automatically sending back−to−back packets.

Programmed I/O (e.g. NE2000, 3c509)

Pro: Doesn't use any constrained system resources, just a few I/O registers, and has no 16M limit.

Con: Usually the slowest transfer rate, the CPU is waiting the whole time, and interleaved packet access is usually difficult to impossible.

Shared memory (e.g. WD80x3, SMC−Ultra, 3c503)

Pro: Simple, faster than programmed I/O, and allows random access to packets. Where possible, the linux drivers compute the checksum of incoming IP packets as they are copied off the card, resulting in a further reduction of CPU usage vs. an equivalent PIO card.

Con: Uses up memory space (a big one for DOS users, essentially a non−issue under Linux), and it still ties up the CPU.

Slave (normal) Direct Memory Access (e.g. none for Linux!)

Pro: Frees up the CPU during the actual data transfer.

Con: Checking boundary conditions, allocating contiguous buffers, and programming the DMA registers makes it the slowest of all techniques. It also uses up a scarce DMA channel, and requires aligned low memory buffers.

Bus Master Direct Memory Access (e.g. LANCE, DEC 21040)

Pro: Frees up the CPU during the data transfer, can string together buffers, can require little or no CPU time lost on the ISA bus. Most of the bus−mastering linux drivers now use a `copybreak' scheme where large packets are put directly into a kernel networking buffer by the card, and small packets are copied by the CPU which primes the cache for subsequent processing.

Con: (Only applicable to ISA bus cards) Requires low−memory buffers and a DMA channel for cards. Any bus−master will have problems with other bus−masters that are bus−hogs, such as some primitive SCSI adaptors. A few badly−designed motherboard chipsets have problems with bus−masters. And a reason for not using *any* type of DMA device is using a 486 processor designed for plug−in replacement of a 386: these processors must flush their cache with each DMA cycle. (This includes the Cx486DLC, Ti486DLC, Cx486SLC, Ti486SLC, etc.)

7.2 Performance Implications of Bus Width

The ISA bus can do 5.3MB/sec (42Mb/sec), which sounds like more than enough for 10Mbps ethernet. In the case of the 100Mbps cards, you clearly need a faster bus to take advantage of the network bandwidth.

ISA Eight bit vs ISA 16 bit Cards

You probably can't buy a new 8 bit ISA ethercard anymore, but you will find lots of them turning up at computer swap meets and the like for the next few years, at very low prices. This will make them popular for ``home−ethernet'' systems. The above holds true for 16 bit ISA cards now as well, since PCI cards are now very common.

Some 8 bit cards that will provide adequate performance for light to average use are the wd8003, the 3c503 and the ne1000. The 3c501 provides poor performance, and these poor 12 year old relics of the XT days should be avoided. (Send them to Alan, he collects them...)

The 8 bit data path doesn't hurt performance that much, as you can still expect to get about 500 to 800kB/s ftp download speed to an 8 bit wd8003 card (on a fast ISA bus) from a fast host. And if most of your net−traffic is going to remote sites, then the bottleneck in the path will be elsewhere, and the only speed difference you will notice is during net activity on your local subnet.

7.3 32 Bit (VLB/EISA/PCI) Ethernet Cards

Note that a 10Mbs network typically doesn't justify requiring a 32 bit interface. See Programmed I/O vs. ... as to why having a 10Mbps ethercard on an 8MHz ISA bus is really not a bottleneck. Even though having the ethercard on a fast bus won't necessarily mean faster transfers, it will usually mean reduced CPU overhead, which is good for multi−user systems. Of course for 100Mbps networks, which are now commonplace, the 32 bit interface is a must to make use of the full bandwidth.

7.4 Writing a Driver

The only thing that one needs to use an ethernet card with Linux is the appropriate driver. For this, it is essential that the manufacturer will release the technical programming information to the general public without you (or anyone) having to sign your life away. A good guide for the likelihood of getting documentation (or, if you aren't writing code, the likelihood that someone else will write that driver you really, really need) is the availability of the Crynwr (nee Clarkson) packet driver. Russ Nelson runs this operation, and has been very helpful in supporting the development of drivers for Linux. *Net−surfers* can try this URL to look up Russ' software.

Russ Nelson's Packet Drivers

Given the documentation, you can write a driver for your card and use it for Linux (at least in theory). Keep in mind that some old hardware that was designed for XT type machines will not function very well in a multitasking environment such as Linux. Use of these will lead to major problems if your network sees a reasonable amount of traffic.

Most cards come with drivers for MS−DOS interfaces such as NDIS and ODI, but these are useless for Linux. Many people have suggested directly linking them in or automatic translation, but this is nearly impossible. The MS−DOS drivers expect to be in 16 bit mode and hook into `software interrupts', both incompatible with the Linux kernel. This incompatibility is actually a feature, as some Linux drivers are considerably better than their MS−DOS counterparts. The `8390' series drivers, for instance, use ping−pong transmit buffers, which are only now being introduced in the MS−DOS world.

(Ping−pong Tx buffers means using at least 2 max−size packet buffers for Tx packets. One is loaded while the card is transmitting the other. The second is then sent as soon as the first finished, and so on. In this way, most cards are able to continuously send back−to−back packets onto the wire.)

OK. So you have decided that you want to write a driver for the Foobar Ethernet card, as you have the programming information, and it hasn't been done yet. (...these are the two main requirements ;−) You should start with the skeleton network driver that is provided with the Linux kernel source tree. It can be found in the file `linux/drivers/net/skeleton.c` in all recent kernels. In 2.4.x (and newer) kernels it has been renamed to `isa−skeleton.c` Also have a look at the Kernel Hackers Guide, at the following URL: KHG

7.5 Driver interface to the kernel

Here are some notes on the functions that you would have to write if creating a new driver. Reading this in conjunction with the above skeleton driver may help clear things up.

Probe

Called at boot to check for existence of card. Best if it can check un−obtrsively by reading from memory, etc. Can also read from I/O ports. Initial writing to I/O ports in a probe is *not good* as it may kill another device. Some device initialization is usually done here (allocating I/O space, IRQs,filling in the dev−>??? fields etc.) You need to know what io ports/mem the card can be configured to, how to enable shared memory (if used) and how to select/enable interrupt generation, etc.

Interrupt handler

Called by the kernel when the card posts an interrupt. This has the job of determining why the card posted an interrupt, and acting accordingly. Usual interrupt conditions are data to be rec'd, transmit completed, error conditions being reported. You need to know any relevant interrupt status bits so that you can act accordingly.

Transmit function

Linked to dev−>hard_start_xmit() and is called by the kernel when there is some data that the kernel wants to put out over the device. This puts the data onto the card and triggers the transmit. You need to know how to bundle the data and how to get it onto the card (shared memory copy, PIO transfer, DMA?) and in the right place on the card. Then you need to know how to tell the card to send the data down the wire, and (possibly) post an interrupt when done. When the hardware can't accept additional packets it should set the dev−>tbusy flag. When additional room is available, usually during a transmit−complete interrupt, dev−>tbusy should be cleared and the higher levels informed with `mark_bh(INET_BH)`.

Receive function

Called by the kernel interrupt handler when the card reports that there is data on the card. It pulls the data off the card, packages it into a sk_buff and lets the kernel know the data is there for it by doing a netif_rx(sk_buff). You need to know how to enable interrupt generation upon Rx of data, how to check any relevant Rx status bits, and how to get that data off the card (again sh mem, PIO, DMA, etc.)

Open function

linked to dev−>open and called by the networking layers when somebody does `ifconfig eth0 up` − this puts the device on line and enables it for Rx/Tx of data. Any special initialization incantations that were not done in the probe sequence (enabling IRQ generation, etc.) would go in here.

Close function (optional)

This puts the card in a sane state when someone does `ifconfig eth0 down`. It should free the IRQs and DMA channels if the hardware permits, and turn off anything that will save power (like the transceiver).

Miscellaneous functions

Things like a reset function, so that if things go south, the driver can try resetting the card as a last ditch effort. Usually done when a Tx times out or similar. Also a function to read the statistics registers of the card if so equipped.

7.6 Technical information from 3Com

If you are interested in working on drivers for 3Com cards, you can get technical documentation from 3Com. Cameron has been kind enough to tell us how to go about it below:

3Com's Ethernet Adapters are documented for driver writers in our `Technical References' (TRs). These manuals describe the programmer interfaces to the boards but they don't talk about the diagnostics, installation programs, etc that end users can see.

The Interface Products Group marketing department has the TRs to give away. To keep this program efficient, we centralized it in a thing called `CardFacts.' CardFacts is an automated phone system. You call it with a touch−tone phone and it faxes you stuff. To get a TR, call CardFacts at 408−727−7021. Ask it for Developer's Order Form, document number 9070. Have your fax number ready when you call. Fill out the order form and fax it to 408−764−5004. Manuals are shipped by Federal Express 2nd Day Service.

There are people here who think we are too free with the manuals, and they are looking for evidence that the system is too expensive, or takes too much time and effort. So far, 3Com customers have been really good about this, and there's no problem with the level of requests we've been getting. We need your continued cooperation and restraint to keep it that way.

7.7 Notes on AMD PCnet / LANCE Based cards

The AMD LANCE (Local Area Network Controller for Ethernet) was the original offering, and has since been replaced by the `PCnet−ISA' chip, otherwise known as the 79C960. Note that the name `LANCE' has stuck, and some people will refer to the new chip by the old name. Dave Roberts of the Network Products Division of AMD was kind enough to contribute the following information regarding this chip:

`Functionally, it is equivalent to a NE1500. The register set is identical to the old LANCE with the 1500/2100 architecture additions. Older 1500/2100 drivers will work on the PCnet−ISA. The NE1500 and NE2100 architecture is basically the same. Initially Novell called it the 2100, but then tried to distinguish between coax and 10BASE−T cards. Anything that was 10BASE−T only was to be numbered in the 1500 range. That's the only difference.

Many companies offer PCnet−ISA based products, including HP, Racal−Datacom, Allied Telesis, Boca Research, Kingston Technology, etc. The cards are basically the same except that some manufacturers have added `jumperless' features that allow the card to be configured in software. Most have not. AMD offers a standard design package for a card that uses the PCnet−ISA and many manufacturers use our design without change. What this means is that anybody who wants to write drivers for most PCnet−ISA based cards can just get the data−sheet from AMD. Call our literature distribution center at (800)222−9323 and ask for the Am79C960, PCnet−ISA data sheet. It's free.

A quick way to understand whether the card is a `stock' card is to just look at it. If it's stock, it should just have one large chip on it, a crystal, a small IEEE address PROM, possibly a socket for a boot ROM, and a connector (1, 2, or 3, depending on the media options offered). Note that if it's a coax card, it will have some transceiver stuff built onto it as well, but that should be near the connector and away from the PCnet−ISA.'

A note to would−be card hackers is that different LANCE implementations do `restart' in different ways. Some pick up where they left off in the ring, and others start right from the beginning of the ring, as if just initialised.

7.8 Multicast and Promiscuous Mode

Another one of the things Donald has worked on is implementing multicast and promiscuous mode hooks. All of the *released* (i.e. **not** ALPHA) ISA drivers now support promiscuous mode.

Donald writes: `I'll start by discussing promiscuous mode, which is conceptually easy to implement. For most hardware you only have to set a register bit, and from then on you get every packet on the wire. Well, it's almost that easy; for some hardware you have to shut the board (potentially dropping a few packet), reconfigure it, and then re−enable the ethercard. OK, so that's easy, so I'll move on something that's not quite so obvious: Multicast. It can be done two ways:

1. Use promiscuous mode, and a packet filter like the Berkeley packet filter (BPF). The BPF is a pattern matching stack language, where you write a program that picks out the addresses you are interested in. Its advantage is that it's very general and programmable. Its disadvantage is that there is no general way for the kernel to avoid turning on promiscuous mode and running every packet on the wire through every registered packet filter. See The Berkeley Packet Filter for more info.
2. Using the built−in multicast filter that most etherchips have.

I guess I should list what a few ethercards/chips provide:

```
Chip/card  Promiscuous  Multicast filter
----------------------------------------
Seeq8001/3c501  Yes     Binary filter (1)
3Com/3c509      Yes     Binary filter (1)
8390            Yes     Autodin II six bit hash (2) (3)
LANCE           Yes     Autodin II six bit hash (2) (3)
i82586          Yes     Hidden Autodin II six bit hash (2) (4)
```

1. These cards claim to have a filter, but it's a simple yes/no `accept all multicast packets', or `accept no multicast packets'.
2. AUTODIN II is the standard ethernet CRC (checksum) polynomial. In this scheme multicast addresses are hashed and looked up in a hash table. If the corresponding bit is enabled, this packet is accepted. Ethernet packets are laid out so that the hardware to do this is trivial −− you just latch six (usually) bits from the CRC circuit (needed anyway for error checking) after the first six octets (the destination address), and use them as an index into the hash table (six bits −− a 64−bit table).
3. These chips use the six bit hash, and must have the table computed and loaded by the host. This means the kernel must include the CRC code.
4. The 82586 uses the six bit hash internally, but it computes the hash table itself from a list of multicast addresses to accept.

Note that none of these chips do perfect filtering, and we still need a middle−level module to do the final filtering. Also note that in every case we must keep a complete list of accepted multicast addresses to recompute the hash table when it changes.

7.9 The Berkeley Packet Filter (BPF)

The general idea of the developers is that the BPF functionality should not be provided by the kernel, but should be in a (hopefully little−used) compatibility library.

For those not in the know: BPF (the Berkeley Packet Filter) is an mechanism for specifying to the kernel networking layers what packets you are interested in. It's implemented as a specialized stack language interpreter built into a low level of the networking code. An application passes a program written in this language to the kernel, and the kernel runs the program on each incoming packet. If the kernel has multiple BPF applications, each program is run on each packet.

The problem is that it's difficult to deduce what kind of packets the application is really interested in from the packet filter program, so the general solution is to always run the filter. Imagine a program that registers a BPF program to pick up a low data−rate stream sent to a multicast address. Most ethernet cards have a hardware multicast address filter implemented as a 64 entry hash table that ignores most unwanted multicast packets, so the capability exists to make this a very inexpensive operation. But with the BPF the kernel must switch the interface to promiscuous mode, receive _all_ packets, and run them through this filter. This is work, BTW, that's very difficult to account back to the process requesting the packets.

---

8. Networking with a Laptop/Notebook Computer

There are several ways to put your laptop on a network. You can use the SLIP code (and run at serial line speeds); you can get a notebook with a supported PCMCIA slot built−in; you can get a laptop with a docking station and plug in an ISA ethercard; or you can use a parallel port Ethernet adapter.

8.1 Using SLIP

This is the cheapest solution, but by far the most difficult. Also, you will not get very high transmission rates. Since SLIP is not really related to ethernet cards, it will not be discussed further here. See the NET−2 Howto.

8.2 PCMCIA Support

Try and determine exactly what hardware you have (ie. card manufacturer, PCMCIA chip controller manufacturer) and then ask on the LAPTOPS channel. Regardless, don't expect things to be all that simple. Expect to have to fiddle around a bit, and patch kernels, etc. Maybe someday you will be able to type `make config' 8−)

At present, the two PCMCIA chipsets that are supported are the Databook TCIC/2 and the intel i82365.

There is a number of programs on tsx−11.mit.edu in /pub/linux/packages/laptops/ that you may find useful. These range from PCMCIA Ethercard drivers to programs that communicate with the PCMCIA controller chip. Note that these drivers are usually tied to a specific PCMCIA chip (ie. the intel 82365 or the TCIC/2)

For NE2000 compatible cards, some people have had success with just configuring the card under DOS, and then booting linux from the DOS command prompt via `loadlin`.

Things are looking up for Linux users that want PCMCIA support, as substantial progress is being made. Pioneering this effort is David Hinds. His latest PCMCIA support package can be obtained from:

PCMCIA Package

Look for a file like `pcmcia-cs-X.Y.Z.tgz` where X.Y.Z will be the latest version number. This is most likely uploaded to the `tsx-11.mit.edu` FTP site as well.

Note that Donald's PCMCIA enabler works as a user−level process, and David Hinds' is a kernel−level solution. You may be best served by David's package as it is much more widely used and under continuous development.

8.3 ISA Ethercard in the Docking Station.

Docking stations for laptops typically cost about $250 and provide two full−size ISA slots, two serial and one parallel port. Most docking stations are powered off of the laptop's batteries, and a few allow adding extra batteries in the docking station if you use short ISA cards. You can add an inexpensive ethercard and enjoy full−speed ethernet performance.

8.4 Pocket / parallel port adaptors.

The `pocket' ethernet adaptors may also fit your need. Note that the transfer speed will not be all that great (perhaps 200kB/s tops?) due to the limitations of the parallel port interface.

Also most tie you down with a wall−brick power supply. You can sometimes avoid the wall−brick with the adaptors by buying or making a cable that draws power from the laptop's keyboard port. (See keyboard power)

See DE−600 / DE−620 and RealTek for two supported pocket adaptors.

---

9. Miscellaneous.

Any other associated stuff that didn't fit in anywhere else gets dumped here. It may not be relevant, and it may not be of general interest but it is here anyway.

9.1 Passing Ethernet Arguments to the Kernel

Here are two generic kernel commands that can be passed to the kernel at boot time (`ether` and `reserve`). This can be done with LILO, loadlin, or any other booting utility that accepts optional arguments.

For example, if the command was `blah' and it expected 3 arguments (say 123, 456, and 789) then, with LILO, you would use:

```
LILO: linux blah=123,456,789
```

For more information on (and a complete list of) boot time arguments, please see the BootPrompt−HOWTO

The `ether` command

The `ether=` argument is used in conjunction with drivers that are directly built into the kernel. The `ether=` argument will have *absolutely no effect* on a modular driver. In its most generic form, it looks something like this:

```
ether=IRQ,BASE_ADDR,PARAM_1,PARAM_2,NAME
```

All arguments are optional. The first non−numeric argument is taken as the NAME.

IRQ: Obvious. An IRQ value of `0' (usually the default) means to autoIRQ. It's a historical accident that the IRQ setting is first rather than the base_addr −− this will be fixed whenever something else changes.

BASE_ADDR: Also obvious. A value of `0' (usually the default) means to probe a card−type−specific address list for an ethercard.

PARAM_1: It was orginally used as an override value for the memory start for a shared−memory ethercard, like the WD80*3. Some drivers use the low four bits of this value to set the debug message level. 0 −− default, 1−7 −− level 1..7, (7 is maximum verbosity) 8 −− level 0 (no messages). Also, the LANCE driver uses the low four bits of this value to select the DMA channel. Otherwise it uses auto−DMA.

PARAM_2: The 3c503 driver uses this to select between the internal and external transceivers. 0 −− default/internal, 1 −− AUI external. The Cabletron E21XX card also uses the low 4 bits of PARAM_2 to select the output media. Otherwise it detects automatically.

NAME: Selects the network device the values refer to. The standard kernel uses the names `eth0', `eth1', `eth2' and `eth3' for bus−attached ethercards, and `atp0' for the parallel port `pocket' ethernet adaptor. The arcnet driver uses `arc0' as its name. The default setting is for a single ethercard to be probed for as `eth0'. Multiple cards can only be enabled by explicitly setting up their base address using these LILO parameters. The 1.0 kernel has LANCE−based ethercards as a special case. LILO arguments are ignored, and LANCE cards are always assigned `eth<n>' names starting at `eth0'. Additional non−LANCE ethercards must be explicitly assigned to `eth<n+1>', and the usual `eth0' probe disabled with something like `ether=0,−1,eth0'. (Yes, this is bug.)

The `reserve` command

This next lilo command is used just like `ether=' above, ie. it is appended to the name of the boot select specified in lilo.conf

```
reserve=IO-base,extent{,IO-base,extent...}
```

In some machines it may be necessary to prevent device drivers from checking for devices (auto−probing) in a specific region. This may be because of poorly designed hardware that causes the boot to *freeze* (such as some ethercards), hardware that is mistakenly identified, hardware whose state is changed by an earlier probe, or merely hardware you don't want the kernel to initialize.

The `reserve` boot−time argument addresses this problem by specifying an I/O port region that shouldn't be probed. That region is reserved in the kernel's port registration table as if a device has already been found in that region. Note that this mechanism shouldn't be necessary on most machines. Only when there is a problem or special case would it be necessary to use this.

The I/O ports in the specified region are protected against device probes. This was put in to be used when some driver was hanging on a NE2000, or misidentifying some other device as its own. A correct device driver shouldn't probe a reserved region, unless another boot argument explicitly specifies that it do so. This implies that `reserve` will most often be used with some other boot argument. Hence if you specify a `reserve` region to protect a specific device, you must generally specify an explicit probe for that device. Most drivers ignore the port registration table if they are given an explicit address.

For example, the boot line

```
LILO: linux reserve=0x300,32 ether=0,0x300,eth0
```

keeps all device drivers except the ethercard drivers from probing 0x300−0x31f.

As usual with boot−time specifiers there is an 11 parameter limit, thus you can only specify 5 reserved regions per `reserve` keyword. Multiple `reserve` specifiers will work if you have an unusually complicated request.

9.2 Using the Ethernet Drivers as Modules

Most of the linux distributions now ship kernels that have very few drivers built−in. The drivers are instead supplied as a bunch of independent dynamically loadable modules. These modular drivers are typically loaded by the administrator with the `modprobe(8)` command, or in some cases they are automatically loaded by the kernel through `kerneld' (in 2.0) or `kmod' (in 2.1) which then calls `modprobe`.

You particular distribution may offer nice graphical configuration tools for setting up ethernet modules. If possible you should try and use them first. The description that follows here gives information on what underlies any fancy configuration program, and what these programs change.

The information that controls what modules are to be used and what options are supplied to each module is usually stored in the file `/etc/conf.modules`. The two main options of interest (for ethernet cards) that will be used in this file are `alias` and `options`. The `modprobe` command consults this file for module information.

The actual modules themselves are typically stored in a directory named `/lib/modules/`uname −r`/net` where the `uname −r` command gives the kernel version (e.g. 2.0.34). You can look in there to see which module matches your card.

The first thing you need in your `conf.modules` file is something to tell `modprobe` what driver to use for the `eth0` (and `eth1` and...) network interface. You use the `alias` command for this. For example, if you have an ISA SMC EtherEZ card which uses the `smc−ultra.o` driver module, you need to `alias` this driver to `eth0` by adding the line:

```
alias eth0 smc-ultra
```

The other thing you may need is an `options` line indicating what options are to be used with a particular module (or module alias). Continuing with the above example, if you only used the single `alias` line with no `options` line, the kernel would warn you (see `dmesg`) that autoprobing for ISA cards is not a good idea. To get rid of this warning, you would add another line telling the module what I/O base the card is configured to, in this case say the hexidecimal address `0x280` for example.

```
options smc-ultra io=0x280
```

Most ISA modules accept parameters like `io=0x340` and `irq=12` on the `insmod` command line. It is *REQUIRED* or at least *STRONGLY ADVISED* that you supply these parameters to avoid probing for the card. Unlike PCI and EISA devices, there is no real safe way to do auto−probing for most ISA devices, and so it should be avoided when using drivers as modules.

A list of all the options that each module accepts can be found in the file:

`/usr/src/linux/Documentation/networking/net−modules.txt`

It is recommended that you read that to find out what options you can use for your particular card. Note that some modules support comma separated value lists for modules that have the capability to handle multiple devices from a single module, such as all the 8390 based drivers, and the PLIP driver. For exmple:

```
        options 3c503 io=0x280,0x300,0x330,0x350 xcvr=0,1,0,1
```

The above would have the one module controlling four 3c503 cards, with card 2 and 4 using external transcievers. Don't put spaces around the `=' or commas.

Also note that a *busy* module can't be removed. That means that you will have to `ifconfig eth0 down` (shut down the ethernet card) before you can remove the module(s).

The command `lsmod` will show you what modules are loaded, whether they are in use, and `rmmod` will remove them.

9.3 Related Documentation

Much of this info came from saved postings from the comp.os.linux groups, which shows that it is a valuable resource of information. Other useful information came from a bunch of small files by Donald himself. Of course, if you are setting up an Ethernet card, then you will want to read the NET−2 Howto so that you can actually configure the software you will use. Also, if you fancy yourself as a bit of a hacker, you can always scrounge some additional info from the driver source files as well. There is usually a paragraph or two in there describing any important points before any actual code starts..

For those looking for information that is not specific in any way to Linux (i.e. what is 10BaseT, what is AUI, what does a hub do, etc.) I strongly recommend making use of the newsgroup *comp.dcom.lans.ethernet* and/or *comp.sys.ibm.pc.hardware.networking*. Newsgroup archives such as those at `dejanews.com` can also be an invaluable source of information. You can grab the newsgroup FAQ from RTFM (which holds all the newsgroup FAQs) at the following URL:

Usenet FAQs

You can also have a look at the `Ethernet−HomePage' so to speak, which is at the following URL:

Ethernet−HomePage

9.4 Disclaimer and Copyright

This document is *not* gospel. However, it is probably the most up to date info that you will be able to find. Nobody is responsible for what happens to your hardware but yourself. If your ethercard or any other hardware goes up in smoke (...nearly impossible!) we take no responsibility. ie. THE AUTHORS ARE NOT RESPONSIBLE FOR ANY DAMAGES INCURRED DUE TO ACTIONS TAKEN BASED ON THE INFORMATION INCLUDED IN THIS DOCUMENT.

A hint to people considering doing a translation. First, translate the SGML source (available via FTP from the HowTo main site) so that you can then generate other output formats. Be sure to keep a copy of the original English SGML source that you translated from! When an updated HowTo is released, get the new SGML source for that version, and then a simple `diff −u old.sgml new.sgml` will show you exactly what has changed so that you can easily incorporate those changes into your translated SMGL source without having to re−read or re−translate everything.

If you are intending to incorporate this document into a published work, please make contact (via e−mail) so that you can be supplied with the most up to date information available. In the past, out of date versions of the Linux HowTo documents have been published, which caused the developers undue grief from being plagued with questions that were already answered in the up to date versions.

9.5 Closing

If you have found any glaring typos, or outdated info in this document, please send an e−mail. It is big, and it is easy to overlook stuff. If you have e−mailed about a change, and it hasn't been included in the next version, please don't hesitate to send it again, as it might have got lost amongst the usual sea of SPAM and junk mail I get.

Thanks!

Paul Gortmaker, `p_gortmaker@yahoo.com`
