LINUX for S/390

Device Drivers and Installation Commands

LINUX kernel 2.2.16

LNUX-1003-03

LINUX for S/390

Device Drivers and Installation Commands

LINUX kernel 2.2.16

LNUX-1003-03

Note

Before using this document, be sure to read the information in “Notices” on page 127.

Fourth Edition – (18 July 2001)

This edition applies to the fourth release of the LINUX for S/390 kernel 2.2.16 patch (made in June 2001) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Summary of changes vii
Edition 4 changes vii
Edition 3 changes vii
Edition 2 changes vii

About this book ix
How this book is organized ix
Who should read this book ix
Assumptions ix

Part 1. LINUX for S/390 Device drivers overview 1

Chapter 1. Common device support 3

Part 2. LINUX for S/390 — S/390 device drivers 5

Chapter 2. LINUX for S/390 DASD device driver 7
DASD overview 7
DASD naming scheme 7
Partitioned DASD 8
DASD features 9
DASD kernel parameter syntax 10
DASD kernel example 11
DASD module parameter syntax 12
DASD module example 12
DASD – Preparing for use 12
DASD restrictions 13

Chapter 3. LINUX for S/390 VM minidisk device driver 15
VM minidisk features 15
VM minidisk kernel parameter syntax 15
VM minidisk kernel example 15
VM minidisk – Preparing disks 15

Chapter 4. LINUX for S/390 XPRAM device driver 19
XPRAM features 19
Note on reusing XPRAM partitions 19
XPRAM kernel parameter syntax 20
XPRAM module parameter syntax 21

Chapter 5. LINUX for S/390 Console device drivers 23
Console features 23
Console kernel parameter syntax 24
Console kernel examples 24
Using the console 24
Console – Use of VInput 26
Console limitations 27

Chapter 6. Channel attached tape device driver 29
Tape driver features 29
Tape character device front-end 30
Tape block device front-end 30

Tape driver kernel parameter syntax 31
Tape driver kernel example 31
Tape driver module parameter syntax 32
Tape driver module example 32
Tape device driver API 33
Tape driver examples 33
Tape driver restrictions 34
Tape driver further information 34

Part 3. LINUX for S/390 Network device drivers 35

Chapter 7. LINUX for S/390 CTC/ESCON device driver 37
CTC/ESCON features 37
CTC/ESCON configuration 37
CTC/ESCON – Preparing the connection 40
CTC/ESCON – Recovery procedure after a crash 42

Chapter 8. LINUX for S/390 IUCV device driver 43
IUCV features 43
IUCV kernel parameter syntax 43
IUCV kernel parameter example 44
IUCV module parameter syntax 44
IUCV module parameter example 44
IUCV – Preparing the connection 45
IUCV – Further information 47
IUCV restrictions 47

Chapter 9. LINUX for S/390 LCS Device Driver 49
LCS features 49
LCS configuration 49
LCS restrictions 51
LCS limitations 51
LCS – Common set up problem 52

Chapter 10. LINUX for S/390 OSA-Express device driver 53
OSA-Express features 53
OSA-Express configuration 53
OSA-Express – Preparing the connection 57
OSA-Express device recognition 58
OSA-Express restrictions 59
OSA-Express queuing 59
OSA-Express IP Address Takeover 60
OSA-Express background – QDIO 61

Part 4. Installation commands and parameters 63

Chapter 11. Useful LINUX commands 65
dasdfmt - Format a DASD 66
ifconfig - Configure a network interface 69
insmod - Load a module into the LINUX kernel 73
modprobe - Load a module with dependencies into the LINUX kernel 75
lsmod - List loaded modules 78
depmod - Create dependency descriptions for loadable kernel modules 79
mke2fs - Create a file system on DASD 81
silo - Make DASD bootable 82

Chapter 12. Kernel parameters 85
ipldelay 86
maxcpus 87
mem 88
noinitrd 89
ramdisk_size 90
ro 91
root 92
vmhalt 93
cio_msg 94

Chapter 13. Overview of the parameter line file 95
Parameters 96

Appendix A. Reference information 97
LCS module parameter syntax 97
OSA-Express driver command syntax 97
LINUX for S/390 Device numbers 98

Appendix B. Kernel building 99
Building the kernel 100
Using 'config' or 'oldconfig' 103
Using 'menuconfig' 106
Kernel parameter options 115

Glossary 123

Notices 127
Trademarks 128

International License Agreement for Non-Warranted Programs 129

GNU General Public Licence, Version 2, June 1991 135
Preamble 135
GNU General Public Licence: Terms and conditions for copying, distribution and modification 136

Index 141

Summary of changes

This revision contains changes to support the LINUX for S/390 kernel loadable module for the LINUX kernel version 2.2.16.

Edition 4 changes

New Information

- Tape driver
- modprobe, lsmod, depmod summarized

Changed Information

This revision also includes maintenance and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

- DASD – Add commands for creating device nodes and more details of naming scheme
- XPRAM – note on reusing partitions
- Gigabit Ethernet section expanded for all OSA Express devices
- Console section expanded

Edition 3 changes

New Information

- Gigabit Ethernet driver restriction

This revision also includes maintenance and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Edition 2 changes

New Information

- CTC/ESCON VM channel subset selection for TCP/IP

Changed Information

- CTC/ESCON module parameter syntax
- VM Minidisk driver revisions
- 'mem' parameter additional option
- Console parameter change for P/390

About this book

This book describes the drivers available to LINUX for the control of S/390 devices and attachments.

The drivers described herein have been developed with version 2.2.16 of the LINUX kernel. If you are using a later version of the kernel, the kernel parameters may be different to those described in this document.

For more specific information about the device driver structure, see the documents in the kernel source tree at `...linux/Documentation/s390`.

When you have installed LINUX including the kernel sources this path will be on your machine. Typically: /usr/src/linux/Documentation/s390.

How this book is organized

The first part of this book contains general information relevant to all LINUX for S/390 device drivers.

Parts two and three consist of chapters specific to individual device drivers. (Part two describes the drivers for S/390 hardware; part three describes the network device drivers.)

Part four contains information on the LINUX and S/390 commands and parameters used in installing.

These chapters are followed by a reference section containing summaries of the command syntax of the drivers, a glossary and an index.

Who should read this book

This book is intended for :

- System administrators who wish to configure a LINUX for S/390 system

Assumptions

The following general assumptions are made about your background knowledge:

- You have an understanding of LINUX and S/390 terminology.
- You are familiar with LINUX device driver software.
- You have an understanding of basic computer architecture, operating systems, and programs.
- You are familiar with the S/390 devices attached to your system. (S/390 knowledge should not be required, as the code specific to the S/390 hardware is provided by IBM.)

Part 1. LINUX for S/390 Device drivers overview

This section describes principles common to different device drivers.

Chapter 1. Common device support

Before LINUX for S/390 can use a device the associated device driver must be available to the LINUX kernel. This can be achieved either by compiling the device driver into the kernel or by invoking the driver as a module. The options for each driver are shown in the following table:

Device driver	Kernel	Module
DASD	yes	yes
VM minidisk	yes	no
XPRAM	yes	yes
Hardware console	yes	no
3215 console	yes	no
Tape	yes	yes
CTC/ESCON	yes	yes
IUCV	yes	no
LCS	no	yes
OSA-Express	no	yes

A description of how to build the kernel including device drivers is given in “Appendix B. Kernel building” on page 99.

The parameters for the kernel resident device drivers are held in the parameter line file which is created during the installation of LINUX .

- If you are using an LPAR or native installation this is parameter `-p` in the `silo` parameter file.
- For a VM installation, include the parameter in the `PARM LINE A` file.

For the format of this file see “Chapter 13. Overview of the parameter line file” on page 95.

Drivers which are not kernel resident are loaded into LINUX with their parameters by means of the `insmod` or `modprobe` command. See “insmod - Load a module into the LINUX kernel” on page 73 or “modprobe - Load a module with dependencies into the LINUX kernel” on page 75 for the syntax.

Because the S/390 architecture differs from that used by the Intel PC and other machines the I/O concepts used by S/390 device drivers are also different.

LINUX was originally designed for the Intel PC architecture which uses two cascaded 8259 programmable interrupt controllers (PIC) that allow a maximum of 15 different interrupt lines. All devices attached to that type of system share those 15 interrupt levels (or IRQs). In addition, the bus systems (ISA, MCA, EISA, PCI, etc.) might allow shared interrupts, different polling methods or DMA processing.

Unlike other hardware architectures, ESA/390 implements a channel subsystem that provides a unified view of the devices attached to the system. Although a large variety of peripheral attachments are defined for the ESA/390 architecture, they are

all accessed in the same manner using I/O interrupts. Each device attached to the system is uniquely identified by a subchannel, and the ESA/390 architecture allows up to 64,000 devices to be attached.

To avoid the introduction of a new I/O concept to the common LINUX code, LINUX for S/390 preserves the IRQ concept and systematically maps the ESA/390 subchannels to LINUX as IRQs. This allows LINUX for S/390 to support up to 64,000 different IRQs, each representing a unique device.

The unified I/O access method incorporated in LINUX for S/390 allows the operating system to implement all of the hardware I/O attachment functionality that each device driver would otherwise have to provide itself. A common I/O device driver is provided which uses a functional layer to provide a generic access method to the hardware. The driver comprises a set of I/O support routines, some of which are common LINUX interfaces, while others are LINUX for S/390 specific:

get_dev_info()
: Allows a device driver to find out what devices are attached (visible) to the system, and to determine their current status.

request_irq()
: Assigns the ownership of a specific device to a device driver.

free_irq()
: Releases the ownership of a specific device.

disable_irq()
: Prevents a specific device from presenting interrupts to the device driver.

enable_irq()
: Allows a device to present I/O interrupts to the device driver.

do_IO()
: Initiates an I/O request.

halt_IO()
: Terminates the I/O request that is currently being processed by the device.

do_IRQ()
: This is an interrupt pre-processing routine that is called by the interrupt entry routine whenever an I/O interrupt is presented to the system. The `do_IO()` routine determines the interrupt status and calls the device specific interrupt handler according to the rules (flags) defined by `do_IO()`.

More information on these commands can be found in the LINUX source directory, `.../Documentation/s390/cds.txt`

Part 2. LINUX for S/390 — S/390 device drivers

The S/390 device drivers are:

- "Chapter 2. LINUX for S/390 DASD device driver" on page 7
- "Chapter 3. LINUX for S/390 VM minidisk device driver" on page 15
- "Chapter 4. LINUX for S/390 XPRAM device driver" on page 19
- "Chapter 5. LINUX for S/390 Console device drivers" on page 23
- "Chapter 6. Channel attached tape device driver" on page 29

Chapter 2. LINUX for S/390 DASD device driver

DASD overview

The DASD device driver in LINUX for S/390 takes care of all real or emulated DASD (Direct Access Storage Device) that can be attached to an S/390 system. The class of devices named DASD includes a variety of physical media, on which data is organized in blocks and/or records which can be accessed (read or written) in random order.

Traditionally these devices are attached to a control unit connected to an S/390 I/O channel. In modern systems these have been largely replaced by emulated DASD, such as the internal disks of the Multiprise family, the volumes of the RAMAC virtual array, or the volumes of the Enterprise Storage Server. These are completely virtual representations of DASD in which the identity of the physical device is hidden.

The driver can either be statically built into the kernel or loaded during run time as a module.

The DASD device driver is capable of accessing an arbitrary number of devices. The default major number for DASD (94) can only address 64 DASD (see below for details), so additional major numbers (typically descending from 254) are allocated dynamically at initialization or run time. The only practical limit to the number of DASD accessible is the range of major numbers available in the dynamic allocation pool.

Each DASD configured to the system uses 4 minor numbers.

- The first minor number always represents the entire device, including IPL and label records.
- The remaining three minor numbers represent partitions of the device as defined in the partition table.

DASD naming scheme

A LINUX 2.2 or 2.4 system is restricted to 256 major device numbers, each holding 64 blocks of 4 minor numbers, giving a maximum of 16,384 DASD even if no numbers are used for other types of device. Every major number used for other devices reduces the maximum number of DASD by 64. The DASD device driver has a built in naming scheme for DASD according to Table 1. (You can override the built in scheme by creating customized nodes in the LINUX `/dev/` subdirectory.) These names are sufficient to access the maximum number of DASD accessible.

Table 1. DASD naming convention

Names	Number	Major/minor numbers (assuming dynamic allocation from 254)
dasda – dasdz	26	94:0 — 94:100
dasdaa – dasdbl	38	94:104 — 94:252
dasdbm – dasdzz	638	254:0 — 245:244
dasdaaa – dasdzzz	17576	245:248 — 131:148
Sum:	18278	

General DASD nodes have the format dasd<*x*>, or dasd<*x*><*p*>, where <*x*> is a letter identifying the device and <*p*> is a number denoting the partition on that device. The first form, dasd<*x*>, is used to address the entire disk. The second, dasd<*x*><*p*>, is used to address the partitions on this device.

For example /dev/dasda refers to the whole of the first disk in the system and /dev/dasda1 to the first partition on that disk.

They are typically created by:

```
mknod -m 660 /dev/dasda  b 94 0
mknod -m 660 /dev/dasda1 b 94 1
mknod -m 660 /dev/dasda2 b 94 2
mknod -m 660 /dev/dasda3 b 94 3
mknod -m 660 /dev/dasdb  b 94 4
mknod -m 660 /dev/dasdb1 b 94 5
....
```

If you have a large number of DASD you may wish to use a script to create them. An example of this for bash is:

```
'cat /proc/dasd/devices |
 sed 's/^.*[(]\([ 0-9]*\)[:]\([ 0-9]*\)[)].*\(dasd[a-z]*\)[:].*$/\1 \2 \3/g' |
 awk ' $1 {
printf "mknod /dev/%s b %d %d; mknod /dev/%s1 b %d %d;",$3,$1,$2,$3,$1,$2+1;
 }''
```

A similar script may be written for csh or ksh.

Partitioned DASD

The DASD device driver is embedded into the LINUX generic support for partitioned disks. This implies that you can have any kind of partition table known to LINUX on your DASD, such as the MSDOS or Amiga partition scheme. However none of the partition schemes built in to LINUX to support platforms other than S/390 will preserve S/390 IPL and label records.

'IBM label' partition scheme:

To ensure compatibility with other S/390 operating systems the IBM-label partition scheme has been added to LINUX . This scheme currently supports LNX (LINUX) and CMS (VM/ESA) labelled disks, as well as unlabeled disks which are treated equivalently to LNX-labelled disks. The disk layout of the different types is shown in Figure 1 on page 9.

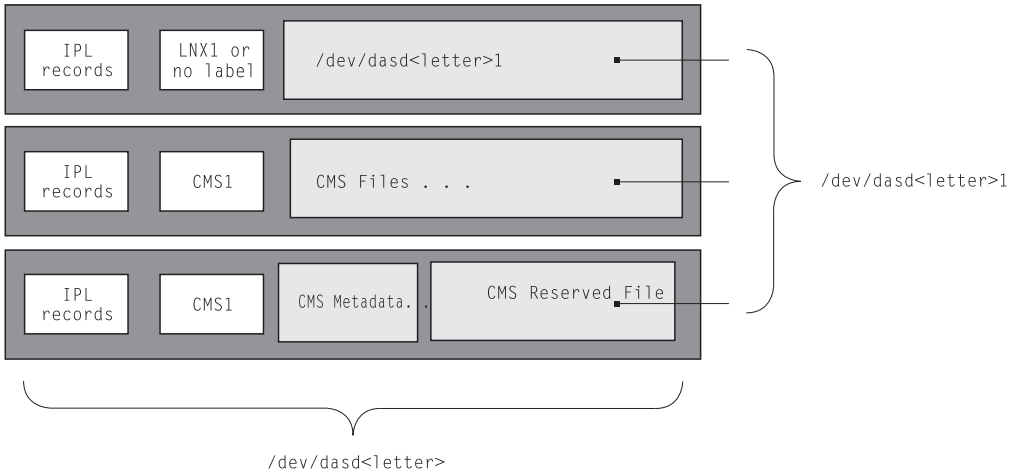


Figure 1. Partition scheme for LNX and CMS labelled disks

The first of these examples shows a disk in an LPAR or native mode, or a full pack minidisk (dedicated DASD) in VM. The second and third examples are VM specific.

LNX1 labelled disk or non-labelled volume:

These disks are implicitly reserved for use by LINUX . The disk layout reserves the IPL and label records for access through the 'entire disk' device. All remaining records are grouped into the first partition.

CMS1 labelled disk:

Handling of these disks depends on the content of the CMS filesystem. If the volume contains a CMS filesystem it will be treated equivalently to a LNX labelled volume. If the volume is a CMS reserved volume [1] the CMS reserved file is represented by the first and only partition. IPL and label records as well as the metadata of the CMS filesystem are reserved for access through the 'entire disk' device.

DASD features

The DASD device driver can access devices according to Table 2 by its built in CCW interface.

Table 2. Supported devices. '' signifies any digit.*

Device format	Control unit type/model	Device type/model
ECKD (Extended Count Key Data)	3990(2105)/**	3380/**
	3990(2105)/**	3390/**
	9343/**	9345/**
FBA (Fixed Block Access)	6310/??	9336/??
	3880/**	3370/**

1. CMS reserved volume means a volume that has been reserved by a 'CMS RESERVE fn ft fm' command.

In addition under a guest operating system in VM/ESA any DASD device supported by VM/ESA is also supported by LINUX for S/390 by accessing the device using the `DIAG250` command.

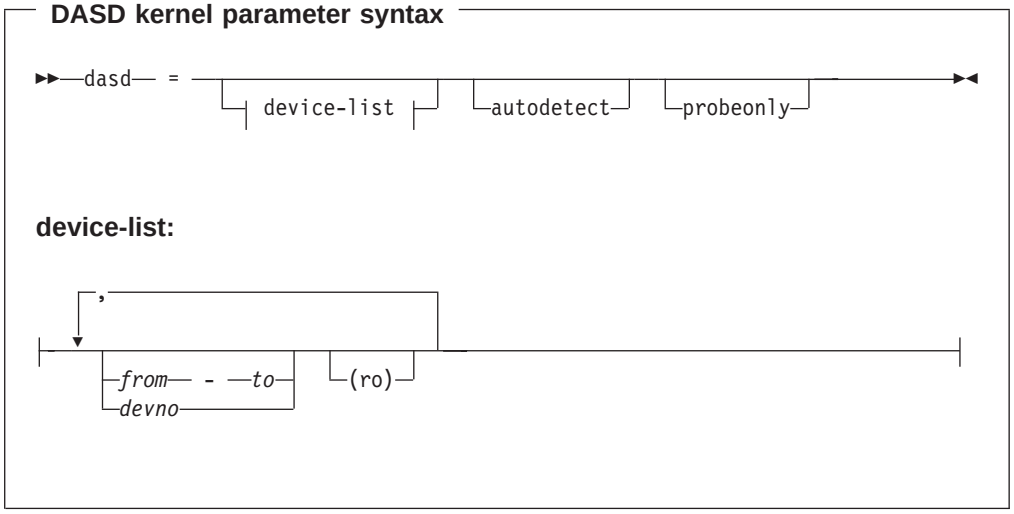
The DASD device driver is also known to work with these devices:

- Multiprise internal disks
- RAMAC
- RAMAC RVA
- Enterprise Storage Server (Seascape) virtual ECKD-type disks

LINUX for S/390 currently implements one partition per volume, which is the whole volume, skipping the first blocks according to the scheme outlined in Figure 1 on page 9.

DASD kernel parameter syntax

The DASD driver is configured by a kernel parameter added to the parameter line:



where:

autodetect
: causes the driver to consider any device operational at the time of IPL as a potential DASD and allocate a device number for it. Nevertheless the devices which are not DASD, or do not respond to the access methods known to the kernel, will not be accessible as DASD. Any 'open' request on such a device will return `ENODEV`. In `/proc/dasd/devices` these devices will be flagged 'unknown'.

probeonly
: causes the DASD device driver to reject any '`open`' syscall with `EPERM`.

autodetect,probeonly
: behaves in the same way as above, but additionally all devices which are accessible as DASD will refuse to be opened, returning `EPERM`. This setting is the default if no '`dasd=...`' parameter is given in the command line or in the module parameter.

from-to
: defines the first and last DASD in a range. All DASD devices with

addresses in the range are selected. It is not necessary for the *from* and *to* addresses to correspond to actual DASD.

devno defines a single DASD address.

(ro) specifies that the given device or range is to be accessed in read-only mode.

The DASD addresses must be given in hexadecimal notation with or without a leading `0x`, for example 0191 or 5a10.

If you supply one or more kernel parameters `dasd=`*`device-list1`* `dasd=`*`device-list2`* **...** the devices are processed in order of appearance in the parameter line. Devices are ignored if they are unknown to the machine, non-operational, or set off-line.[2]

If autodetection is turned on a DASD device is allocated in LINUX for every device operational at the time of initialization of the driver, in order of ascending subchannel numbers.

Note that the autodetection option may cause confusing results if you change your I/O configuration between two IPLs, or if you are running as a guest operating system in VM/ESA, because the devices might appear with different major/minor combinations in the new IPL .

DASD kernel example

```
dasd=192-194,5a10(ro)
```

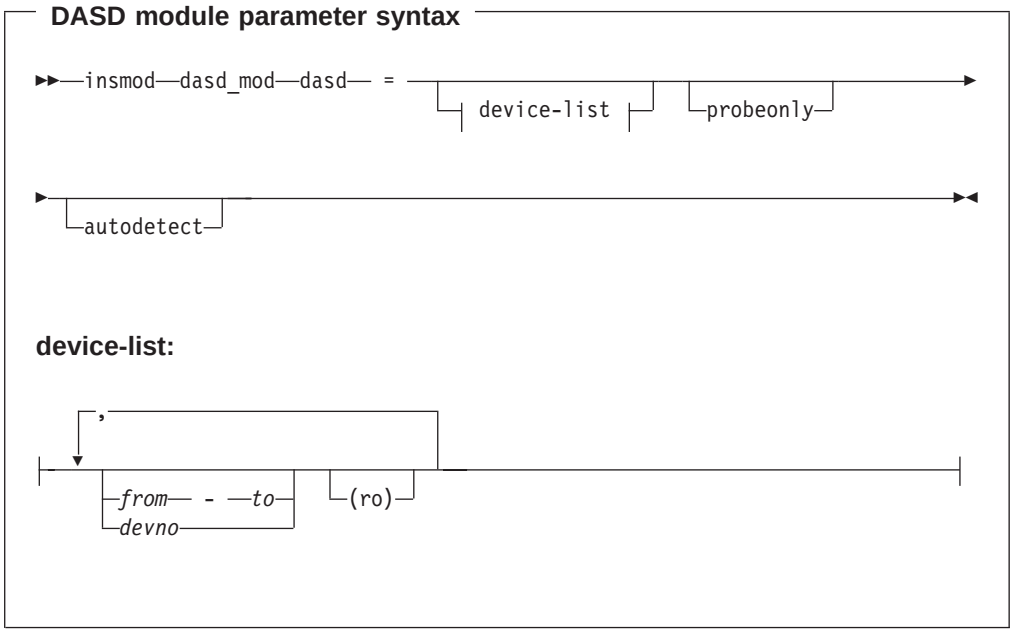
This reserves major/minor numbers and nodes as follows:

```
94  0  /dev/dasda   - for the entire device  192
94  1  /dev/dasda1  - first partition on     192
94  2  /dev/dasda2  - reserved (not used)
94  3  /dev/dasda3  - reserved (not used)

94  4  /dev/dasdb   - for the entire device  193
94  5  /dev/dasdb1  - first partition on     193
94  6  /dev/dasdb2  - reserved (not used)
94  7  /dev/dasdb3  - reserved (not used)

94  8  /dev/dasdc   - for the entire device  194
94  9  /dev/dasdc1  - first partition on     194
94 10  /dev/dasdc2  - reserved (not used)
94 11  /dev/dasdc3  - reserved (not used)

94 12  /dev/dasdd   - for the entire device 5a10 (read only)
94 13  /dev/dasdd1  - first partition on    5a10 (read only)
94 14  /dev/dasdd2  - reserved (not used)
94 15  /dev/dasdd3  - reserved (not used)
```

2. Currently there is no check for duplicate occurrences of the same device number.

DASD module parameter syntax

The following are the DASD driver module parameters:



where:

dasd_mod
is the name of the device driver module

dasd is the start of the parameters

and all other parameters are the same as the DASD kernel parameters described in "DASD kernel parameter syntax" on page 10.

DASD module example

```
insmod dasd_mod dasd=192-194,5a10(ro)
```

The details are the same as "DASD kernel example" on page 11.

DASD – Preparing for use

1) Low level format

Before using an ECKD type DASD as a LINUX for S/390 disk the device must be formatted. This should be done from LINUX for S/390 by issuing an `ioctl` called `BIODASDFORMAT` on the file descriptor of the opened volume `/dev/dasd<letter>`. The utility `dasdfmt` is provided as an interface to this `ioctl` with additional checking.

Caution: Using `dasdfmt` or the raw `ioctl` can potentially destroy your running LINUX for S/390 system, forcing you to reinstall from scratch.

See the help given by `dasdfmt -help` and “dasdfmt - Format a DASD” on page 66 for further information. The `dasdfmt` utility calls several processes sequentially. Take care to allow sufficient time for each process to end before attempting to enter an additional command.

We recommend you set `blksize` to 1024 or higher (ideally 4096) because the `ext2fs` file system uses 1KB blocks and 50% of capacity will be unusable if the DASD blocksize is 512 bytes.

The formatting process can take a long time (hours) for large DASD.

2) Make a file system

Before using a DASD as a LINUX for S/390 data disk, you must create a file system on it. (A DASD for use as a swap device or paging space only needs to be defined as such.) Using `mk`*`xx`*`fs` (replacing *xx* with the appropriate identifier for the file system – for example use `mke2fs` for an ext2 file system) you can create the file system of your choice on that volume or partition .

It is recommended that you build your file system on the partition of the DASD (`/dev/dasda1`, `/dev/dasdb1`, and so on), rather than the whole volume. This incurs a cost of 3 blocks of disk space, but it will allow you to introduce a real partition table on the device without losing access your data.

Note that the blocksize of the file system must be larger than or equal to the blocksize given to the `dasdfmt` command. It is recommended that the two blocksize values are equal.

You must enable `CONFIG_DASD`, `CONFIG_DASD_ECKD`, `CONFIG_DASD_FBA` and `CONFIG_DASD_MDSK` in the configuration of your current kernel to access IBM DASD.

DASD restrictions

- Note that the `dasdfmt` utility can only format volumes containing a standard record zero on all tracks. If your disk does not fulfill this requirement (for example if you re-use an old volume, or access a brand new disk or one having an unknown history), you should additionally use a device support facility such as ICKDSF (in OS/390, VM/ESA, VSE/ESA or stand-alone) before doing the `dasdfmt` for the low-level format.
- The size of any swap device or file may not exceed 2 GB. Similarly, the limit for the main memory that can be defined is slightly less than 2 GB.

Chapter 3. LINUX for S/390 VM minidisk device driver

Under VM it is possible to divide DASDs into logical partitions, known as minidisks. These minidisks have a unique 16 bit identification – the virtual device number. Each virtual device can be formatted from VM and used with the CMS file system. Also, it is possible to create a reserved minidisk, which appears to CMS as a single large file the size of the whole virtual device. This reserved file can be written to under CMS and accessed with a special opcode `DIAG 250` as a block device.

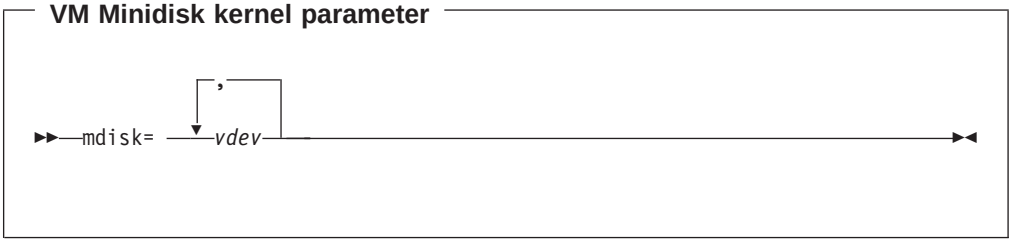
Note: It is also possible under VM to attach an entire DASD to a LINUX guest (as a 'full pack minidisk'). A DASD attached in this manner is controlled by the LINUX for zSeries DASD device driver (see "DASD overview" on page 7) and not by the VM minidisk device driver.

VM minidisk features

A reserved minidisk must be formatted with a blocksize of either 512, 1024, 2048 or 4096 bytes. It may have any partition size up to the physical disk limit.

VM minidisk kernel parameter syntax

The VM Minidisk driver is configured by a kernel parameter added to the LINUX parameter line file (`PARM LINE A`):



where:

vdev virtual device number as hex number (without the leading 0x)

It is possible to have more than one `mdisk=` statement in the `PARM LINE A` file; the assignment to the device minor numbers will follow the order of the statements.

VM minidisk kernel example

The command:

```
mdisk=193,194
```

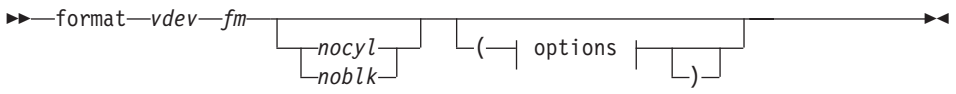
allocates two minidisks to LINUX with device numbers 0x193 and 0x194

VM minidisk – Preparing disks

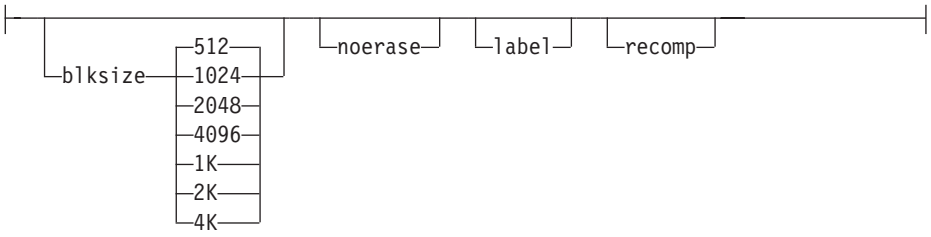
A reserved minidisk is created with the following steps in CMS:

1. Format minidisk.
2. Reserve minidisk.

Formatting a minidisk in CMS



options:



Where:

vdev is the unit address (hexadecimal with no leading 0x)

fm is the CMS disk access letter (one character, A-R or T-Z)

nocyl is the number of cylinders to be allocated (non-FB-512 devices)

noblk is the number of blocks to be allocated (FB-512 devices only)

blksize
is the size of blocks to be formatted

noerase
means that the blocks are not to be cleared to zeroes (FB-512 devices only)

label is the label to be assigned to the minidisk (1 to 6 alphanumeric characters). If this is the only parameter given then the disk is re-labelled without re-formatting.

recomp
changes the number of cylinders/blocks available on a previously formatted minidisk

Reserving a minidisk in CMS

Where:

filename filetype
is a valid CMS file name (each part is 1 to 8 characters)

filemode
is the CMS disk access letter specified in the `format` command

Example:

```
format 192 b (blksize 4096
reserve mnda mnda b
```

Making a file system

Before using a VM minidisk as a LINUX for zSeries data disk, you must create a file system on it. (A VM minidisk for use as a swap device or paging space only needs to be defined as such.) Using `mk__fs` (replacing __ with the appropriate identifier for the file system – for example use `mke2fs` for an ext2 file system) you can create the file system of your choice on that volume or partition .

Note that the blocksize of the file system must be larger than or equal to the blocksize given to the `format` command. It is recommended that the two blocksize values are equal.

Chapter 4. LINUX for S/390 XPRAM device driver

The S/390 architecture supports the access of only 2 GB (gigabytes) of main memory. To overcome this limitation additional memory can be declared and accessed as expanded memory. The S/390 architecture allows applications to access up to 16 TB (terabytes) of expanded storage (although the current hardware can be equipped with at most 64 GB of real memory). The memory in the expanded storage range can be swapped in or out of the main memory in 4 KB blocks.

An IPL (boot) of LINUX for S/390 does not reset expanded storage, so it is persistent through IPLs and could be used, for example, to store diagnostic information. It is of course reset by an IML (power off/on).

The XPRAM device driver is a block device driver that enables LINUX for S/390 to access the expanded storage. Thus XPRAM can be used as a basis for fast swap devices and/or fast file systems.

XPRAM features

- Automatic detection of expanded storage.

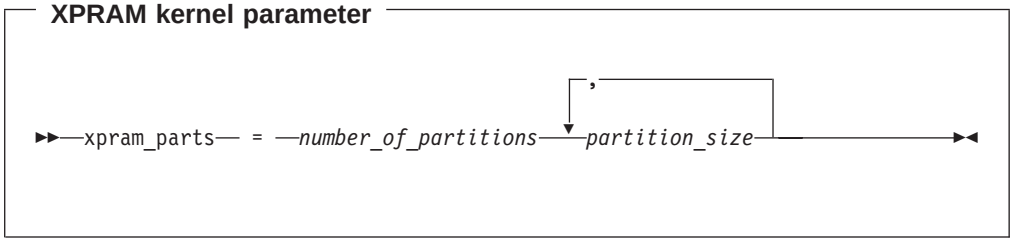
  (If expanded storage is not available, XPRAM fails gracefully with a log message reporting the lack of expanded storage.)
- Storage can be subdivided into up to 32 partitions.
- Device driver major number: 35.
- Partition minor numbers: 0 through 31.
- Hard sector size: 4096 bytes.

Note on reusing XPRAM partitions

It is possible to reuse the filesystem or swap device on an XPRAM device or partition if the XPRAM kernel or module parameters for the new device or partition match the parameters of the previous use of XPRAM. If you change the XPRAM parameters for a new use of XPRAM you must make a new filesystem (for example with `mke2fs`) or swap device for all partitions that have changed. A device or partition has changed if its size has changed. All partitions following one which has changed are treated as changed as well (even if their sizes have not been changed).

XPRAM kernel parameter syntax

The kernel parameter is optional. If omitted the default is to define the whole expanded storage as one partition. The syntax is:



where *number_of_partitions* defines how many partitions the expanded storage is split into. The i-th *partition_size* defines the size of the i-th partition. Blank entries are inserted if necessary to fill *number_of_partitions* values. Each size may be blank, specified as a decimal value, or a hexadecimal value preceded by `0x`, and may be qualified by a magnitude:

- `k` or `K` for kilo (1024) is the default
- `m` or `M` for Mega (1024*1024)
- `g` or `G` for Giga (1024*1024*1024)

The size value multiplied by the magnitude defines the partition size in bytes. The default size is `0`.

Any partition defined with a non-zero size is allocated the amount of memory specified by its size parameter.

Any remaining memory is divided as equally as possible among any partitions with a zero or blank size parameter, subject to the two constraints that blocks must be allocated in multiples of 4K and addressing constraints may leave un-allocated areas of memory between partitions.

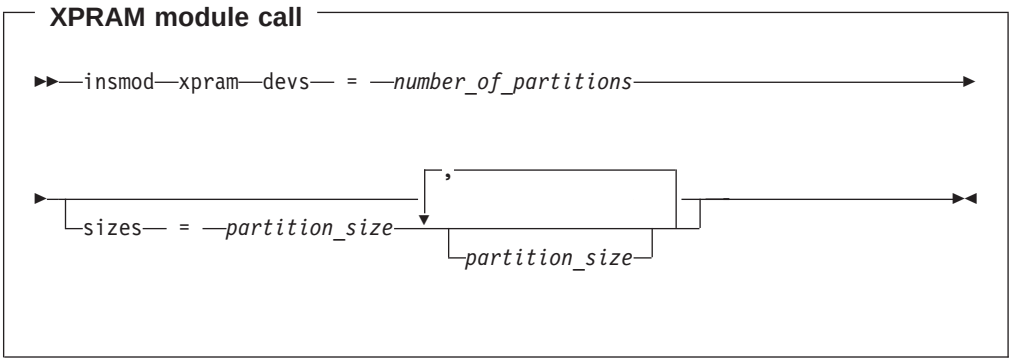
XPRAM kernel example

```
xpram_parts=4,0x800M,0,0,0x1000M
```

This allocates the extended storage into four partitions. Partition 1 has 2 GB (hex 800M), partition 4 has 4 GB, and partitions 2 and 3 use equal parts of the remaining storage. If the total amount of extended storage was 16 GB, then partitions 3 and 4 would each have approximately 5 GB.

XPRAM module parameter syntax

If it is not included in the kernel XPRAM may be loaded as a module. The syntax of the module parameters passed to `insmod` or `modprobe`differs from the kernel parameter syntax:



where:

- *partition_size* is a non-negative integer that defines the size of the partition in KB. Only decimal values are allowed and no magnitudes are accepted.

XPRAM module example

```
insmod xpram devs=4 sizes=2097152,8388608,4194304,2097152
```

This allocates a total of 16 GB of extended storage into four partitions, of (respectively) size 2 GB, 8 GB, 4 GB, and 2 GB.

Chapter 5. LINUX for S/390 Console device drivers

The S/390 hardware requires a line-mode terminal (the hardware console) for overall system control. The LINUX for S/390 console device drivers enable LINUX to use this console for basic LINUX control as well.

You can use a 3215 or a 3270 console instead of the hardware console if LINUX is running under VM/ESA. You can use a 3215 console if LINUX is running on a P/390.

The S/390 **system console** is the device which gives the S/390 operator access to the SE (Service Element) which is in overall control of the S/390 system. This can be a real device physically attached to the S/390, or it can be emulated in software, for example by running an HMC (Hardware Management Console) in a web browser window.

A S/390 **terminal** is any device which gives a S/390 user access to applications running on the S/390 system. This could be a real device such as a 3270 linked to the S/390 through a controller, or again it can be a terminal emulator on a networked device.

Note that both 'terminal' and 'console' have special meanings in LINUX which should not be confused with the S/390 usage. The LINUX console and the LINUX terminals are different applications which both run on S/390 **terminal**s.

The drivers for the 3215, 3270 and hardware consoles can be compiled into the LINUX kernel. If more than one console is present the default console driver will be chosen at run time according to the environment:

- In an LPAR or native environment, the hardware console will be made the default.
- In VM/ESA either the 3215 or the 3270 console driver will be made the default, depending on the guest's console settings (the "CONMODE" field in the output of "#CP QUERY TERMINAL").
- On a P/390 the 3215 console will be made the default.

The default driver can be overridden with the "conmode=" kernel parameter (see "Console kernel parameter syntax" on page 24).

The intended use of the console device drivers is solely to launch LINUX . When LINUX is running, the user should access LINUX for S/390 via Telnet, because the console is a line-mode terminal and is unable to support applications such as `vi`. Therefore it is strongly recommended that you assign *dumb* to the *TERM* environment variable so that at least applications like `less` give appropriate output.

Note that there are different options that must be selected during kernel configuration to enable the LINUX terminal on the hardware console or to enable the LINUX console on the 3215 or 3270 console.

Console features

- Provides a line mode typewriter terminal.
- Console output on the first terminal.

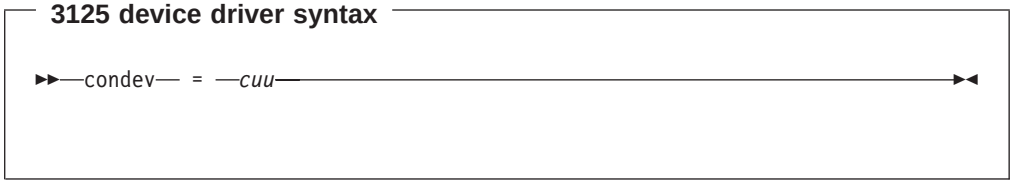
Console kernel parameter syntax

The hardware console device driver does not require any parameters.

The 3215 console device driver does not require any parameters if it is used under VM/ESA. If it is used with a P/390 system, you have to specify the `condev` kernel parameter. This supplies the device driver with the subchannel number of the 3215 device. The reason that this parameter is needed is that there is no guaranteed method of recognizing a 3215 device attached to a P/390.

The kernel parameter syntax is:

3125 device driver syntax



where *cuu* is the device 'Control Unit and Unit' address, and may be expressed in hexadecimal form (preceded by `0x`) or in decimal form.

Note: Early releases of the driver will not accept this parameter in hexadecimal form.

Console kernel examples

```
condev=0x001f
```

or

```
condev=31
```

Both of these formats tell the device driver to use device number hex 1F for the 3215 terminal.

Using the console

The console is a line mode terminal. The user enters a complete line and presses enter to let the system know that a line has been completed. The device driver then issues a read to get the completed line, adds a new line and hands over the input to the generic TTY routines.

Console special characters

The console does not have a control key. That makes it impossible to enter control characters directly. To be able to enter at least some of the more important control characters, the character `'^'` has a special meaning in the following cases:

- The two character input line `^c` is interpreted as a `Ctrl+C`. This is used to cancel the process that is currently running in the foreground of the terminal.
- The two character input line `^d` is interpreted as a `Ctrl+D`. This is used to generate an end of file (EOF) indication.
- The two character input line `^z` is interpreted as a `Ctrl+Z`. This is used to stop a process.

- The two characters ^n at the end of an input line suppresses the automatic generation of a new line. This makes it possible to enter single characters, for example those characters that are needed for yes/no answers in the ext2 file system utilities.
- The two characters '^-' followed by a third character invoke the so called "magic sysrequest" function. Various debugging and emergency functions are performed specified by the third character. This feature can be switched on or off during runtime by echoing '1' or '0' to /proc/sys/kernel/sysrq. The third character can be:
  - 'b' – re-IPL immediately,
  - 's' – emergency sync all filesystems,
  - 'u' – emergency remount all mounted filesystems readonly,
  - 't' – show task info,
  - 'm' – show memory,
  - '0' to '9' – set console log level,
  - 'e' – terminate all tasks,
  - 'i' – kill all tasks except init,
  - 'l' – kill all tasks including init.

If you are running under VM without a 3215 console you will have to use the CP VINPUT command to simulate the **ENTER** and **SPACE** keys.

The **ENTER** key is simulated by entering:

```
#CP VInput VMSG \n
```

The **SPACE** key is simulated by entering:

```
#CP VInput VMSG  \n    (two blanks followed by \n).
```

If the special characters do not appear to work, make sure you have the correct codepage in your terminal emulator. One known to work is codepage 037 (United States).

VM console line edit characters

When running under VM, the control program (CP) defines five characters as line editing symbols. Use the CP QUERY TERMINAL command to see the current settings. The defaults for these depend on the terminal emulator you are using, and can be reassigned by the CP system operator or by yourself using the CP TERMINAL command to change the setting of LINEND, TABCHAR, CHARDEL, LINEDEL or ESCAPE. The most common values are:

LINEND #
: The end of line character (this allows you to enter several logical lines at once).

TABCHAR |
: The logical tab character.

CHARDEL @
: The character delete symbol (this deletes the preceding character).

LINEDEL [(ASCII terminals) or ¢ (EBCDIC terminals)
: The line delete symbol (this deletes everything back to and including the previous LINEND symbol or the start of the input).

ESCAPE ″
: The escape character (this allows you to enter a line edit symbol as a normal character).

To enter the line edit symbols # | @ [" (or # | @ ¢ ") you need to type the character pairs "# "| "@ "["" (or "# "| "@ "¢ ""). Note in particular that to enter the quote character (″) you must type it twice (″″).

Example:

If you should type in the character string:

```
#CP HALT#CP ZIPL 190[#CP IPL 1@290 PARM VMHALT=""MSG OP REBOOT"#IPL 290""
```

the actual commands received by CP will be:

```
CP HALT
CP IPL 290 PARM VMHALT="MSG OP REBOOT#IPL 290"
```

Console 3270 emulation

If you are accessing the VM console using the `x3270` emulator, then you should add the following settings to the `.XDefaults` file in order to get the correct code translation:

```
! X3270 keymap and charset settings for Linux
x3270.charset: us-intl
x3270.keymap: circumfix
x3270.keymap.circumfix: :<key>asciicircum: Key("^")\n
```

Console – Use of VInput

Note: 'VInput' is a VM CP command. It may be abbreviated to 'VI' but should not be confused with the LINUX command 'vi'.

If you use the hardware console driver running under VM it is important to consider how the input is handled. Instead of writing into the suitable field within the graphical user interface at the service element or HMC you have to use the `VInput` command provided by VM. The following examples are written at the input line of a 3270 terminal or terminal emulator (for example, x3270).

Note that, in the examples, capitals within `VInput` and its parameters processed by VM/CP indicate the characters you have to type. The lower case letters are optional and are shown for readability. These examples assume that you enter the `CP READ` mode first. If you are not in this mode you must prefix all of the examples with the command `#CP`.

```
VInput VMSG LS -L
```

(the bash will call `ls -l` after this command was sent via `VInput` to the hardware console as a non-priority command - `VMSG`).

```
VInput PVMSG ECHO *
```

(the bash will execute `echo *` after this command was sent via `VInput` to the hardware console as a priority command - `PVMSG`).

The hardware console driver is capable to accept both if supported by the hardware console within the specific machine or virtual machine. Please inspect your boot

messages to check the hardware console's capability of coping with non-priority or priority commands respectively on your specific machine. Remember that the hardware console is unable to make its own messages available via `dmesg`.

Features of `VInput`.

1. Use '"" to output a single '".

   `VInput` example: `VInput PVMSG echo ""Hello world, here is ""$0`

   (on other systems: `echo "Hello world, here is "$0`)

2. Do not use `#` within `VInput` commands.

   This character is interpreted as an end of line character by `VM CP`, and terminates the `VInput` command. If you need the # character it must be preceded by the escape character (`"#`).

3. All characters in lower case are converted by VM to upper case.

   If you type `VInput VMSG echo $PATH`, the driver will get `ECHO $PATH` and will convert it into `echo $path`. LINUX and the bash are case sensitive and cannot execute such a command. To resolve this problem, the hardware console uses an escape character (%) under VM to distinguish between upper and lower case characters. This behavior and the escape character (%) are adjustable at build-time by editing the driver sources, or at run time by use of the `ioctl` interface. Some examples:

   - input: `VInput VMSG ECHO $PATH`

     output: `echo $path`

   - input: `VInput vmsg echo $%PATH%`

     output: `echo $PATH`

   - input: `VInput pvmsg echo ""%H%ello, here is ""$0 #name of this process`

     output: `VINPUT PVMSG ECHO "%H%ELLO, HERE IS "$0`

     `NAME OF THIS PROCESS`

     `HCPCMD001E Unknown CP command: NAME`

     `echo "Hello, here is "$0`

     `Hello, here is -bash`

Console limitations

- The 3215 driver only works in combination with VM/ESA. In a single image or in LPAR mode the 3215 terminal device driver initialization function just exits without registering the driver.
- Due to a problem with the translation of code pages (500, 037) on the host, the pipe command character (|) cannot be intercepted by the console. If you need to use this command execute it from a Telnet session.
- Displaying large files might cause some missing sections within the output because of the latency of the hardware interface employed by the device.
- In native or LPAR environments, you occasionally have to use the **Delete** button of the GUI on the Service Element or Hardware Management Console to enable further output. This is relevant to:
  - SE version 1.6.1 or older on G5, G6, and Multiprise 3000.
  - SE version 1.5.2 or older on G3, G4, and Multiprise 2000.
- Messages concerning the hardware console operation generated by the hardware console driver cannot be provided to the `syslog` and are therefore unavailable with `dmesg`.

- Output from the head/top is deleted if the amount exceeds approximately 30 kilobytes per LPAR (or image) on SE or HMC.
- Applications such as `vi` are not supported because of the console's line-mode nature.

Chapter 6. Channel attached tape device driver

The LINUX for S/390 tape device driver manages channel attached tape drives which are compatible with IBM 3480 or IBM 3490 magnetic tape subsystems. Various models of these devices are handled (for example the 3490E).

Tape driver features

The device driver supports a maximum of 128 tape devices.

No official LINUX device major number is assigned to the S/390 tape device. The driver allocates major numbers dynamically and lists them on initialization. Typically major number 254 will be allocated. (This will be used for both the character device front-end and the block device front-end.) Minor numbers will be allocated in pairs from zero.

The driver may search for all tape devices attached to the LINUX machine, or it may be given a list of device addresses to use.

If it is not given a list the numbers allocated are volatile – the number allocated to any particular physical device may change if the system is rebooted or the device driver is reloaded. In particular a device brought online during a LINUX session will be allocated the next available number at the time it comes online, but at the next reboot it will be given a number according to the sequence of device addresses.

If a `"tape="` parameter is present at system startup or module load, all tape devices in the ranges of the specified parameter list will be used. The devices are then numbered (sequentially from zero) according to the order in which their subchannel numbers appear in the list.

In both cases the associations between subchannel numbers and device numbers are listed in the file `/proc/tapedevices`.

Tape character device front-end

You will usually read or write to the tape device using the character device front-end of the driver. In fact two front-ends are provided for each physical device. One of these is used in multi-step procedures to leave the tape in position for the subsequent step at the close of each step. The other is used in single-step procedures, or in the last step of multi-step procedures, to automatically rewind the tape at the end of the step.

The node names for these devices are constructed from the standard label `tibm`, with a prefix indicating the close function `r` (rewind) or `n` (non-rewind), and a suffix from the device number (starting at zero). Thus the names given to the first two devices are `/dev/rtibm0`,

```
rtibm0 -> r     rewind
          tibm  label
          0     device number
```

`/dev/ntibm0`, `/dev/rtibm1` and `/dev/ntibm1`.

You can use the character device front-end in the same way as any other LINUX tape device. You can write to it and read from it using normal LINUX facilities such as GNU `tar`. You can perform control operations (such as rewinding the tape or skipping a file) with the standard tool `mt`.

Most LINUX tape software should work with both the rewinding and non-rewinding devices.

Tape block device front-end

You can also use the tape driver as a block device, but this is restricted to read-only mode.

This device could be used for the installation of software in the same way as tapes are used under other operation systems on the S/390 platform. (This is similar to the way most LINUX software distributions are shipped on compact disk using the ISO9660 filesystem).

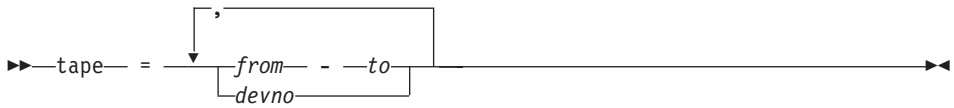
One block device node is allocated to each physical device. They follow a similar naming convention to the character devices. Without `devfs` the prefix `b` is used – `/dev/btibm0` for the first device, `/dev/btibm1` for the second and so on.

You are advised to use only the ISO9660 filesystem on LINUX for S/390 tapes as this filesystem is optimized for CDROM devices, which – just like 3480 or 3490 tape devices – cannot perform fast seeks.

Tape driver kernel parameter syntax

You do not need to give the tape device driver any kernel parameters if you want to use tape auto-detection. If you want to specify the physical tape devices to be used you must configure the tape driver by adding a parameter to the kernel parameter line:

Tape driver kernel parameter syntax



where:

from-to
: defines the first and last tape device in a range. All valid tape devices with addresses in this range are selected. It is not necessary for the *from* and *to* addresses to correspond to actual devices.

devno
: defines a single tape device.

The tape addresses must be given in hexadecimal notation (without a leading `0x`), for example 0181 or 5a01.

If you supply one or more kernel parameters, for example `tape=`*`from-to`*`,tape=`*`devno`*`,...`, the devices are processed in the order in which they appear in the parameter line. Devices are ignored if they are unknown to the device driver, non-operational, or set offline. You should specify no more than 128 devices in the parameter line as this is the maximum number of devices manageable by the driver.[3]

Note that the auto-detection option may cause confusing results if you change your I/O configuration between two IPLs, or if you are running as a guest operating system in VM/ESA, because the devices might appear with different names (major/minor combinations) in the new IPL.

Tape driver kernel example

The kernel parameter could be:

```
tape=181-184,19f
```

This reserves devices as follows:

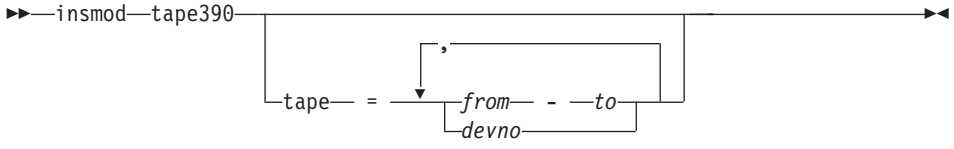
```
0181     will be    /dev/ntibm0   /dev/rtibm0   /dev/btibm0
0182     will be    /dev/ntibm1   /dev/rtibm1   /dev/btibm1
0183     will be    /dev/ntibm2   /dev/rtibm2   /dev/btibm2
0184     will be    /dev/ntibm3   /dev/rtibm3   /dev/btibm3
019f     will be    /dev/ntibm4   /dev/rtibm4   /dev/btibm4
```

3. Currently there is no check for duplicate occurrences of the same device number.

Tape driver module parameter syntax

The syntax of the module call to load the tape device driver is:

Tape module parameter syntax



where:

tape390
is the name of the device driver module

and the rest of the parameters are the same as those of the tape driver kernel syntax.

Tape driver module example

```
insmod tape390 tape=181-184,19f
```

The details are the same as “Tape driver kernel example” on page 31.

Tape device driver API

The tape device driver uses the **posix** compliant tape interface similar to the LINUX SCSI tape device driver.

Some differences in the MTIO interface do exist due to the different hardware:

- `MTSETDENSITY` has no effect as the recording density is automatically detected.
- `MTSETDRVBUFFER` has no effect as the drive automatically switches to unbuffered mode if buffering is unavailable.
- `MTLOCK` and `MTUNLOCK` have no effect as the tape device hardware does not support media locking.
- `MTLOAD` waits until a tape is inserted rather than loading a tape automatically.
- The drives do not support a load command, but if `MTUNLOAD` is used the next tape in the stacker will be inserted automatically.
- `MTCOMPRESSION` controls the IDRC (Improved Data Recording Capability). This is activated if the `COUNT` argument is non-zero or deactivated if it is zero. On system startup the IDRC is activated by default.
- `MTSETPART` and `MTMKPART` have no effect as the devices do not support partitioning.
- The contents of the structure returned by `MTIOCGET` are incomplete as some SCSI specific data is not applicable.

Tape driver examples

Example 1 – Creating a single-volume tape

In this example a tape with an ISO9660 filesystem is created using the first tape device. For this the ISO9660 filesystem support must be built into your system kernel.

Use the `mt` command to issue tape commands, and the `mkisofs` command to create an ISO9660 filesystem:

- Create a LINUX directory (`somedir`) and fill it with the contents of the filesystem

  ```
  mkdir somedir
  cp contents somedir
  ```

- Insert a tape
- Ensure the tape is positioned at the beginning

  ```
  mt -f /dev/ntibm0 rewind
  ```

- Set the blocksize of the character driver. (The blocksize 2048 bytes is commonly used on ISO9660 CD-roms.)

  ```
  mt -f /dev/ntibm0 setblk 2048
  ```

- Write the filesystem to the character device driver

  ```
  mkisofs -o /dev/ntibm0 somedir
  ```

- Rewind the tape again

  ```
  mt -f /dev/ntibm0 rewind
  ```

- Now you can mount your new filesystem as a block device:

  ```
  mount -t iso9660 -o ro,block=2048 /dev/btibm0 /mnt
  ```

Example 2 – Creating a multivolume tape

In this example files are backed up onto a multivolume tape using the LINUX facility `tar`.

- Insert a tape media in the tape device (here: `/dev/ntibm0`).
- If necessary, rewind and erase the tape:

```
mt -f /dev/ntibm0 rewind
mt -f /dev/ntibm0 erase
mt -f /dev/ntibm0 rewind
```

- Open a new telnet session to trace the content of `/var/log/messages`

```
tail -f /var/log/messages &
```

- In the first telnet session backup the files to tape using the tar command with the option `-M` (multi-volume), for example:

```
tar -cvMf /dev/ntibm0 /file_specs
```

- If more tape volumes are required you will be prompted to prepare the next medium. Go to a third telnet session and enter the command:

```
mt -f /dev/ntibm0 offl
```

- Insert a new tape manually (if not using a tape unit magazine with autoload).
- Wait for a message of the form:

```
Apr 30 16:27:53 boeaet22 kernel: T34xx:A medium was inserted into the tape.
```

  in `/var/log/messages` (see 34). When you see this message hit the return key in the `tar` session.
- Repeat the last four steps with further tapes until the backup is complete.

Tape driver restrictions

The driver is unable to detect manual operations on the tape device, in particular manual tape unloads, and these operations will lead to errors in reading and writing. The driver provides `ioctl` functions to control the device and these must be used, either through the API or by using the LINUX `mt` utility.

Tape driver further information

Basic LINUX tape control is handled by the `mt` utility, which is described in the `mt` man page. Note that the sections on SCSI tape devices are inapplicable to S/390 devices.

Part 3. LINUX for S/390 Network device drivers

These chapters describe the device drivers available to connect S/390 systems to your network.

The drivers described are:

- "Chapter 7. LINUX for S/390 CTC/ESCON device driver" on page 37
- "Chapter 8. LINUX for S/390 IUCV device driver" on page 43
- "Chapter 9. LINUX for S/390 LCS Device Driver" on page 49
- "Chapter 10. LINUX for S/390 OSA-Express device driver" on page 53

Licence conditions

Some of these drivers are subject to licence conditions as reflected in: "International License Agreement for Non-Warranted Programs" on page 129.

Chapter 7. LINUX for S/390 CTC/ESCON device driver

A CTC connection or an ESCON connection is the typical high speed connection between mainframes. The data packages and the protocol of both connections are the same. The difference between them is the physical channel used to transfer the data.

Both types of connection may be used to connect a mainframe, an LPAR, or a VM guest to another mainframe, LPAR or VM guest, where the peer LPAR or VM guest may reside on the same or on a different system.

A third type of connection is virtual CTC which is a software connection between two VM guests on the same VM system and which is faster then a physical connection.

The LINUX for S/390 CTC device driver supports all three types of connection and can be used to establish a point-to-point TCP/IP connection between two LINUX for S/390 systems or between a LINUX for S/390 system and another operating system such as VM/ESA, VSE/ESA or OS/390.

CTC/ESCON features

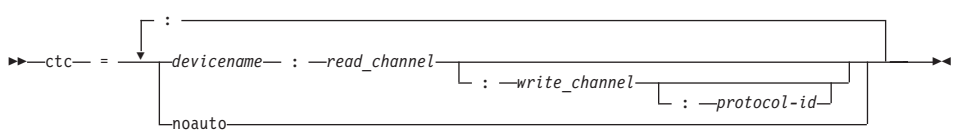
- Any number of CTC and/or ESCON connections available.
- Autosense mode available (the driver will pick all available channels starting with the lowest subchannel IDs).
- If built monolithically (not as a module) the parameters can be used to describe a maximum of 16 devices. If more channels are available and 'noauto' is not specified the additional channels are auto-sensed and used in ascending order.

CTC/ESCON configuration

Kernel parameter syntax

The default for this driver is to select channels in order (automatic channel selection). If you need to use the channels in a different order, or do not want to use automatic channel selection, you can specify alternatives using the `ctc=` kernel parameter.

CTC kernel parameter



Note: The entire parameter is repeated (separated by spaces) for each `CTC/ESCON` device.

Where:

devicename
is `ctc` or `escon` concatenated with the channel number, for example `ctc1` or `escon99`.

read_channel
is the read channel address (in hexadecimal preceded by 0x).

write_channel
is the write channel address (in hexadecimal preceded by 0x). If omitted the default is the read channel address plus 1.

protocol-id
is the protocol number for CTC or ESCON. This can take the values:

- 0 for compatibility mode (the default; used with non-LINUX peers other than OS/390 and z/OS)
- 1 for extended mode,
- 2 meaning ″CTC-based tty″ (this is only supported on LINUX -LINUX connections),
- 3 for compatibility mode with OS/390 and z/OS.

Using `noauto` as the device name disables automatic channel selection. If the only parameter given is `noauto` the CTC driver is disabled. This might be necessary, for example, if your installation uses 3271 devices or other such devices that use the CTC device type and model, but operate with a different protocol.

Kernel example

For one network device (CTC):

Figure 2. Connection of two systems via CTC

```
ctc=ctc0:0x600
```

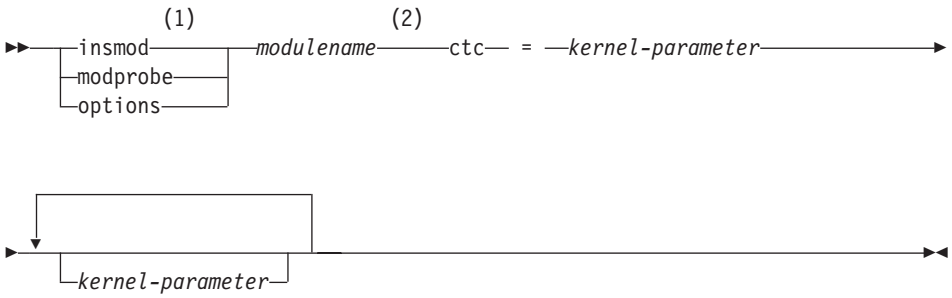
Or for two network devices (CTC + ESCON):

```
ctc=ctc0:0x601:0x600:escon3:0x605:0x608,
```

Module parameter syntax

These parameters can be passed to the CTC/ESCON driver module by `insmod`, or can be specified in the parameter file `"/etc/modules.conf"` or `"/etc/conf.modules"` (the file name depends on the LINUX distribution).

CTC module options



Notes:

1 **insmod** or **modprobe** on the command line or **options** in the parameter file.

2 When using **insmod**, *modulename* includes the path to the module's object file (for example `/lib/modules/.../ctc.o`). When using **modprobe** *modulename* is only the module name (`ctc`).

Where:

kernel-parameter
is as defined above in "Kernel parameter syntax" on page 37

Note: If the parameter line file is used the CTC driver may be loaded by typing `modprobe ctc` on the command line.

Module example

For one network device (CTC):

Figure 3. Connection of two systems via CTC

Command line example:

```
insmod ctc ctc=ctc0,0x0600,0x0601
```

or

```
insmod /lib/modules/ctc.o ctc=ctc0:0x0600
```

Parameter file example:

```
options ctc ctc=ctc0:0x0600
```

Or for two network devices (CTC + ESCON):

Command line example:

```
insmod /lib/modules/ctc.o ctc=ctc0:0x0601:0x0600:escon3:0x0605:0x0608
```

or

Parameter file example:

```
options ctc ctc=ctc0:0x0601:0x0600:escon3:0x0605:0x0608
```

CTC/ESCON – Preparing the connection

1. Connection

   Prior to activation a channel connection is required. This can be a real or virtual connection :

   - Real Channels

     Connect the systems with a pair of channels to the remote system. Verify that the read channel of one is connected to the write channel of the other.

   - LPAR to LPAR Channels

     Select a pair of channels on each system. Verify that the read channel of one is connected to the write channel of the other and vice-versa.

   - VM Channels

     a. Obtain a subnet from your TCP/IP communications staff. It is important that the subnet used by your LINUX guests is not the same as that used by VM/ESA on the LAN. The LINUX system is a separate network and should be treated as such.

     b. Take one address from that subnet and assign it to VM.

     c. Define two virtual channels to your user ID. The channels may be defined in the VM User Directory using directory control `SPECIAL` statements, for example:

        ```
        special 0c04 ctca
        special 0c05 ctca
        ```

        or by using the CP commands:

        ```
        define ctca as 0c04
        define ctca as 0c05
        ```

        from the console of the running CMS machine (preceded by `#CP` if necessary), or from an `EXEC` file (such as `PROFILE EXEC A`).

     d. Add the necessary VM TCP/IP routing statements (BsdRoutingParms or Gateway). Use an MTU size of 9216 and a point-to-point host route (subnet mask 255.255.255.255). If you use dynamic routing, but do not wish to run routed or gated on LINUX , update the VM `ETC GATEWAYS` file to include ″permanent″ host entries for each LINUX guest.

     e. Bring these updates online by using `OBEYFILE` or by recycling `TCPIP` and/or `ROUTED` as needed.

     Connect the virtual channels to the channels of the VM TCP/IP target user ID. You must couple the LINUX read channel to the VM TCP/IP write channel and vice versa. The coupling can be done with the following CP commands (following the previous example)

     ```
     couple 0c04 to tcpip 0c05
     couple 0c05 to tcpip 0c04
     ```

The VM TCP/IP channel numbers depend on the customisation on the remote side. In this example, the CTC read channel `0c04` is connected to the VM TCP/IP write channel `0c05`. Similarly, CTC write (`0c05`) is connected to VM TCP/IP read (`0c04`).

You can write the `define` and `couple` commands into the CMS `PROFILE EXEC A` script. The LINUX for S/390 virtual machine must always be IPLed as CMS before IPLing as LINUX in order for these commands to take effect.

Instead of connecting to the VM TCP/IP user ID, you can connect to any other virtual machine in which a LINUX for S/390, OS/390, or VSE system is running.

2. Definitions on the remote side

   Set up the TCP/IP on the remote side, as described in the reference manuals. This will vary depending on which operating system is used on the remote side.

   Note: It is important that you have `IOBUFFERSIZE 32678` defined because the LINUX for S/390 `CTC` driver works with 32k internally. This is configurable for each device by writing the value to the buffersize file for that device (`/proc/net/ctc/<devicename>/buffersize`), for example

   ```
   echo 32768 > /proc/net/ctc/ctc0/buffersize
   ```

3. Activation on the remote side

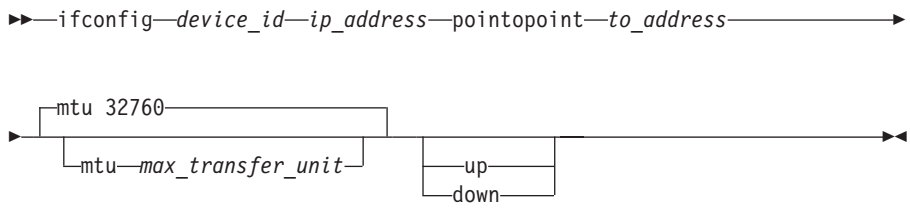
   Activate the channels on the remote side. This again will vary depending on the operating system used on the remote side.

4. Activation on the LINUX for S/390 side

   The network devices are activated with the `ifconfig` command. It is necessary to define the right MTU size for the channel device, otherwise it will not work properly. Please use the same MTU size (default 1500) that is defined on the remote side:

   The syntax of this command is:

CTC ifconfig command



Where:

device_id
: identifies the device. (`ctc0` to `ctc`*n* or `escon0` to `escon`*n*)

ip_address
: is the IP address of the local side.

to_address
: is the IP address of the remote side.

max_transfer_unit
: is the size of the largest IP packet which may be transmitted

up activates the interface

down deactivates the interface

An example of the use of `ifconfig` is:

```
ifconfig ctc0 10.0.51.3 pointopoint 10.0.50.1 mtu 32760
```

If you are using a CTC-based tty connection you must create a device node with major number 43 in the LINUX `/dev` directory:

```
mknod /dev/ttyZ0 c 43 0
mknod /dev/ttyZ1 c 43 1
```

and so on

No network device setup is needed in this case. The CTC-based tty emulates a standard serial port including the usual handshake lines (RTS/CTS/DTR/DSR/CD). To establish a connection, simply open the previously created device (`/dev/ttyZx`) on both peers using a standard terminal emulator or activate a standard `getty` on it.

Notes

Device major number 43 is reserved on PC architecture for `/dev/isdn`. This number has been allocated to CTC/ESCON on LINUX for S/390 because on S/390 there is no ISDN support. The connection is established when the tty device is opened. Following closure of the tty device, shutdown of the connection is delayed for about ten seconds. This delay has been implemented to avoid unnecessary initialization sequences if programs quickly open and close the device . For this reason, if the driver is loaded as a module, it can only be unloaded after first closing all CTC-based ttys and then waiting for this delay to expire.

CTC/ESCON – Recovery procedure after a crash

In a native LINUX for S/390 system if one side of a CTC connection crashes it is not possible to simply reconnect to the other side after a reboot. The correct procedure is:

1. Stop the CTC connection on the LINUX for S/390 side using (for instance):

   ```
   ifconfig escon0 down
   ```

2. Activate the channels on the remote side.
3. Activate the channels on the LINUX for S/390 side, for example:

   ```
   ifconfig escon0 10.0.0.1 pointopoint 10.0.50.1 mtu 32760
   ```

Chapter 8. LINUX for S/390 IUCV device driver

The Inter-User Communication Vehicle (IUCV) is a VM/ESA communication facility that enables a program running in one virtual machine to communicate with another virtual machine, or with a control program, or even with itself. The communication takes place over a predefined linkage called a path.

The LINUX for S/390 IUCV device driver is a network device, which uses IUCV to connect LINUX kernels running on different VM user IDs, or to connect a LINUX kernel to another VM guest such as a TCP/IP service machine.

IUCV features

The following features are supported:

- Multiple output paths from a LINUX guest
- Multiple input paths to a LINUX guest
- Simultaneous transmission and reception of multiple messages on the same or different paths
- Network connections via a TCP/IP service machine gateway

IUCV kernel parameter syntax

The driver must be loaded with the IDs of the guest machines you want to connect to:

IUCV kernel parameter

Parameter:

userid Name of the target VM guest machine

IUCV kernel parameter example

The following picture shows the possible connection of two LINUX for S/390 machines:

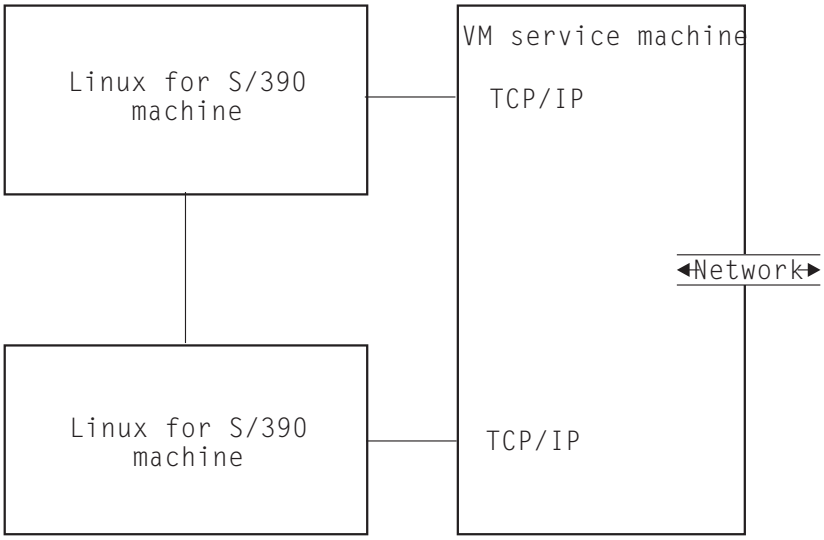


Figure 4. Connection of two systems using IUCV

The command

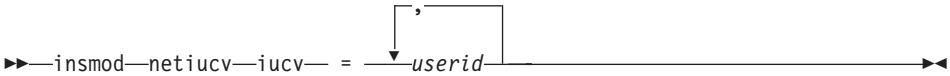
```
iucv=vmtcpid,linux2
```

connects the LINUX system to the TCP service machine and the other LINUX system.

IUCV module parameter syntax

The driver must be loaded with the IDs of the guest machines you want to connect to:

IUCV module parameter



Parameter:

userid Name of the target VM guest machine

IUCV module parameter example

The example of “IUCV kernel parameter example” could be set up by starting the IUCV module with:

```
insmod netiucv iucv=vmtcpid,linux2
```

IUCV – Preparing the connection

This is an additional task that you must perform before you can use the IUCV network link. If LINUX is being used as a network hub instead of VM TCP/IP, the concepts discussed remain the same, though the syntax will be different.

The following steps must be undertaken in VM:

1. Obtain a subnet from your TCP/IP communications staff. It is important that the subnet used by your LINUX guests not be the same as that used by VM on the LAN. It is a separate network and should be treated as such.
2. Take one address from that subnet and assign it to VM. Update your `PROFILE TCPIP` file with a home entry, device, link, and start statements for each guest, for example:

```
Home
 vm_ip_address link_name1
 vm_ip_address link_name2

Device device_name1 IUCV 0 0 linux_virtual_machine1 A
Link link_name1 IUCV 0  device_name1

Device device_name2 IUCV 0 0 linux_virtual_machine2 A
Link link_name2 IUCV 0  device_name2

Start device_name1
Start device_name2
```

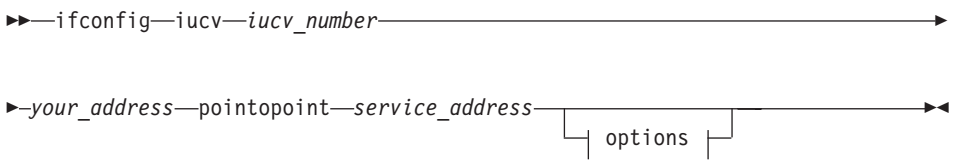
3. Add the necessary VM TCP/IP routing statements (BsdRoutingParms or Gateway). Use an MTU size of 9216 and a point-to-point host route (subnet mask 255.255.255.255). If you use dynamic routing, but do not wish to run routed or gated on LINUX , update the VM `ETC GATEWAYS` file to include ″permanent″ host entries for each LINUX guest.
4. Bring these updates online by using `OBEYFILE` or by recycling `TCPIP` and/or `ROUTED` as needed.
5. Add the statement

   ```
   IUCV ALLOW
   ```

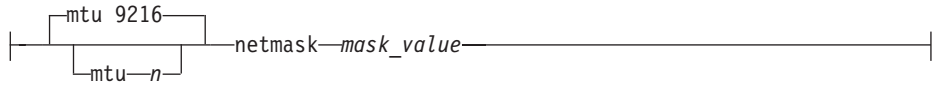
   to your VM user directory entry.

The LINUX commands needed to start communications through a TCP/IP service machine are:

TCP/IP ifconfig command



options:



Parameters:

iucv_number
Path number (for example 0)

your_address
TCP/IP address of your machine

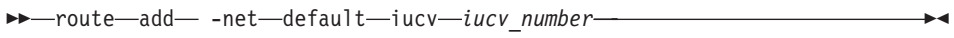
pointopoint
required to establish a point-to-point connection to a service machine

service_address
Address of the TCP/IP service machine to connect to

n maximum transfer unit size. The default is 9216, which is suitable for use with the S/390 Virtual Image Facility for LINUX (VIF). The maximum value is 32764.

mask_value
Mask to identify addresses served by this connection

route command



Parameters:

iucv_number
Path number defined above

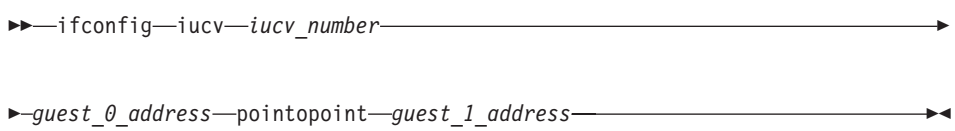
inetd command

Notes:

1 Not required if the IUCV driver is started during boot.

The commands needed to start direct communications to another guest are:

user-to-user ifconfig command



Parameters:

iucv_number
Path number (for example 0)

guest_0_address
TCP/IP address of your machine

guest_1_address
TCP/IP address of target machine

IUCV – Further information

The standard definitions in the VM TCP/IP configuration files apply.

For more information of the VM TCP/IP configuration see: *VM/ESA TCP/IP Planning and Customization* , SC24-5847-01.

IUCV restrictions

- This device driver is only available to LINUX for S/390 systems running as guests under VM/ESA.

Chapter 9. LINUX for S/390 LCS Device Driver

This driver is subject to licence conditions as reflected in: "International License Agreement for Non-Warranted Programs" on page 129.

This LINUX network driver supports LAN Channel Station (LCS) Ethernet and Token Ring access through the OSA-2™ card.

The LCS network interface has two channels, one read channel and one write channel. This is very similar to the S/390 `CTC` interface (see "Chapter 7. LINUX for S/390 CTC/ESCON device driver" on page 37). The read channel must have model type 0x3088 and an even `cuu` number. The write channel also has a model type of 0x3088 and has a `cuu` number one greater than the read `cuu` number. Only certain `cuu` types are supported so as not to clash with a `CTC` control unit type.

The driver always has a read outstanding on the read subchannel. This is used to receive command replies and network packets (these are differentiated by checking the type field in the `LCS` header structure). Any network packets that arrive during the startup and shutdown sequence have to be discarded. During normal network I/O, the driver will intermittently retry reads in order to permanently keep a read outstanding on the read channel. (This is in case an `-EBUSY` or similar occurs, in which case the driver would stop receiving network packets.)

The default configuration is to use software statistics, with IP checksumming off (this improves performance) and to have network hardware checking using a CRC32 check (CRC64 for FDDI) which should guarantee integrity for normal use. However, financial institutions or similar might want the additional security of IP checksumming.

Additional `CUU` model types can be added later so that new `LCS` compatible cards will be supported even if not available when the driver was developed.

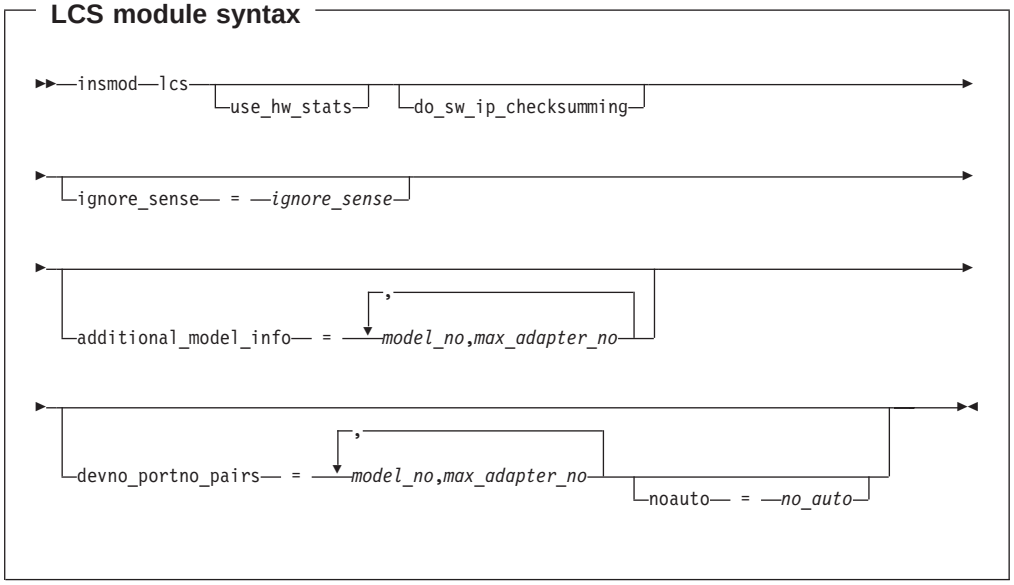
LCS features

- Supports Ethernet and Token Ring
- Auto detects whether card is connected to Token Ring or Ethernet
- Can be configured from the `insmod` parameters .

LCS configuration

Module parameter syntax

The following are the `LCS` device driver module parameters:



The meanings of these keywords are:

use_hw_stats
Get network statistics from the `LANSTAT LCS` primitive as opposed to doing it in software. This is not recommended as it is incompatible with many network drivers

do_sw_ip_checksumming
Perform IP checksumming on inbound packets in the software. Normally not required because Ethernet CRC32 is usually more than adequate (except perhaps for financial institutions).

ignore_sense
You should set `ignore_sense` to '1' if you want to boot devices which do not report a valid `sense_id`.

additional_model_info
This is made up of sets of model number / maximum relative adapter number pairs (see example below).

devno_portno_pairs
This takes `devno,rel_adapter_no(port)` pairs. A relative adapter number of `-1` indicates that you should not use this adapter. This can be used to force certain CHPIDs to use a particular port number if the LCS protocol returns an incorrect one.

noauto
Set `noauto=1` if you want to set auto-detection off. You must then configure LCS devices explicitly with the `devno_portno_pairs` module parameter.

Module parameter example

```
insmod lcs additional_model_info=0x70,3,0x71,5
          devno_portno_pairs=0x1c00,0,0x1c02,1,0x1d00,-1
```

This tells the LCS device driver to:

- look for 3 ports on a 3088 model 70 and 5 ports on a model 71
- only use `port 0` if available for the device numbers `0x1c00` and `0x1c01`

- only use `port 1` if available for the device numbers `0x1c02` and `0x1c03`
- under no circumstances use the device at `0x1d00` and `0x1d01` as an LCS device.

LCS restrictions

- LINUX for S/390 cannot run with a root file system mounted via NFS when the network connectivity is established via LCS, because the LCS driver is delivered as an object-code-only module.
- To use OSA-2 devices when running LINUX for S/390 on a basic mode machine (no LPARs) you may need to specify an `ipldelay=`*xyz* boot parameter. We recommend a value between `2m` and `5m` for *xyz* for the OSA-2 card to settle down after `LOAD`.
- Currently, there is only support for up to 16 Token Ring or Ethernet devices. However, we strongly recommend that you do not share devices with production systems.

LCS limitations

- FDDI is untested and the code shipped with kernel patch 2.2.16 is unlikely to work on FDDI
- Because LCS does not appear to tell the driver that a port is busy, it is sometimes necessary to force the driver not to use the ports. The `devno_portno_pairs` or `lcs_devno` kernel parameters are used to do this.
- The most problematic area for the code is starting up and shutting down the driver.

  This is primarily due to the fact that network packets can be received during the startup process before receiving the `lanstat` command to get the `mac` address. This can happen earlier if the card wasn't previously shut down properly.

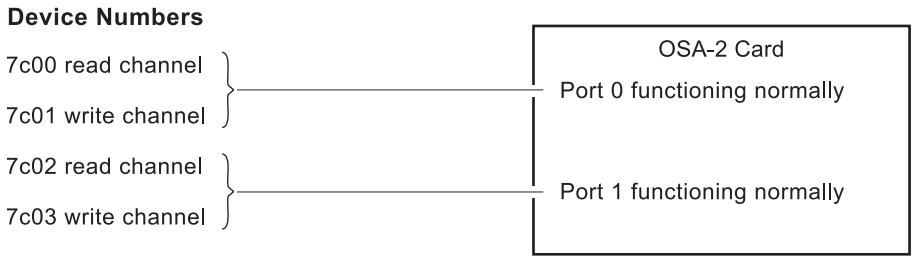
  If the card is being very troublesome, use `ifconfig` to switch it on and off.

  If this fails, compile the driver as a module. Use `insmod` and `rmmod`, as these are guaranteed to call the startup and shutdown routines, whereas the kernel keeps a reference count (doing `ifconfig` up twice will call the startup routine only once).

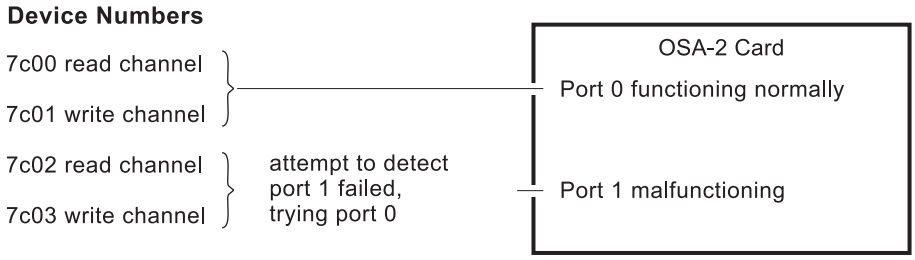
LCS – Common set up problem

The LCS device driver sometimes has a set up problem. The same port on the OSA/2 card is taken twice by the LCS driver. This port can be communicated to via two or more pairs of device numbers. The driver attempts to determine the port number from the low byte of the device number, however the LCS microcode does not indicate that the port is already in use. If the first attempt is wrong (port already in use), the driver may use the same port twice with different pairs of device numbers. This can be better explained via the following diagrams.

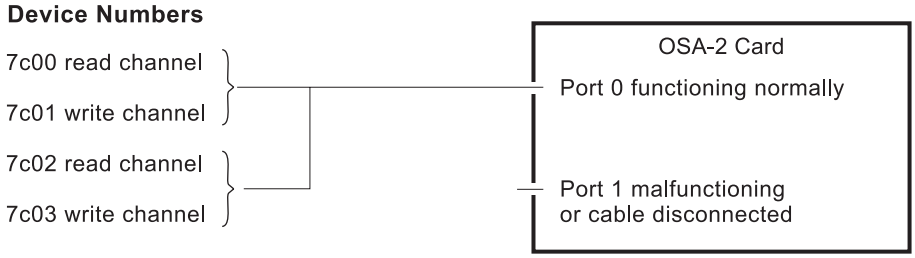
Normal case:



Abnormal case - part 1:



Abnormal case - part 2 (two or more connections on the same port):



The common symptoms of this problem are:

1. Duplicate `mac` addresses (most common)
2. Two or more network interfaces detected of the same type despite the fact that port *X* is configured as Token Ring and port *Y* as Ethernet
3. Disconnecting network cable from one port disables network traffic on two or more ports.

If this problem occurs, use the `devno_portno_pairs` parameter to force the problematical device numbers to specific ports.

Chapter 10. LINUX for S/390 OSA-Express device driver

This driver is subject to licence conditions as reflected in: "International License Agreement for Non-Warranted Programs" on page 129.

This LINUX network driver supports the OSA-Express cards. These enable the S/390 to connect to Gigabit Ethernet networks, or to Ethernet or Fast Ethernet networks through a switch. [4]

OSA-Express features

The following features are supported:

- Autosensing of devices
- Primary and secondary routers
- Priority queueing
- Individual device configuration.
- IP Address Takeover

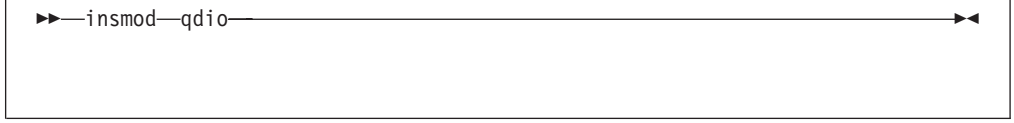
OSA-Express configuration

Module parameter syntax

Before the OSA-Express device driver can be loaded the `QDIO` protocol driver must be loaded.

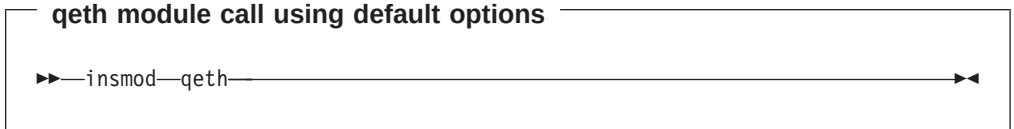
The command to load the protocol driver is:

qdio module call



When the `QDIO` protocol has been loaded the OSA-Express driver (`qeth`) is loaded with the command:

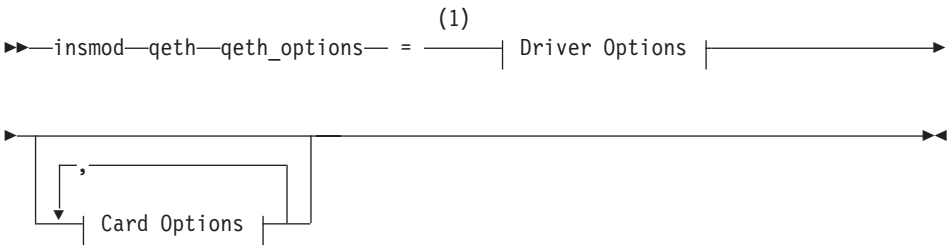
qeth module call using default options



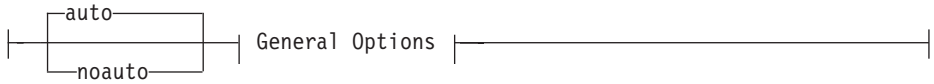
or:

4. Please read the licence on page "International License Agreement for Non-Warranted Programs" on page 129.

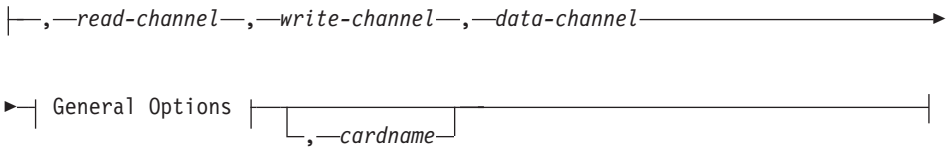
qeth module call specifying options



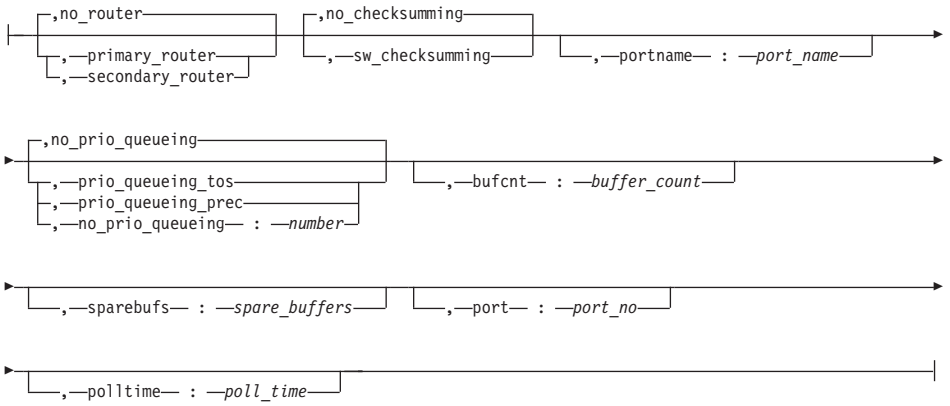
Driver Options:



Card Options:



General Options:



Notes:

1 All options except the first used must be preceded by a comma.

Note: All characters must be entered in lower case as shown, except in hexadecimal numbers where either case may be used.

The meaning of the parameters of this command is as follows:

auto | noauto
Specifies whether the driver is to search for all subchannels (auto), or is only to use device numbers specified in *card options*.

read-channel,write-channel,data-channel
identify the card addresses.

cardname
identifies the card. Note that the cardname (if used) must follow the general options.

primary_router | secondary_router | no_router
Specifies whether the device is used to interconnect networks. A ″Primary router″ is the principal connection between two networks; a ″Secondary router″ is used as backup in case of problems with the primary. Both of these options require the LINUX system to be configured as a router. The default for this parameter is ″No router″ – the OSA-Express card will only be used to connect the LINUX for S/390 system to a single network.

It is possible to add routing status dynamically. This is done with the command:

```
echo primary_router ifname > /proc/qeth
```

or

```
echo secondary_router ifname > /proc/qeth
```

ifname is the name of the interface in LINUX , for example `eth0`.

It is not possible to reset routing status with the current hardware.

sw_checksumming | no_checksumming
Specifies whether error detection is to be performed by the driver, or is not required.

port_name
Identifies the port for sharing by other OS images, for example the 'PORTNAME' dataset used by OS/390. *port_name* is 1 to 8 characters long.

prio_queueing_tos | prio_queueing_prec | no_prio_queueing | no_prio_queueing: *number*
Specifies the type of priority queuing to be used. See "OSA-Express queuing" on page 59 for details. **no_prio_queueing** is equivalent to **no_prio_queueing:***2*

buffer_count
Specifies the number of inbound buffers used. Valid values for *buffer_count* are 16 to 128. The default is 128.

This may be used to overcome problems with memory shortage. The size of each buffer is 64 kilobytes.

spare_buffers
Specifies the number of spare buffers to reserve. The default is none. These buffers are pre-allocated and can be used as a safety valve if excessive load fills the normal buffer pool.

port_no
Specifies the port number on the CHPID. The default port number is 0.

poll_time
Specifies the maximum duration of background polling (in microseconds) used by QDIO. The default is 5000.

<card options> are used to override the global options for a particular device. These are also comma-separated lists. The card is identified by its three device numbers (in decimal, or in hexadecimal prefixed with 0x). These may be followed by any of the <*driver options*> keywords except **auto** or **noauto**.

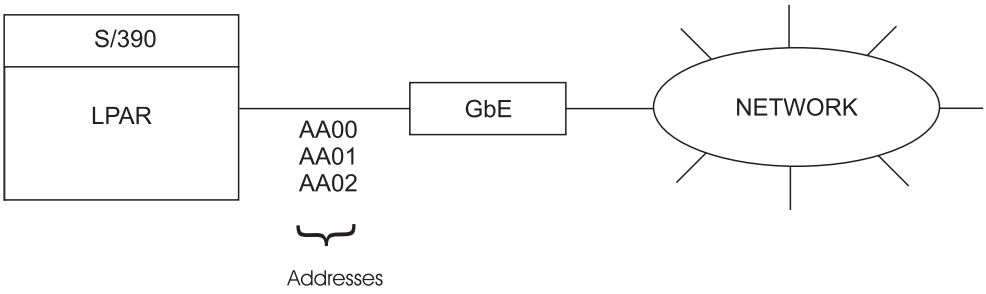
<*driver options*> which are valid as <*card options*> will apply to all cards detected, unless overridden in any <*card options*>.

Examples

1: Basic configuration

In this example a single Gigabit Ethernet card is being used to connect a LINUX for S/390 system to a network.

Hardware configuration – Gigabit Ethernet connecting LINUX for S/390 to a network.



Software configuration – Gigabit Ethernet connecting LINUX for S/390 to a network.

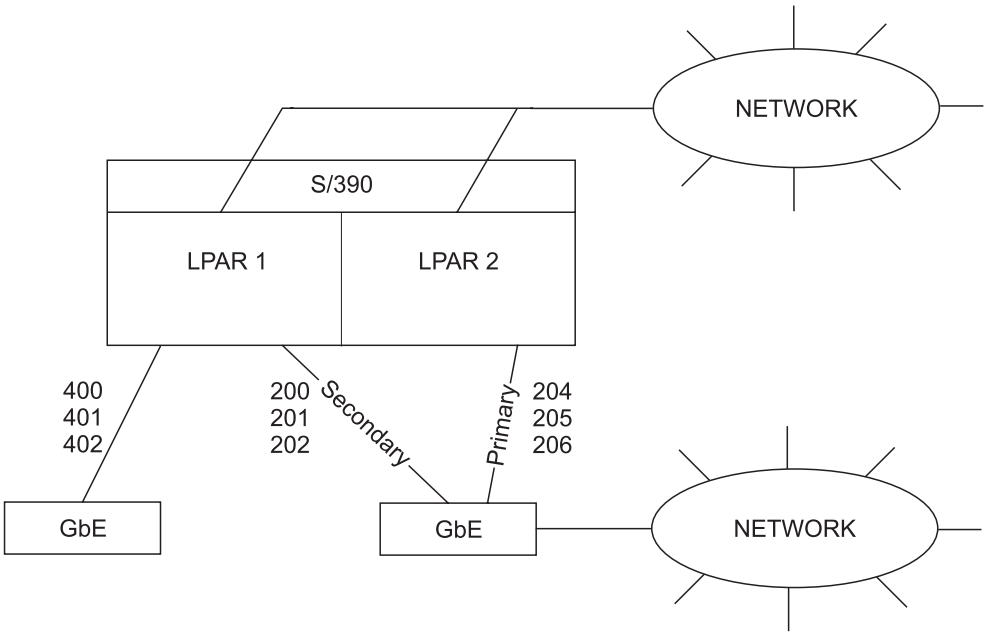
The load commands for this configuration are:

```
insmod qdio
insmod qeth qeth_options=noauto,0xAA00,0xAA01,0xAA02
```

2: Router configuration

This example shows how LINUX systems running on different LPARs in an S/390 may use Gigabit Ethernet cards to communicate with a network or to act as a router between networks.

Hardware configuration – Gigabit Ethernet and LINUX for S/390 as router.



In this example it is assumed that LINUX is configured as a router in both LPARs.

Software configuration – Gigabit Ethernet and LINUX for S/390 as router.

LPAR 1 – uses the first subchannel group as a network client, and the second subchannel group as a backup router for LPAR 2:

```
insmod qdio
insmod qeth qeth_options=noauto,0x400,0x401,0x402,0x200,0x201,0x202,secondary_router
```

LPAR 2 – uses the third subchannel group as a primary router:

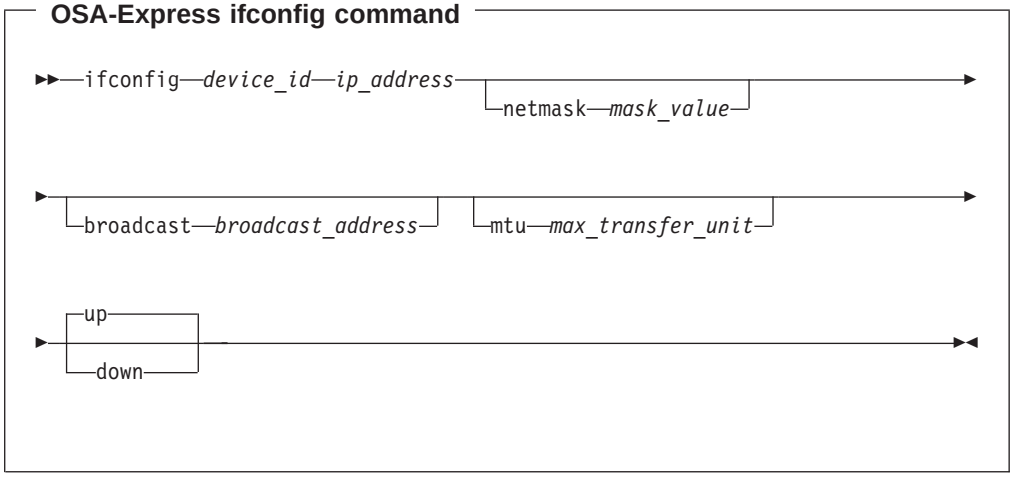
```
insmod qdio
insmod qeth qeth_options=0x204,0x205,0x206,primary_router
```

OSA-Express – Preparing the connection

Activating the OSA-Express connection

The network devices can be activated with the `ifconfig` command. It is necessary to define the right MTU size for the channel device, otherwise it will not work properly. You must use the same MTU size (default 1492) that is defined on the remote side:

The syntax of this command is:



Where:

device_id
identifies the interface

ip_address
is the IP address of the remote side.

mask_value
is the IP network mask for this interface

broadcast_address
is the address used to send to all devices on the connection

max_transfer_unit
is the size of the largest block which may be carried

up activates the interface

down deactivates the interface

An example of the use of `ifconfig` is:

```
ifconfig eth0 192.168.100.11 netmask 255.255.255.0
             broadcast 192.168.100.255 mtu 1492 up
```

or, more simply:

```
ifconfig eth0 192.168.100.11
```

OSA-Express device recognition

With autosensing on any device addresses specified in the driver parameter are checked, followed by a scan of all other device subchannels in ascending order. With autosensing switched off only the device addresses specified in the driver parameter are checked. This may be necessary to prevent the card from taking other device numbers if the specified numbers cannot be used.

A subchannel is checked to see if:

- it is a **QDIO** eligible subchannel

- the channel is not already in use
- it is on an unused CHPID (unless explicitly specified in options)
- the device type and model are known

When one subchannel has been verified a search for two more is carried out on the same CHPID. The same criteria apply. When a group of three subchannels has been found, the driver signals a request for these and the card is set up.

OSA-Express restrictions

- Currently the I/O Layer does not provide CHPID information, so all devices are seen as on one CHPID. This means only one device will automatically be detected.
- The MTU range is 576 – 56000. Standard sizes used are 576, 1492, 1500, 8992 and 9000.
- The QDIO based Ethernet Driver `qeth.o` does not work with full autosensing in any situation. It uses different heuristics for finding subchannels when fully autosensing compared to the case where device numbers are specified. In the latter case, the specified device numbers are always used and the `qeth` driver recognizes the device. When fully autosensing, the driver is sometimes not able to recognize a group of three subchannels, as it does not try all permutations of the subchannels. The problem will appear when:
  1. there are only three subchannels
  2. the device numbers of these are consecutive and start with an odd number, for example 0x1103, 0x1104 and 0x1105

  In this case full autosensing does not work, but

```
insmod qeth qeth_options=0x1103,0x1104,0x1105
```

  will work. To avoid this problem either attach four subchannels to each VM guest or LPAR using the card, or ensure that the subchannels start with an even number.

OSA-Express queuing

The OSA-Express card has four output queues (queues 0 to 3) in central storage. The card gives these queues different priorities (queue 0 having the highest priority) which is relevant mainly to high traffic situations. When there is little traffic queuing has no impact on processing.

The device driver can put data on one or more of the queues. By default it uses queue 2 for all data. However, the driver can look at outgoing IP packets and select a queue for the data according to the IP type of service (if `prio_queueing_tos` is specified in the options) or IP precedence (if `prio_queueing_prec` is specified in the options) fields. These fields are part of the IP datagram header and can be set with a `setsockopt` call.

Some applications use these fields to tag data. The mapping the driver performs between IP type of service is as follows:

IP type of service	queue used when IP TOS queueing is switched on
not important	3

IP type of service	queue used when IP TOS queueing is switched on
low latency	0
high throughput	1
high reliability	2
everything else	2

When IP precedence was selected as queueing type, the two most significant bits of each IP header precedence field are used to determine the queue for this packet.

OSA-Express IP Address Takeover

It is possible to add and remove ranges of IP addresses for the OSA-Express card by writing to the `/proc/qeth_ipa_takeover` file. When a command is written to this file the driver calls on the `OSA` "Address Takeover" mechanism. This overrides any previous allocation of the specified address to another LPAR or card. If another LPAR on the same card has already registered for that IP address this association will be removed.

The registered addresses are held in this file in plain text and can be read to see the current associations.

Only one command at a time can be written to the file. Subsequent commands in the sane write action are ignored.

The following commands are available:

- `inv4`
- `inv6`
- `add4 <addr>/<mask_bits>[:<interface>]`
- `add6 <addr>/<mask_bits>[:<interface>]`
- `del4 <addr>/<mask_bits>[:<interface>]`
- `del6 <addr>/<mask_bits>[:<interface>]`

`inv4` and `inv6` toggle the IPA takeover behavior for all interfaces: if `inv4` is input once all addresses which have been specified with `add4` are unset using the takeover mechanism, but all other IPv4 addresses are set.

`add4` or `add6` adds an address range. `del4` or `del6` deletes an address range. <*addr*> is a hexadecimal IP address in 8 bytes or 32 bytes. <*mask_bits*> specifies the number of bits which are set in the network mask. <*interface*> is optional and specifies the interface name to which the address range is bound.

For example

```
echo add4 c0a80100/24 > /proc/qeth_ipa_takeover
```

activates all addresses in the 192.168.10 subnet for address takeover.

Note that the address is not actually taken over until a corresponding `ifconfig` command is executed; for example

```
ifconfig 192.168.10.5
```

sets the IP address 192.168.10.5 on the card and removes it from other LPARs, if necessary.

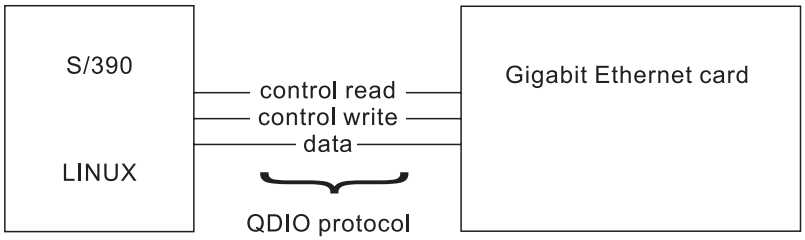
OSA-Express background – QDIO

The QDIO protocol governs the interface between the S/390 and the OSA-Express card.

The OSA-Express is optimized for low latency and SMP environments (rather than high throughput).

For OSA-Express devices three I/O channels must be available to the driver. One channel is for control reads, one for control writes, and the third is for data.

Example card layout



Part 4. Installation commands and parameters

This section describes configuration parameters for LINUX for S/390 and the tools available for configuration.

These are described in the chapters:

- Useful LINUX commands:
  - dasdfmt - Format a DASD,
  - ifconfig - Configure a network interface,
  - insmod - Load a module into the LINUX kernel,
  - modprobe - Load a module with dependencies into the LINUX kernel,
  - lsmod - List loaded modules,
  - depmod - Create dependency descriptions for loadable kernel modules,
  - mke2fs - Create a file system,
  - silo - Make DASD bootable.
- Kernel parameters:
  - ipldelay,
  - maxcpus,
  - mem,
  - noinitrd,
  - ramdisk_size,
  - ro,
  - root,
  - vmhalt,
  - cio_msg.
- Overview of the parameter line file.

Chapter 11. Useful LINUX commands

This chapter describes commands which have been created to install and configure LINUX for S/390 (dasdfmt).

It also summarizes standard LINUX commands used during the installation, configuration and initial startup of LINUX for S/390. These are:

- ifconfig,
- insmod,
- modprobe,
- lsmod,
- depmod,
- mke2fs,
- and silo.

dasdfmt - Format a DASD

Purpose

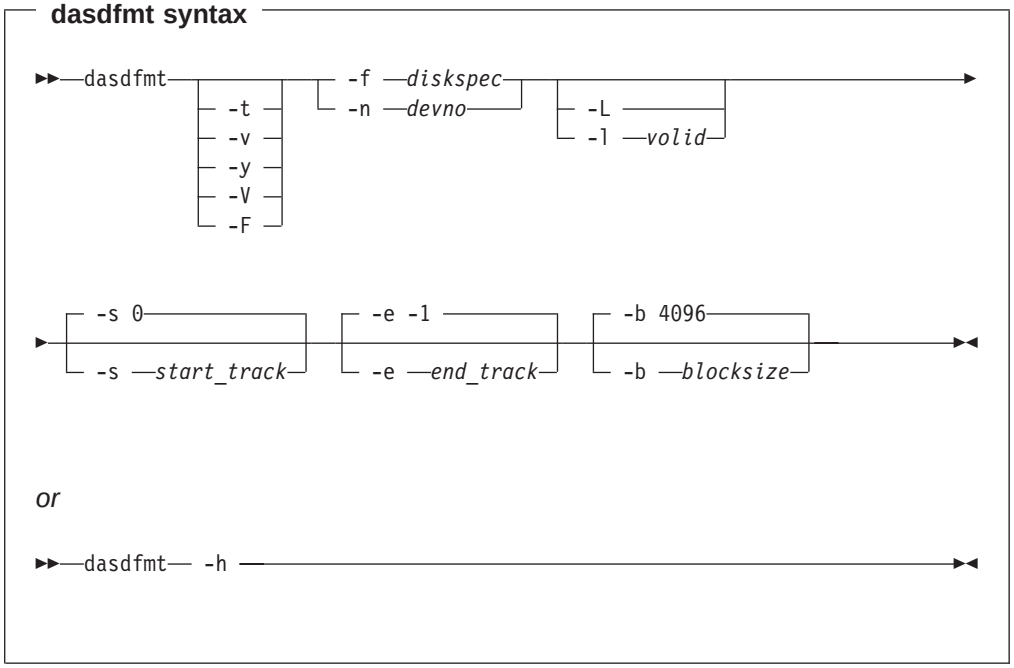
This tool is used to give a low-level format to direct access storage devices (DASD). Note that this is a software format. To give a hardware format to raw DASD you must use another S/390 device support facility such as ICKDSF, either in stand-alone mode or through another operating system.

`dasdfmt` uses an `ioctl` call to the DASD driver to format tracks. A start and end track for formatting can be specified, as well as a blocksize (hard sector size). Remember that the formatting process can take quite a long time.

Usage

Prerequisites:

Format



The parameters are:

-f *diskspec*
: Specifies the device node in the file system. This must be the whole device, not a partition.

-n *devno*
: Specifies the four-character hexadecimal device address of the disk to format, for example `-n 01a3`.

The following parameters are necessary, however if you do not specify their values you are prompted for them. You can use the default values by pressing the <enter> key:

-b *block_size*
Specifies the blocksize. The minimum blocksize is 512 bytes and increases in powers of 2 (512, 1024, 2048, 4096 and so on). The default blocksize is 4096.

The following parameters are optional:

-h Prints out an overview of the syntax. Any other parameters will be ignored.

-t or **--test**
(test mode) Analyses parameters and prints out what would happen, but does not modify the disk.

-v Prints out extra information messages.

-y Starts formatting immediately without prompting for confirmation.

-F Formats the device without checking if it is mounted or in use as swap space.

-L or **--no_label**
Valid for `-d ldl` only, where it suppresses the default `LNX1` label.

-V Prints the version number of `dasdfmt` and exits.

Examples

To format two whole disks with a blocksize of 4096, one the device nodes /dev/dasda and the other at address 0x0293:

```
dasdfmt -b 4096 -f /dev/dasda
dasdfmt -b 4096 -n 0293
```

ifconfig - Configure a network interface

Usage

`ifconfig` is used to configure the kernel-resident network interfaces. It is used at startup time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.

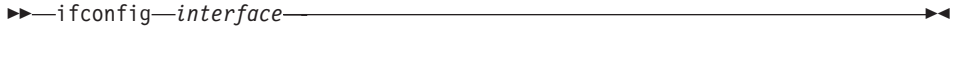
- If no arguments are given, `ifconfig` displays the status of the currently active interfaces.
- If a single interface argument is given, it displays the status of the given interface only
- Otherwise, it configures an interface. The configurable interfaces for S/390 are:
  - `iucv`
  - `ctc`*i* (*i* = 0 to 255)
  - `escon`*j* (*j* = 0 to 255)

Note: Since kernel release 2.2 there are no explicit interface statistics for alias interfaces anymore. The statistics printed for the original address are shared with all alias addresses on the same device. If you want per-address statistics you should add explicit accounting rules for the address using the `ipchains` command.

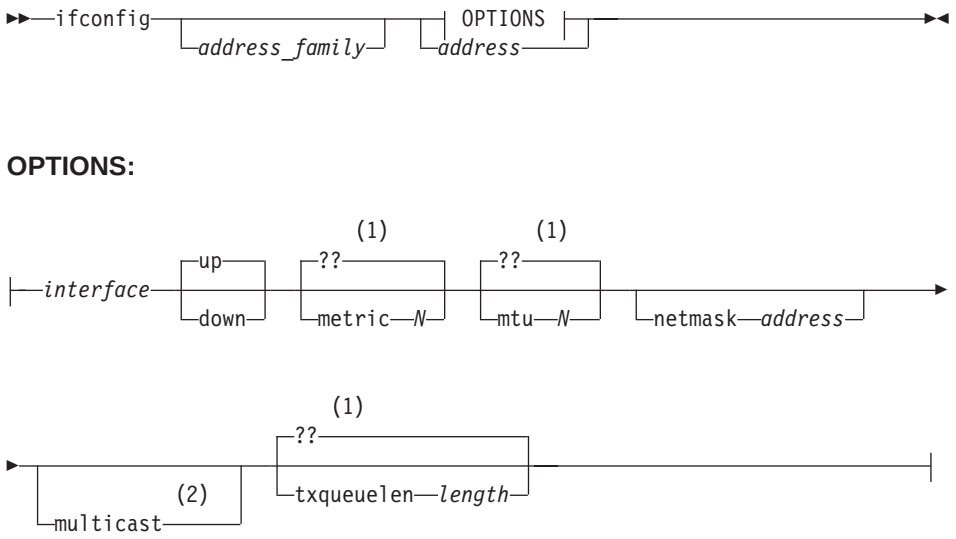
Format

Display active interface status

Display status of given interface



Activate or shut down an interface



Notes:

1 Distribution dependent

2 This should not normally be needed as the drivers set the flag correctly themselves.

address_family

If the first argument after the interface name is recognized as the name of a supported address family, that address family is used for decoding and displaying all protocol addresses.

On S/390, supported address families include:

- inet

interface

The name of the interface. This is usually a driver name followed by a unit number, for example eth0 for the first Ethernet interface.

On S/390 the supported interfaces include:

- `escon0 - escon255`
- `ctc0 - ctc255`
- `iucv0`
- `lo0` (loopback device)
- `eth0`
- `tr0`

up This flag causes the interface to be activated. It is implicitly specified if an address is assigned to the interface.

down This flag causes the driver for this interface to be shut down.

metric N
This parameter sets the interface metric.[5]

mtu N This parameter sets the maximum transfer unit (MTU) of an interface.

netmask addr
Set the IP network mask for this interface. This value defaults to the usual class A, B or C network mask (as derived from the interface IP address), but it can be set to any value.

multicast
Set the multicast flag on the interface. This should not normally be needed as the drivers set the flag correctly themselves.

address
The IP address to be assigned to this interface.

txqueuelen length
Set the length of the transmit queue of the device. It is useful to set this to small values for slower devices with a high latency (modem links, ISDN) to prevent fast bulk transfers from disturbing interactive traffic like `telnet` too much.

Files:

```
/proc/net/socket
/proc/net/dev
```

5.

Metric: Cost of an OSPF interface. The cost is a routing metric that is used in the OSPF link-state calculation. To set the cost of routes exported into OSPF, configure the appropriate routing policy.

- Range: 1 through 65,535
- Default: 1

All OSPF interfaces have a cost, which is a routing metric that is used in the OSPF link-state calculation. Routes with lower total path metrics are preferred over those with higher path metrics. When several equal-cost routes to a destination exist, traffic is distributed equally among them. The cost of a route is described by a single dimensionless metric that is determined using the following formula:

```
cost = ref-bandwidth / bandwidth
```

Where `ref-bandwidth` is the reference bandwidth. Its default value is 100 Mbps (which you specify as 100000000), which gives a metric of 1 for any bandwidth that is 100 Mbps or greater.

Examples

To start or modify an ESCON interface in LINUX:

```
ifconfig escon0 x.x.x.x pointopoint y.y.y.y netmask 255.255.255.255 mtu mmmmm
```

where:

x.x.x.x
is the IP address of the LINUX side

y.y.y.y is the IP address of the remote partner OS/390

mmmmm
is the MTU size which could be up to 32760 - make sure the other side of the channel uses the same MTU size

The ESCON CTC device addresses are defined in the kernel parameter file, or when loading the module:

```
...... ctc=0,0xddd,0xyyy,escon0
```

Another example, showing how to set up an ethernet connection:

```
ifconfig eth0 192.168.100.11 netmask 255.255.255.0 broadcast 192.168.100.255 mtu 1492 up
```

or, simply:

```
ifconfig eth0 192.168.100.11
```

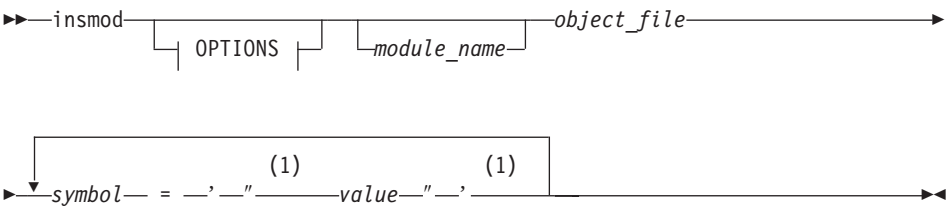
insmod - Load a module into the LINUX kernel

Usage

`insmod` installs a loadable module in the running kernel. It tries to link a module into the kernel by resolving global symbols in the module with values from the kernel's symbol table. If the object file name is given without extension, `insmod` will search for the module in common default directories. The environment variable MODPATH can be used to override this default.

Format

Load a module into the kernel



OPTIONS:

Notes:

1 Both quotes are needed to prevent the shell stripping the quote characters from the parameter. This can also be accomplished using the escape character '\'.

object_file
Name of module source file. (Name by which module is invoked.)

module_name
Explicit name of module if not derived from the name of the source file.

symbol
Name of parameter specific to module.

value Value of parameter to be passed to module.

-f Attempt to load the module even if the version of the running kernel and the version of the kernel for which the module was compiled do not match.

-k Set the auto-clean flag on the module. This flag will be used to remove modules that have not been used in some period of time (usually one minute).

-m Output a load map, making it easier to debug the module in the event of a kernel panic.

-p Probe the module to see if it could be successfully loaded. This includes locating the object file in the module path, checking version numbers, and resolving symbols.

-s Output everything to `syslog` instead of the terminal.

-X Export the external symbols of the module.

-x Do not export the external symbols of the module.

-v Verbose mode.

Examples

DASD module

```
insmod dasd_mod dasd=192-194,5a10
```

XPRAM module

```
insmod xpram devs=4 sizes=2097152,8388608,4194304,2097152
```

CTC module

```
insmod ctc setup=\"ctc=0,0x0600,0x0601,ctc0\"
```

LCS module

```
insmod lcs additional_model_info=0x70,3,0x71,5
          devno_portno_pairs=0x1c00,0,0x1c02,1,0x1d00,-1
```

QDIO module

```
insmod qdio
```

OSA Express module

```
insmod qeth qeth_options=noauto,0x400,0x401,0x402,0x200,0x201,0x202,secondary_router
```

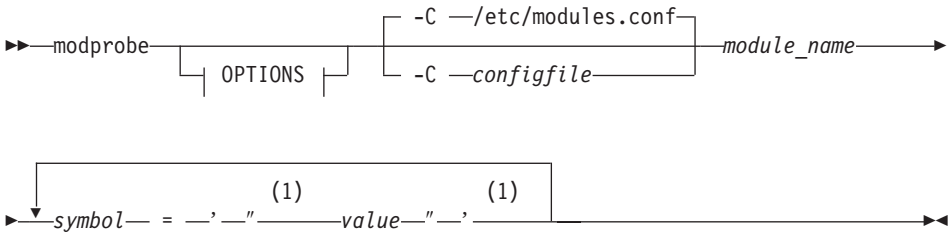
modprobe - Load a module with dependencies into the LINUX kernel

Usage

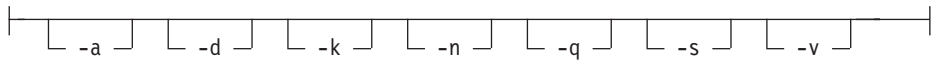
`modprobe` installs a loadable module and all its dependencies in the running kernel. The dependency information is created by `depmod` (see “depmod - Create dependency descriptions for loadable kernel modules” on page 79). It tries to link a module into the kernel by resolving global symbols in the module with values from the kernel's symbol table. If there are still unresolved symbols it will try to satisfy these by loading further modules. If the object file name is given without extension, `modprobe` will search for the module in common default directories. The environment variable `MODPATH` can be used to override this default.

Format

Load a module with dependencies into the kernel



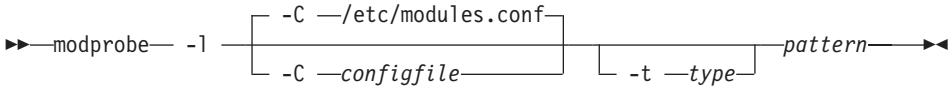
OPTIONS:



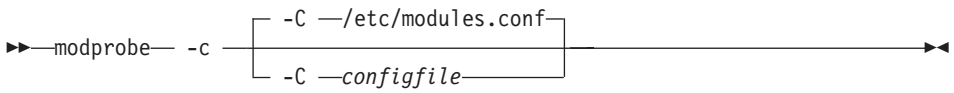
Notes:

1 Both quotes are needed to prevent the shell stripping the quote characters from the parameter. This can also be accomplished using the escape character '\'.

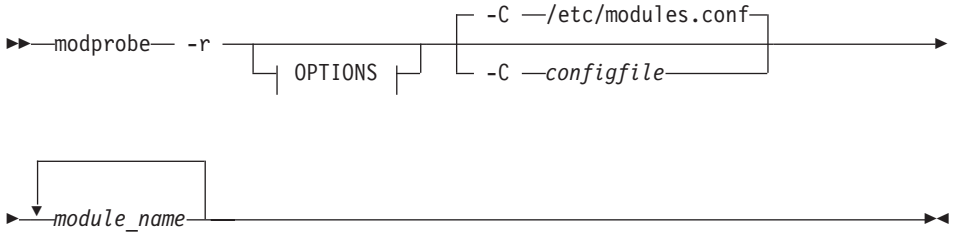
List matching modules



Show configuration



Remove module (stacks) or do autoclean



OPTIONS:

The parameters for the `modprobe` command are:

module_name
: Explicit name of module. (Name by which module is invoked.)

symbol
: Name of parameter specific to module.

value
: Value of parameter to be passed to module.

pattern
: Module name pattern, which may include wildcard characters.

-a
: Load all matching modules (default is to stop after first success).

-d
: Show debugging information.

-k
: Set the auto-clean flag on the module. This flag will be used to remove modules that have not been used in some period of time (usually one minute).

-n
: Probe the module to see if it (and its dependencies) could be successfully loaded, but do not load the module.

-q
: Suppress error messages.

-s
: Send the report to `syslog` instead of `stderr`.

-t
: Only consider modules of type <*type*>.

-v
: Verbose mode.

Comments

Note that this list is not comprehensive. See `man modprobe` for information on the full set of parameters.

Examples

OSA Express module

```
modprobe qeth
```

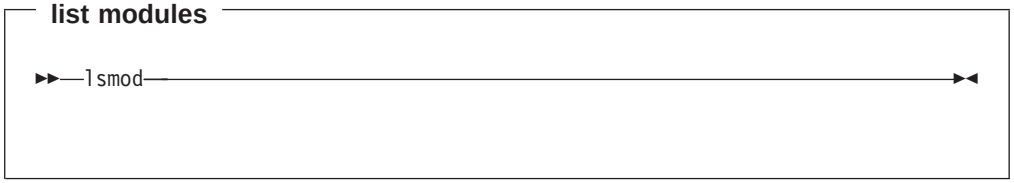
This will attempt to load the `qeth` network driver. It will find a dependency on `qdio`, load the `qdio` base support automatically, and then load `qeth`.

lsmod - List loaded modules

Usage

`lsmod` lists all loaded modules in the running kernel.

Format



Examples

```
# lsmod
Module                  Size  Used by
qeth                  135680   1  (autoclean)
qdio                   22992   1  (autoclean) [qeth]
#
```

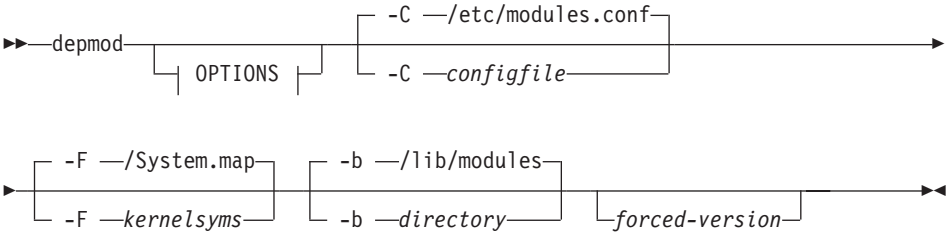
depmod - Create dependency descriptions for loadable kernel modules

Usage

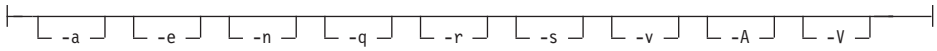
`depmod` creates a dependency file based on the symbols found in a set of modules. This information is used by `modprobe` (qv).

Format

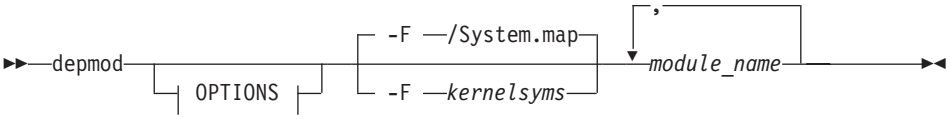
Create dependencies for all files in a directory



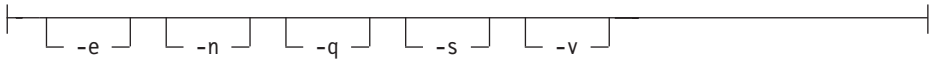
OPTIONS:



Create dependencies for files



OPTIONS:



module_name
: Explicit name of module

-a Search for modules in all directories specified in `/etc/modules.conf` or <*configfile*>.

-b Use `/lib/modules` or <*directory*> as the starting point for the subtree containing the modules.

-e Show all the unresolved symbols for each module.

-n Write the dependency file on `stdout` instead of in the `/lib/modules` tree.

-q (quiet) Suppress error messages about missing symbols.

-s Write all error messages via the `syslog` daemon instead of `stderr`.

-v Show the name of each module as it is analyzed.

-A Like `depmod -a`, but compare file timestamps and only update the dependency file if anything has changed.

-C Use the file <*configfile*> instead of `/etc/modules.conf`.

-F Use the symbol information found in <*kernelsyms*>.

-V Show the release version of `depmod`.

Examples

```
depmod -e -n mymodule
```

displays the unresolved references in *mymodule* on `stdout`.

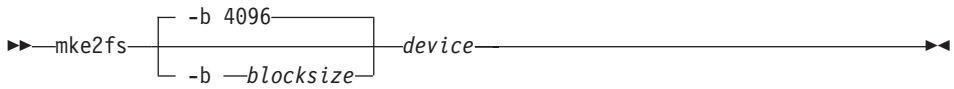
mke2fs - Create a file system on DASD

Usage

This utility creates an ext2 file system on a DASD hard disk. The device must already have a low-level format.

Format

mke2fs syntax



-b *blocksize*
Specifies the blocksize. Default is 4096. The blocksize needs to be a multiple of the blocksize specified on the **dasdfmt** command. This allows you to use block sizes larger than the hardware maximum of 4096.

device Specifies the device node.

Examples

```
mke2fs -b 4096 /dev/dasda1
```

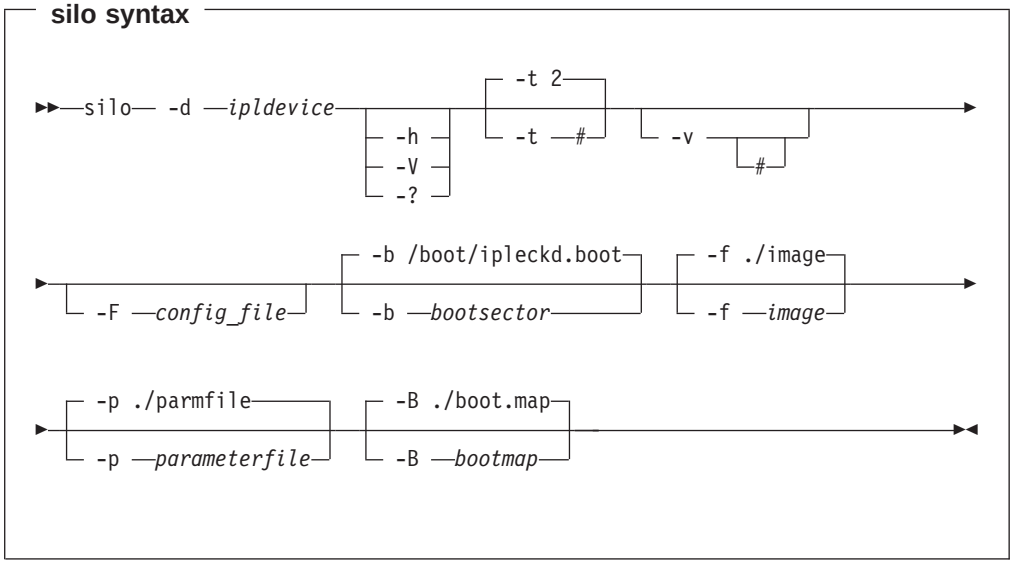
silo - Make DASD bootable

Usage

This tool is used to make DASDs (direct access storage devices) bootable. It takes a kernel image, a parameter file, a boot sector file, and the device node as input. In addition the file `/etc/silo.conf`, or the file specified by using the `-F` file name parameter, is parsed for further options.

Format

silo syntax



Note that the defaults for these parameters can be overwritten by entering keywords in the config-file. The format used for each parameter keyword is shown in mono-spaced text in the following descriptions.

-d ipldevice
Set `ipldevice=`*devicenode* to set the boot device to a specific device node. The device node specified must be the node of the 'full' device and not one of a partition.

-h Prints out a short usage message.

-V Print version number and exit silo.

-? Prints out a short usage message. Equivalent to –h

-t[#] By default, silo runs with a `testlevel` of 2, which means that no modifications are made to the disk. A testing level of 1 means that a bootmap is generated with a temporary file name, but the boot records of the disk are not modified. The disk is made bootable only with a testing level of 0 or below. Set `testlevel=`*level* to decrease the testing level from the default by the value of *level*. Use the short forms `-t` to decrease the testing level by one, or`-t`#, to decrease the testing level by #.

-v[#] Set `verbose=`*level* to specify the level of verbosity used. Increases verbosity, or sets verbosity to #, respectively.

-F config-file
There are some defaults for the most common parameters compiled into the binary. You can overwrite these defaults by your own using

/etc/silo.conf or another config-file specified by -F config-file. All values set by defaults or the config-file can be overwritten using the command line options of silo.

An example of the contents of this file is:

```
image = image ipldevice = /dev/dasdb parmfile = parmfile
   bootsect = ipleckd.boot testlevel = -2
```

-b bootsector
Set bootsect=*bootsect* to specify the name of the bootsector to be used as boot record for that volume. The default name is /boot/ipleckd.boot.

-f image
Set image=*image* to specify the name of the image that is going to be loaded from that volume. The default name is ./image.

-p parameterfile
Set parmfile=*parameter file* to specify the name of the parameter file holding the kernel parameters to be used during setup of the kernel. The default name is ./parmfile.

-B bootmap
Set map=*bootmap* to specify the name of the bootmap used to hold the map information needed during booting. The default name is ./boot.map. In test-only mode this name is replaced by a temporary name.

Chapter 12. Kernel parameters

There are two different ways of passing parameters to LINUX :

- Passing parameters to your kernel at startup time. (The parameter line)
- Configuring your boot loader to always pass those parameters.

The kernel can only handle a parameter line file that is no larger than 896 bytes.

The parameters which affect LINUX for S/390 in particular are:

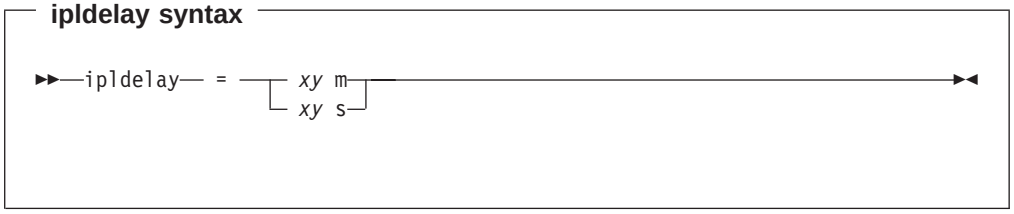
- ipldelay
- maxcpus
- mem
- noinitrd
- ramdisk_size
- ro
- root
- vmhalt
- cio_msg

ipldelay

Usage

When you do a power on reset (POR), some activation and loading is done. This can cause LINUX not to find the OSA-2 card. If you have problems with your OSA-2 card after booting, you might want to insert a delay to allow the POR, microcode load and initialization to take place in the OSA-2 card. The recommended delay time is two minutes. For example, `30s` means a delay of thirty seconds between the boot and the initialization of the OSA-2 card, `2m` means a delay of two minutes. The value `xy` must be a number followed by either `s` or `m`.

Format



Examples

This example delays the initialization of the card by 2 minutes:

```
ipldelay=2m
```

This example delays the initialization of the card by 30 seconds:

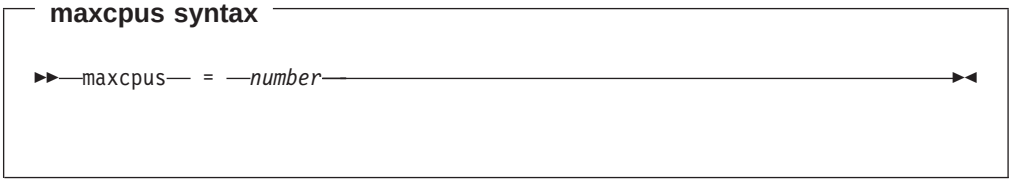
```
ipldelay=30s
```

maxcpus

Usage

Specifies the maximum number of CPUs that LINUX can use.

Format



Examples

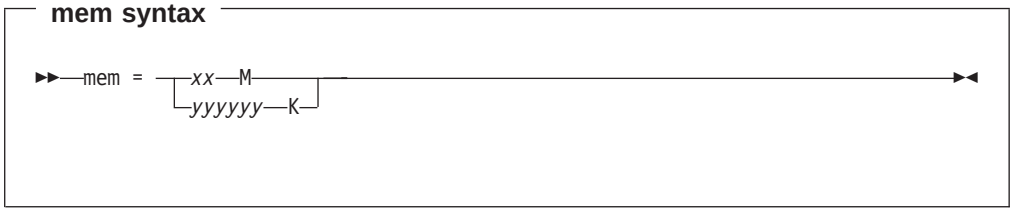
```
maxcpus=2
```

mem

Usage

Restricts memory usage to the size specified. This must be used to overcome initialization problems on a P/390.

Format



Examples

```
mem=64M
```

Restricts the memory LINUX can use to 64MB.

```
mem=123456K
```

Restricts the memory LINUX can use to 123456KB.

noinitrd

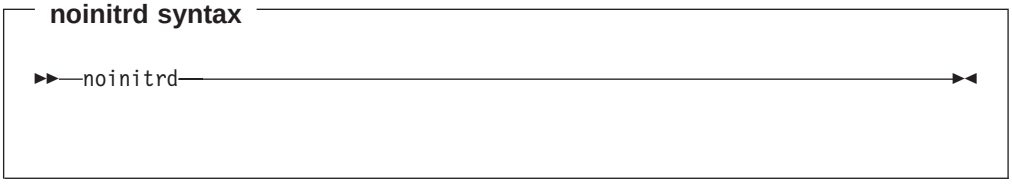
Usage

The `noinitrd` statement is required when the kernel was compiled with initial RAM disk support enabled. This command bypasses using the initial ramdisk.

This can be useful if the kernel was used with a RAM disk for the initial startup, but the RAM disk is not required when booted from a DASD

Format

noinitrd syntax

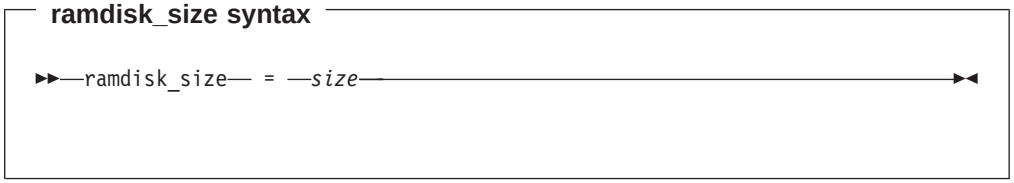


ramdisk_size

Usage

Specifies the size of the ramdisk in kilobytes.

Format



Examples

```
ramdisk_size=32000
```

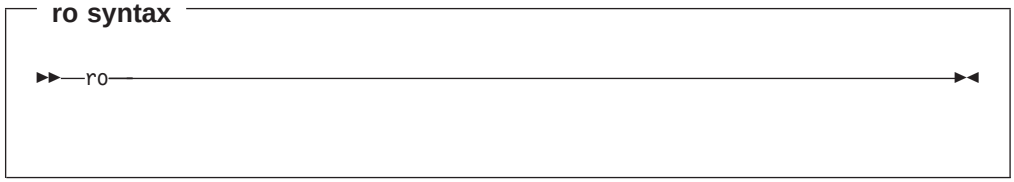
ro

Usage

Mounts the root file system read-only.

Format

ro syntax

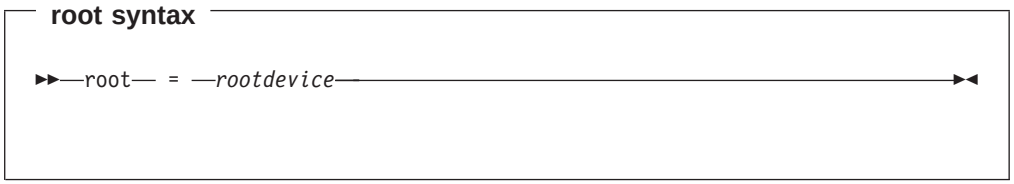


root

Usage

Tells LINUX what to use as the root when mounting the root file system.

Format



Examples

This makes LINUX use `/dev/dasda1` when mounting the root file system:

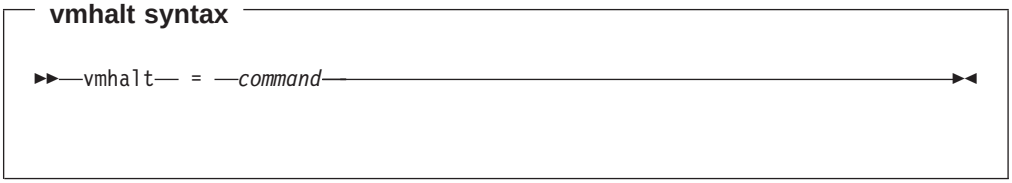
```
root=/dev/dasda1
```

vmhalt

Usage

Specifies a command to be issued after a shutdown on VM.

Format



Examples

This example specifies that an initial program load of CMS should follow a shutdown on VM:

```
vmhalt="i cms"
```

cio_msg

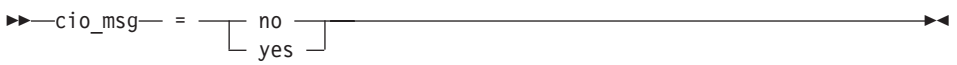
Usage

Specifies whether I/O messages are to be sent to the console on boot-up.

These messages are usually suppressed (`cio_msg=no`) because on large machines with many attached devices the I/O layer generates a large number of these messages which can flood the console for a significant period of time. If you do need those messages (for example for debugging) you can switch them on manually using `cio_msg=yes`.

Format

cio_msg syntax



Examples

This example switches I/O messages to the console on boot:

```
cio_msg=yes
```

Chapter 13. Overview of the parameter line file

The parameter line file contains kernel parameters which are read by LINUX during the boot process. This chapter describes the format of the parameters in this file.

Parameters

Comments

The parameters allowed in the parameter line are described in “Chapter 12. Kernel parameters” on page 85 and/or together with the description of the device drivers.

Usage

The parameter line file contains data to be passed to the kernel for evaluation at startup time. The location from which the kernel reads this file varies with the IPL method as shown below:

IPL method	Location of parameter line file
DASD	Installed into the boot sector using `silo` (option `–p`)
Tape	Second file on tape
VM reader	Second file in reader
CD-ROM	Third entry in the `.ins` file (with load address 0x00010480)

Format

The kernel parameter file consists of a single line containing at most 896 bytes. The line may either be encoded in ASCII or in EBCDIC. It contains a list of kernel options (see kernel parameters, device driver parameters) separated by blanks.

For IPL from a VM reader the kernel parameter file must be broken into fixed length records of 80 bytes. Note that a record end does not separate two options. Therefore if an option ends at the end of a record the next record should begin with a blank character.

Examples

Here is an example of a parameter line file:

```
dasd=E0C0-E0C2 root=/dev/ram0 ro ipldelay=2m
```

This defines three DASD, a read-only root file system, and a two-minute delay to allow network connection.

Note that when loading from tape using an ASCII encoded parameter file (such as one generated on a UNIX or PC system) you must make sure that your parameter file does not span more than one line, is not larger than 896 bytes, and contains no special characters (for example tabs or new lines).

Appendix A. Reference information

LCS module parameter syntax 97
OSA-Express driver command syntax 97
LINUX for S/390 Device numbers 98

LCS module parameter syntax

This driver is subject to licence conditions as reflected in: "International License Agreement for Non-Warranted Programs" on page 129.

The following are the LCS device driver module parameters:

use_hw_stats	Get network statistics from the `LANSTAT LCS` primitive.
do_sw_ip_checksumming	Perform checksum on inbound packets.
ignore_sense	Boot devices which do not report a valid `sense_id`
additional_model_info	Model/maximum relative adapter number pairs.
devno_portno_pairs	Matching pairs of device numbers and port numbers.
noauto	`noauto=1` disables auto-detection.

For a description of the parameters see "Module parameter syntax" on page 49.

OSA-Express driver command syntax

This driver is subject to licence conditions as reflected in: "International License Agreement for Non-Warranted Programs" on page 129.

There is a single keyword parameter for the OSA-Express driver:

```
qeth_options
```

This parameter is used as follows:

(Note that all characters must be in the case shown, except in hexadecimal numbers where case is irrelevant.)

```
qeth_options=[<driver options>,][<card options>,[<card options>,...]]
```

<driver options> is a comma separated list that sets the driver defaults. It can contain the following keywords:

auto	turns on autosensing
noauto	turns off autosensing
no_router	does not prepare the device as router (default)
primary_router	makes the device a primary router
secondary_router	makes the device a secondary router
sw_checksumming	checksumming is to be performed by the software
hw_checksumming	checksumming is to be performed by the hardware
no_checksumming	checksumming is not to be used

prio_queueing_tos	priority queueing based on the IP type of service field
prio_queueing_prec	priority queueing based on the IP precedence field
no_prio_queueing	switch off priority queueing
no_prio_queueing: <*number***>**	switch off priority queueing and set the default queue to <*number*>

<card options> are used to override the global options for a particular device. These are also comma-separated lists and each list consists of three device numbers (decimal, or hex starting with 0x), an optional device name, and any of the driver options keywords except **auto** or **noauto**.

For a description of the parameters see “Module parameter syntax” on page 53.

LINUX for S/390 Device numbers

Device numbers currently allocated to S/390 devices are:

Device	Major number	Minor numbers
DASD	94 + dynamic	0,4,8..252. – Volume, 1,5,9,...253 – Partitions
VM Minidisk	95	0–255
XPRAM	35	0–31

Appendix B. Kernel building

Building the kernel 100
Preliminary steps 100
Configuring the parameters 101
Checking the configuration 102
Checking the dependencies. 102
Compiling the kernel 102
Installing the modules 103
Finishing off 103
Using 'config' or 'oldconfig' 103
Sample output listing 104
Cross-reference to configuration options 105
Using 'menuconfig' 106
File handling 106
Main menu 107
Code maturity level option 108
Processor type and features 108
Loadable module support 108
General setup 109
S/390 block device drivers 109
S/390 network device support 110
S/390 terminal and console options 110
Networking options 111
QoS and/or Fair queueing 111
Filesystems. 112
Network file systems 112
Partition types 112
Kernel hacking 113
Load an alternate configuration file 113
Save configuration to an alternate file 113
Exit 'menuconfig' 113
Kernel parameter options. 115
IEEE FPU emulation 115
Built-in IPL record support 116
IPL method generated into head.S 116
Enable /proc/deviceinfo entries 116
Support for VM minidisk 116
Support for VM minidisk synchronous read-write 117
Support for DASD devices 117
Support for ECKD disks 117
Support for FBA disks 118
Support for DIAG access to CMS reserved minidisk 118
XPRAM device support 118
CTC/ESCON device support 119
IUCV device support 119
OSA Express device support 119
Support for 3215 line mode terminal 120
Support for console output on 3215 line mode terminal. 120
Support for 3270 console 120
Support for hardware console 121
Console output on hardware console 121

Tape device support 121

Building the kernel

Before deciding to change the details in the kernel source code, consider whether installing one of the LINUX images provided in the LINUX for S/390 kernel patches will be a more appropriate solution to your requirements.

Your build system must have the following software installed (as a minimum):

- kernel source 2.2.16 with the LINUX for S/390 patch
- gcc version 2.95.2 or newer supporting LINUX for S/390
- binutils version 2.9.1 or newer supporting LINUX for S/390
- glibc 2.1.2 or newer supporting LINUX for S/390
- sed
- bash
- make version 3.77 or newer.

The following assumptions are made:

- You are confident in your ability to modify the kernel parameters without severely damaging the system
- You cannot locate a pre-compiled kernel image that meets your requirements (that is, a suitable kernel does not already exist)
- You are able to make an emergency IPL tape available (or preferably a complete backup on tape).

If you decide to modify your LINUX for S/390 kernel, you should use the instructions outlined in the following sections. In this way you will be sure of completing all of the tasks necessary to ensure the system runs properly when you have finished. For example, you might have to install and link additional modules after you have compiled and installed the kernel.

1. “Preliminary steps”
2. “Configuring the parameters” on page 101
3. “Checking the configuration” on page 102
4. “Checking the dependencies” on page 102
5. “Compiling the kernel” on page 102
6. “Installing the modules” on page 103
7. “Finishing off” on page 103

Preliminary steps

Before working with the kernel, there are a number of precautions that you should take:

- Make a backup copy of the current kernel and all modules corresponding to this kernel. It is important to make a backup even if you are replacing your kernel with a new version. This is because the new system might not run properly and you can use the backup to reload the old system

- Decide whether you want to modify the complete kernel, or only change some modules. If you only change some modules, you might not have to build the kernel.

If you are upgrading or replacing the kernel, obtain the new kernel or patch and load it into the directory `/usr/src`. This will probably create a new directory `usr/src/linux`, which will overwrite your last version of the kernel source. You will need to check the symbolic links to the `/usr/include` directory to ensure the following two links are valid:

- `ln -sf /usr/src/linux/include/linux /usr/include/linux`
- `ln -sf /usr/src/linux/include/asm /usr/include/asm`

Your first step in modifying the kernel, is to change to the `/usr/src/linux` directory and enter the command `make distclean`. This cleans up the LINUX for S/390 distribution, resetting all options to their default values and removing all object files from the system. If you enter `make clean` instead of `make distclean`, you will only delete the object files.

Configuring the parameters

To configure the parameters, make sure you are in the `/usr/src/linux` directory and enter the command `make config`.

In `make config`, you select what you want to include in the resident kernel and what features you want to have available as dynamically loadable modules. See “Using ’config’ or ’oldconfig’” on page 103 for an example of a `make config` listing. You will generally select the minimal resident set that is needed to boot:

- The type of file system used for your root partition (for example, ext2)
- Normal hard disk drive support (for example, DASD)
- Network support
- TCP/IP support.

The set of modules is constantly increasing, and you will be able to select the option (M) when responding to the prompts shown in `make config` for those features that the current kernel can offer as loadable modules. You can completely enable or disable the use of your set of modules by using the `CONFIG_MODVERSIONS` option during `make config`.

`make config` requires `bash` to allow it work. Bash will be searched for in `$BASH`, `/bin/bash` and `/bin/sh` (in that order), so it must be located in one of these directories for it to work. `make config` must be performed even if you are only upgrading to the next patch. New configuration options are added in each release, and odd problems will turn up if the configuration files are not set up as expected. If you want to upgrade your existing configuration with minimal work, use `make oldconfig`, which will keep your old kernel and only ask you questions about new or modified options.

Alternative configuration commands are:

- `make menuconfig` — Text based menus, radiolists and windows, see “Using ’menuconfig’” on page 106
- `make oldconfig` — Same as `make config` except all questions based on the contents of your existing `./.config` file
- `make xconfig` — This X windows based configuration tool is currently not available with LINUX for S/390.

Notes on `make config`:

- Keeping unnecessary drivers in the LINUX for S/390 kernel will make it bigger, and can cause problems: for example, unnecessary networking options might confuse some drivers.
- Selecting the kernel hacking option and changing the source code directly usually result in a bigger or slower LINUX for S/390 kernel (or both). Thus you should probably answer (N) to the questions for development, experimental, or debugging features.

Checking the configuration

There are a pair of scripts that check the source tree for problems. These scripts do not have to be run each time you build the kernel, but it is a good idea to check for these types of errors and discrepancies at regular intervals.

`make checkconfig` checks the source tree for missing instances of `#include <linux>`. This needs to be done occasionally, because the C preprocessor will silently give bad results if these symbols haven't been included (it treats undefined symbols in preprocessor directives as defined to 0). Superfluous uses of `#include <linux>` are also reported, but you can ignore these, because smart `CONFIG_*` dependencies make them harmless. You can run `make checkconfig` without configuring the kernel. Also, `make checkconfig` does not modify any files.

`make checkhelp` checks the source tree for options that are in `Config.in` files but are not documented in `scripts/Configure.help`. Again, this needs to be done occasionally. If you have hacked the kernel and changed configuration options or are adding new ones, it is a good idea to `make checkhelp` (and add help as necessary) before you publish your patch. Also, `make checkhelp` does not modify any files.

Checking the dependencies

All of the source dependencies must be set each time you configure a new LINUX for S/390 kernel.

Enter `make dep` to set up all the dependencies correctly. `make dep` is a synonym for the long form, `make depend`. This command performs two tasks:

- It computes dependency information about which `.o` files depend on which `.h` files. It records this information in a top-level file named `.hdepend` and in one file per source directory named `.depend`.
- If you have `CONFIG_MODVERSIONS` enabled, `make dep` computes symbol version information for all of the files that export symbols (note that both resident and modular files can export symbols). If you do not enable `CONFIG_MODVERSIONS`, you only have to run `make dep` once, right after the first time you configure the kernel. The `.hdepend` files and the `.depend` file are independent of your configuration. If you do enable `CONFIG_MODVERSIONS`, you must run `make dep` because the symbol version information depends on the configuration.

Compiling the kernel

Enter `make image` to create a LINUX for S/390 kernel image. This compiles the source code and leaves the kernel image in the current directory (usually `/usr/src/linux/arch/s390/boot`).

Note that `make zImage` and `make bzImage` are not supported by the LINUX for S/390 kernel.

Compiling for IPL from tape

If you want to make a boot tape, you must transfer a set of files to the IPL tape. The files, `image.tape.bin` (renamed as `image.txt`), `parm.line`, and `initrd.bin` (renamed as `initrd.txt`) are the ones used during the installation process.

If you want to IPL from tape, ensure that the following configuration settings are used:

- CONFIG_IPL is set
- CONFIG_IPL_TAPE is set
- CONFIG_BLK_DEV_RAM is set
- CONFIG_BLK_DEV_INITRD is set.

Additionally you should keep the default configuration settings to make sure that all requirements to get a running LINUX for S/390 kernel are met.

Installing the modules

If you configured any of the parts of the LINUX for S/390 kernel as modules by selecting (M) in the kernel parameter option, you will have to create the modules and then link them to the kernel.

You create the modules by entering the command `make modules`. This will compile all of the modules and update the `linux/modules` directory. This directory will now contain a set of symbolic links, pointing to the various object files in the kernel tree.

After you have created all your modules, you must enter `make modules_install`. This will copy all of the newly made modules into subdirectories under `/lib/modules/`*`kernel_release`*`/`, where *`kernel_release`* is 2.2.16 (the current kernel version).

As soon as you have rebooted the newly made kernel, you can use the utilities `insmod` and `rmmod` to install and remove modules without recompiling the kernel. Read the man-pages for insmod and rmmod to find out how to configure and remove a module.

Finishing off

You should always keep a backup LINUX for S/390 kernel available in case something goes wrong. This backup must also include the modules corresponding to that kernel. If you are installing a new kernel with the same version number as your working kernel, make a backup of your modules' directory before you do a `make modules_install`.

In order to boot your new kernel, you'll need to copy the kernel image (found in `/usr/src/linux/arch/s390/boot/image` after compilation) to the place where your regular bootable kernel is located. This will be on your IPL tape.

To see how to create a tape from which you can perform an IPL, refer to the installation manual for your LINUX for S/390 distribution.

Using 'config' or 'oldconfig'

Use `make config` to configure all of the LINUX for S/390 kernel options, or use `make oldconfig` to keep your current kernel options and change only those options that are new or have been modified in the latest kernel release.

Use `make config` or `make oldconfig` when you only have access to a line based console. If you can use a screen based console, you might find `make menuconfig` gives you a better interface (see “Using ’menuconfig’” on page 106).

The following output listing shows an example of the complete configuration script that results from a `make config`. See “Kernel parameter options” on page 115 for more information about individual options.

Sample output listing

Note:

This is for illustration only. The prompts and responses on your system may not match these. The responses typed are shown in **bold** type. (The responses shown are the defaults unless they are **underlined**; the default results would occur if just the ENTER key was pressed each time.)

```
[root@host /usr/src/linux# make config
rm -f include/asm
( cd include ; ln -sf asm-s390 asm)
/bin/sh scripts/Configure arch/s390/config.in
#
# Using defaults found in .config
#
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [Y/n/?] Y
*
* Processor type and features
*
Symmetric multi-processing support (CONFIG_SMP) [Y/n/?] Y
IEEE FPU emulation (CONFIG_IEEEFPU_EMULATION) [Y/n/?] Y
*
* Loadable module support
*
Enable loadable module support (CONFIG_MODULES) [Y/n/?] Y
Set version information on all symbols for modules (CONFIG_MODVERSIONS) [N/y/?] N
Kernel module loader (CONFIG_KMOD) [Y/n/?] Y
*
* General setup
*
Fast IRQ handling (CONFIG_FAST_IRQ) [Y/n/?] Y
Built-in IPL record support (CONFIG_IPL) [Y/n/?] Y
IPL method generated into head.S (tape, vm_reader) [vm_reader] wm_reader
  defined CONFIG_IPL_VM
Networking support (CONFIG_NET) [Y/n/?] Y
System V IPC (CONFIG_SYSVIPC) [Y/n/?] Y
BSD Process Accounting (CONFIG_BSD_PROCESS_ACCT) [N/y/?] N
Sysctl support (CONFIG_SYSCTL) [Y/n/?] Y
Kernel support for ELF binaries (CONFIG_BINFMT_ELF) [Y/m/n/?] Y
*
* S/390 block device drivers
*
Loopback device support (CONFIG_BLK_DEV_LOOP) [Y/m/n/?] Y
Network block device support (CONFIG_BLK_DEV_NBD) [N/y/m/?] N
Multiple devices driver support (CONFIG_BLK_DEV_MD) [N/y/?] N
RAM disk support (CONFIG_BLK_DEV_RAM) [Y/m/n/?] Y
   Initial RAM disk (initrd) support (CONFIG_BLK_DEV_INITRD) [Y/n/?] Y
XPRAM disk support (CONFIG_BLK_DEV_XPRAM) [N/y/m/?] N
Support for VM minidisk (VM only) (CONFIG_MDISK) [N/y/?] Y
   Support for synchronous read-write (CONFIG_MDISK_SYNC) [N/y/?] (NEW) N
Support for DASD devices (CONFIG_DASD) [Y/m/n/?] Y
*
* DASD disciplines
*
   Support for ECKD Disks (CONFIG_DASD_ECKD) [Y/n/?] Y
   Support for FBA Disks (CONFIG_DASD_FBA) [Y/n/?] Y
Support for S/390 tape devices (CONFIG_S390_TAPE) [N/y/m/?] N
Support for DIAG access to CMS reserved minidisk (CONFIG_DASD_MDSK) [N/y/?] N
*
* S/390 Network device support
*
Channel Device Configuration (Temporary Option) (CONFIG_CHANDEV) [N/y/?] N
Network device support (CONFIG_NETDEVICES) [Y/n/?] Y
*
* S/390 Network devices
*
LAN Channel Station Interface (CONFIG_LCS) [Y/m/n/?] Y
CTC device support (CONFIG_CTC) [Y/n/?] Y
IUCV device support (VM only) (CONFIG_IUCV) [Y/n/?] Y
Dummy net driver support (CONFIG_DUMMY) [N/y/m/?] N
Ethernet (10 or 100Mbit) (CONFIG_NET_ETHERNET) [Y/n/?] Y
Token Ring driver support (CONFIG_TR) [Y/n/?] Y
*
* S/390 Terminal and Console options
*
```

```
Support for 3215 line mode terminal (CONFIG_3215) [Y/n/?] Y
Support for console on 3215 line mode terminal (CONFIG_3215_CONSOLE) [Y/n/?] Y
Support for HWC line mode terminal (CONFIG_HWC) [Y/n/?] Y
console on HWC line mode terminal (CONFIG_HWC_CONSOLE) [Y/n/?] Y
*
* Networking options
*
Packet socket (CONFIG_PACKET) [Y/m/n/?] Y
Kernel/User netlink socket (CONFIG_NETLINK) [Y/n/?] Y
Routing messages (CONFIG_RTNETLINK) [N/y/?] N
Netlink device emulation (CONFIG_NETLINK_DEV) [N/y/m/?] N
Network firewalls (CONFIG_FIREWALL) [N/y/?] N
Socket Filtering (CONFIG_FILTER) [N/y/?] N
Unix domain sockets (CONFIG_UNIX) [Y/m/n/?] Y
TCP/IP networking (CONFIG_INET) [Y/n/?] Y
IP: multicasting (CONFIG_IP_MULTICAST) [N/y/?] N
IP: advanced router (CONFIG_IP_ADVANCED_ROUTER) [N/y/?] N
IP: kernel level autoconfiguration (CONFIG_IP_PNP) [N/y/?] N
IP: optimize as router not host (CONFIG_IP_ROUTER) [N/y/?] N
IP: tunneling (CONFIG_NET_IPIP) [N/y/m/?] N
IP: GRE tunnels over IP (CONFIG_NET_IPGRE) [N/y/m/?] N
IP: aliasing support (CONFIG_IP_ALIAS) [N/y/?] N
IP: TCP syncookie support (not enabled per default) (CONFIG_SYN_COOKIES) [N/y/?] N
*
* (it is safe to leave these untouched)
*
IP: Reverse ARP (CONFIG_INET_RARP) [N/y/m/?] N
IP: Allow large windows (not recommended if <16Mb of memory) (CONFIG_SKB_LARGE) [Y/n/?] Y
The IPv6 protocol (EXPERIMENTAL) (CONFIG_IPV6) [N/y/m/?] N
*
*
*
The IPX protocol (CONFIG_IPX) [N/y/m/?] N
Appletalk DDP (CONFIG_ATALK) [N/y/m/?] N
CCITT X.25 Packet Layer (EXPERIMENTAL) (CONFIG_X25) [N/y/m/?] N
LAPB Data Link Driver (EXPERIMENTAL) (CONFIG_LAPB) [N/y/m/?] N
Bridging (EXPERIMENTAL) (CONFIG_BRIDGE) [N/y/?] N
802.2 LLC (EXPERIMENTAL) (CONFIG_LLC) [N/y/?] N
Acorn Econet/AUN protocols (EXPERIMENTAL) (CONFIG_ECONET) [N/y/m/?] N
WAN router (CONFIG_WAN_ROUTER) [N/y/m/?] N
Fast switching (read help!) (CONFIG_NET_FASTROUTE) [N/y/?] N
Forwarding between high speed interfaces (CONFIG_NET_HW_FLOWCONTROL) [N/y/?] N
CPU is too slow to handle full bandwidth (CONFIG_CPU_IS_SLOW) [N/y/?] N
*
* QoS and/or fair queueing
*
QoS and/or fair queueing (CONFIG_NET_SCHED) [N/y/?] N
*
* Filesystems
*
Quota support (CONFIG_QUOTA) [N/y/?] N
Kernel automounter support (CONFIG_AUTOFS_FS) [N/y/m/?] N
ADFS filesystem support (read only) (EXPERIMENTAL) (CONFIG_ADFS_FS) [N/y/m/?] N
Amiga FFS filesystem support (CONFIG_AFFS_FS) [N/y/m/?] N
Apple Macintosh filesystem support (experimental) (CONFIG_HFS_FS) [N/y/m/?] N
DOS FAT fs support (CONFIG_FAT_FS) [N/y/m/?] N
ISO 9660 CDROM filesystem support (CONFIG_ISO9660_FS) [N/y/m/?] N
Minix fs support (CONFIG_MINIX_FS) [N/y/m/?] N
NTFS filesystem support (read only) (CONFIG_NTFS_FS) [N/y/m/?] N
OS/2 HPFS filesystem support (read only) (CONFIG_HPFS_FS) [N/y/m/?] N
/proc filesystem support (CONFIG_PROC_FS) [Y/n/?] Y
QNX filesystem support (EXPERIMENTAL) (CONFIG_QNX4FS_FS) [N/y/m/?] N
ROM filesystem support (CONFIG_ROMFS_FS) [N/y/m/?] N
Second extended fs support (CONFIG_EXT2_FS) [Y/m/n/?] Y
System V and Coherent filesystem support (CONFIG_SYSV_FS) [N/y/m/?] N
UFS filesystem support (CONFIG_UFS_FS) [N/y/m/?] N
SGI EFS filesystem support (read only) (experimental) (CONFIG_EFS_FS) [N/y/m/?] N
*
* Network File Systems
*
Coda filesystem support (advanced network fs) (CONFIG_CODA_FS) [N/y/m/?] N
NFS filesystem support (CONFIG_NFS_FS) [Y/m/n/?] Y
NFS server support (CONFIG_NFSD) [N/y/m/?] N
SMB filesystem support (to mount WfW shares etc.) (CONFIG_SMB_FS) [N/y/m/?] N
NCP filesystem support (to mount NetWare volumes) (CONFIG_NCP_FS) [N/y/m/?] N
*
* Partition Types
*
BSD disklabel (BSD partition tables) support (CONFIG_BSD_DISKLABEL) [N/y/?] N
Macintosh partition map support (CONFIG_MAC_PARTITION) [N/y/?] N
SMD disklabel (Sun partition tables) support (CONFIG_SMD_DISKLABEL) [N/y/?] N
Solaris (x86) partition table support (CONFIG_SOLARIS_X86_PARTITION) [N/y/?] N
Unixware slices support (EXPERIMENTAL) (CONFIG_UNIXWARE_DISKLABEL) [N/y/?] N
*
* Kernel hacking
*
Kernel profiling support (CONFIG_PROFILE) [N/y/?] N
Remote GDB kernel debugging (CONFIG_REMOTE_DEBUG) [N/y/?] N

*** End of Linux kernel configuration.
*** Check the top-level Makefile for additional configuration.
*** Next, you may run 'make dep'.
```

Cross-reference to configuration options

The LINUX for S/390 specific configuration options shown in the sample output listing are described in the following sections:

- "IEEE FPU emulation" on page 115
- "Built-in IPL record support" on page 116
- "IPL method generated into head.S" on page 116
- "Support for VM minidisk" on page 116
- "Support for VM minidisk synchronous read-write" on page 117
- "Support for DASD devices" on page 117
- "Support for ECKD disks" on page 117
- "Support for FBA disks" on page 118
- "Support for DIAG access to CMS reserved minidisk" on page 118
- "XPRAM device support" on page 118
- "CTC/ESCON device support" on page 119
- "IUCV device support" on page 119
- "OSA Express device support" on page 119
- "Support for 3215 line mode terminal" on page 120
- "Support for console output on 3215 line mode terminal" on page 120
- "Support for 3270 console" on page 120
- "Support for hardware console" on page 121
- "Console output on hardware console" on page 121
- "Tape device support" on page 121.

Using 'menuconfig'

You can use `make menuconfig` to edit individual LINUX for S/390 kernel options out of sequence and you can discard your settings at any time. You need a screen based console device to use `make menuconfig`. If you only have access to a line based console, you must use `make config`, see "Using 'config' or 'oldconfig'" on page 103 for more details.

Some options can be built directly into the kernel. Other options can be made into dynamically loadable modules. Some options can be completely removed altogether. There are also certain kernel parameters which are not really selectable options (Y) or (N), but instead values must be entered for them or selected from a list (decimal or hexadecimal numbers or possibly text).

The menu items begin with various types of bracket to indicate that the features:

- `[*]` are configured to be built in to the kernel.
- `[ ]` are configured to be removed from the kernel.
- `<M>` are configured to be modularized.
- `< >` are module capable features.
- `<*>` could have been modules, but have been built into the kernel.

Modules are linked to the kernel after booting.

Some menu items contain subordinate options that are displayed only when the menu item has been selected.

File handling

You can revise your settings up until you save the configuration. You will be given a last chance to confirm them prior to exiting `menuconfig` – see "Exit 'menuconfig'" on page 113.

If `menuconfig` quits with an error while saving your configuration, you can look in the file `/usr/src/linux/.menuconfig.log` for information which can help you determine the failure cause.

You can also save the current configuration to a file of your choice, or load a previously saved configuration – see “Save configuration to an alternate file” on page 113 and “Load an alternate configuration file” on page 113.

`menuconfig` supports the use of alternative configuration files for those who, for various reasons, find it necessary to switch between different kernel configurations. At the end of the main menu you will find two options. One is for saving the current configuration to a file of your choosing. The other option is for loading a previously saved alternative configuration. Even if you don't use alternative configuration files, but during a session you decide to restore your previously saved settings from `.config`, you can use the `Load Alternative...` to do so without restarting `menuconfig`.

Note:

These examples are for illustration only. The prompts and responses on your system may not match these. The headers and footers have been removed from all screens except the first for clarity.

Main menu

The following example shows the different sets of configuration options:

```
Linux Kernel v2.2.16 Configuration
──────────────────────────────────────────────────────────────────────────────
                                  Main Menu
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ┌──────────────────────────────────────────────────────────────────────┐
  │         Code maturity level options  --->                            │
  │         Processor type and features  --->                            │
  │         Loadable module support  --->                                │
  │         General setup  --->                                          │
  │         S/390 block device drivers  --->                             │
  │         S/390 Network device support  --->                           │
  │         S/390 Terminal and Console options  --->                     │
  │         --- Character devices                                        │
  │         [*] Unix98 PTY support                                       │
  │         (256) Maximum number of Unix98 PTYs in use (0-2048)          │
  │         Networking options  --->                                     │
  │         Filesystems  --->                                            │
  │         Kernel hacking  --->                                         │
  │         ---                                                          │
  │         Load an Alternate Configuration File                         │
  │         Save Configuration to an Alternate File                      │
  └──────────────────────────────────────────────────────────────────────┘
                     <Select>    < Exit >    < Help >
```

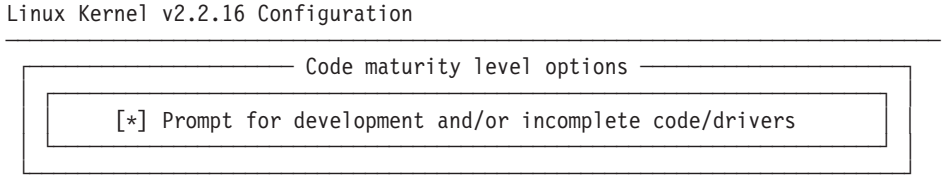
The options on this menu are described in detail in the following sections:

- “Code maturity level option” on page 108
- “Processor type and features” on page 108
- “Loadable module support” on page 108
- “General setup” on page 109

- "S/390 block device drivers" on page 109
- "S/390 network device support" on page 110
- "S/390 terminal and console options" on page 110
- "Networking options" on page 111
- "QoS and/or Fair queueing" on page 111
- "Filesystems" on page 112
- "Network file systems" on page 112
- "Partition types" on page 112
- "Kernel hacking" on page 113
- "Save configuration to an alternate file" on page 113
- "Load an alternate configuration file" on page 113
- "Exit 'menuconfig'" on page 113.

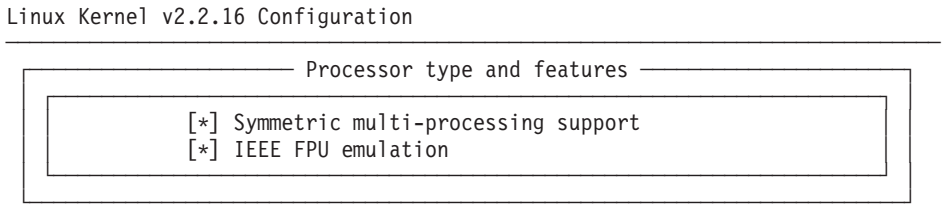
Code maturity level option

The Code Maturity Level option is intended to reduce the number of options that are displayed. This can help by not showing the code/drivers that are either incomplete or still under development:



Processor type and features

The Processor Type and Features options allow you to configure the kernel to match your S/390 hardware installation.

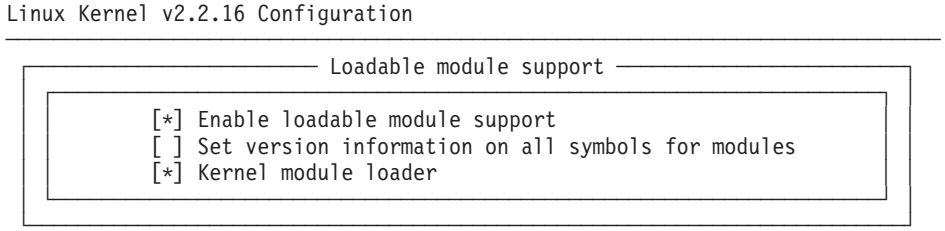


The LINUX for S/390 Processor Type and Features menu option is described in the following section:

- "IEEE FPU emulation" on page 115.

Loadable module support

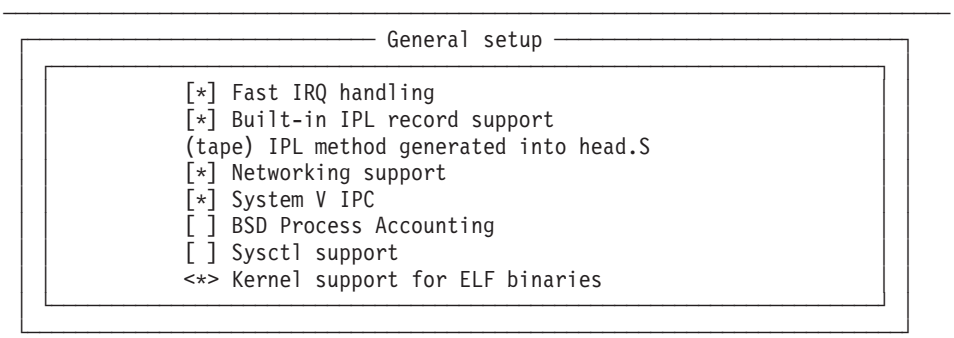
The Loadable Module Support option lets you use dynamically loaded modules that are linked to your basic kernel.



General setup

The General Setup options configure your kernel's interaction with certain external programs, this includes:

- kernel networking support
- the synchronization and exchange of information between kernel and program
- instructing the kernel to write process accounting information to a file
- dynamically changing certain kernel parameters and variables on the fly.

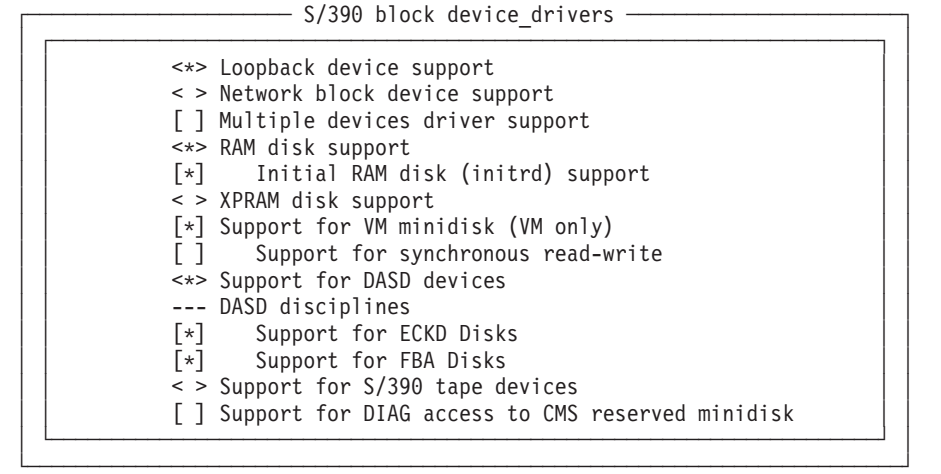


The LINUX for S/390 General Setup menu options are described in the following sections:

- "Built-in IPL record support" on page 116
- "IPL method generated into head.S" on page 116.

S/390 block device drivers

The S/390 Block Device Drivers options allow the kernel to be set up to use the various block devices available with your S/390 system.



Note that the `Support for DIAG access to CMS reserved minidisk` option is available only when the `Support for VM minidisk` option is not selected.

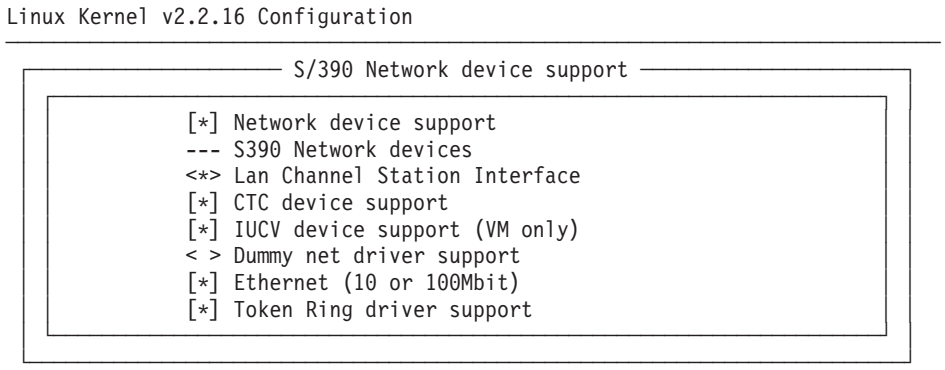
The S/390 Block Device Drivers menu options unique to LINUX for S/390 are described in the following sections:

- "Support for VM minidisk" on page 116
- "Support for VM minidisk synchronous read-write" on page 117

- "Support for DASD devices" on page 117
- "Support for ECKD disks" on page 117
- "Support for FBA disks" on page 118
- "Support for DIAG access to CMS reserved minidisk" on page 118
- "XPRAM device support" on page 118.

S/390 network device support

The S/390 Network Device Support options allow the kernel to be set up to use the various network devices available with your S/390 system.

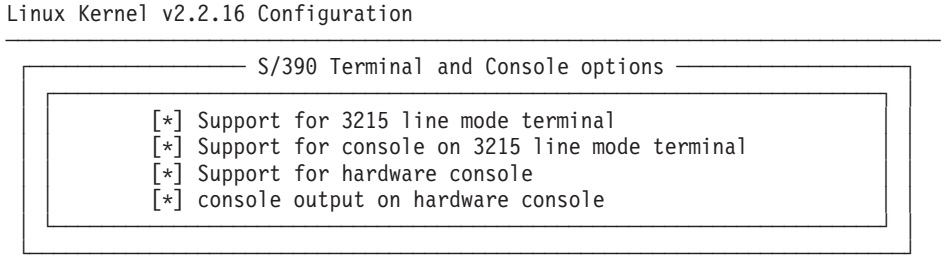


The S/390 Network device support menu options unique to LINUX for S/390 are described in the following sections:

- "CTC/ESCON device support" on page 119
- "OSA Express device support" on page 119
- "IUCV device support" on page 119.

S/390 terminal and console options

The S/390 Terminal and Console options allow the kernel to be set up to use the various line mode terminals (or emulators) available with your S/390 system.



The S/390 Terminal and Console options menu options unique to LINUX for S/390 are described in the following sections:

- “Support for 3215 line mode terminal” on page 120
- “Support for console output on 3215 line mode terminal” on page 120
- “Support for hardware console” on page 121
- “Console output on hardware console” on page 121.

Networking options

The Networking options allow the kernel to interact with the network devices attached to your S/390 and also lets you optimize the performance of communications within and outside your system.

```
Linux Kernel v2.2.16 Configuration
──────────────────────────────────────────────────────────────────────────────
                          Networking options
┌──────────────────────────────────────────────────────────────────────────┐
│ <*> Packet socket                                                        │
│ [*] Kernel/User netlink socket                                           │
│ [ ] Routing messages                                                     │
│ < > Netlink device emulation                                             │
│ [ ] Network firewalls                                                    │
│ [ ] Socket Filtering                                                     │
│ <*> Unix domain sockets                                                  │
│ [*] TCP/IP networking                                                    │
│ [ ] IP: multicasting                                                     │
│ [ ] IP: advanced router                                                  │
│ [ ] IP: kernel level autoconfiguration                                   │
│ [ ] IP: optimize as router not host                                      │
│ < > IP: tunneling                                                        │
│ < > IP: GRE tunnels over IP                                              │
│ [ ] IP: aliasing support                                                 │
│ [ ] IP: TCP syncookie support (not enabled per default)                  │
│ --- (it is safe to leave these untouched)                                │
│ < > IP: Reverse ARP                                                      │
│ [*] IP: Allow large windows (not recommended if <16Mb of memory)         │
│ < > The IPv6 protocol (EXPERIMENTAL)                                     │
│ ---                                                                      │
│ < > The IPX protocol                                                     │
│ < > Appletalk DDP                                                        │
│ < > CCITT X.25 Packet Layer (EXPERIMENTAL)                               │
│ < > LAPB Data Link Driver (EXPERIMENTAL)                                 │
│ [ ] Bridging (EXPERIMENTAL)                                              │
│ [ ] 802.2 LLC (EXPERIMENTAL)                                             │
│ < > Acorn Econet/AUN protocols (EXPERIMENTAL)                            │
│ < > WAN router                                                           │
│ [ ] Fast switching (read help!)                                          │
│ [ ] Forwarding between high speed interfaces                             │
│ [ ] CPU is too slow to handle full bandwidth                             │
│ QoS and/or fair queueing  --->                                           │
└──────────────────────────────────────────────────────────────────────────┘
```

QoS and/or Fair queueing

The QoS and/or Fair Queueing option allows fine tuning of the network device packet handling algorithms.

```
Linux Kernel v2.2.16 Configuration
──────────────────────────────────────────────────────────────────────────────
                        QoS and/or fair queueing
┌──────────────────────────────────────────────────────────────────────────┐
│                  [ ] QoS and/or fair queueing                            │
└──────────────────────────────────────────────────────────────────────────┘
```

Filesystems

The Filesystems options allow you to define the filesystems used to access your storage devices (hard disk, CDROM etc.).

```
Linux Kernel v2.2.16 Configuration
──────────────────────────────────────────────────────────────────────────────
                              Filesystems
┌──────────────────────────────────────────────────────────────────────────┐
│ [ ] Quota support                                                        │
│ < > Kernel automounter support                                           │
│ < > ADFS filesystem support (read only) (EXPERIMENTAL)                   │
│ < > Amiga FFS filesystem support                                         │
│ < > Apple Macintosh filesystem support (experimental)                    │
│ < > DOS FAT fs support                                                   │
│ < > ISO 9660 CDROM filesystem support                                    │
│ < > Minix fs support                                                     │
│ < > NTFS filesystem support (read only)                                  │
│ < > OS/2 HPFS filesystem support (read only)                             │
│ [*] /proc filesystem support                                             │
│ < > QNX filesystem support (EXPERIMENTAL)                                │
│ < > ROM filesystem support                                               │
│ <*> Second extended fs support                                           │
│ < > System V and Coherent filesystem support                             │
│ < > UFS filesystem support                                               │
│ < > SGI EFS filesystem support (read only) (experimental)                │
│ Network File Systems  --->                                               │
│ Partition Types  --->                                                    │
└──────────────────────────────────────────────────────────────────────────┘
```

Network file systems

The Network File Systems options let you decide which network filesystem is most appropriate for your system.

```
Linux Kernel v2.2.16 Configuration
──────────────────────────────────────────────────────────────────────────────
                           Network File Systems
┌──────────────────────────────────────────────────────────────────────────┐
│ < > Coda filesystem support (advanced network fs)                        │
│ <*> NFS filesystem support                                               │
│ < > NFS server support                                                   │
│ < > SMB filesystem support (to mount WfW shares etc.)                    │
│ < > NCP filesystem support (to mount NetWare volumes)                    │
└──────────────────────────────────────────────────────────────────────────┘
```

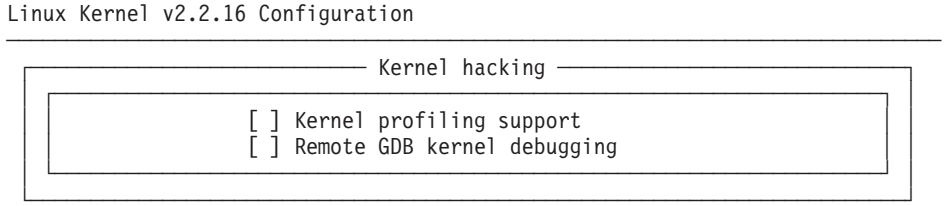
Partition types

The Partition Types options lets you gain access to the hard disk partitions of other device types.

```
Linux Kernel v2.2.16 Configuration
──────────────────────────────────────────────────────────────────────────────
                            Partition Types
┌──────────────────────────────────────────────────────────────────────────┐
│      [ ] BSD disklabel (BSD partition tables) support                    │
│      [ ] Macintosh partition map support                                 │
│      [ ] SMD disklabel (Sun partition tables) support                    │
│      [ ] Solaris (x86) partition table support                           │
│      [ ] Unixware slices support (EXPERIMENTAL)                          │
└──────────────────────────────────────────────────────────────────────────┘
```

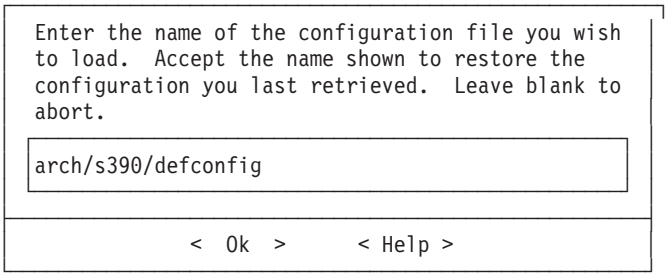
Kernel hacking

The Kernel Hacking options allow you access and control over the kernel in certain situations. These options should be used with care!



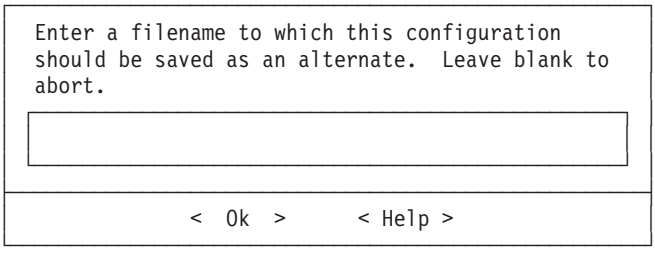
Load an alternate configuration file

For various reasons, you might want to keep different kernel configurations available on a single S/390 system. Entering a file name here will allow you to later retrieve, modify and use the current configuration as an alternate to whatever configuration options you have selected at that time.



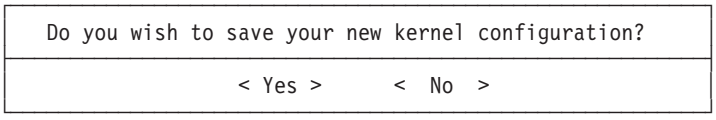
Save configuration to an alternate file

For various reasons, you might want to keep several different kernel configurations available on a single S/390 system. If you have saved a previous configuration in a file other than the kernel's default, entering the name of the file here will allow you to modify that configuration.



Exit 'menuconfig'

When you have completed your modifications to the kernel, you are asked whether you want to save the new kernel configuration. Normally, you would respond by selecting the `Yes` option, but you might have made modifications that you do not want to keep. In that case you can exit the configuration process without saving the changes by selecting the `No` option.



If you have elected to save your configuration this will be confirmed with the messages:

```
Saving your kernel configuration...

*** End of Linux kernel configuration.
*** Check the top-level Makefile for additional configuration.
*** Next, you must run 'make dep'.
```

Kernel parameter options

The LINUX for S/390 specific kernel parameter options are described in the following sections:

- "IEEE FPU emulation"
- "Built-in IPL record support" on page 116
- "IPL method generated into head.S" on page 116
- "Enable /proc/deviceinfo entries" on page 116
- "Support for VM minidisk" on page 116
- "Support for VM minidisk synchronous read-write" on page 117
- "Support for DASD devices" on page 117
- "Support for ECKD disks" on page 117
- "Support for FBA disks" on page 118
- "Support for DIAG access to CMS reserved minidisk" on page 118
- "XPRAM device support" on page 118
- "CTC/ESCON device support" on page 119
- "IUCV device support" on page 119
- "OSA Express device support" on page 119
- "Support for 3215 line mode terminal" on page 120
- "Support for console output on 3215 line mode terminal" on page 120
- "Support for 3270 console" on page 120
- "Support for hardware console" on page 121
- "Console output on hardware console" on page 121
- "Tape device support" on page 121.

IEEE FPU emulation

Configuration option

CONFIG_IEEEFPU_EMULATION

Capable of being a module? -- (Module Name)

No

Value Required by LINUX for S/390

Dependent on S/390 version, see Description

Description

From S/390 versions G5 and G6 (or newer), the S/390 systems are capable of calculating floating point numbers in IEEE-format.

LINUX for S/390 provides an IEEE floating point emulation. This can be configured on (enter Y) or off (enter N) during kernel compilation time. If you have IEEE-emulation configured on, floating point arithmetic can be performed on any S/390 system, however it will be slow on those systems using the emulation.

On S/390 versions G3, G4 (or older ones) you must run with IEEE-emulation configured on (Y).

On S/390 versions G5, G6 (or newer) you can make use of the hardware IEEE-arithmetic. VM/ESA has to be enabled to allow it to use and provide the hardware IEEE-arithmetic. This occurs automatically when you are running VM 2.4 or VM 2.3 with a PTF applied that enables the

IEEE-arithmetic. If you do not have VM 2.4 or did not apply the PTF to VM 2.3, then you still can (and have to) rely on the IEEE-emulation as on the older S/390 systems.

Built-in IPL record support

Configuration option
CONFIG_IPL

Capable of being a module? -- (Module Name)
No

Value Required by LINUX for S/390
Yes

Description
With this option turned on an IPL loader is generated at the start of the kernel image. That makes it possible to 'boot' from the kernel image directly without the need of a separate loader. This makes sense for a medium that is sequentially read from the start at IPL time like a (VM) reader or a tape. The type of the loader generated to the head of the kernel image is chosen by the 'IPL method generated into head.S' selection.

IPL method generated into head.S

Configuration option
CONFIG_IPL_VM

Capable of being a module? -- (Module Name)
No

Value Required by LINUX for S/390
See Description

Description
There are two loaders available for the generation into the kernel. 'tape' selects the loader for an IPL from a tape device, 'vm_reader' selects the loader for an IPL from a VM virtual reader.

Enable /proc/deviceinfo entries

Configuration option
CIO_PROC_DEVINFO

Capable of being a module? -- (Module Name)
No

Value Required by LINUX for S/390
No

Description
With many devices attached the `proc` filesystem runs out of inodes. Creating of the `/proc/deviceinfo/` entries is now disabled by default. If it is required it can be switched on again by setting this parameter to 'yes'.

Support for VM minidisk

Configuration option
CONFIG_MDISK

Capable of being a module? -- (Module Name)
No

Value Required by LINUX for S/390
No

Description
This option is used under VM only. With this flag enabled (Y) your S/390 can use a reserved minidisk under VM. VM internally uses the DIAG 250 to access the minidisk. When this boot parameter is enabled, several parameters (the virtual device ID, the size, offset and blocksize) must be passed to the kernel parameter file for each device. The minidisk must be formatted and reserved under VM/CMS.

When you are running a native installation, you would use CONFIG_DASD to configure your DASD.

Support for VM minidisk synchronous read-write

Configuration option
CONFIG_MDISK_SYNC

Capable of being a module? -- (Module Name)
No

Value Required by LINUX for S/390
No, VM only

Description
This option is used under VM only. You can make the minidisk operation synchronous. Normally a DIAG 250 is issued to start an I/O operation and the finish of the operation is reported with an external interrupt. With this flag enabled (Y), the DIAG 250 is issued synchronously.

Support for DASD devices

Configuration option
CONFIG_DASD

Capable of being a module? -- (Module Name)
`dasd_mod.o`

Value Required by LINUX for S/390
See Description

Description
This is used mainly in native installations.

Enable this option (Y) to support access to S/390 disks. These are known as DASD (Direct Access Storage Devices). You must enable this option to have disk access on a native or LPAR system. Running under VM/ESA you can choose `CONFIG_MDISK` instead.

When enabled (Y), this option lets you specify additional options for DASD access, see “Support for ECKD disks”.

Support for ECKD disks

Configuration option
CONFIG_DASD_ECKD

Capable of being a module? -- (Module Name)
dasd_eckd_mod.o

Value Required by LINUX for S/390
See Description

Description
This is used mainly in native installations.

Enable this option (Y) if you have ECKD-type DASDs such as an IBM 3380 or 3390. ECKD devices are the most commonly used devices in S/390 , so you should enable this option unless you are very sure you do not have any ECKD devices.

This option is subordinate to `CONFIG_DASD`, see “Support for DASD devices” on page 117.

Support for FBA disks

Configuration option
CONFIG_DASD_FBA

Capable of being a module? -- (Module Name)
dasd_fba_mod.o

Value Required by LINUX for S/390
See Description

Description
This is used mainly under VM/ESA for the virtual disk in storage VFB-512. Enable this option (Y) if you want to access your FBA devices.

This option is subordinate to `CONFIG_DASD`, see “Support for DASD devices” on page 117.

Support for DIAG access to CMS reserved minidisk

Configuration option
CONFIG_DASD_MDSK

Capable of being a module? -- (Module Name)
No

Value Required by LINUX for S/390
See Description

Description
This is applicable under VM/ESA only. Enable this option (Y) if you want to access your disks by means of VM/ESA’s `DIAG250` opcode instead of channel processing. You might want to do this if channel access to your device is currently unsupported under LINUX for S/390 or you require the additional capabilities of VM/ESA such as cross-guest DASD caching or enhanced error recovery.

This option is a sub-option of CONFIG_DASD, see “Support for DASD devices” on page 117.

XPRAM device support

Configuration option
CONFIG_BLK_DEV_XPRAM

Capable of being a module? -- (Module Name)
`xpram.o`

Value Required by LINUX for S/390
See Description

Description
This is used to allow more than 2 GB of main storage to be accessed by LINUX for S/390. Enable this option (Y) to support access to expanded storage of up to 16 TB (although current hardware currently supports only 64 GB). The expanded storage can be subdivided into partitions.

See “XPRAM kernel parameter syntax” on page 20 for more information.

CTC/ESCON device support

Configuration option
CONFIG_CTC

Capable of being a module? -- (Module Name)
ctc.o

Value Required by LINUX for S/390
No

Description
If you want to use channel connections under LINUX , enter (Y) here. This gives you the opportunity to make TCP/IP connections via virtual, parallel or ESCON channels between LINUX for S/390 and other S/390 operating systems (LINUX for S/390, OS/390, VM/ESA and VSE/ESA).

Read the CTC/ESCON device driver description on page37 for more information.

IUCV device support

Configuration option
CONFIG_IUCV

Capable of being a module? -- (Module Name)
netiucv.o

Value Required by LINUX for S/390
No

Description
This is a VM/ESA only device driver. Enter (Y) to enable a fast communication link between VM guests. At boot time the user ID of the guest needs to be passed to the kernel. Using `ifconfig` a point-to-point connection can be established to the LINUX for S/390 system running on the other VM guest. Note that both kernels need to be compiled with this option and both need to be booted with the user ID of the other VM guest.

OSA Express device support

Configuration option
CONFIG_NET_ETHERNET

Capable of being a module? -- (Module Name)
qdio.o, qeth.o

Value Required by LINUX for S/390
No

Description

If you want to use OSA Express connections under LINUX , enter (Y) here.

Read OSA Express device driver on page 53 for more information.

Support for 3215 line mode terminal

Configuration option

CONFIG_TN3215

Capable of being a module? -- (Module Name)

No

Value Required by LINUX for S/390

No

Description

The 3215 console driver is used to read and write to a 3215 line mode console. Real 3215 devices are no longer available in an S/390 environment, so the 3215 driver can only be used under VM/ESA. On a native S/390 system the initialization function of the 3215 driver returns without registering the driver to the system.

Entering (Y) to this option switches on the compilation of parts 1 and 2 of the 3215 terminal driver. The option makes it possible to use “Support for console output on 3215 line mode terminal”.

Support for console output on 3215 line mode terminal

Configuration option

CONFIG_TN3215_CONSOLE

Capable of being a module? -- (Module Name)

No

Value Required by LINUX for S/390

No

Description

This option is subordinate to “Support for 3215 line mode terminal”.

This option enables console output on the first 3215 console in the system. It prints kernel errors and kernel warnings to the 3215 terminal in addition to the normal output on the TTY device.

Support for 3270 console

Configuration option

CONFIG_TN3270

Capable of being a module? -- (Module Name)

No

Value Required by LINUX for S/390

No

Description

The 3270 console driver is used to read and write to a 3270 console.

Entering (Y) to this option switches on the compilation of the 3270 terminal driver.

Support for hardware console

Configuration option
CONFIG_HWC

Capable of being a module? -- (Module Name)
No

Value Required by LINUX for S/390
See Description

Description
The hardware console is an alternative terminal, usually required for a native LINUX for S/390 installation although it is also run under VM/ESA.

You would normally enter (Y) for this option in a native installation if your hardware configuration includes a hardware console. In a VM/ESA installation, without a hardware console, you would normally enter (N).

Read the Device driver description “Chapter 5. LINUX for S/390 Console device drivers” on page 23 for more information.

Console output on hardware console

Configuration option
CONFIG_HWC_CONSOLE

Capable of being a module? -- (Module Name)
No

Value Required by LINUX for S/390
No

Description
This option is subordinate to “Support for hardware console”.

This option enables console output on the first hardware console in the system. It prints kernel errors and kernel warnings to the hardware console in addition to the normal output on the TTY device.

Tape device support

Configuration option
CONFIG_S390_TAPE

Capable of being a module? -- (Module Name)
tape390.o

Value Required by LINUX for S/390
No

Description
If you want to use the tape device driver with LINUX enter (m) here.

Glossary

This glossary includes IBM product terminology as well as selected other terms and definitions. Additional information can be obtained in:

- The American National Standard Dictionary for Information Systems , ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036.
- The ANSI/EIA Standard–440-A, Fiber Optic Terminology. Copies may be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue, N.W., Washington, DC 20006.
- The Information Technology Vocabulary developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1).
- The IBM Dictionary of Computing , New York: McGraw-Hill, 1994.
- Internet Request for Comments: 1208, Glossary of Networking Terms
- Internet Request for Comments: 1392, Internet Users' Glossary
- The Object-Oriented Interface Design: IBM Common User Access Guidelines, Carmel, Indiana: Que, 1992.

A

autosensing. Listing the addresses of devices attached to a card by issuing a query command to the card.

C

cdl. compatible disk layout. A disk structure for LINUX for S/390 which allows access from other S/390 operating systems. This replaces the older **ldl**.

chandev. channel device layer. A unified programming interface to devices attached to the S/390 via the channel subsystem.

channel subsystem. The programmable input/output processors of the S/390, which operate in parallel with the cpu.

checksum. An error detection method using a check byte appended to message data

CHPID. channel path identifier. In a channel subsystem, a value assigned to each installed channel path of the system that uniquely identifies that path to the system.

CRC. cyclic redundancy check. A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

CSMA/CD. carrier sense multiple access with collision detection

CTC. channel to channel. A method of connecting two computing devices.

CUU. control unit and unit address. A form of addressing for S/390 devices using device numbers.

D

DASD. direct access storage device. A mass storage medium on which a computer stores data.

device driver. (1) A file that contains the code needed to use an attached device. (2) A program that enables a computer to communicate with a specific peripheral device; for example, a printer, a videodisc player, or a CD-ROM drive. (3) A collection of subroutines that control the interface between I/O device adapters and the processor.

E

ECKD. extended count-key-data device. A disk storage device that has a data transfer rate faster than some processors can utilize and that is connected to the processor through use of a speed matching buffer. A specialized channel program is needed to communicate with such a device.

ESCON. enterprise systems connection. A set of IBM products and services that provide a dynamically connected environment within an enterprise.

Ethernet. A 10-Mbps baseband local area network that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and delayed retransmission. Ethernet uses CSMA/CD.

F

Fast Ethernet. Ethernet network with a bandwidth of 100-Mbps

FBA. fixed block architecture. A type of DASD on Multiprise 3000 or P/390 or emulated by VM.

FDDI. fiber distributed data interface. An American National Standards Institute (ANSI) standard for a 100-Mbps LAN using optical fiber cables.

FTP. file transfer protocol. In the Internet suite of protocols, an application layer protocol that uses TCP and Telnet services to transfer bulk-data files between machines or hosts.

G

Gigabit Ethernet. An ethernet network with a bandwidth of 1000-Mbps

G3, G4, G5 and G6. The generation names of the S/390 CMOS based product family.

H

hardware console. A service-call logical processor that is the communication feature between the main processor and the service processor.

HMC. hardware management console. A console used to monitor and control hardware such as the S/390 microprocessors.

HFS. hierarchical file system. A system of arranging files into a tree structure of directories.

I

IOCS. input / output channel subsystem. See channel subsystem.

IP. internet protocol. In the Internet suite of protocols, a connectionless protocol that routes data through a network or interconnected networks and acts as an intermediary between the higher protocol layers and the physical network.

IP address.. The unique 32-bit address that specifies the location of each device or workstation on the Internet. For example, 9.67.97.103 is an IP address.

IPL. initial program load (or boot). (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage at the beginning of a work day or after a system malfunction. (3) The process of loading system programs and preparing a system to run jobs.

IPv6. IP version 6. The next generation of the Internet Protocol.

IPX. Internetwork Packet Exchange. (1) The network protocol used to connect Novell servers, or any workstation or router that implements IPX, with other workstations. Although similar to the Internet Protocol (IP), IPX uses different packet formats and terminology.

IPX address. The 10-byte address, consisting of a 4-byte network number and a 6-byte node address, that is used to identify nodes in the IPX network. The node address is usually identical to the medium access control (MAC) address of the associated LAN adapter.

IUCV. inter-user communication vehicle. A VM facility for passing data between virtual machines and VM components.

K

kernel. The part of an operating system that performs basic functions such as allocating hardware resources.

kernel module. A dynamically loadable part of the kernel, such as a device driver or a file system.

kernel image. The kernel when loaded into memory.

L

LAN. local area network.

LCS. LAN channel station. A protocol used by OSA.

ldl. LINUX disk layout. A basic disk structure for LINUX for S/390. Now replaced by `cdl`.

LDP. Linux Documentation Project. An attempt to provide a centralized location containing the source material for all open source LINUX documentation. Includes user and reference guides, HOW TOs, and FAQs. The homepage of the Linux Documentation Project is **http://www.linuxdoc.org**

LINUX . a version of UNIX which runs on a wide range of machines from wristwatches through personal and small business machines to enterprise systems.

LINUX for S/390. the port of LINUX to the IBM S/390 architecture.

LPAR. logical partition of an S/390.

M

MAC. medium access control. In a LAN this is the sub-layer of the data link control layer that supports medium-dependent functions and uses the services of the physical layer to provide services to the logical link control (LLC) sub-layer. The MAC sub-layer includes the method of determining when a device has access to the transmission medium.

Mbps. million bits per second.

MTU. maximum transmission unit. The largest block which may be transmitted as a single unit.

Multicast. A protocol for the simultaneous distribution of data to a number of recipients, for example live video transmissions.

Multiprise. An enterprise server of the S/390 family.

N

NIC. network interface card. The physical interface between the S/390 and the network.

O

OS. operating system. (1) Software that controls the execution of programs. An operating system may provide services such as resource allocation, scheduling, input/output control, and data management. (2) A set of programs that control how the system works. (3) The software that deals with the most basic operations that a computer performs.

OSA-2. Open Systems Adapter-2. A common S/390 network interface card.

OSA Express. an S/390 network interface card used with Gigabit Ethernet and other devices.

OSPF. open shortest path first. A function used in route optimization in networks.

P

POR. power-on reset

R

router. A device or process which allows messages to pass between different networks.

S

S/390. System/390. The family of IBM enterprise servers that demonstrate outstanding reliability, availability, scalability, security, and capacity in today's network computing environments.

SA/SE. stand alone support element. See SE.

SE. support element. (1) An internal control element of a processor that assists in many of the processor operational functions. (2) A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex.

SNA. systems network architecture. The IBM architecture that defines the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. The layered structure of SNA allows the ultimate origins and destinations of information (the users) to be independent of and unaffected by the specific SNA network services and facilities that are used for information exchange.

Sysctl. system control programming manual control (frame). A means of dynamically changing certain LINUX kernel parameters during operation.

T

TCP. transmission control protocol. A communications protocol used in the Internet and in any network that follows the Internet Engineering Task Force (IETF) standards for internetwork protocol. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It uses the Internet Protocol (IP) as the underlying protocol.

TCP/IP. transmission control protocol/internet protocol. (1) The Transmission Control Protocol and the Internet Protocol, which together provide reliable end-to-end connections between applications over interconnected networks of different types. (2) The suite of transport and application protocols that run over the Internet Protocol.

Telnet. A member of the Internet suite of protocols which provides a remote terminal connection service. It

allows users of one host to log on to a remote host and interact as if they were using a terminal directly attached to that host.

Token Ring. (1) According to IEEE 802.5, network technology that controls media access by passing a token (special packet or frame) between media-attached stations. (2) A FDDI or IEEE 802.5 network with a ring topology that passes tokens from one attaching ring station (node) to another.

U

UNIX. An operating system developed by Bell Laboratories that features multiprogramming in a multiuser environment. The UNIX operating system was originally developed for use on minicomputers but has been adapted for mainframes and microcomputers.

V

volume. A data carrier that is usually mounted and demounted as a unit, for example a tape cartridge or a disk pack. If a storage unit has no demountable packs the volume is the portion available to a single read/write mechanism.

Numbers

3215. IBM console printer-keyboard.

3270. IBM information display system.

3370, 3380 or 3390. IBM direct access storage device (disk).

3480 or 3490. IBM magnetic tape subsystem.

9336 or 9345. IBM direct access storage device (disk).

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Advanced Peer-to-Peer Networking	OS/2
APPN	OS/390
CICS	OSA
Common User Access	PowerPC
e-business	RACF
ECKD	RAMAC
ESA/390	S/390
ESCON	Seascape
IBM	System/390
Micro Channel	VM/ESA
Multiprise	VSE/ESA
MVS	VTAM

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

LINUX is a registered trademark of Linus Torvalds and others.

Microsoft, Windows NT, and MSDOS are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

International License Agreement for Non-Warranted Programs

Part 1 - General Terms

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THE PROGRAM. IBM WILL LICENSE THE PROGRAM TO YOU ONLY IF YOU FIRST ACCEPT THE TERMS OF THIS AGREEMENT. BY USING THE PROGRAM YOU AGREE TO THESE TERMS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PROMPTLY RETURN THE UNUSED PROGRAM TO THE PARTY (EITHER IBM OR ITS RESELLER) FROM WHOM YOU ACQUIRED IT TO RECEIVE A REFUND OF THE AMOUNT YOU PAID.

The Program is owned by International Business Machines Corporation or one of its subsidiaries (IBM) or an IBM supplier, and is copyrighted and licensed, not sold.

The term ″Program″ means the original program and all whole or partial copies of it. A Program consists of machine-readable instructions, its components, data, audio-visual content (such as images, text, recordings, or pictures), and related licensed materials.

This Agreement includes Part 1 - General Terms and Part 2 - Country-unique Terms and is the complete agreement regarding the use of this Program, and replaces any prior oral or written communications between you and IBM. The terms of Part 2 may replace or modify those of Part 1.

1. Licence

   Use of the Program

   IBM grants you a nonexclusive licence to use the Program.

   You may 1) use the Program to the extent of authorizations you have acquired and 2) make and install copies to support the level of use authorized, providing you reproduce the copyright notice and any other legends of ownership on each copy, or partial copy, of the Program.

   If you acquire this Program as a program upgrade, your authorization to use the Program from which you upgraded is terminated.

   You will ensure that anyone who uses the Program does so only in compliance with the terms of this Agreement.

   You may not 1) use, copy, modify, or distribute the Program except as provided in this Agreement; 2) reverse assemble, reverse compile, or otherwise translate the Program except as specifically permitted by law without the possibility of contractual waiver; or 3) sublicense, rent, or lease the Program.

   Transfer of Rights and Obligations

   You may transfer all your license rights and obligations under a Proof of Entitlement for the Program to another party by transferring the Proof of Entitlement and a copy of this Agreement and all documentation. The transfer of your license rights and obligations terminates your authorization to use the Program under the Proof of Entitlement.

2. Proof of Entitlement

   The Proof of Entitlement for this Program is evidence of your authorization to use this Program and of your eligibility for future upgrade program prices (if announced) and potential special or promotional opportunities.

3. Charges and Taxes

IBM defines use for the Program for charging purposes and specifies it in the Proof of Entitlement. Charges are based on extent of use authorized. If you wish to increase the extent of use, notify IBM or its reseller and pay any applicable charges. IBM does not give refunds or credits for charges already due or paid.

If any authority imposes a duty, tax, levy or fee, excluding those based on IBM's net income, upon the Program supplied by IBM under this Agreement, then you agree to pay that amount as IBM specifies or supply exemption documentation.

4. No Warranty

   SUBJECT TO ANY STATUTORY WARRANTIES WHICH CAN NOT BE EXCLUDED, IBM MAKES NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, THE WARRANTY OF NON-INFRINGEMENT AND THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY. IBM MAKES NO WARRANTY REGARDING THE CAPABILITY OF THE PROGRAM TO CORRECTLY PROCESS, PROVIDE AND/OR RECEIVE DATE DATA WITHIN AND BETWEEN THE 20TH AND 21ST CENTURIES.

   The exclusion also applies to any of IBM's subcontractors, suppliers, or program developers (collectively called ″Suppliers″).

   Manufacturers, suppliers, or publishers of non-IBM Programs may provide their own warranties.

5. Limitation of Liability

   NEITHER IBM NOR ITS SUPPLIERS WILL BE LIABLE FOR ANY DIRECT OR INDIRECT DAMAGES, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST SAVINGS, OR ANY INCIDENTAL, SPECIAL, OR OTHER ECONOMIC CONSEQUENTIAL DAMAGES, EVEN IF IBM IS INFORMED OF THEIR POSSIBILITY. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE EXCLUSION OR LIMITATION MAY NOT APPLY TO YOU.

6. General

   Nothing in this Agreement affects any statutory rights of consumers that cannot be waived or limited by contract.

   IBM may terminate your licence if you fail to comply with the terms of this Agreement. If IBM does so, you must immediately destroy the Program and all copies you made of it.

   You agree to comply with applicable export laws and regulations.

   Neither you nor IBM will bring a legal action under this Agreement more than two years after the cause of action arose unless otherwise provided by local law without the possibility of contractual waiver or limitation.

   Neither you nor IBM is responsible for failure to fulfill any obligations due to causes beyond its control.

   IBM does not provide program services or technical support, unless IBM specifies otherwise.

   The laws of the country in which you acquire the Program govern this Agreement, except 1) in Australia, the laws of the State or Territory in which the transaction is performed govern this Agreement; 2) in Albania, Armenia, Belarus, Bosnia/Herzegovina, Bulgaria, Croatia, Czech Republic, Georgia, Hungary, Kazakhstan, Kirghizia, Former Yugoslav Republic of Macedonia (FYROM), Moldova, Poland, Romania, Russia, Slovak Republic, Slovenia, Ukraine, and Federal Republic of Yugoslavia, the laws of Austria govern this Agreement; 3) in the United Kingdom, all disputes relating to this Agreement will be governed by

English Law and will be submitted to the exclusive jurisdiction of the English courts; 4) in Canada, the laws in the Province of Ontario govern this Agreement; and 5) in the United States and Puerto Rico, and People's Republic of China, the laws of the State of New York govern this Agreement.

Part 2 - Country-unique Terms

AUSTRALIA:

No Warranty (Section 4):

The following paragraph is added to this Section:

Although IBM specifies that there are no warranties, you may have certain rights under the Trade Practices Act 1974 or other legislation and are only limited to the extent permitted by the applicable legislation.

Limitation of Liability (Section 3):

The following paragraph is added to this Section:

Where IBM is in breach of a condition or warranty implied by the Trade Practices Act 1974, IBM's liability is limited to the repair or replacement of the goods, or the supply of equivalent goods. Where that condition or warranty relates to right to sell, quiet possession or clear title, or the goods are of a kind ordinarily acquired for personal, domestic or household use or consumption, then none of the limitations in this paragraph apply.

GERMANY:

No Warranty (Section 4):

The following paragraphs are added to this Section:

The minimum warranty period for Programs is six months.

In case a Program is delivered without Specifications, we will only warrant that the Program information correctly describes the Program and that the Program can be used according to the Program information. You have to check the usability according to the Program information within the ″money-back guaranty″ period.

Limitation of Liability (Section 5):

The following paragraph is added to this Section:

The limitations and exclusions specified in the Agreement will not apply to damages caused by IBM with fraud or gross negligence, and for express warranty.

INDIA:

General (Section 6):

The following replaces the fourth paragraph of this Section:

If no suit or other legal action is brought, within two years after the cause of action arose, in respect of any claim that either party may have against the other, the

rights of the concerned party in respect of such claim will be forfeited and the other party will stand released from its obligations in respect of such claim.

IRELAND:

No Warranty (Section 4):

The following paragraph is added to this Section:

Except as expressly provided in these terms and conditions, all statutory conditions, including all warranties implied, but without prejudice to the generality of the foregoing, all warranties implied by the Sale of Goods Act 1893 or the Sale of Goods and Supply of Services Act 1980 are hereby excluded.

ITALY:

Limitation of Liability (Section 5):

This Section is replaced by the following:

Unless otherwise provided by mandatory law, IBM is not liable for any damages which might arise.

NEW ZEALAND:

No Warranty (Section 4):

The following paragraph is added to this Section:

Although IBM specifies that there are no warranties, you may have certain rights under the Consumer Guarantees Act 1993 or other legislation which cannot be excluded or limited. The Consumer Guarantees Act 1993 will not apply in respect of any goods or services which IBM provides, if you require the goods and services for the purposes of a business as defined in that Act.

Limitation of Liability (Section 5):

The following paragraph is added to this Section:

Where Programs are not acquired for the purposes of a business as defined in the Consumer Guarantees Act 1993, the limitations in this Section are subject to the limitations in that Act.

PEOPLE'S REPUBLIC OF CHINA:

Charges (Section 3):

The following paragraph is added to the Section:

All banking charges incurred in the People's Republic of China will be borne by you and those incurred outside the People's Republic of China will be borne by IBM.

UNITED KINGDOM:

Limitation of Liability (Section 5):

The following paragraph is added to this Section at the end of the first paragraph:

The limitation of liability will not apply to any breach of IBM's obligations implied by Section 12 of the Sales of Goods Act 1979 or Section 2 of the Supply of Goods and Services Act 1982.

GNU General Public Licence, Version 2, June 1991

DISCLAIMER

Elements of LINUX for S/390 which utilise internal details of the S/390 systems remain the intellectual property and copyright of IBM, notwithstanding the licence below. This right will be waived for specific elements in the case that the documentation specific to the element indicates that it is published under the GNU General Public Licence.

LINUX for S/390 is licensed under the GNU General Public Licence which is reproduced below:

Free Software Foundation, Inc.
59 Temple Place,
Suite 330,
Boston,
MA 02111-1307
USA

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and

2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU General Public Licence: Terms and conditions for copying, distribution and modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The ″Program″, below, refers to any such program or work, and a ″work based on the Program″ means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term ″modification″.) Each licensee is addressed as ″you″.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an

appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source

code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and ″any later version″, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

    NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

    END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the ″copyright″ line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

```
This program is free software; you can redistribute it and/or modify it under the
terms of the GNU General Public License as published by the Free Software Foundation;
either version 2 of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with this
program; if not, write to the Free Software Foundation, Inc., 59 Temple Place,
Suite 330, Boston, MA 02111-1307 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it under certain
conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision'
(which makes passes at compilers) written by James Hacker.
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Index

Special Characters

| OSA-Express 53

Numerics

3215 line mode terminal 23
3270 23
3380 9
3390 9
9345 9

A

autosensing 58, 97

B

basic mode 51
block device
  tape 30

C

character device
  tape 30
checksum 49, 50, 55, 97
chpid 59
cio_msg 94
codepage 25
configuration
  CTC 119
  ESCON 119
  GbE 119
  Gigabit Ethernet 119
  OSA Express 119
  tape 121
connections
  CTC 40
  ESCON 40
console 23
control characters 24
CRC 49, 50
CTC 49
  configuration 119
  connection 40
  device driver 37
  device support 119
  features 37
  kernel example 38
  kernel parameter 37
  module example 39
  module options 39
  recovery 42
  syntax 37, 38
CUU 49

D

DASD device driver 7
dasdfmt 66
device driver 53
  CTC 37
  DASD 7
  ESCON 37
  tape 29
  VM minidisk 15
  XPRAM 19
device major number 29
device name 98
device number 56, 98
device support
  CTC 119
  ESCON 119
  GbE 119
  Gigabit Ethernet 119
  OSA Express 119
  tape 121

E

ECKD 9, 10
edit characters
  VM console 25
Enterprise Storage Server 10
ESCON
  configuration 119
  connection 40
  device driver 37
  device support 119
  features 37
  kernel example 38
  module example 39
  recovery 42
  syntax 37, 38
ethernet 49, 51
examples
  CTC kernel 38
  CTC module 39
  ESCON kernel 38
  ESCON module 39
  tape driver 33

F

FBA 9
FDDI 51
features
  CTC 37
  ESCON 37
filesystem
  tape 30

G

GbE
  configuration 119
  device support 119
General Public Licence
  applying 139
  GNU 135
  terms and conditions 136
Gigabit Ethernet
  configuration 119
  device support 119
GNU
  applying 139
  General Public Licence 135
  terms and conditions 136

H

Hardware console 23

I

i/o message suppression 94
insmod 39
ipldelay 86
ISO9660 filesystem 30
IUCV 43

K

kernel parameter
  CTC 37
  tape device driver 31
kernel source tree ix

L

LCS
  device driver 49
  driver parameters 49, 97
line edit characters
  VM console 25

M

mac 52
maxcpus 87
maximum tape devices 29
mem 88
modprobe 39
module options
  CTC 39
module parameter
  tape device driver 32
Multiprise 10

N

NFS 51
noinitrd 89
notices 127

O

options 39
OSA-2 49, 51
OSA Express
  configuration 119
  device support 119
OSA-Express 53

P

P/390 23, 88
parameter file 39
parameter line 96
problems 52

Q

qdio 53, 58, 61
qeth 53
qeth_options 97
queueing 53, 98

R

RAMAC 10
recovery
  CTC 42
  ESCON 42
ro 91
root 92
routing 53, 55, 97
RVA 10

S

Seascape 10
silo 82
smp 61
special characters
  VM console 25
subchannel 58
syntax
  CTC 37, 38
  ESCON 37, 38

T

tape
  configuration 121
  device support 121
tape block device 30
tape character device 30
tape control operations 30

tape device driver 29
kernel parameter 31
module parameter 32
tape driver examples 33
tape filesystem 30
tape restrictions 34
TCP/IP 37, 43
terms and conditions
GNU General Public Licence 136
token ring 49, 51
trademarks 128

V

VInput 26
VM console
line edit characters 25
VM minidisk
device driver 15
vmhalt 93

X

x3270 26
XPRAM
device driver 19

LNUX-1003-03