

Offline Version of KERNEL-PROJECT .kickino.org



(CVS-)Revision: 1.2

Date: 2005-09-11

Copyright (C) 2005 by Anton Brondz, Sebastian Wieseler [GPL]

Contents

1	Introduction	3
2	Configuration	3
2.1	Get Linux	3
2.2	Untar the tarball	3
2.3	Rules of the makefile	4
3	Compilation	6
4	Patching	8
5	Boodloader	9
5.1	LILO	9
5.2	GRUB	10
6	Masthead	13
6.1	Copyright	13
6.2	Legal	13
6.3	Powered by	13

1 Introduction

This short HOWTO is meant for users new to the Linux kernel. It gives you a short, but good guide to everything from getting the Linux kernel to building and editing your bootloader's configuration and finally rebooting and enjoying your new kernel. We have provided good explanations to all the options and the process of compiling and building. Happy reading!

2 Configuration

2.1 Get Linux

You can download the current version of Linux from <http://kernel.org/>. There you can get almost any version of the Linux kernel.

E.g. for getting the 2.6.XX.YY - you can download <http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.XX.YY.tar.bz2> or <http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.XX.YY.tar.gz> at your choice. You should use 'wget' for doing this or any other download manager.

2.2 Untar the tarball

```
$ su
# cd /usr/src/
# gzip -cd linux-2.4.XX.tar.gz — tar xvf -
# ln -s linux-2.4.XX linux
# cd linux
```

Where XX is the version number of your kernel.

To configure and compile the kernel, you need to use the Makefile that is distributed with the kernel sources. The figure below shows the possible parameters for the Makefile.

(/usr/src/linux/README)

2.3 Rules of the makefile

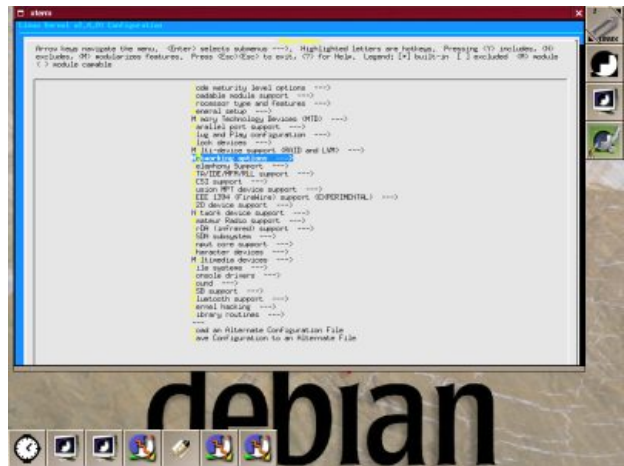
MAKE RULE	DESCRIPTION
make psdocs make pdfdoc make htmdocsc creates	documentation in the appropriate format (PostScript/PDF/HTML)
make mrproper	forces delete of old object files and dependencies
make clean	deletes old object files
make config	configure kernel (plain ASCII)
make menuconfig	configure kernel (ASCII-menus)
make xconfig	configure kernel (X Window-based)
make oldconfig	takes an existing <code>./config</code> as config defaults
make dep	setup all dependencies
make bzImage	create a "bzip2" compressed kernel image
make bzdisk	create a "bzip2" compressed kernel boot disk
make install	installs new kernel only if you use the LILO boot loader
make modules modules_install	create and install kernel modules see Documentation/modules.txt

When configuring, you can choose between `make config`, `make menuconfig`, `make xconfig`, `make gconfig` and `make oldconfig`.

This picture shows make xconfig



... and this one make menuconfig.



So as you can see, you can configure Linux comfortably using graphical dialogs. More information on the individual kernel configurations can be found under <http://www.linuxfibel.de/kconf.htm> (German)

3 Compilation

After configuring your kernel, you need to compile it with:

```
$ su
# cd /usr/src/linux
```

```
# make dep
# make clean
```

You're now able to choose from one of the following options:

```
# make bzImage
# make zImage
# make bzdisk
# make install
```

Use *bzImage* to create a compressed kernel image
zImage for a compressed kernel,
bzdisk to create a boot disk and
install if you want to use LILO (all further steps then become unnecessary).

```
# make modules
# make modules_install
```

The time required by your computer to compile Linux is processor and machine dependant.

Once this process has been completed, you can copy your new kernel to */boot/*.

Please **don't** forget to make a backup of your old kernel though. The backup commands are as follows:

```
# cp /boot/vmlinuz /boot/vmlinuz.old
# cp /boot/System.map /boot/System.map.old

# cp arch/i386/boot/bzImage /boot/
# cp System.map /boot/
# cp .config /boot/
```

After copying the kernel, you still need to test it.

Depending on your bootloader you still need to set various options in order to boot your new kernel.

```
# shutdown -r now
```

In the case where you want to overwrite your existing kernel, you need to use the following commands:

```
# cp /usr/src/kernel-old-version/.config /usr/src/kernel-new-version/  
# cd /usr/src/kernel-new-version/  
# make oldconfig  
# make dep && make clean bzImage modules modules_install
```

4 Patching

This chapter shows possibilities to upgrade to follow-up kernel versions without having to download a new kernel archive in it's entirety.

Upgrading is done utilizing the small program **patch**, originally developed by Larry Wall.

You will just have to download a rather small file (called "patch") that contains the differences between the two kernel versions. You can get it, for instance, from kernel.org.

The following lines show how to backup your current kernel tree and how to apply the patch.

```
$ su
# cd /usr/src/linux
# make clean
# cd ..
# tar zcvf old-tree.tar.gz linux

# zcat patch.gz patch -p0
```

5 Boodloader

This section is about LILO and GRUB.

5.1 LILO

(Linux Loader) The lilo configuration file is */etc/lilo.conf*.
For your new kernel, it could look like this:

```
# /etc/lilo.conf - See: 'lilo(8)' and 'lilo.conf(5)',
# ----- 'install-mbr(8)', '/usr/share/doc/lilo/'
# and '/usr/share/doc/mbr/'.
# +-----+
# -- !! Reminder !! --
# --
# -- Don't forget to run 'lilo' after you make changes to this --
# -- conffile, '/boot/bootmess.txt', or install a new kernel. The --
# -- computer will most likely fail to boot if a kernel-image --
# -- post-install script or you don't remember to run 'lilo'. --
# --
# +-----+

# [...]

# Boot GNU/Linux per default
#
default=GNU/Linux

image="/boot/bzImage"
label="GNU/Linux"
restricted
# append="hdc=ide-scsi"
# further kernel optionens

image="/boot/bzImage.old"
label="GNU/LinuxOLD"
restricted
```

Further information on LILO can be found on lilo.org

5.2 GRUB

(Grand Unified Bootloader) At first, you should ensure there is an already complete installation of GRUB on your hard disk.

To write GRUB to a floppy you want to boot from, please do the following:

```
$ su

# cd /usr/share/grub/i386-pc
or at different distros:
# cd /usr/lib/grub/i386-pc

# dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
# dd if=stage2 of=/dev/fd0 bs=512 seek=1
153+1 records in
153+1 records out
```

There is a GRUB option that lets you boot from a floppy while the kernel itself is read from hard disk.

This is done like this:

```
# grub
grub> root (hd0,0)
grub> setup(fd0)
grub> quit
```

To install GRUB permanently on your hard disk, use this command:

```
# grub-install /dev/hda
Probing devices to guess BIOS drives. This may take a long time.
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script 'grub-install'.

(fd0) /dev/fd0
(hd0) /dev/hda
(hd1) /dev/hdb
```

...

Depending on your configuration, you'll have to select the correct hardware type (*/dev/hda*).

A warning: Using *grub install* to install on several hard disks sometimes leads to problems!

To write GRUB to a partition, you would enter:

```
# grub
grub> root (hd0,0)
grub> find /boot/grub/stage1
grub> setup (hd0) (MBR-Installation)
grub> setup (hd0,0) (Installation in Bootsektor von '0')
```

To start GNU/Linux directly, enter the following commands at the boot prompt:

```
grub> kernel /bzImage root=/dev/hda1
grub> boot
```

Or create a permanent GRUB configuration that lets you choose from a boot menu.

This file is named *menu.lst* and should be in your */boot* directory (*/boot/grub/menu.lst*). It could look like this:

```
#
# Sample boot menu configuration file
#

# Per default, boot the first entry
default 0

# Wait 30 seconds, then boot the default entry
timeout 30

# If the first entry doesn't work, use the second
fallback 1

# GNU/Hurd (first harddisc, second partition)
title GNU/Hurd
root (hd0,1)
kernel /boot/gnumach.gz root=hd0s1
module /boot/serverboot.gz

# GNU/Linux (second harddisc, first partition)
title GNU/Linux
kernel (hd1,0)/bzImage root=/dev/hdb1

# Windows NT or Windows95 (first harddisc)
title Windows NT / Windows 95 boot menu
root (hd0,0)
makeactive
chainloader +1
# Because of DOS, if Windows NT was installed
# chainload /bootsect.dos
```

6 Masthead

6.1 Copyright

Offline version of kernel-project
Copyright (c) 2005 Anton Brondz, Sebastian Wieseler

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., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Crystalized Tux by gg3po/Tony (found on <http://kde-look.org>) available under GPL

All other pictures (C) Sebastian Wieseler 2005 - GPLed

6.2 Legal

Linux is a registered trademark of Linus Torvalds.
All other trademarks are property of their respective owners.

6.3 Powered by

ℒ_AT_EX
kernel-project.kickino.org
savannah.nongnu.org