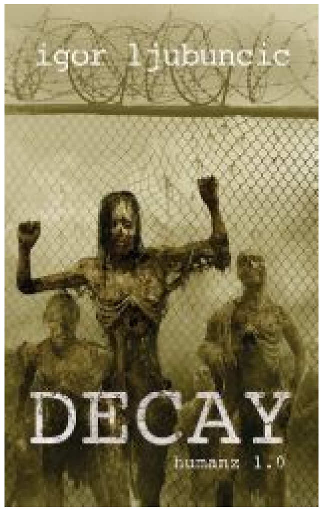
GRUB 2 bootloader - Full tutorial

Updated: December 4, 2009; November 30, 2017



Dedoimedo definite GRUB 2.00 multi-boot tutorial featured in the 117th issue of the Linux User & Developer Magazine! You really should take a closer look.

GRUB 2 bootloader is in continuous development. Some information may change over time, as features are added or removed and fixes introduced. For latest updates, please take a look at Updates section below.

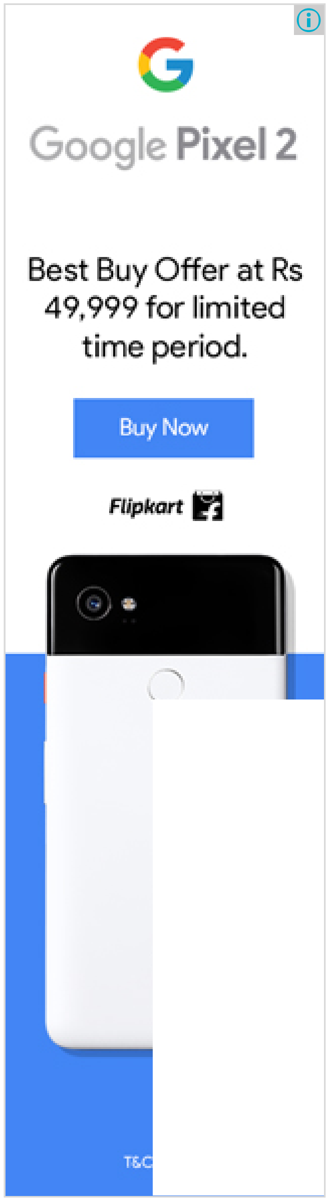
This tutorial focuses on GRUB version 2, the next generation of the popular bootloader. If you are looking for the original (legacy) GRUB tutorial, please take a look at this article.

Welcome to the GRUB 2 bootloader tutorial! You must have read my GRUB legacy guide. In the last two and a half years, it alone has garnered some half a million views, proving to be quite popular and useful to computer users worldwide.

I want to recreate the same success with my GRUB 2 tutorial. My goal is to provide people running any flavor of UNIX-like operating systems or multi-booting their computers and using GRUB as their bootloader with a simple, no-nonsense, step-by-step, proven and working tutorial that should allow them to quickly, easily and painlessly control the boot sequence of their systems.

In this tutorial, you will learn how to work with GRUB 2, add and remove menu entries, customize titles and boot options, dual-boot and triple-boot operating systems, combine legacy GRUB and GRUB 2, and we will even see how Windows fits into this scheme. After that, we will learn how to recover from errors and mistakes. Follow me.

Table of Contents



Warning

Warning! GRUB 2 is still mostly beta software. Although it already ships with Ubuntu flavors, it is not yet production quality per se. You can probably use it without any troubles, although there is a slight, remote yet possible chance of damage.

You need to be aware of this. Furthermore, whenever handling delicate tasks like the critical functions of the system, disk partitioning, boot sequence, imaging, etc, you should always be prepared for the worst. This means:

- You must have a solid, proven backup procedure for all your data.
- You must possess the tools and the knowledge to quickly recover from disasters. This includes being able to restore a previous system image, fix broken configurations, restore the bootloaders, and more.
- You must be confident in what you are doing.

Now that we know this, let us proceed cheerfully and safely. Just remember that GRUB 2 is still beta. Although the same can be claimed for Google Mail, which was beta for some six years or so, you must exercise caution. What's more, the contents and relevance of contents in this tutorial might yet change as GRUB 2 makes into the production, so stay tuned for any updates.

GRUB 2 roadmap

This is something you should consider before trying GRUB 2. When will GRUB 2 become the de facto bootloader for UNIX-like operating systems? Currently, GRUB legacy is doing fine and will continue for many more years. Given the long-term support by companies like RedHat and Novell for their server distributions, GRUB legacy is going to remain the key player for at least 5-10 years.

On desktops, the adoption rate may be faster, but do not expect any miracles too soon. Nevertheless, it does not hurt to start exploring. Be aware that you may encounter some compatibility issues down the road, especially with more conservative distributions that do not embrace new technologies too quickly.

At the time being, GRUB 2 is only used by the Ubuntu family, which makes about a third to one half of the Linux desktop market. That makes this tutorial rather relevant, as about one in every two or three Linux home users will probably be interested in learning more about GRUB 2 and its uses. Let's now move on to the actual mechanics.

GRUB 2 introduction

Before you dig in, I strongly advise you to read my original GRUB tutorial. This will help you understand this article better. GRUB 2 introduces many new changes. GRUB 2 has better portability and modularity, supports non-ASCII characters, dynamic loading of modules, real memory management, and more. All these are pretty much irrelevant for most users. What you need to know are the changes in the configuration files and the way GRUB 2 operates.

New layout

Old GRUB files were (and still are!) located under /boot/grub/, including the menu.lst file that was read during boot and which contents were displayed to the user in the form of the GRUB menu. GRUB 2 places its files in three core locations:

/boot/grub/grub.cfg - This is the main configuration file that replaces menu.lst. Unlike menu.lst, this file cannot be edited by hand! I strongly advise against trying to tamper with this file, using chattr command or anything of the sort. Let it be.

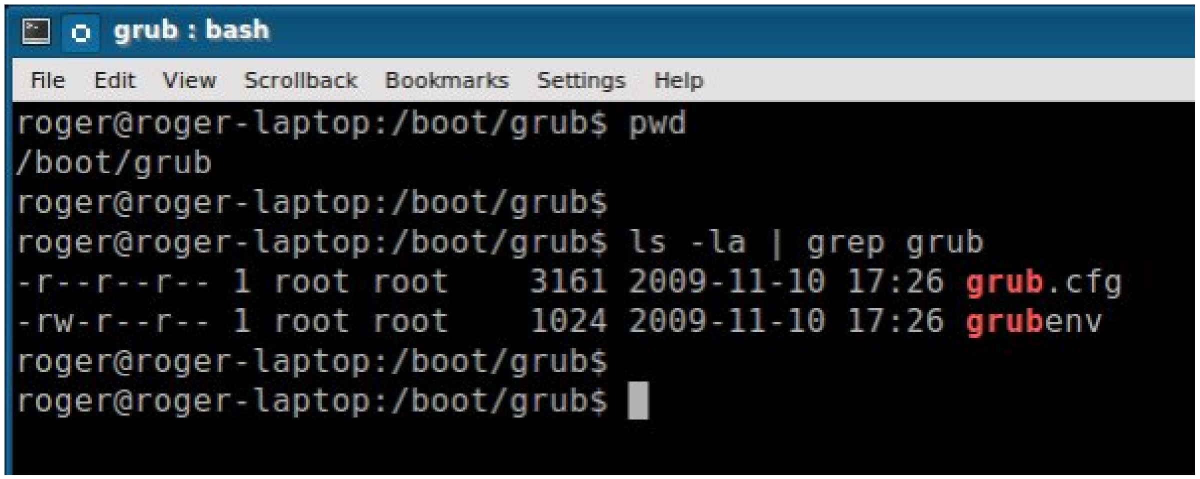
/etc/grub.d/ - This new directory contains GRUB scripts. These scripts are building blocks from which the grub.cfg file is built. When the relevant GRUB command is executed, the scripts are read in a certain sequence and grub.cfg is created.

/etc/default/grub - This file contains the GRUB menu settings that are read by the GRUB scripts and written into grub.cfg. It is the customization part of the GRUB, similar to the old menu.lst, except the actual boot entries.

This means that if you want to change the GRUB menu, you will have to edit existing scripts or create new ones, then update the menu. This is more similar to LILO than GRUB legacy, which allow editing the menu on the fly.

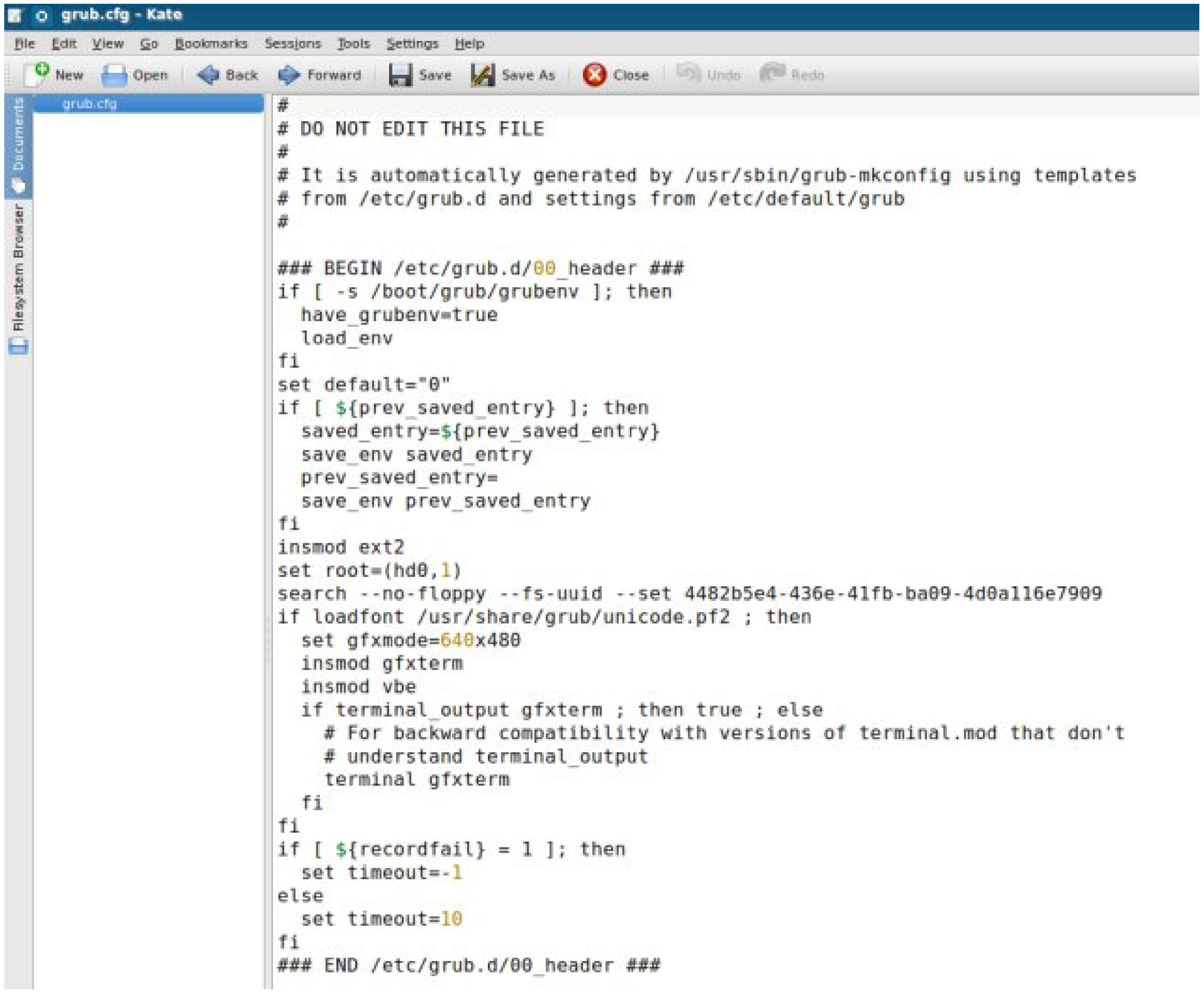
Sample grub.cfg file

The file is located under /boot/grub/:



```
roger@roger-laptop:/boot/grub$ pwd
/boot/grub
roger@roger-laptop:/boot/grub$
roger@roger-laptop:/boot/grub$ ls -la | grep grub
-r--r--r-- 1 root root    3161 2009-11-10 17:26 grub.cfg
-rw-r--r-- 1 root root    1024 2009-11-10 17:26 grubenv
roger@roger-laptop:/boot/grub$
roger@roger-laptop:/boot/grub$
```

This is what the grub.cfg file looks like:

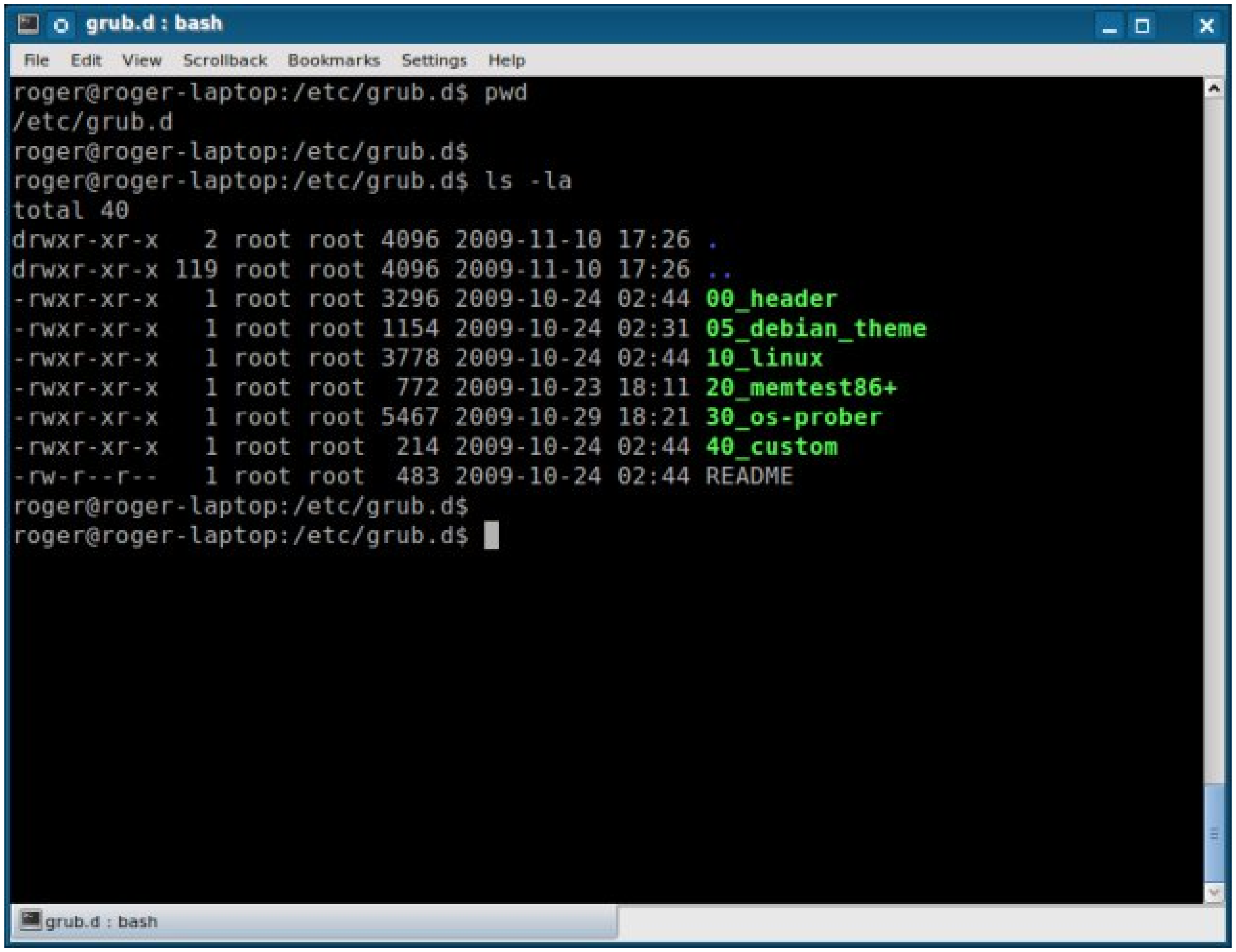

grub.cfg - Kate
File Edit View Go Bookmarks Sessions Tools Settings Help
New Open Back Forward Save Save As Close Undo Redo
Documents
Filesystem Browser
grub.cfg

```
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by /usr/sbin/grub-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#

### BEGIN /etc/grub.d/00_header ###
if [ -s /boot/grub/grubenv ]; then
  have_grubenv=true
  load_env
fi
set default="0"
if [ ${prev_saved_entry} ]; then
  saved_entry=${prev_saved_entry}
  save_env saved_entry
  prev_saved_entry=
  save_env prev_saved_entry
fi
insmod ext2
set root=(hd0,1)
search --no-floppy --fs-uuid --set 4482b5e4-436e-41fb-ba09-4d0a116e7909
if loadfont /usr/share/grub/unicode.pf2 ; then
  set gfxmode=640x480
  insmod gfxterm
  insmod vbe
  if terminal_output gfxterm ; then true ; else
    # For backward compatibility with versions of terminal.mod that don't
    # understand terminal_output
    terminal gfxterm
  fi
fi
if [ ${recordfail} = 1 ]; then
  set timeout=-1
else
  set timeout=10
fi
### END /etc/grub.d/00_header ###
```

It is not really interesting, because it's just a shell script. Most people will probably not want to read its contents or be able to fully understand them. This file is still worth a quick look any time you update the GRUB menu, to make sure the correct entries are included as you expected. But only if you're comfortable with scripts!

Sample /etc/grub.d/ directory

These are the contents of the directory on a fresh installation, in this case Kubuntu:



```
roger@roger-laptop:/etc/grub.d$ pwd
/etc/grub.d
roger@roger-laptop:/etc/grub.d$
roger@roger-laptop:/etc/grub.d$ ls -la
total 40
drwxr-xr-x   2 root root 4096 2009-11-10 17:26 .
drwxr-xr-x 119 root root 4096 2009-11-10 17:26 ..
-rwxr-xr-x   1 root root 3296 2009-10-24 02:44 00_header
-rwxr-xr-x   1 root root 1154 2009-10-24 02:31 05_debian_theme
-rwxr-xr-x   1 root root 3778 2009-10-24 02:44 10_linux
-rwxr-xr-x   1 root root  772 2009-10-23 18:11 20_memtest86+
-rwxr-xr-x   1 root root 5467 2009-10-29 18:21 30_os-prober
-rwxr-xr-x   1 root root  214 2009-10-24 02:44 40_custom
-rw-r--r--   1 root root  483 2009-10-24 02:44 README
roger@roger-laptop:/etc/grub.d$
roger@roger-laptop:/etc/grub.d$
```

Let's review the scripts:

00_header is the script that loads GRUB settings from /etc/default/grub, including timeout, default boot entry, and others. We will talk more about these soon.

05_debian_theme defines the background, colors and themes. The name of this script is definitely going to change to when other distributions adopt GRUB 2.

10_linux loads the menu entries for the installed distribution.

20_memtest86+ loads the memtest utility.

30_os-prober is the script that will scan the hard disks for other operating systems and add them to the boot menu.

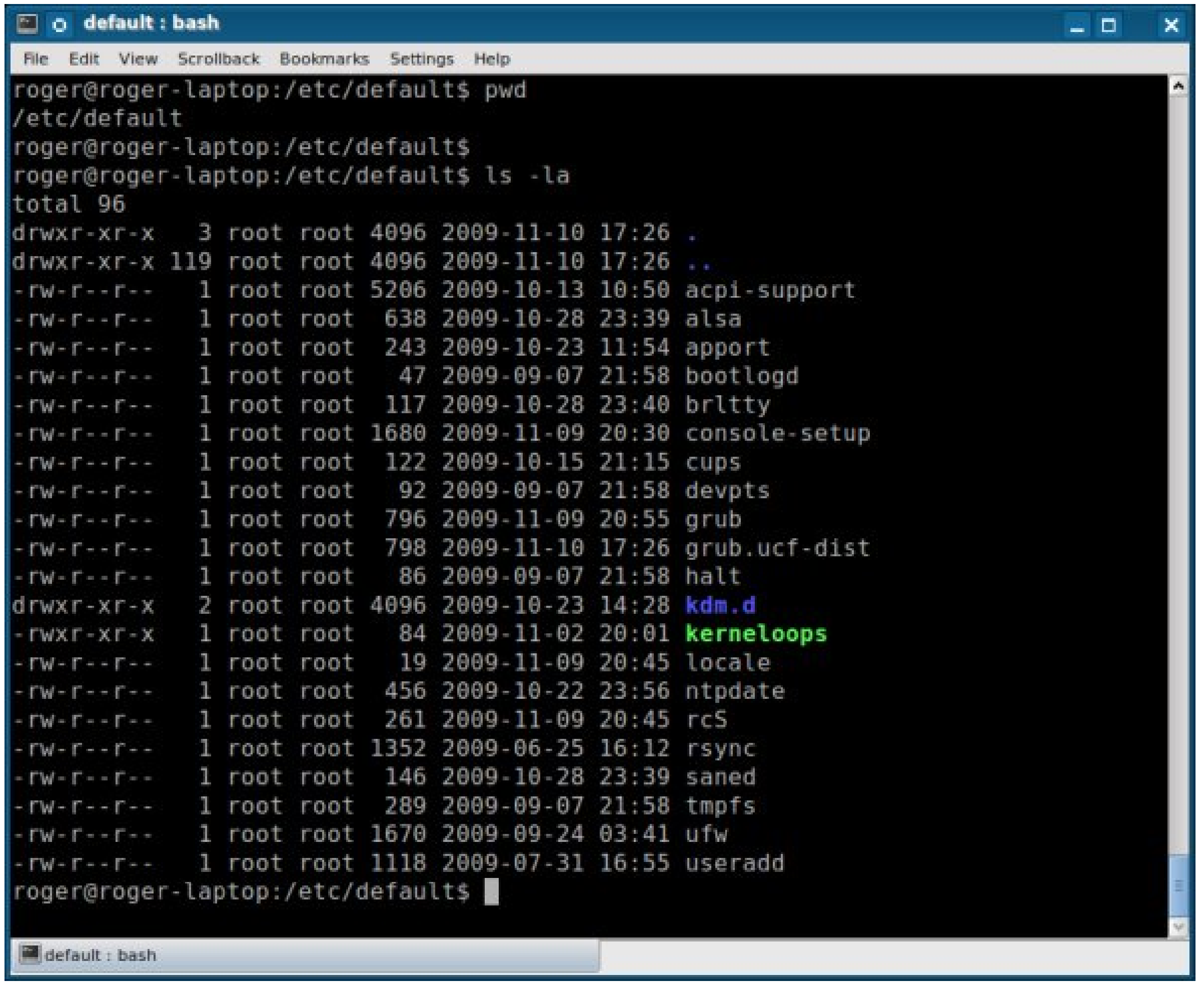
40_custom is a template that you can use to create additional entries to be added to the boot menu.

Have you noticed the numbering in the script names? Well, this is somewhat similar to the order of Start/Kill scripts used in different runlevels. The numbering defines precedence. This means that 10_linux will be executed before 20_memtest86+ and therefore placed higher in the boot menu order.

The scripts are not very interesting. Like the grub.cfg file, they are not intended to be edited, save for 40_custom. You need to very careful when working with these scripts. We will soon see what they look like and learn how to use them.

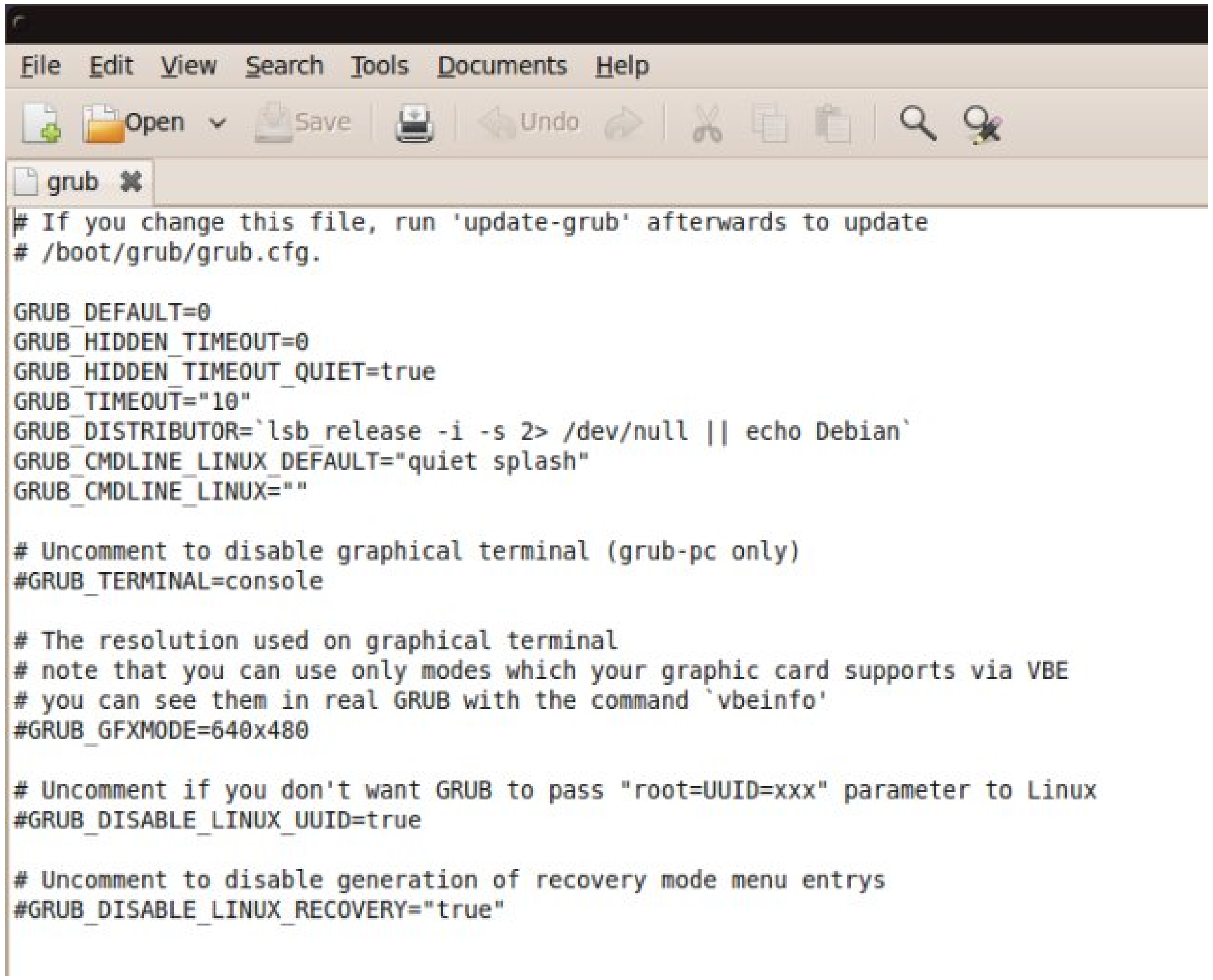
Sample /etc/default/grub

The file is located under /etc/default:



```
roger@roger-laptop:/etc/default$ pwd
/etc/default
roger@roger-laptop:/etc/default$
roger@roger-laptop:/etc/default$ ls -la
total 96
drwxr-xr-x   3 root root 4096 2009-11-10 17:26 .
drwxr-xr-x 119 root root 4096 2009-11-10 17:26 ..
-rw-r--r--   1 root root 5206 2009-10-13 10:50 acpi-support
-rw-r--r--   1 root root  638 2009-10-28 23:39 alsa
-rw-r--r--   1 root root  243 2009-10-23 11:54 apport
-rw-r--r--   1 root root   47 2009-09-07 21:58 bootlogd
-rw-r--r--   1 root root  117 2009-10-28 23:40 brltty
-rw-r--r--   1 root root 1680 2009-11-09 20:30 console-setup
-rw-r--r--   1 root root  122 2009-10-15 21:15 cups
-rw-r--r--   1 root root   92 2009-09-07 21:58 devpts
-rw-r--r--   1 root root  796 2009-11-09 20:55 grub
-rw-r--r--   1 root root  798 2009-11-10 17:26 grub.ucf-dist
-rw-r--r--   1 root root   86 2009-09-07 21:58 halt
drwxr-xr-x   2 root root 4096 2009-10-23 14:28 kdm.d
-rwxr-xr-x   1 root root   84 2009-11-02 20:01 kerneloops
-rw-r--r--   1 root root   19 2009-11-09 20:45 locale
-rw-r--r--   1 root root  456 2009-10-22 23:56 ntpdate
-rw-r--r--   1 root root  261 2009-11-09 20:45 rcS
-rw-r--r--   1 root root 1352 2009-06-25 16:12 rsync
-rw-r--r--   1 root root  146 2009-10-28 23:39 saned
-rw-r--r--   1 root root  289 2009-09-07 21:58 tmpfs
-rw-r--r--   1 root root 1670 2009-09-24 03:41 ufw
-rw-r--r--   1 root root 1118 2009-07-31 16:55 useradd
roger@roger-laptop:/etc/default$
```

This directory contains many other files so do not assume it is just used for GRUB. The grub file is a text file that is parsed by the 00_header script. You can make your changes here, if you want. We will talk about these later, in the Customization section.



```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.

GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT="10"
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entrys
#GRUB_DISABLE_LINUX_RECOVERY="true"
```

Now we know what the core files are. Let's see how we use them.

Basic usage

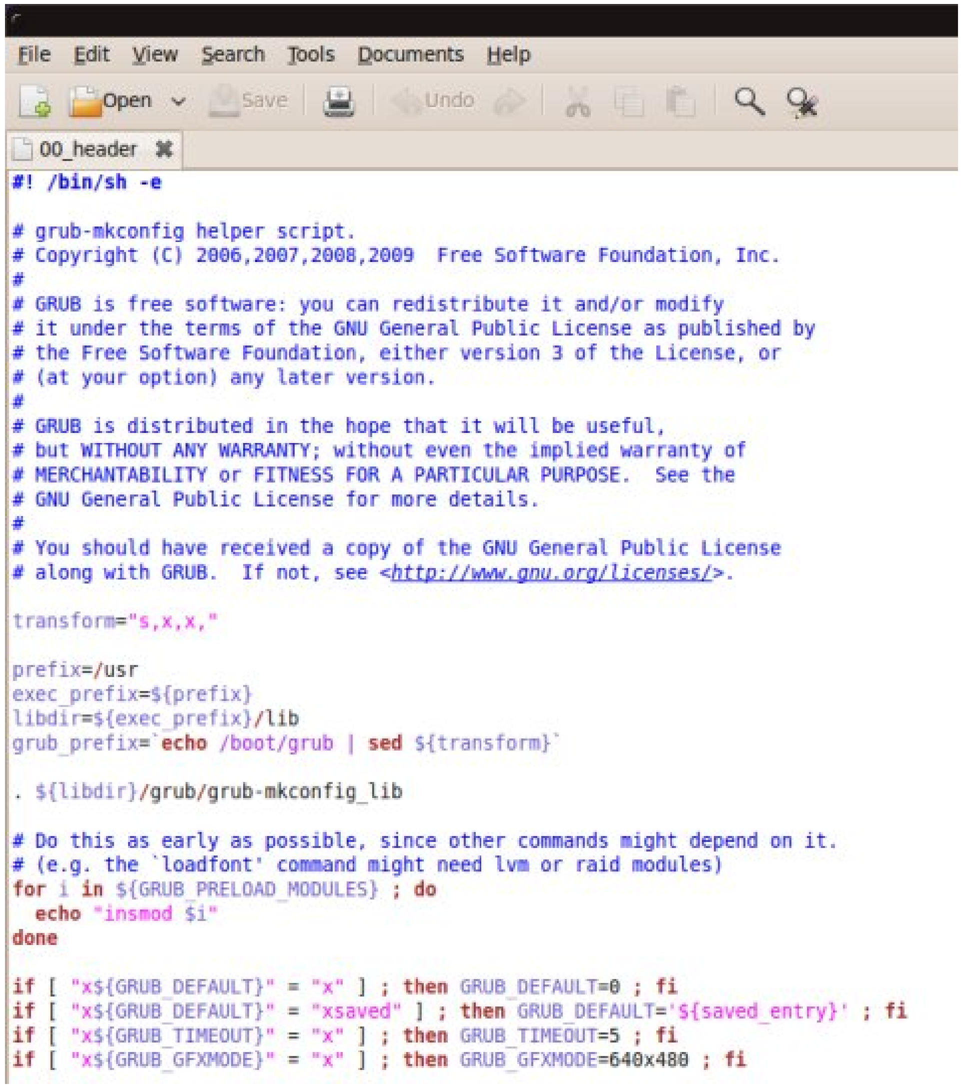
It's time to put our theoretical knowledge to some real action.

How GRUB 2 works?

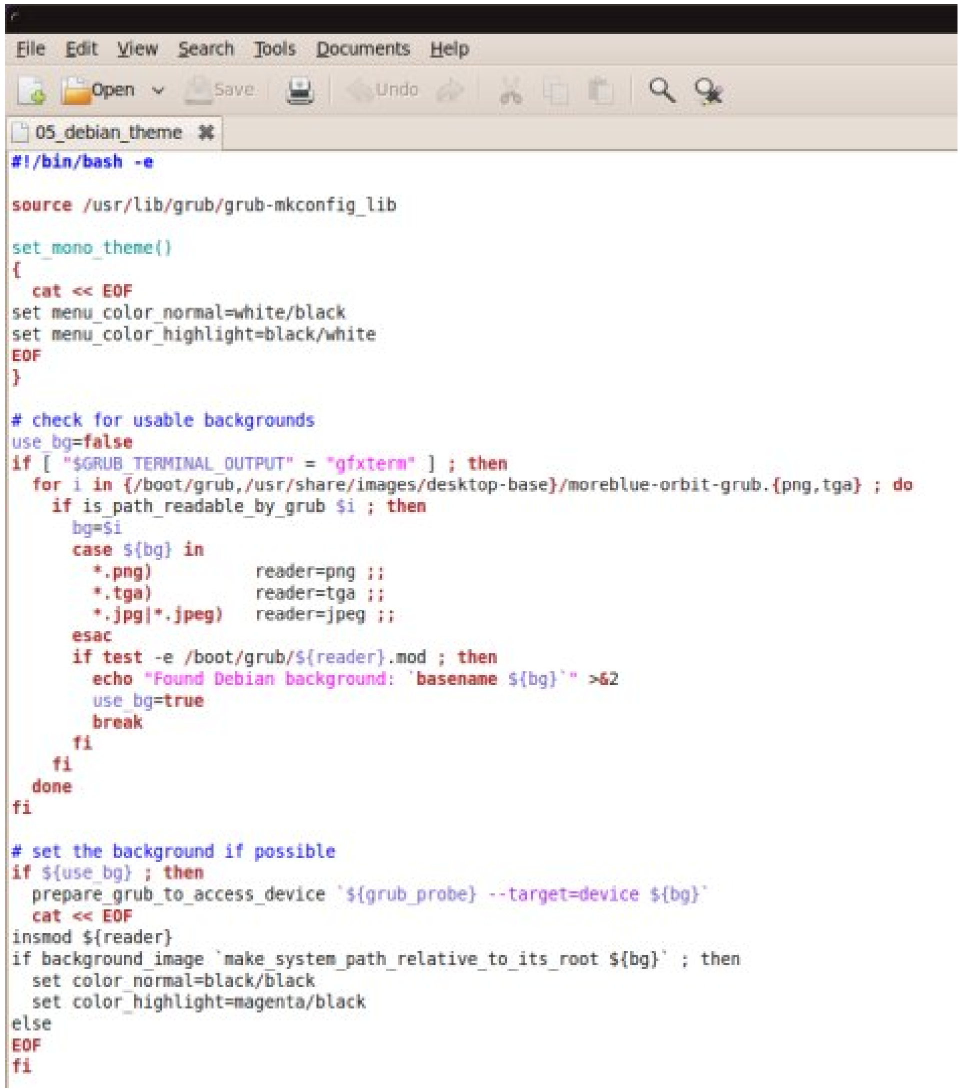
GRUB 2 works like this: /etc/default/grub contains customization; /etc/grub.d/ scripts contain GRUB menu information and operating system boot scripts. When the update-grub command is run, it reads the contents of the grub file and the grub.d scripts and creates the grub.cfg file.

That's all. To change the grub.cfg file, you need to edit the grub file or the scripts under grub.d. Scripts are meant to be executed. This means that they have the execute bit turned on. If you turn the execute bit off, they will not run.

This means that you can place as many files as you want into the grub.d directory, as long as they are not executable shell scripts that update-grub can read. If you want to use them, you will activate the executable bit, or vice versa, turn it off. Let's examine the scripts. For instance, 00_header and 05_debian_theme:

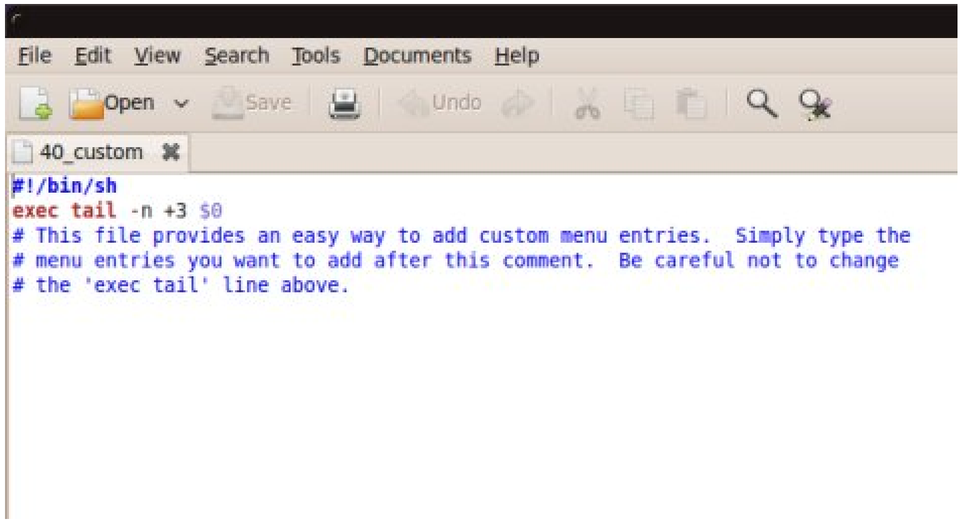


```
#! /bin/sh -e

# grub-mkconfig helper script.
# Copyright (C) 2006,2007,2008,2009  Free Software Foundation, Inc.
#
# GRUB 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 3 of the License, or
# (at your option) any later version.
#
# GRUB 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 GRUB.  If not, see <http://www.gnu.org/licenses/>.

transform="s,x,x,"

prefix=/usr
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
grub_prefix=`echo /boot/grub | sed ${transform}`

. ${libdir}/grub/grub-mkconfig_lib

# Do this as early as possible, since other commands might depend on it.
# (e.g. the `loadfont' command might need lvm or raid modules)
for i in ${GRUB_PRELOAD_MODULES} ; do
  echo "insmod $i"
done

if [ "x${GRUB_DEFAULT}" = "x" ] ; then GRUB_DEFAULT=0 ; fi
if [ "x${GRUB_DEFAULT}" = "xsaved" ] ; then GRUB_DEFAULT='${saved_entry}' ; fi
if [ "x${GRUB_TIMEOUT}" = "x" ] ; then GRUB_TIMEOUT=5 ; fi
if [ "x${GRUB_GFXMODE}" = "x" ] ; then GRUB_GFXMODE=640x480 ; fi
```



```
#!/bin/bash -e

source /usr/lib/grub/grub-mkconfig_lib

set_mono_theme()
{
  cat << EOF
set menu_color_normal=white/black
set menu_color_highlight=black/white
EOF
}

# check for usable backgrounds
use_bg=false
if [ "$GRUB_TERMINAL_OUTPUT" = "gfxterm" ] ; then
  for i in {/boot/grub,/usr/share/images/desktop-base}/moreblue-orbit-grub.{png,tga} ; do
    if is_path_readable_by_grub $i ; then
      bg=$i
      case ${bg} in
        *.png)          reader=png ;;
        *.tga)          reader=tga ;;
        *.jpg|*.jpeg)   reader=jpeg ;;
      esac
      if test -e /boot/grub/${reader}.mod ; then
        echo "Found Debian background: `basename ${bg}`" >&2
        use_bg=true
        break
      fi
    fi
  done
fi

# set the background if possible
if ${use_bg} ; then
  prepare_grub_to_access_device `${grub_probe} --target=device ${bg}`
  cat << EOF
insmod ${reader}
if background_image `make_system_path_relative_to_its_root ${bg}` ; then
  set color_normal=black/black
  set color_highlight=magenta/black
else
EOF
fi
```

40_custom makes a little more sense, but it still does not tell us how we can customize the scripts. Don't worry, we will soon learn everything.



```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
```

Add new GRUB script

To add a new boot option, you will have to follow a basic syntax:

Create a new file that has a XX_ prefix in the name, where XX is a sequence of numbers. If you want the new entry to be placed above others, use lower numbers, if you want it to be placed below others, use higher numbers.

For example, 11_something will be placed after the default entries by the operating system, whereas 08_something will be placed before the 10_linux entries. The next step is to write the actual content. Here's a sample:

```
#!/bin/sh -e
echo "Some string"
cat << EOF
menuentry "Something" {
set root=(hdX,Y)
-- boot parameters --
}
EOF
```

Let's examine the file. It's a shell script, as declared in the first line.

echo "Some string" is a string that you will see when running update-grub. If you do not want to see the echo command printed, you can redirect it to standard error or /dev/null:

```
echo "Some string" > &2
```

Example: echo "Adding openSUSE 11.2"

cat << EOF defines the start of the actual boot entry.

menuentry "Something" is the name that will show in the menu. Example: Linux.

set root=(hdX,Y) - we're back to old school, setting the root device.

Critical! GRUB 2 uses **PARTITION** notation that starts with 1 and not 0 like GRUB legacy! This is terribly important to remember!

In other words, devices are still numbered from 0, but partitions start with 1. For example, this means that sda1 is now (hd0,1) and NOT (hd0,0) as before!

-- boot parameters -- will really change from one OS to another. In Linux, you may want to use something like:

```
linux /boot/vmlinuz
initrd /boot/initrd.img
```

But in Windows, you would probably use:

```
chainloader (hdX,Y)+1
```

Therefore, a complete script example would look something like:

```
#!/bin/sh -e
echo "Adding my custom Linux to GRUB 2"
cat << EOF
menuentry "My custom Linux" {
set root=(hd0,5)
linux /boot/vmlinuz
initrd /boot/initrd.img
}
EOF
```

Or for Windows, something like:

```
#!/bin/sh -e
echo "Adding Windows 8 to GRUB 2 menu"
cat << EOF
menuentry "Windows 8" {
set root=(hd0,1)
chainloader (hd0,1)+1
}
EOF
```

EOF ends the GRUB entry.

Now we have a file ready. But we need to make it executable.

```
chmod +x XX_new_os_script
```

GRUB 2 commands

One more thing I'd like to emphasize here is the data contained in the cat << EOF section. As we've seen just now, the cat command defines the start of the code in the script that will be added to the GRUB menu literally and NOT interpreted by the shell. In other words, anything that goes between cat << EOF and EOF are GRUB commands.

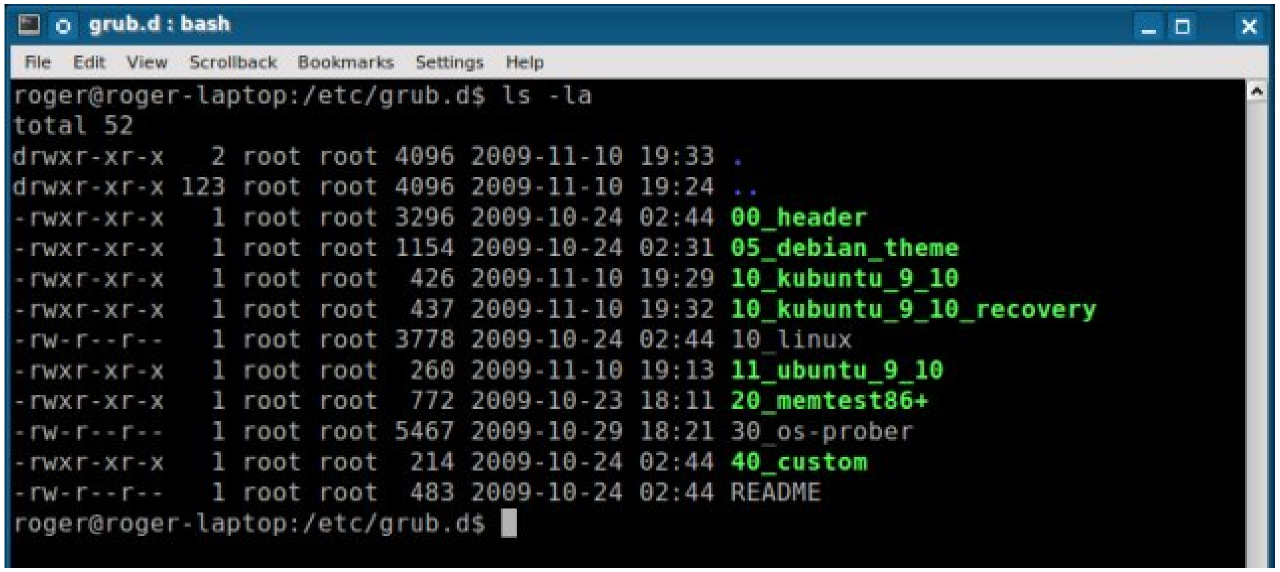
We have used a number of different commands in this script. Some of these you may have seen before in GRUB legacy, some may appear new and strange. Not to worry, there's a full list, including a comparison to GRUB legacy. Some of the commands have been replaced and others added. For more details, please check GRUB 2 command list.

Update GRUB

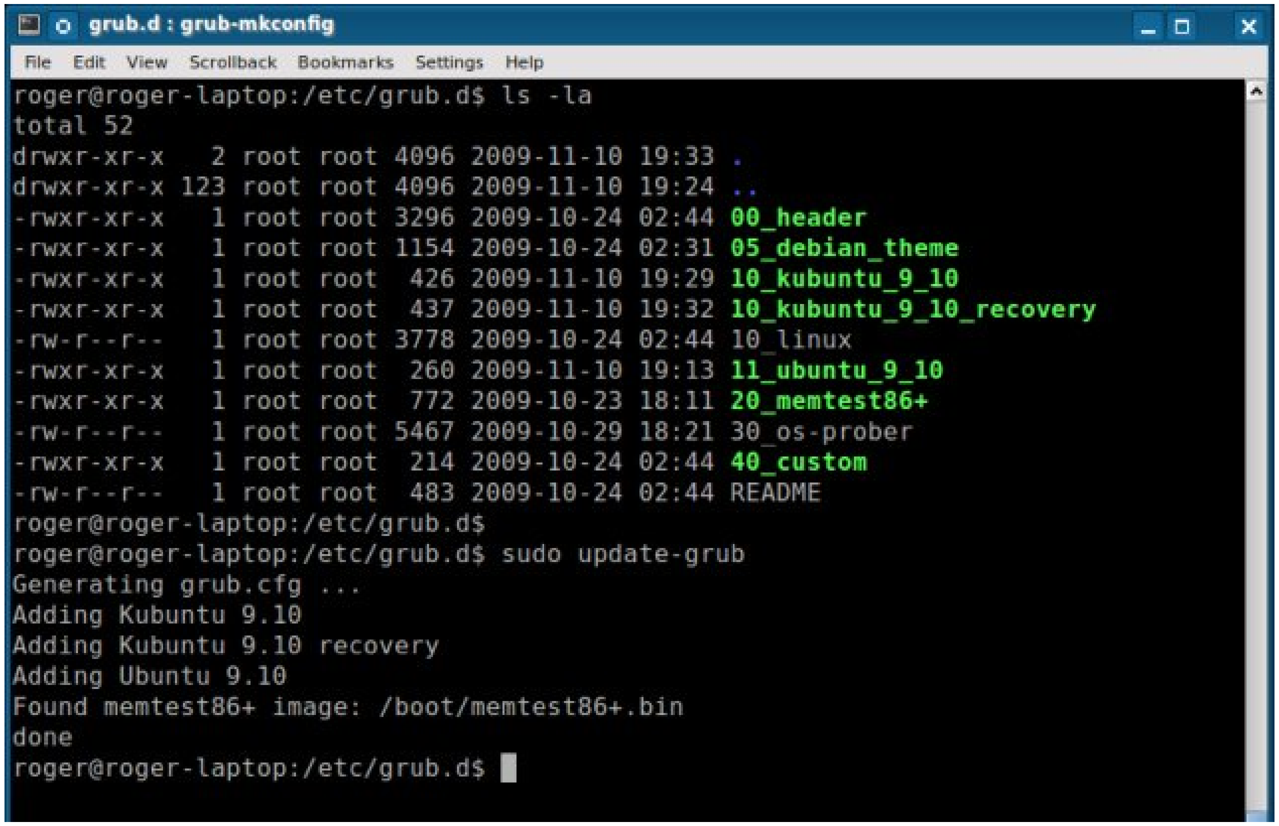
The new script is in place, but the GRUB menu (grub.cfg) has not been updated yet. We need to run the update-grub command to make it happen.

```
update-grub
```

Here's an example from a dual-boot configuration, which we will examine more deeply later on. I've added two Kubuntu entries and one Ubuntu entry to the Ubuntu GRUB menu. I have done this in order to get rid of the default 10_linux, which is ugly and uses generic kernel names in the titles, plus it calls both Ubuntu and Kubuntu the same. But let's move slowly. Our first task is to add the new scripts. After we ascertain that they boot fine, we will get rid of the duplicates.

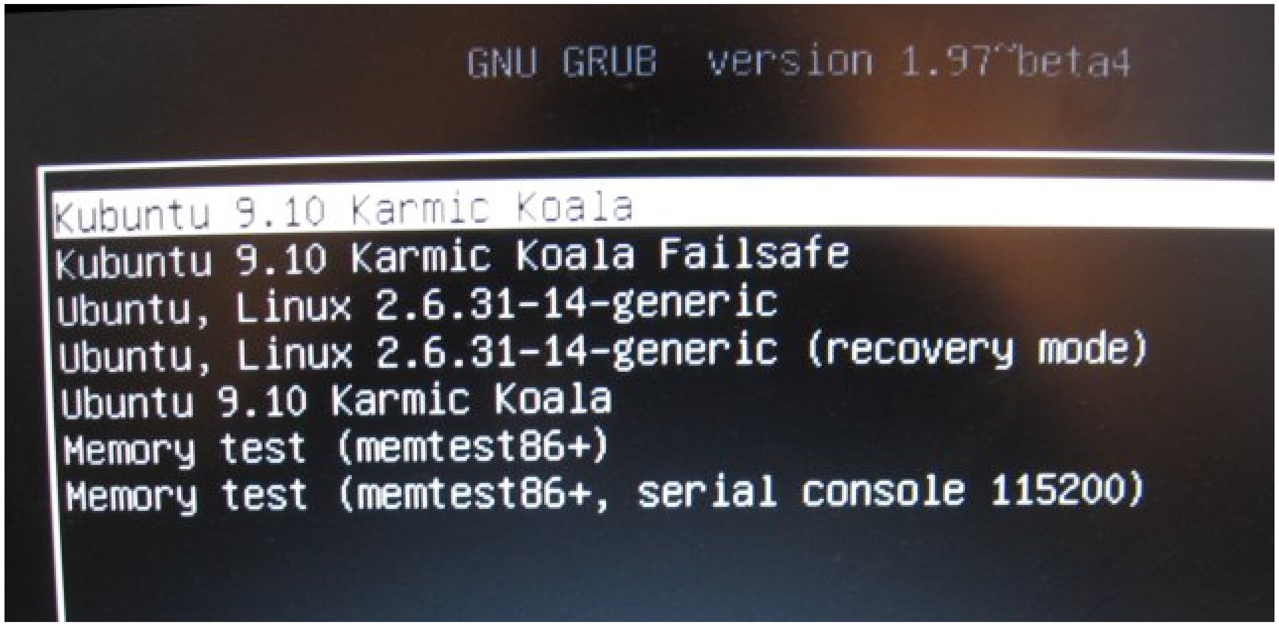


```
roger@roger-laptop:/etc/grub.d$ ls -la
total 52
drwxr-xr-x   2 root root 4096 2009-11-10 19:33 .
drwxr-xr-x 123 root root 4096 2009-11-10 19:24 ..
-rwxr-xr-x   1 root root 3296 2009-10-24 02:44 00_header
-rwxr-xr-x   1 root root 1154 2009-10-24 02:31 05_debian_theme
-rwxr-xr-x   1 root root  426 2009-11-10 19:29 10_kubuntu_9_10
-rwxr-xr-x   1 root root  437 2009-11-10 19:32 10_kubuntu_9_10_recovery
-rw-r--r--   1 root root 3778 2009-10-24 02:44 10_linux
-rwxr-xr-x   1 root root  260 2009-11-10 19:13 11_ubuntu_9_10
-rwxr-xr-x   1 root root  772 2009-10-23 18:11 20_memtest86+
-rw-r--r--   1 root root 5467 2009-10-29 18:21 30_os-prober
-rwxr-xr-x   1 root root  214 2009-10-24 02:44 40_custom
-rw-r--r--   1 root root  483 2009-10-24 02:44 README
roger@roger-laptop:/etc/grub.d$
```



```
roger@roger-laptop:/etc/grub.d$ ls -la
total 52
drwxr-xr-x   2 root root 4096 2009-11-10 19:33 .
drwxr-xr-x 123 root root 4096 2009-11-10 19:24 ..
-rwxr-xr-x   1 root root 3296 2009-10-24 02:44 00_header
-rwxr-xr-x   1 root root 1154 2009-10-24 02:31 05_debian_theme
-rwxr-xr-x   1 root root  426 2009-11-10 19:29 10_kubuntu_9_10
-rwxr-xr-x   1 root root  437 2009-11-10 19:32 10_kubuntu_9_10_recovery
-rw-r--r--   1 root root 3778 2009-10-24 02:44 10_linux
-rwxr-xr-x   1 root root  260 2009-11-10 19:13 11_ubuntu_9_10
-rwxr-xr-x   1 root root  772 2009-10-23 18:11 20_memtest86+
-rw-r--r--   1 root root 5467 2009-10-29 18:21 30_os-prober
-rwxr-xr-x   1 root root  214 2009-10-24 02:44 40_custom
-rw-r--r--   1 root root  483 2009-10-24 02:44 README
roger@roger-laptop:/etc/grub.d$
roger@roger-laptop:/etc/grub.d$ sudo update-grub
Generating grub.cfg ...
Adding Kubuntu 9.10
Adding Kubuntu 9.10 recovery
Adding Ubuntu 9.10
Found memtest86+ image: /boot/memtest86+.bin
done
roger@roger-laptop:/etc/grub.d$
```

After rebooting, we have this - I apologize for the uncomeliness:

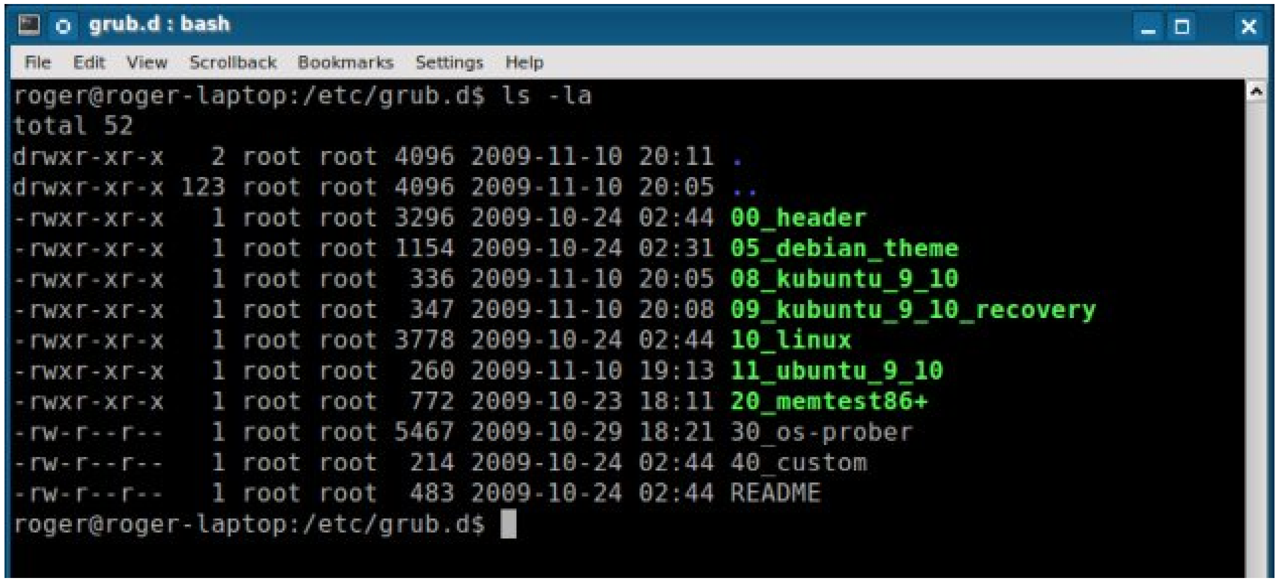


```
GNU GRUB  version 1.97~beta4

Kubuntu 9.10 Karmic Koala
Kubuntu 9.10 Karmic Koala Failsafe
Ubuntu, Linux 2.6.31-14-generic
Ubuntu, Linux 2.6.31-14-generic (recovery mode)
Ubuntu 9.10 Karmic Koala
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)
```

Once we are using the new scripts boot properly, we can then chmod -x the 10_linux and get rid of the generic 2.6.31-14 entries, keeping our menu nice and dandy.

Change script boot order by changing numbers

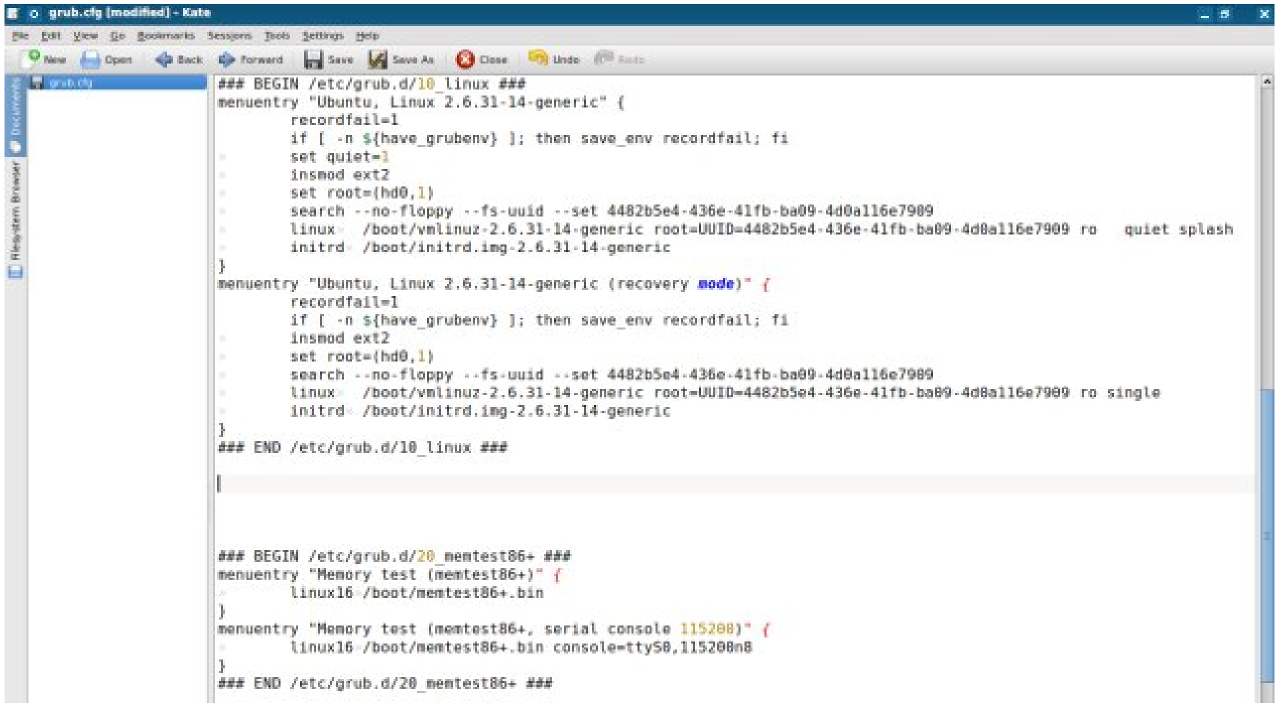
You can also change the numbers if you want. For instance, I changed the Kubuntu entries to 08 and 09, to make them boot first, before default entries and the custom Ubuntu script.



```
roger@roger-laptop:/etc/grub.d$ ls -la
total 52
drwxr-xr-x   2 root root 4096 2009-11-10 20:11 .
drwxr-xr-x 123 root root 4096 2009-11-10 20:05 ..
-rwxr-xr-x   1 root root 3296 2009-10-24 02:44 00_header
-rwxr-xr-x   1 root root 1154 2009-10-24 02:31 05_debian_theme
-rwxr-xr-x   1 root root  336 2009-11-10 20:05 08_kubuntu_9_10
-rwxr-xr-x   1 root root  347 2009-11-10 20:08 09_kubuntu_9_10_recovery
-rwxr-xr-x   1 root root 3778 2009-10-24 02:44 10_linux
-rwxr-xr-x   1 root root  260 2009-11-10 19:13 11_ubuntu_9_10
-rwxr-xr-x   1 root root  772 2009-10-23 18:11 20_memtest86+
-rw-r--r--   1 root root 5467 2009-10-29 18:21 30_os-prober
-rw-r--r--   1 root root  214 2009-10-24 02:44 40_custom
-rw-r--r--   1 root root  483 2009-10-24 02:44 README
roger@roger-laptop:/etc/grub.d$
```

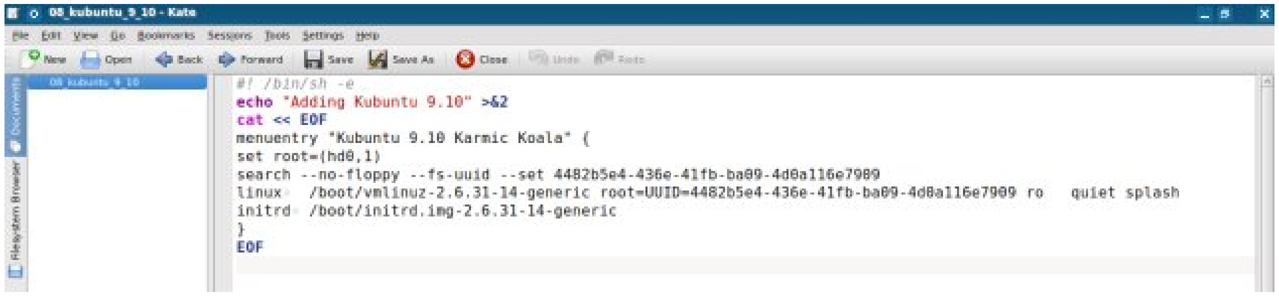
Replacing default entries

This is something you may want to do, just as we have shown above. In order to make sure your new scripts boot correctly, you can refer to a working example of grub.cfg to make sure you use the right linux and initrd lines. Here's an example of what I have:


grub.cfg [modified] - Kate
File Edit View Go Bookmarks Sessions Tools Settings Help
New Open Back Forward Save Save As Close Undo Redo
grub.cfg
Documents
Filesystem Browser

```
### BEGIN /etc/grub.d/10_linux ###
menuentry "Ubuntu, Linux 2.6.31-14-generic" {
        recordfail=1
        if [ -n ${have_grubenv} ]; then save_env recordfail; fi
        set quiet=1
        insmod ext2
        set root=(hd0,1)
        search --no-floppy --fs-uuid --set 4482b5e4-436e-41fb-ba09-4d0a116e7909
        linux   /boot/vmlinuz-2.6.31-14-generic root=UUID=4482b5e4-436e-41fb-ba09-4d0a116e7909 ro   quiet splash
        initrd  /boot/initrd.img-2.6.31-14-generic
}
menuentry "Ubuntu, Linux 2.6.31-14-generic (recovery mode)" {
        recordfail=1
        if [ -n ${have_grubenv} ]; then save_env recordfail; fi
        insmod ext2
        set root=(hd0,1)
        search --no-floppy --fs-uuid --set 4482b5e4-436e-41fb-ba09-4d0a116e7909
        linux   /boot/vmlinuz-2.6.31-14-generic root=UUID=4482b5e4-436e-41fb-ba09-4d0a116e7909 ro single
        initrd  /boot/initrd.img-2.6.31-14-generic
}
### END /etc/grub.d/10_linux ###




### BEGIN /etc/grub.d/20_memtest86+ ###
menuentry "Memory test (memtest86+)" {
        linux16 /boot/memtest86+.bin
}
menuentry "Memory test (memtest86+, serial console 115200)" {
        linux16 /boot/memtest86+.bin console=ttyS0,115200n8
}
### END /etc/grub.d/20_memtest86+ ###
```

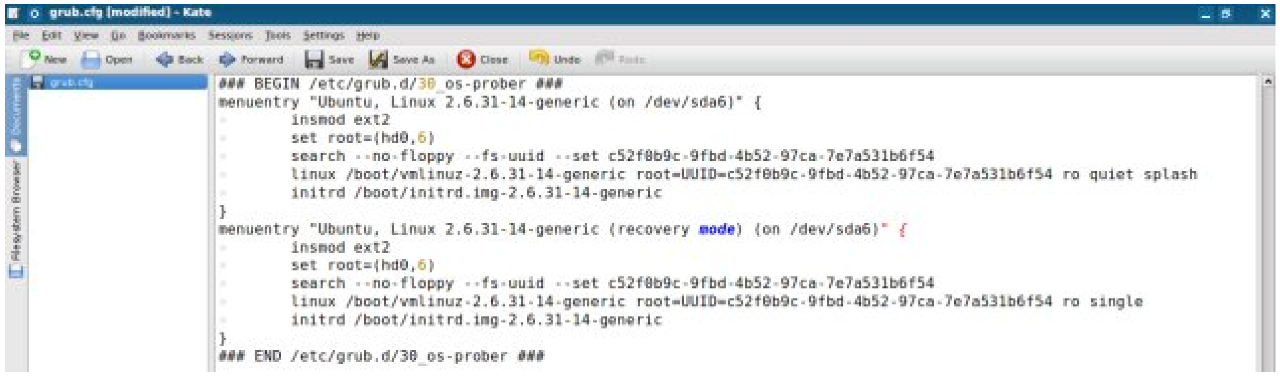
And accordingly, I've created the new script:


08_kubuntu_9_10 - Kate
File Edit View Go Bookmarks Sessions Tools Settings Help
New Open Back Forward Save Save As Close Undo Redo
08_kubuntu_9_10
Documents
Filesystem Browser

```
#! /bin/sh -e
echo "Adding Kubuntu 9.10" >&2
cat << EOF
menuentry "Kubuntu 9.10 Karmic Koala" {
set root=(hd0,1)
search --no-floppy --fs-uuid --set 4482b5e4-436e-41fb-ba09-4d0a116e7909
linux   /boot/vmlinuz-2.6.31-14-generic root=UUID=4482b5e4-436e-41fb-ba09-4d0a116e7909 ro   quiet splash
initrd  /boot/initrd.img-2.6.31-14-generic
}
EOF
```

OS Prober

OS Prober can also help you. It will find additional entries on your hard disks and add them to the menu. You can use the added information to create your own scripts. Again, refer to the grub.cfg file, os-prober section, for more data:


grub.cfg [modified] - Kate
File Edit View Go Bookmarks Sessions Tools Settings Help
New Open Back Forward Save Save As Close Undo Redo
grub.cfg
Documents
Filesystem Browser

```
### BEGIN /etc/grub.d/30_os-prober ###
menuentry "Ubuntu, Linux 2.6.31-14-generic (on /dev/sda6)" {
        insmod ext2
        set root=(hd0,6)
        search --no-floppy --fs-uuid --set c52f0b9c-9fbd-4b52-97ca-7e7a531b6f54
        linux /boot/vmlinuz-2.6.31-14-generic root=UUID=c52f0b9c-9fbd-4b52-97ca-7e7a531b6f54 ro quiet splash
        initrd /boot/initrd.img-2.6.31-14-generic
}
menuentry "Ubuntu, Linux 2.6.31-14-generic (recovery mode) (on /dev/sda6)" {
        insmod ext2
        set root=(hd0,6)
        search --no-floppy --fs-uuid --set c52f0b9c-9fbd-4b52-97ca-7e7a531b6f54
        linux /boot/vmlinuz-2.6.31-14-generic root=UUID=c52f0b9c-9fbd-4b52-97ca-7e7a531b6f54 ro single
        initrd /boot/initrd.img-2.6.31-14-generic
}
### END /etc/grub.d/30_os-prober ###
```

Reinstall GRUB

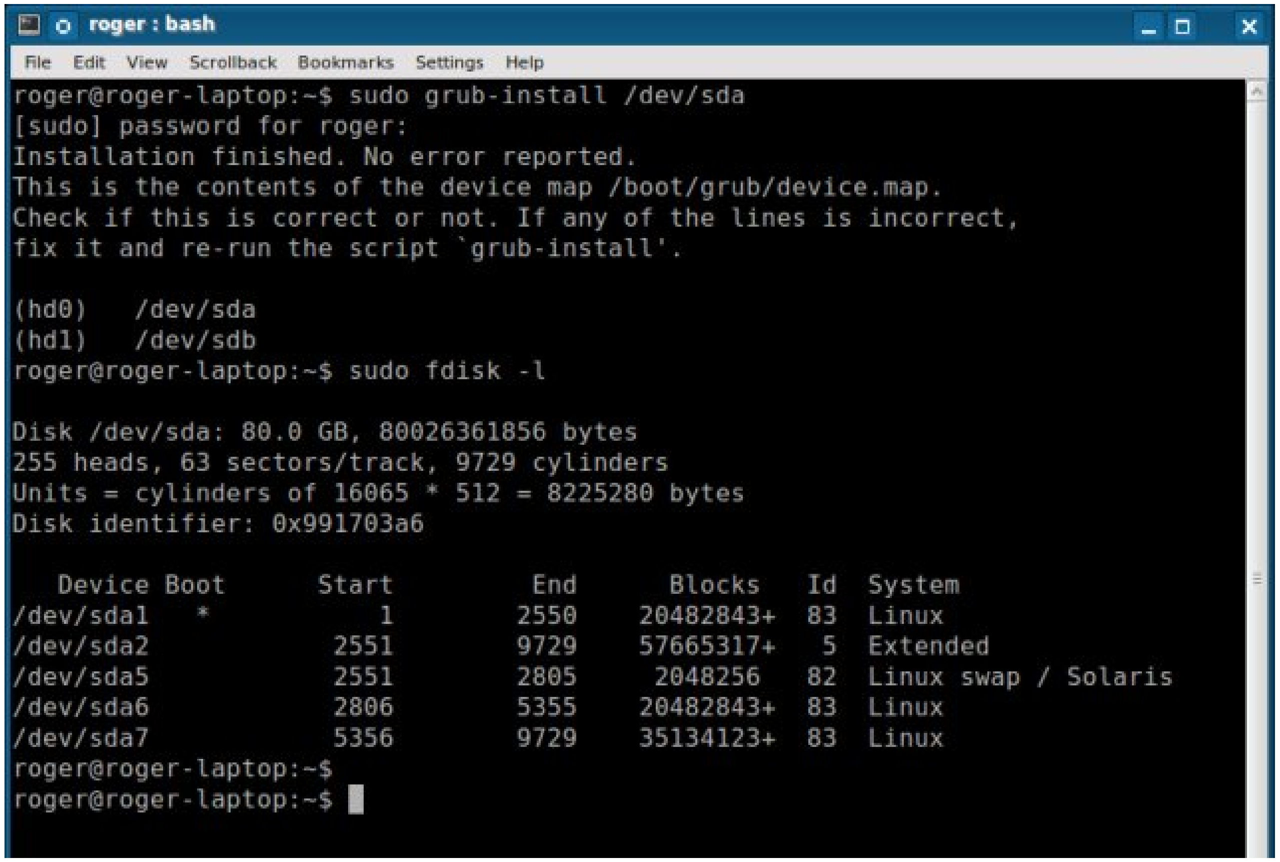
GRUB 2 can be installed even while you are booted in the OS. You do not need a live environment for that. Just execute the grub-install command against the device or the partition you desire.

```
grub-install <target>
```

<target> can be /dev/hda, /dev/sdb, /dev/sdc4, and so forth. It is important that you pay attention to the output produced by the command. If you have external disks connected at the time you run the command, it will add these disks to the list of mapped devices. There should be no big harm in that, but if some entries are incorrect, remove them.

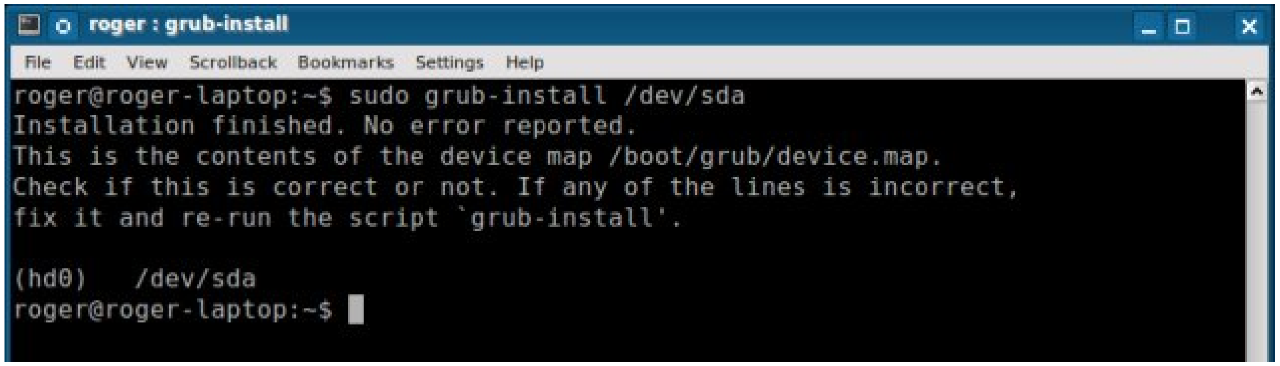
grub-install reports incorrect devices

Here's an example of that:



```
roger@roger-laptop:~$ sudo grub-install /dev/sda
[sudo] password for roger:
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'.

(hd0)   /dev/sda
(hd1)   /dev/sdb
roger@roger-laptop:~$ sudo fdisk -l

Disk /dev/sda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x991703a6

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        2550    20482843+  83  Linux
/dev/sda2            2551        9729    57665317+   5  Extended
/dev/sda5            2551        2805     2048256   82  Linux swap / Solaris
/dev/sda6            2806        5355    20482843+  83  Linux
/dev/sda7            5356        9729    35134123+  83  Linux
roger@roger-laptop:~$
roger@roger-laptop:~$
```

You will need to open the /boot/grub/device.map file, delete the wrong entry, in our case the one pointing to /dev/sdb and rerun the grub-install command.



```
roger@roger-laptop:~$ sudo grub-install /dev/sda
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'.

(hd0)   /dev/sda
roger@roger-laptop:~$
```

Mini summary

That's it for now. You have the tools you need to start working. Still, to make things clearer, I've prepared a short summary that concludes the introduction and basic usage sections. Here we go. GRUB 2 has three main parts:

1. /etc/default/grub - the file containing GRUB 2 menu settings.
2. /etc/grub.d/ - the directory containing GRUB 2 menu creating scripts.
3. /boot/grub/grub.cfg - the GRUB 2 configuration file, not editable.

update-grub command reads the /etc/grub.d directory and looks for executable scripts inside it. The scripts are read, in the order of their numbering, and written into the grub.cfg file, along with the menu settings read from the /etc/default/grub file.

Boot entries come from several sources - the default that comes with the distribution, other operating systems probed on the connected disks and custom scripts written by the user, following a strict syntax. The scripts are written as shell (sh).

You can add/remove entries by simply chmod-ing the scripts; no need to delete them. GRUB 2 can be reinstalled anytime you want, even while booted in the OS. Good so far? Excellent. You see, it's rather simple. Now, let's see a few real-life cases. Afterwards, we will customize GRUB 2 and learn how to recover from serious errors and misconfigurations.

Real life multi-boot cases

In this section, we will discuss several common multi-boot cases that the average user might encounter, including GRUB and GRUB 2 mix, Windows and Linux side by side, chainloading, and some others.

Some of the experiments were conducted on real physical systems, so some screenshots of boot menus come from actual photos, so they may look slightly twisted and less appealing that you would like, but this is in order to reassure you that working with GRUB 2 is as simple as any other administrative task. It just requires some patience and care.

Dual boot: two operating systems with GRUB 2

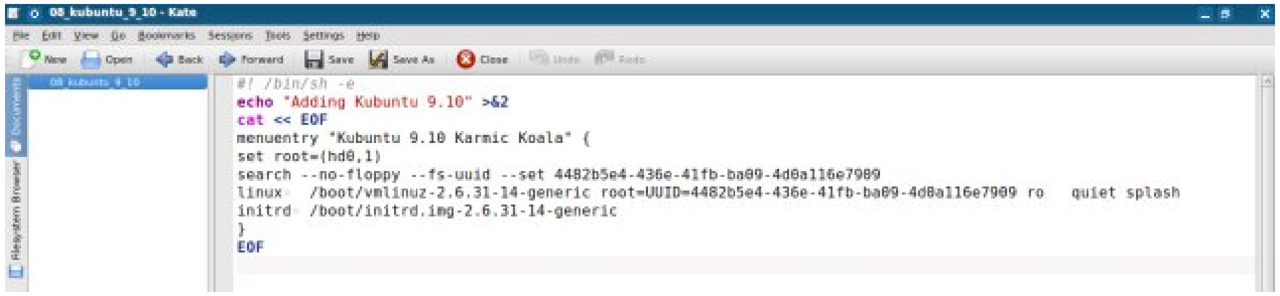
This is probably the simplest dual-boot configuration. Both operating systems use GRUB 2, so they can easily interact with one another. The two systems are Ubuntu 9.10 and Kubuntu 9.10, both formatted with Ext4.

Ubuntu was installed first, with GRUB 2 installed to the MBR of /dev/sda. Kubuntu was installed second, again to MBR, overwriting the previous instance. However, the OS probe script found and added Ubuntu successfully. If you've read my [Kubuntu](https://www.dedoimedo.com) tutorial, you will notice that Kubuntu entries are also labeled Ubuntu, which can be confusing if you have several, different Ubuntu entries on your machine. In this section, we will:

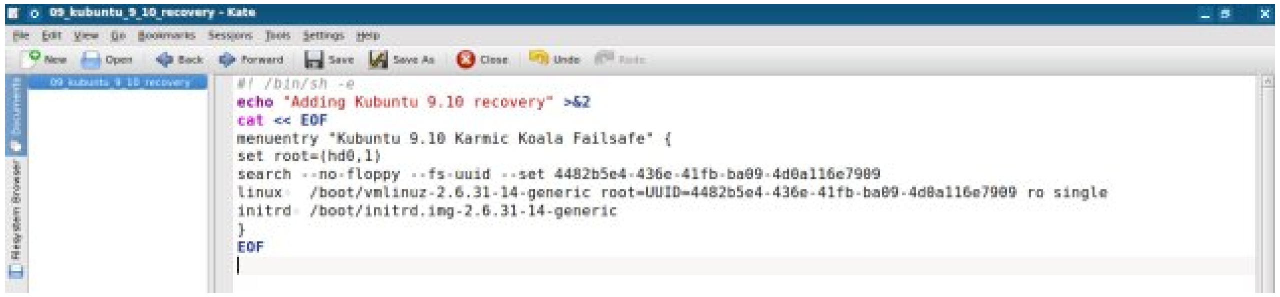
- Learn how to edit existing entries and make them more presentable.
- Add new entries and reorder existing ones.

This will give us the initial clues into how GRUB 2 works and what users have to do to make it work. After that, we will examine bootloading GRUB 2 ready systems from the legacy GRUB and vice versa. And then, we will add Windows, too.

We have seen snippets of this testcase earlier. We have our default entries, which do not look well and want them replaced. So we refer to grub.cfg file, examine the syntax used by 10_linux and we copy the relevant bits of data into our new scripts, called 08_9_10_kubuntu and 09_9_10_kubuntu_recovery. Just like we did earlier.

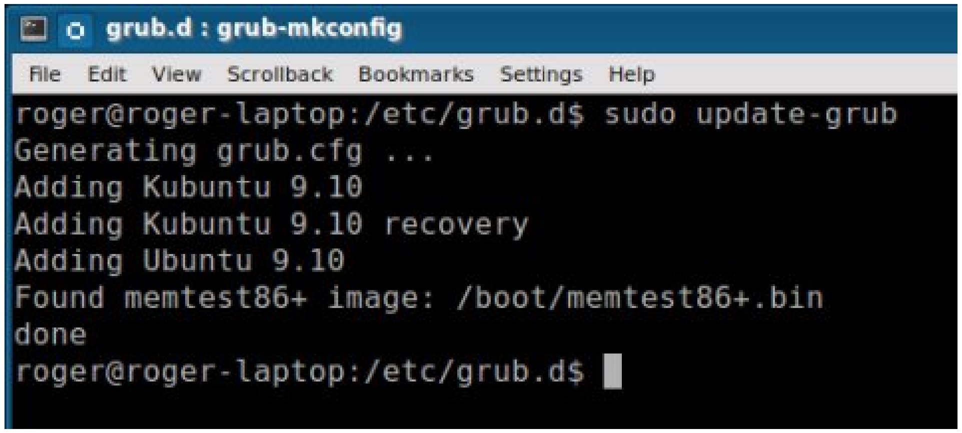

08_kubuntu_9_10 - Kate
File Edit View Go Bookmarks Sessions Tools Settings Help
New Open Back Forward Save Save As Close Undo Redo
08_kubuntu_9_10
Documents
Filesystem Browser

```
#! /bin/sh -e
echo "Adding Kubuntu 9.10" >&2
cat << EOF
menuentry "Kubuntu 9.10 Karmic Koala" {
set root=(hd0,1)
search --no-floppy --fs-uuid --set 4482b5e4-436e-41fb-ba09-4d0a116e7909
linux   /boot/vmlinuz-2.6.31-14-generic root=UUID=4482b5e4-436e-41fb-ba09-4d0a116e7909 ro   quiet splash
initrd  /boot/initrd.img-2.6.31-14-generic
}
EOF
```

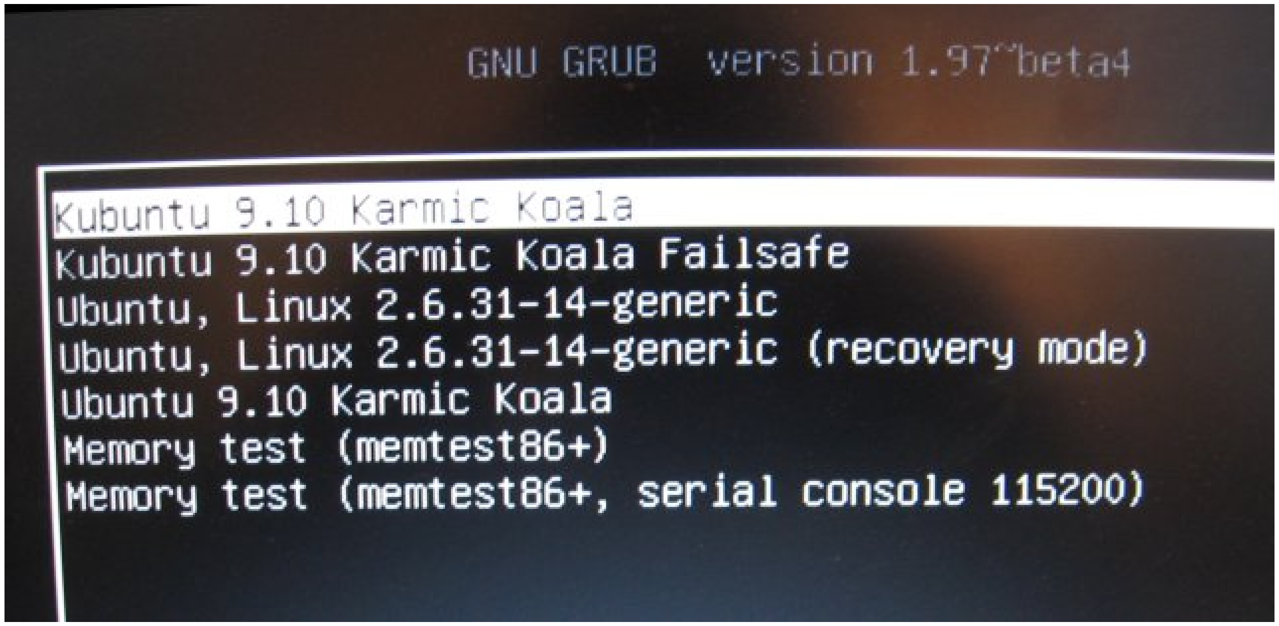

09_kubuntu_9_10_recovery - Kate
File Edit View Go Bookmarks Sessions Tools Settings Help
New Open Back Forward Save Save As Close Undo Redo
09_kubuntu_9_10_recovery
Documents
Filesystem Browser

```
#! /bin/sh -e
echo "Adding Kubuntu 9.10 recovery" >&2
cat << EOF
menuentry "Kubuntu 9.10 Karmic Koala Failsafe" {
set root=(hd0,1)
search --no-floppy --fs-uuid --set 4482b5e4-436e-41fb-ba09-4d0a116e7909
linux   /boot/vmlinuz-2.6.31-14-generic root=UUID=4482b5e4-436e-41fb-ba09-4d0a116e7909 ro single
initrd  /boot/initrd.img-2.6.31-14-generic
}
EOF
```

After that, we need to chmod +x the scripts and update the GRUB. Very simple.



```
roger@roger-laptop:/etc/grub.d$ sudo update-grub
Generating grub.cfg ...
Adding Kubuntu 9.10
Adding Kubuntu 9.10 recovery
Adding Ubuntu 9.10
Found memtest86+ image: /boot/memtest86+.bin
done
roger@roger-laptop:/etc/grub.d$
```



```
GNU GRUB  version 1.97~beta4

Kubuntu 9.10 Karmic Koala
Kubuntu 9.10 Karmic Koala Failsafe
Ubuntu, Linux 2.6.31-14-generic
Ubuntu, Linux 2.6.31-14-generic (recovery mode)
Ubuntu 9.10 Karmic Koala
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)
```

After rebooting and verifying that everything works, we can get rid of the 10_linux, having replaced its entries with the custom scripts. Job done, very easy!

Dual boot: two operating systems with GRUB legacy & GRUB 2 mix

This is a very interesting case. Let's say you have two operating systems, like Ubuntu 9.10, which uses the new GRUB 2 and boots from Ext4, and a more classic distro like openSUSE 11.2, which does use Ext4, but still boots the old GRUB.

We will learn how to:

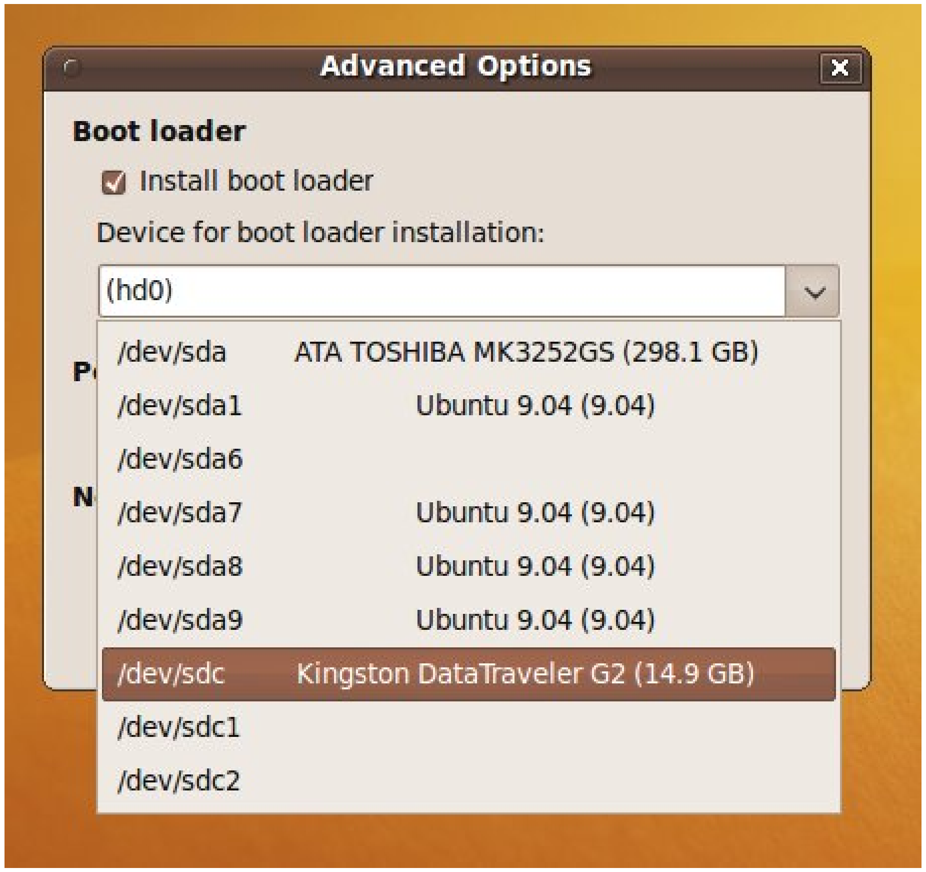
- Boot Ubuntu from the context of the openSUSE GRUB.
- Boot openSUSE from the context of the Ubuntu GRUB.
- Edit entries in both systems to make things work.

Decision: which one is default?

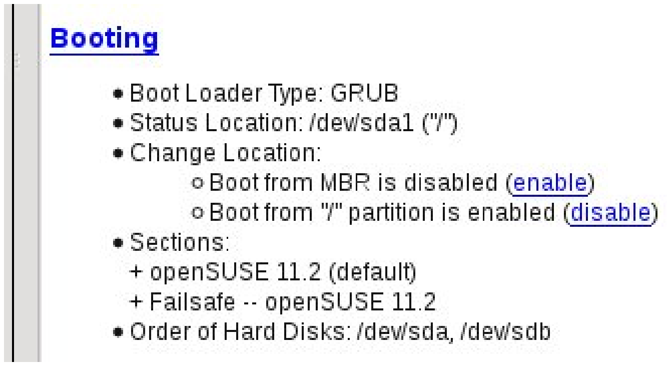
This is something you need to decide. You have two bootloader options and you need to choose the default one. If you're asking me, at the time being, you should use GRUB legacy as your default bootloader, because it is production quality and has known, established support channels. Running GRUB 2 will place you in a minority. Therefore, if you are going to use GRUB legacy for booting, then you should:

- Install GRUB into the MBR when installing the distribution that ships it, like openSUSE in our example. Pay attention to these details when installing the distro!
- Install GRUB 2 into the root partition where the distribution using it is installed, in this case Ubuntu. Pay attention to these details when installing the distro.

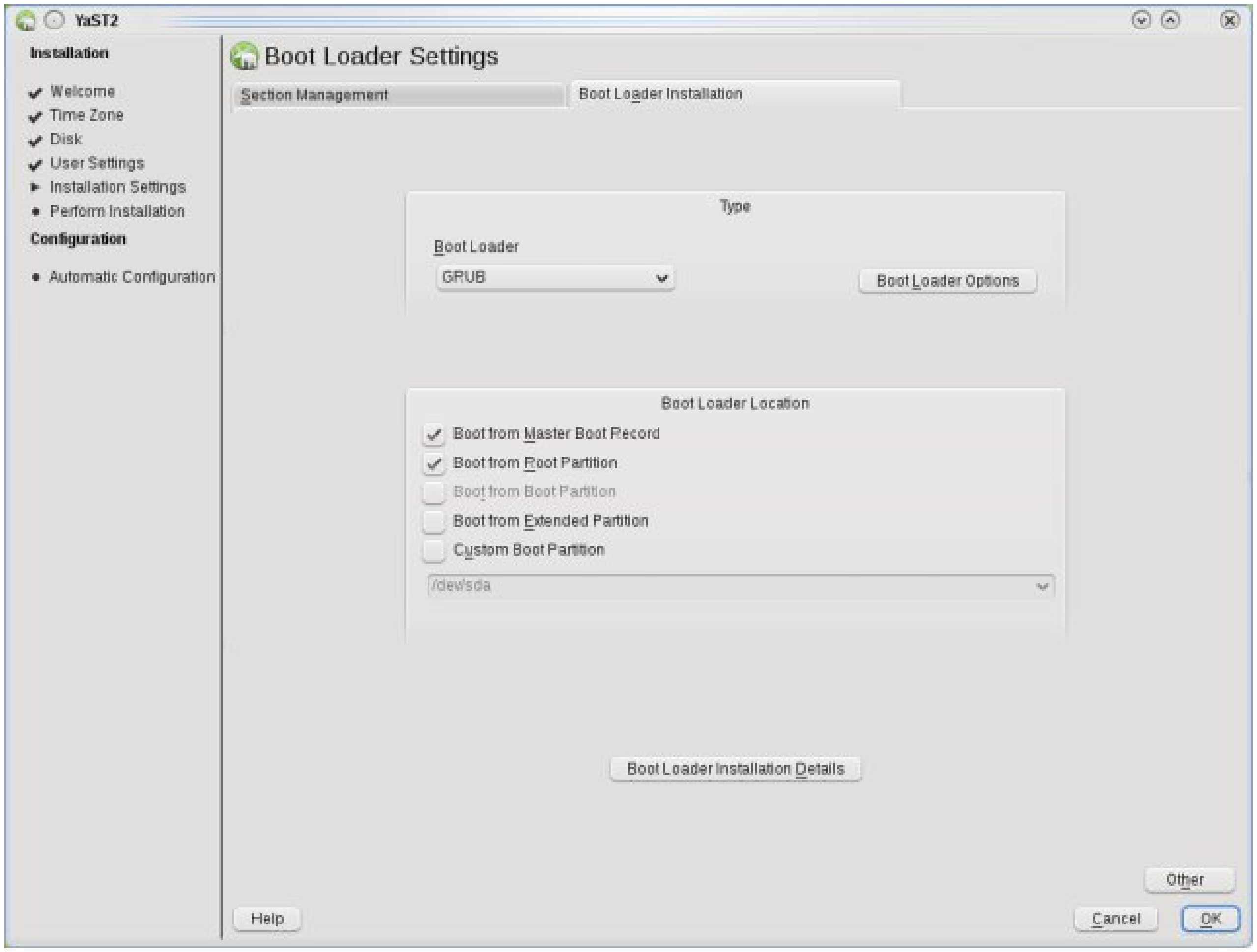
Otherwise, if you plan on using GRUB 2 as your default, you need to do things the other way around.



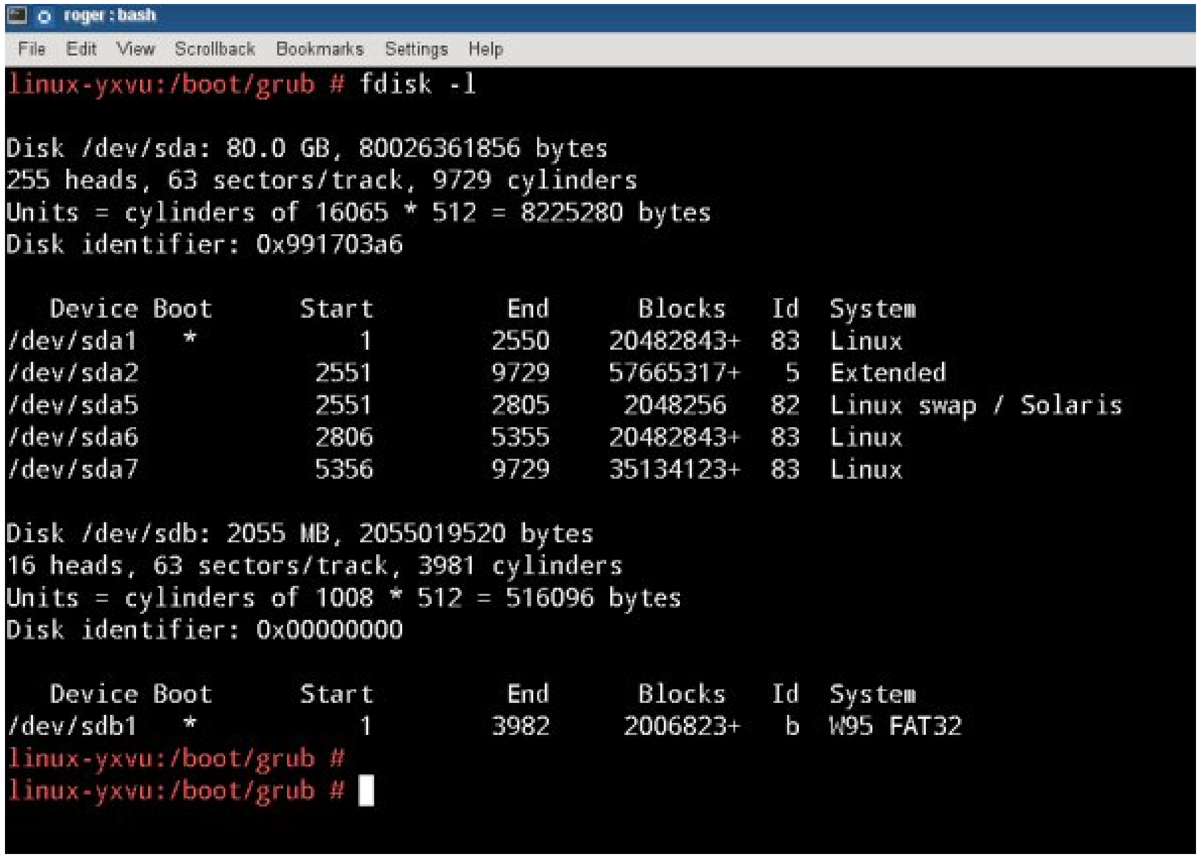
/dev/sda ATA TOSHIBA MK3252GS (298.1 GB)
/dev/sda1 Ubuntu 9.04 (9.04)
/dev/sda6
/dev/sda7 Ubuntu 9.04 (9.04)
/dev/sda8 Ubuntu 9.04 (9.04)
/dev/sda9 Ubuntu 9.04 (9.04)
/dev/sdc Kingston DataTraveler G2 (14.9 GB)
/dev/sdc1
/dev/sdc2


Booting

- Boot Loader Type: GRUB
- Status Location: /dev/sda1 ("/")
- Change Location:
  - Boot from MBR is disabled (enable)
  - Boot from "/" partition is enabled (disable)
- Sections:
  + openSUSE 11.2 (default)
  + Failsafe -- openSUSE 11.2
- Order of Hard Disks: /dev/sda, /dev/sdb



If you are not really sure, review the partition table.



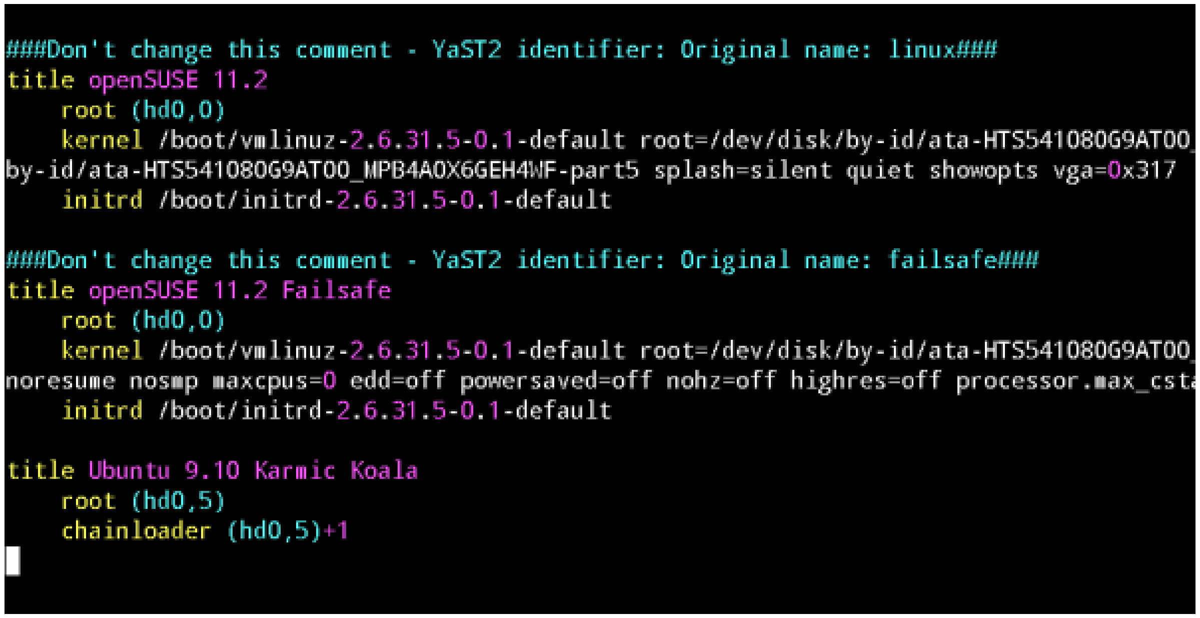
```
linux-yxvu:/boot/grub # fdisk -l

Disk /dev/sda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x991703a6

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        2550    20482843+  83  Linux
/dev/sda2            2551        9729    57665317+   5  Extended
/dev/sda5            2551        2805     2048256   82  Linux swap / Solaris
/dev/sda6            2806        5355    20482843+  83  Linux
/dev/sda7            5356        9729    35134123+  83  Linux

Disk /dev/sdb: 2055 MB, 2055019520 bytes
16 heads, 63 sectors/track, 3981 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1   *           1        3982     2006823+   b  W95 FAT32
linux-yxvu:/boot/grub #
linux-yxvu:/boot/grub #
```

In our example, Ubuntu was installed first, with root under /dev/sda6 and home under /dev/sda7. openSUSE was installed next, to /dev/sda1. Since we want to use the GRUB legacy bootloader, then:

- Ubuntu bootloader needs to be installed to /dev/sda6.
- openSUSE bootloader needs to be installed to MBR (/dev/sda).

Now that this is done, let's boot and see what happens. After you install openSUSE, you will notice that the lovely green GRUB menu contains only openSUSE entries. The reason is, GRUB legacy does not know how to handle the GRUB 2 layout directly and therefore cannot find and append entries to the menu. You will have to do this manually.

Boot into openSUSE and open the menu.lst file in a text editor. Back it up first! We do not really know yet how to work with GRUB 2 entries from within GRUB legacy. So we will assume that it's the same thing like Windows. We will chainload the other operating system, passing the command to the other bootloader. Will this work? I do not know, but let's try! We're using our GRUB experience and trying to adapt to the new situation.

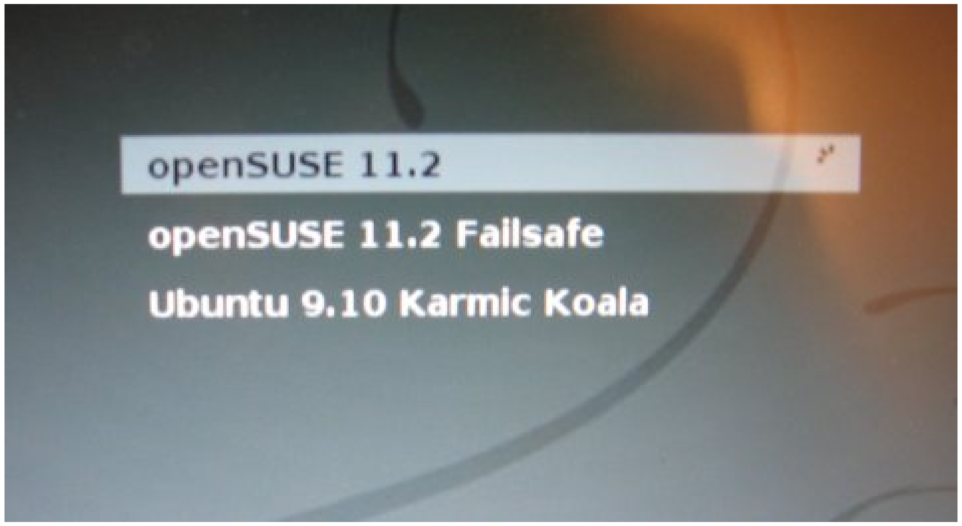


```
###Don't change this comment - YaST2 identifier: Original name: linux###
title openSUSE 11.2
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.31.5-0.1-default root=/dev/disk/by-id/ata-HTS541080G9AT00_
by-id/ata-HTS541080G9AT00_MPB4A0X6GEH4WF-part5 splash=silent quiet showopts vga=0x317
    initrd /boot/initrd-2.6.31.5-0.1-default

###Don't change this comment - YaST2 identifier: Original name: failsafe###
title openSUSE 11.2 Failsafe
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.31.5-0.1-default root=/dev/disk/by-id/ata-HTS541080G9AT00_
noresume nosmp maxcpus=0 edd=off powersaved=off nohz=off highres=off processor.max_csta
    initrd /boot/initrd-2.6.31.5-0.1-default

title Ubuntu 9.10 Karmic Koala
    root (hd0,5)
    chainloader (hd0,5)+1
```

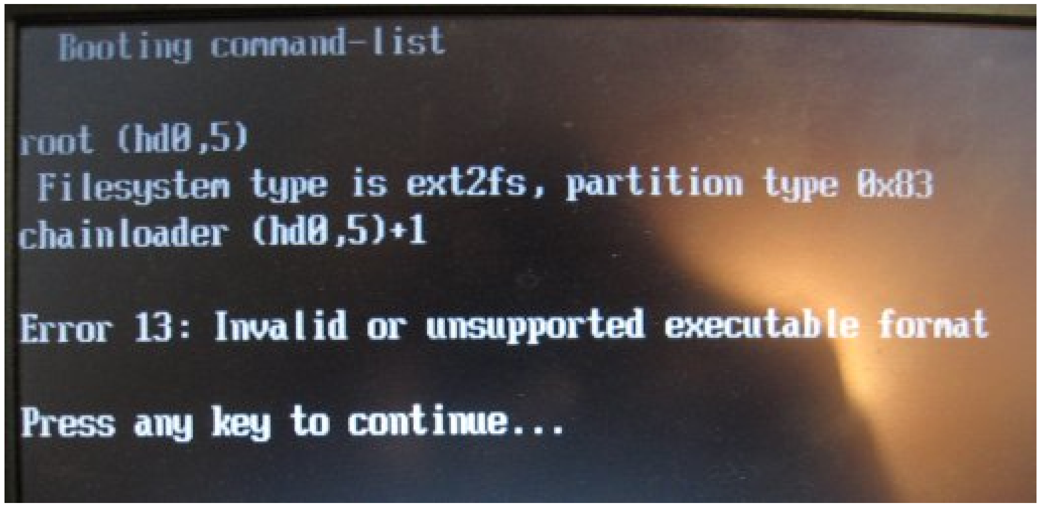
For those who cannot see what's in the image, we are adding the following entry to the menu.lst file:

```
title Ubuntu 9.10 Karmic Koala
root (hd0,5)
chainloader (hd0,5)+1
```

After booting, you will see the menu (once again, excuse the ugliness):



If you try to boot the Karmic Koala entry, you will hit this unwholesome GRUB error 13:



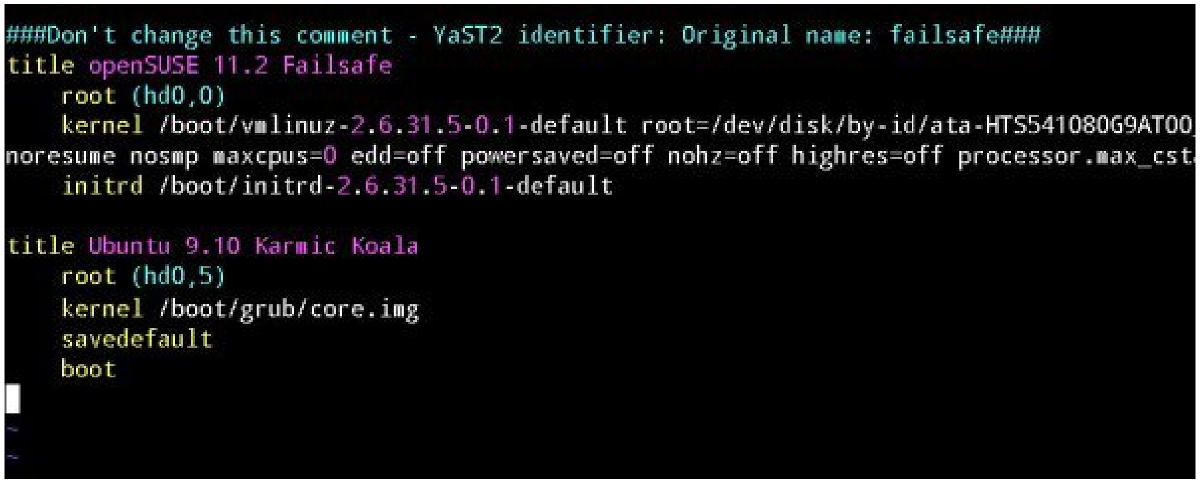
```
 Booting command-list

root (hd0,5)
 Filesystem type is ext2fs, partition type 0x83
chainloader (hd0,5)+1

Error 13: Invalid or unsupported executable format

Press any key to continue...
```

This kind of error is common for Linux users trying to boot Windows and Mac partitions. For Windows, the solution is to try to remap the partitions, hide partitions or make them active, which usually solves the problem.

Mac-wise, the problem is similar to what we are facing here. GRUB legacy that comes with openSUSE does not handle well the Ubuntu partitions + GRUB 2 mechanism. We will need a different method to get things done. For more information, check GRUB error messages.

Booting Ubuntu (with GRUB 2) from openSUSE the right way

Here's what you need to do to make things work. Go back to openSUSE and open the menu.lst file again. We need to make it look like this:



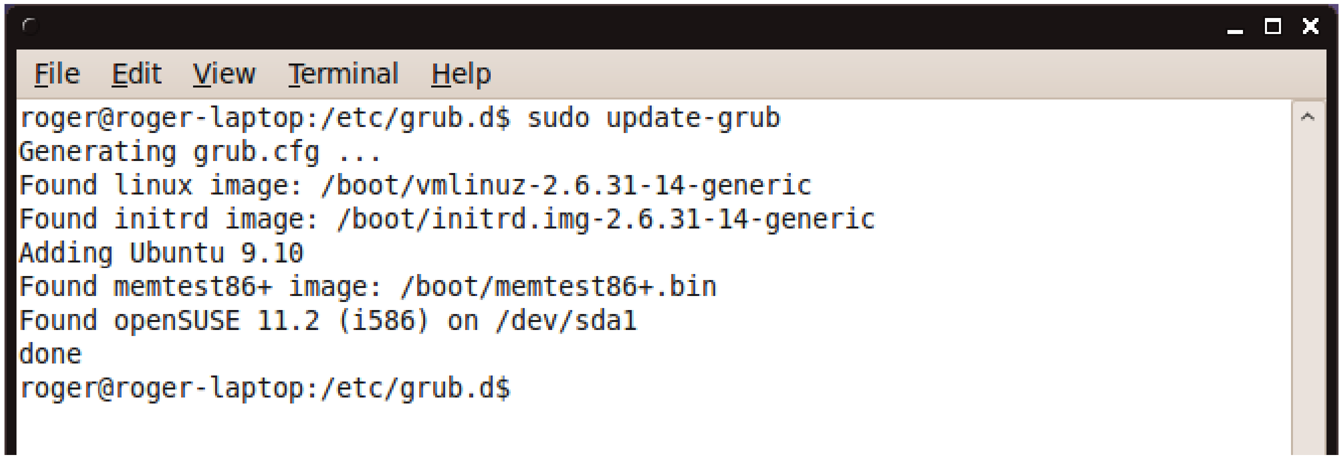
```
###Don't change this comment - YaST2 identifier: Original name: failsafe###
title openSUSE 11.2 Failsafe
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.31.5-0.1-default root=/dev/disk/by-id/ata-HTS541080G9AT00
noresume nosmp maxcpus=0 edd=off powersaved=off nohz=off highres=off processor.max_cst
    initrd /boot/initrd-2.6.31.5-0.1-default

title Ubuntu 9.10 Karmic Koala
    root (hd0,5)
    kernel /boot/grub/core.img
    savedefault
    boot

~
~
```

For those who cannot see what's in the image, we are adding the following entry to the menu.lst file:

```
title Ubuntu 9.10 Karmic Koala
root (hd0,5)
kernel /boot/grub/core.img
savedefault
boot
```

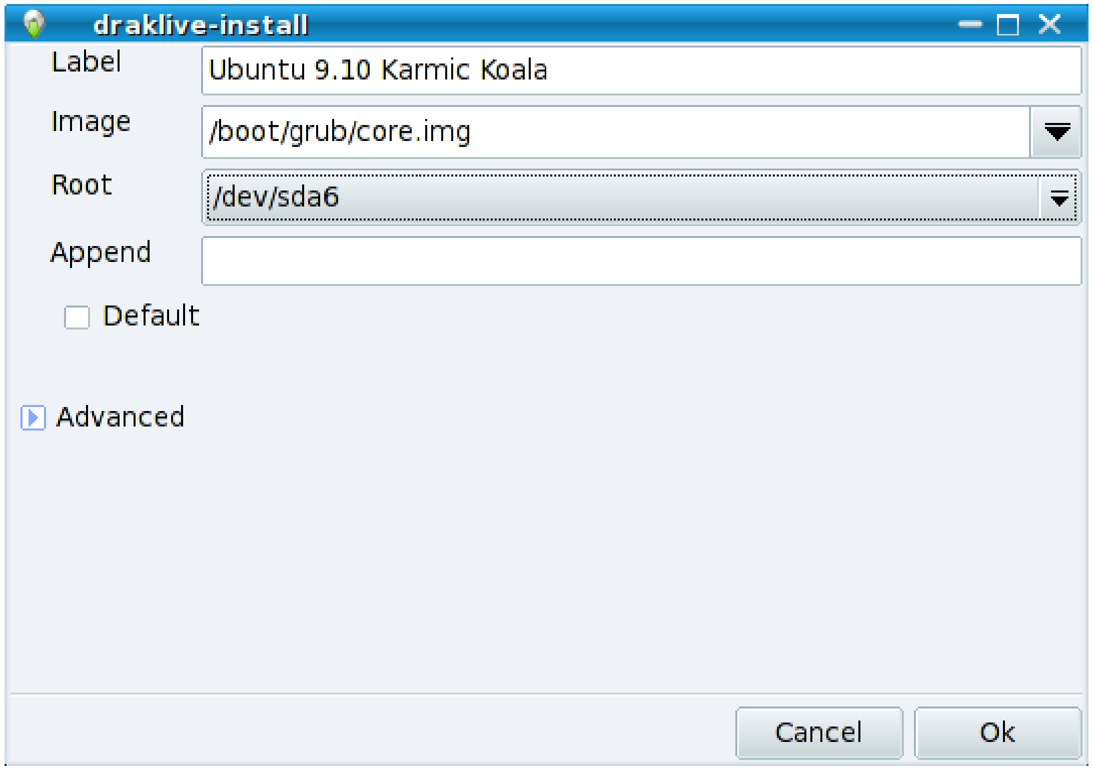
You may be wondering what we did here. The answer is, core.img file is a small kernel image that allows you to boot the right drivers and properly identify and initialize the real kernel. You can think of this file as a sort of a transitional initrd.img. After rebooting, you will have the GRUB 2 menu. Select the desired entry and boot.

Once inside Ubuntu, you can do a little trick if you want, that is, rerun the update-grub script. It will now pick up the installed openSUSE as well and add it to the menu. You will then have a recursive GRUB menu that calls the other that calls the other. This is not necessary, but can be fun and exercises the power of knowledge and control.



```
roger@roger-laptop:/etc/grub.d$ sudo update-grub
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-2.6.31-14-generic
Found initrd image: /boot/initrd.img-2.6.31-14-generic
Adding Ubuntu 9.10
Found memtest86+ image: /boot/memtest86+.bin
Found openSUSE 11.2 (i586) on /dev/sda1
done
roger@roger-laptop:/etc/grub.d$
```

Another interesting case is Mandriva 2010 and Ubuntu 9.10 dual boot. When installing Mandriva, you have the option to manually edit the GRUB bootloader menu even before the first boot. The truth is, many distributions let you configure the boot menu to some degree. If you know what you need, you can save time by setting up the menu entries during the installation. Here's what editing the GRUB menu following Mandriva installation looks like:



Label: Ubuntu 9.10 Karmic Koala
Image: /boot/grub/core.img
Root: /dev/sda6
Append:
Default [ ]

Basically, we have done the exact same thing we did with openSUSE. For more details, please take a look at the review above.

Dual boot: Windows 7 and Ubuntu

This will probably interest quite a few users. In general, it's the same as before. However, to spice things up a little, I have also added another element of interest here, that of the GRUB to GRUB 2 upgrade.

Test case: a machine dual booting Windows 7 and Ubuntu Jaunty, as shown in my new dual boot guide. Jaunty uses GRUB legacy, but it can be upgraded to GRUB 2, which is what we're going to do. So, we will first go through the upgrade procedure, paying attention to important details and then, we will make sure our Ubuntu and Windows installations boot fine.

Upgrade from GRUB legacy to GRUB 2

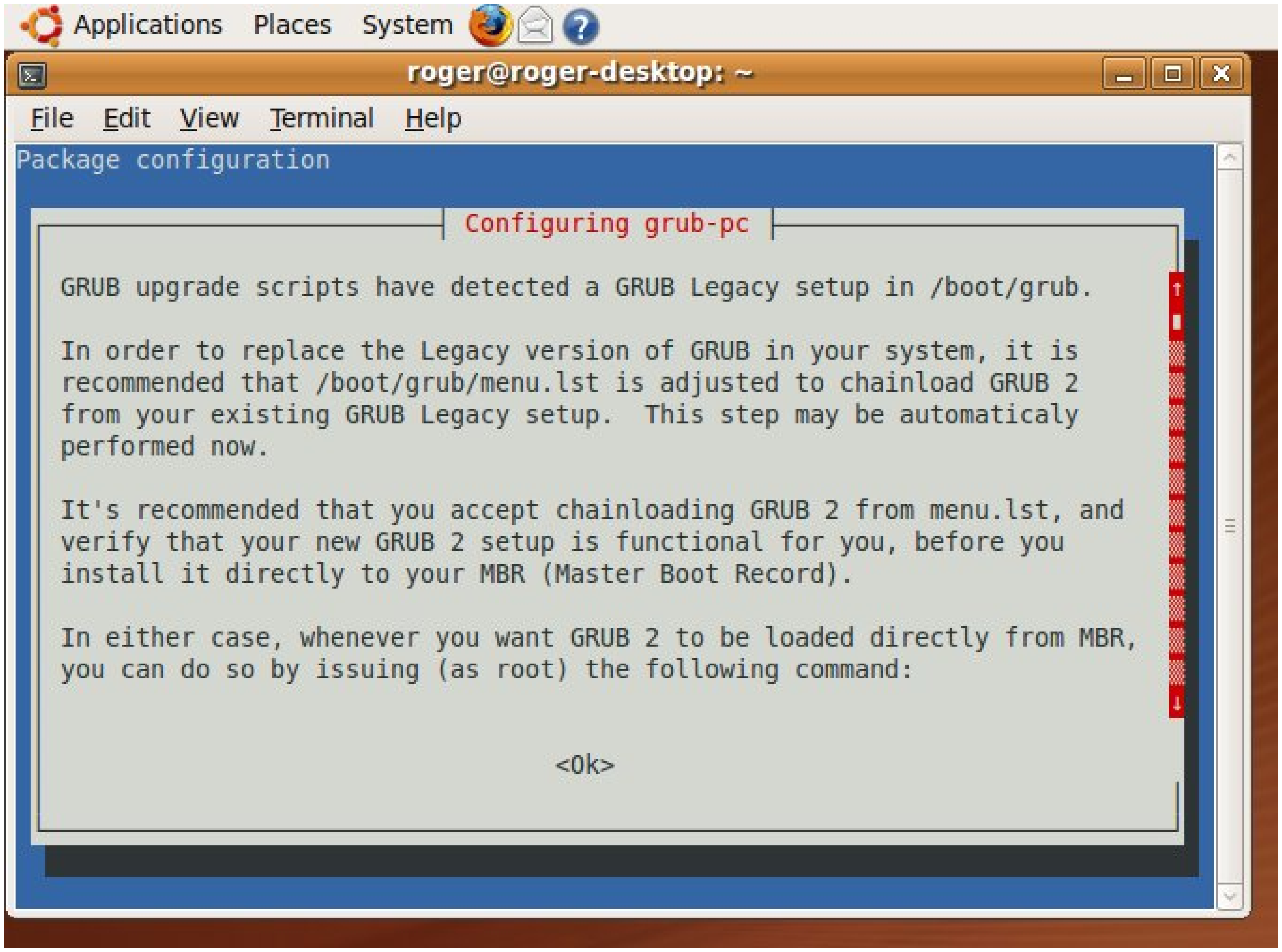
The first thing is to install the GRUB 2 package:

```
sudo apt-get install grub2
```

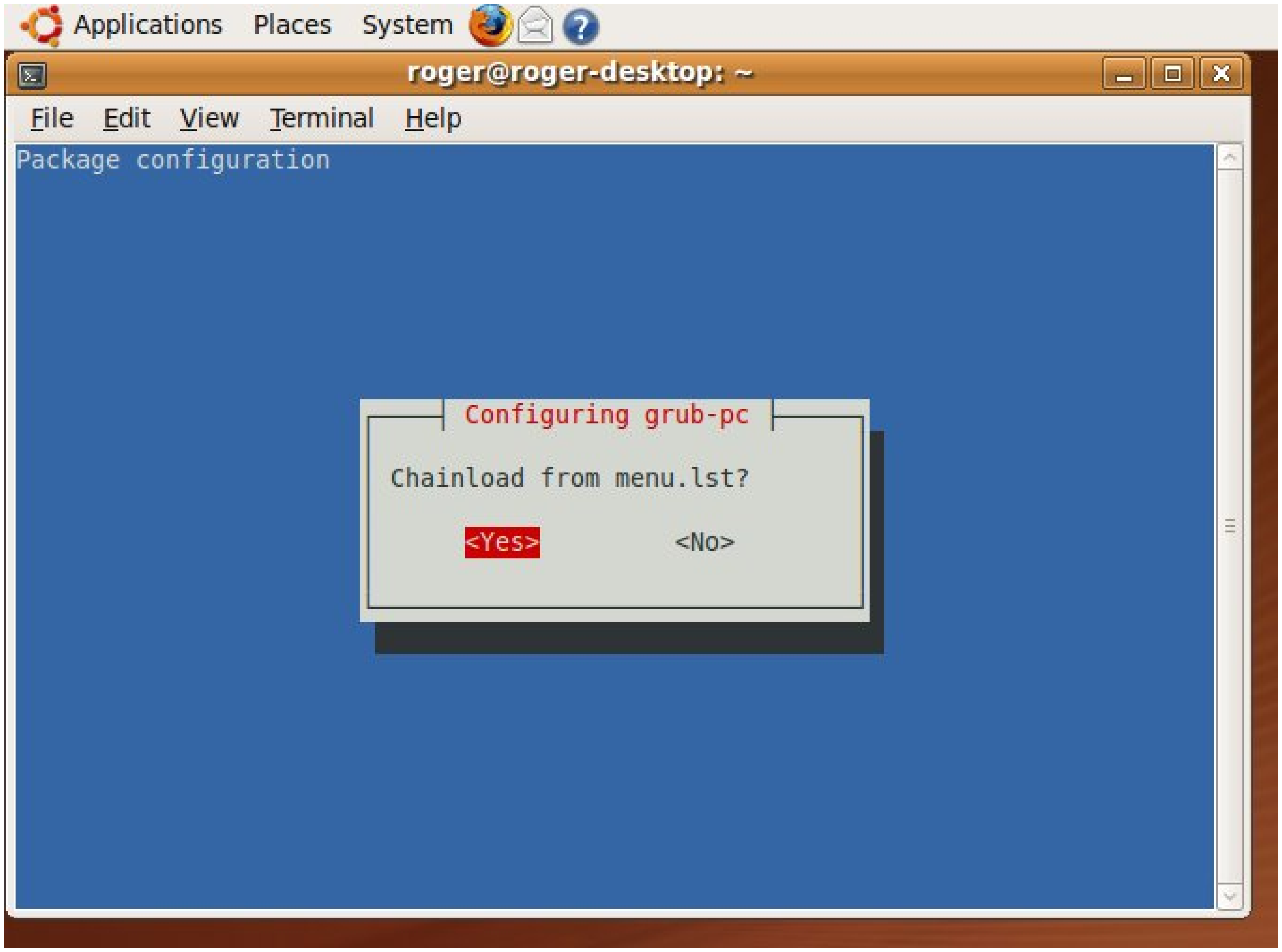
Watch the terminal. You will soon have to answer a few prompts.

Chainload first

The first thing you will have to decide upon is whether to install GRUB2 into MBR right away or test first by installing to the root partition where Ubuntu is installed and chainloading from GRUB legacy. We have done this earlier, by calling on the core.img file, so we know how to do this. For the sake of elegance, we won't skip this step.

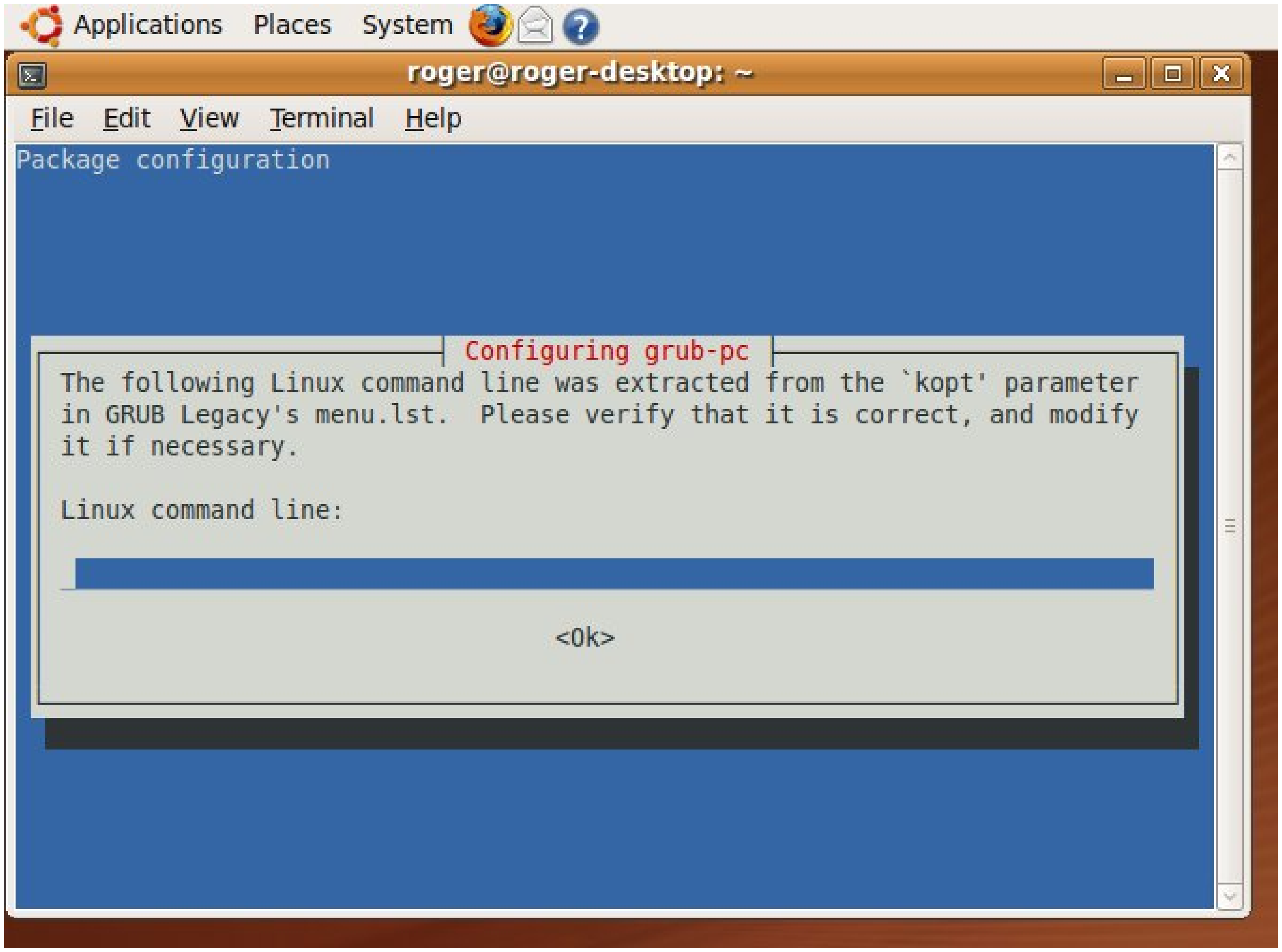


```
Package configuration

                    ┤ Configuring grub-pc ├

 GRUB upgrade scripts have detected a GRUB Legacy setup in /boot/grub.

 In order to replace the Legacy version of GRUB in your system, it is
 recommended that /boot/grub/menu.lst is adjusted to chainload GRUB 2
 from your existing GRUB Legacy setup.  This step may be automaticaly
 performed now.

 It's recommended that you accept chainloading GRUB 2 from menu.lst, and
 verify that your new GRUB 2 setup is functional for you, before you
 install it directly to your MBR (Master Boot Record).

 In either case, whenever you want GRUB 2 to be loaded directly from MBR,
 you can do so by issuing (as root) the following command:


                              <Ok>
```



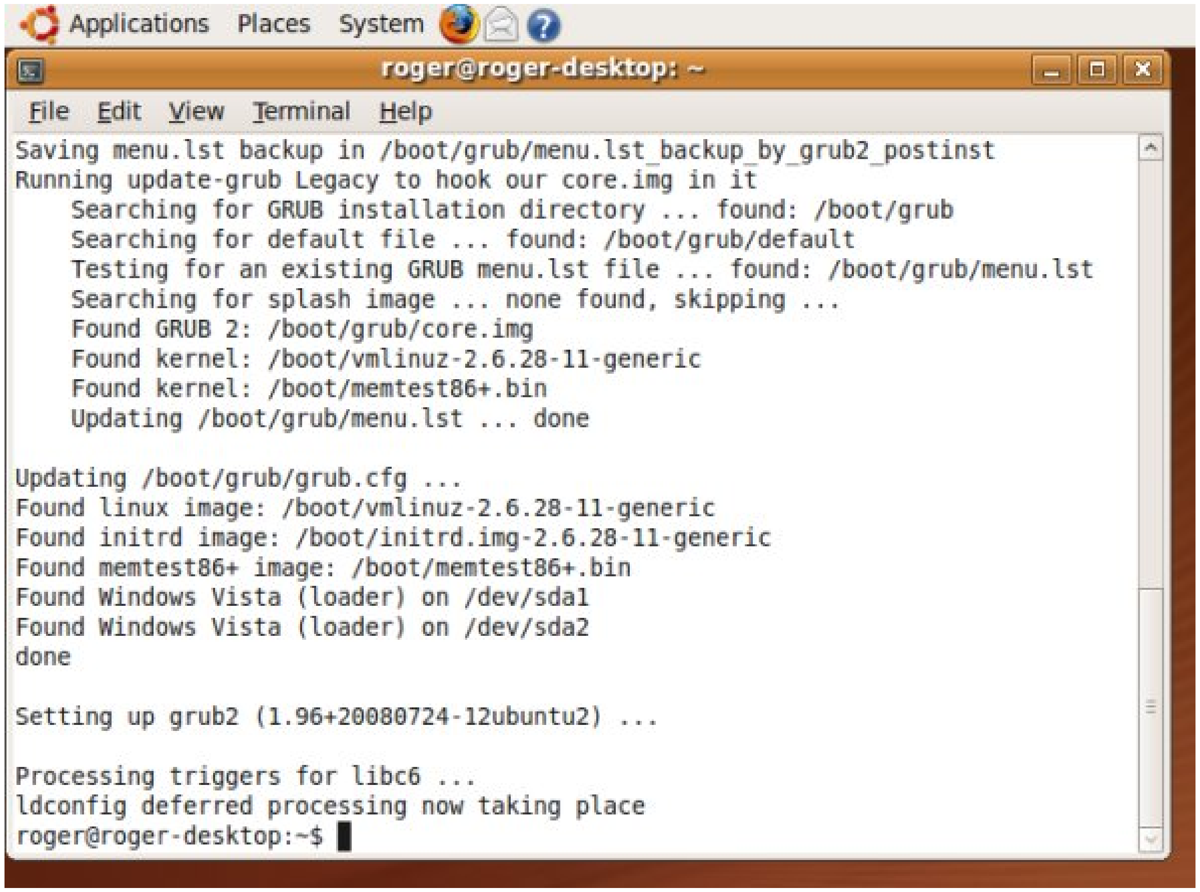
```
Package configuration

┤ Configuring grub-pc ├

Chainload from menu.lst?

<Yes>          <No>
```

The next step is to verify the command-line parameters. Most people won't need to make any change here.



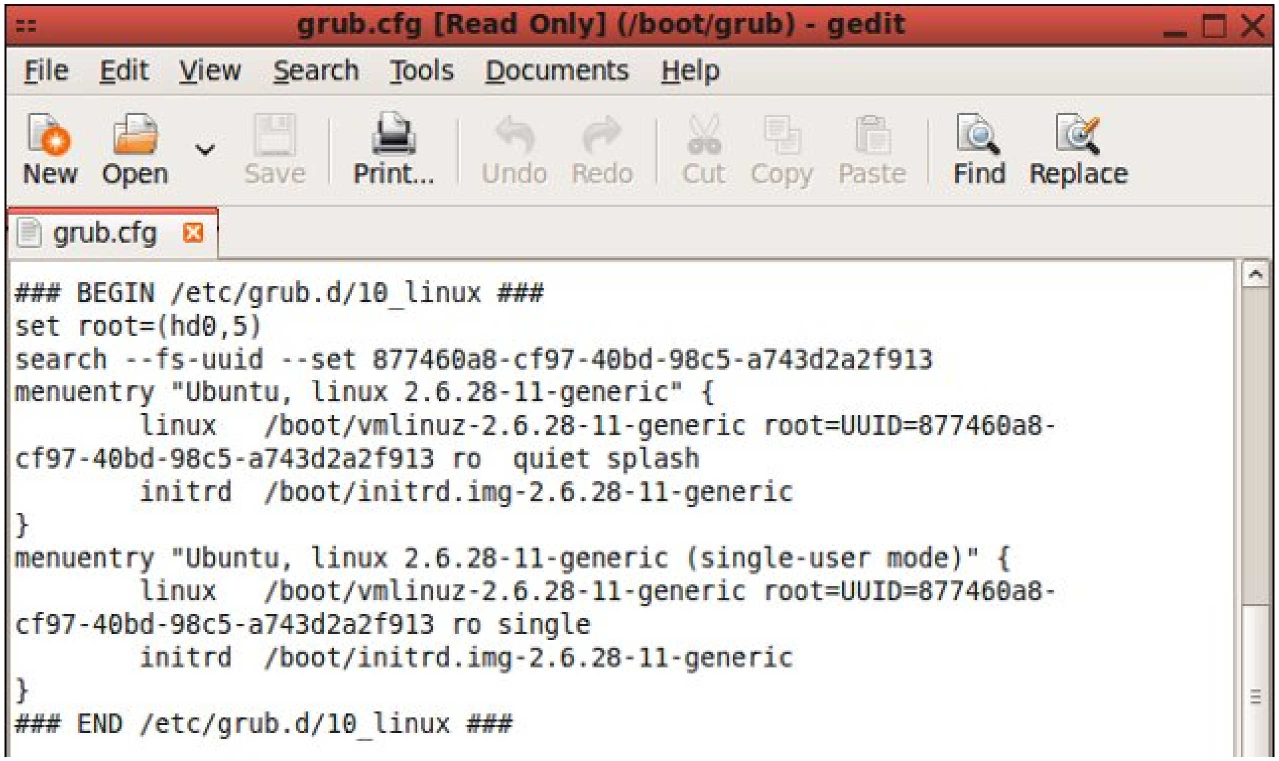
```
Package configuration

                 Configuring grub-pc
 The following Linux command line was extracted from the `kopt' parameter
 in GRUB Legacy's menu.lst.  Please verify that it is correct, and modify
 it if necessary.

 Linux command line:

 _

                              <Ok>
```

GRUB 2 is now installed:



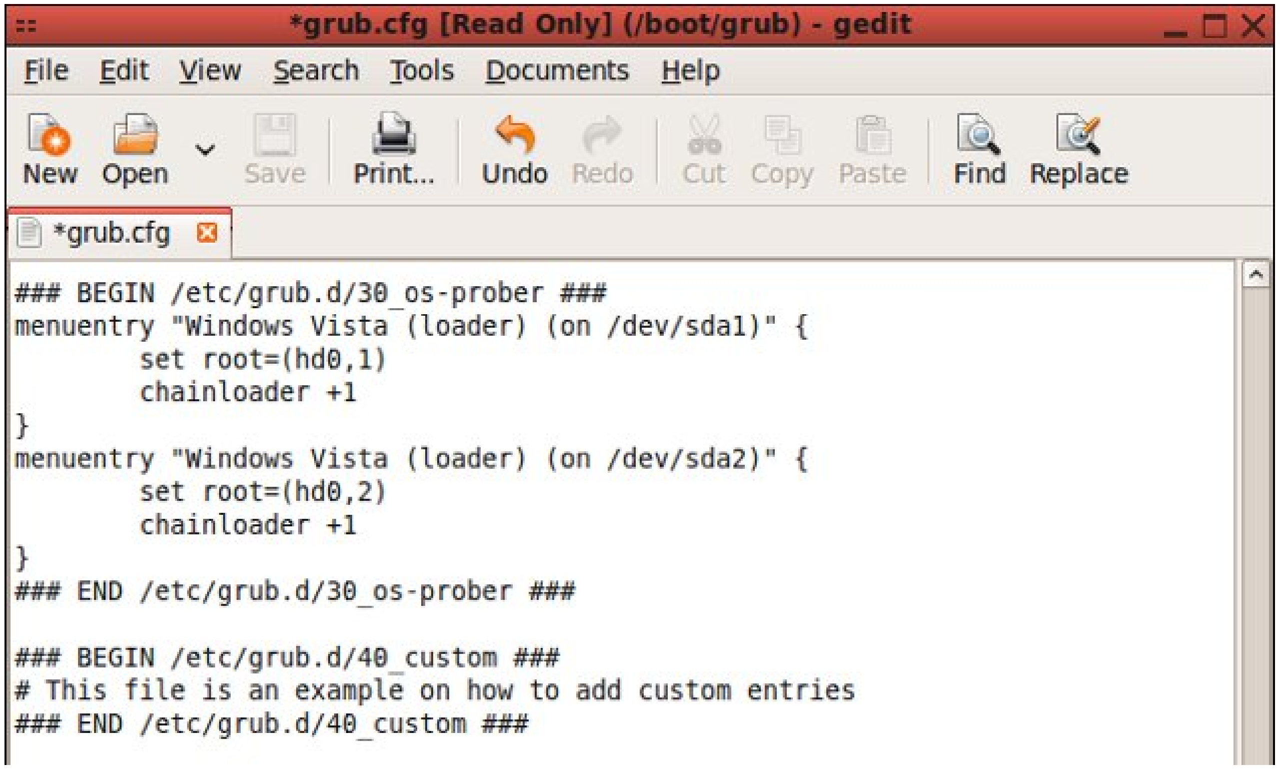
```
Saving menu.lst backup in /boot/grub/menu.lst_backup_by_grub2_postinst
Running update-grub Legacy to hook our core.img in it
    Searching for GRUB installation directory ... found: /boot/grub
    Searching for default file ... found: /boot/grub/default
    Testing for an existing GRUB menu.lst file ... found: /boot/grub/menu.lst
    Searching for splash image ... none found, skipping ...
    Found GRUB 2: /boot/grub/core.img
    Found kernel: /boot/vmlinuz-2.6.28-11-generic
    Found kernel: /boot/memtest86+.bin
    Updating /boot/grub/menu.lst ... done

Updating /boot/grub/grub.cfg ...
Found linux image: /boot/vmlinuz-2.6.28-11-generic
Found initrd image: /boot/initrd.img-2.6.28-11-generic
Found memtest86+ image: /boot/memtest86+.bin
Found Windows Vista (loader) on /dev/sda1
Found Windows Vista (loader) on /dev/sda2
done

Setting up grub2 (1.96+20080724-12ubuntu2) ...

Processing triggers for libc6 ...
ldconfig deferred processing now taking place
roger@roger-desktop:~$
```

Before we reboot, make sure the grub.cfg has been built correctly and that it contains the right entries, pointing to the right devices and partitions. Use fdisk if you need to verify that everything is in order. Ubuntu entry:


grub.cfg [Read Only] (/boot/grub) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
grub.cfg

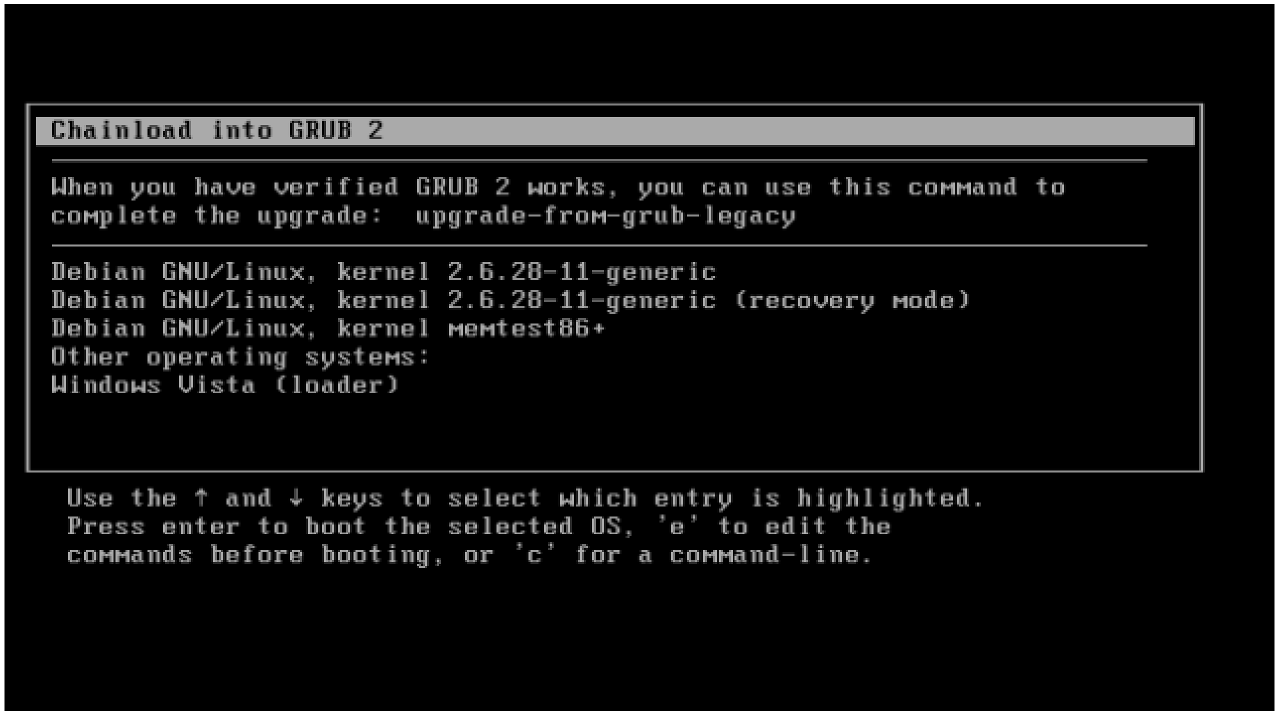
```
### BEGIN /etc/grub.d/10_linux ###
set root=(hd0,5)
search --fs-uuid --set 877460a8-cf97-40bd-98c5-a743d2a2f913
menuentry "Ubuntu, linux 2.6.28-11-generic" {
        linux   /boot/vmlinuz-2.6.28-11-generic root=UUID=877460a8-
cf97-40bd-98c5-a743d2a2f913 ro  quiet splash
        initrd  /boot/initrd.img-2.6.28-11-generic
}
menuentry "Ubuntu, linux 2.6.28-11-generic (single-user mode)" {
        linux   /boot/vmlinuz-2.6.28-11-generic root=UUID=877460a8-
cf97-40bd-98c5-a743d2a2f913 ro single
        initrd  /boot/initrd.img-2.6.28-11-generic
}
### END /etc/grub.d/10_linux ###
```

Windows entry:


*grub.cfg [Read Only] (/boot/grub) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
*grub.cfg

```
### BEGIN /etc/grub.d/30_os-prober ###
menuentry "Windows Vista (loader) (on /dev/sda1)" {
        set root=(hd0,1)
        chainloader +1
}
menuentry "Windows Vista (loader) (on /dev/sda2)" {
        set root=(hd0,2)
        chainloader +1
}
### END /etc/grub.d/30_os-prober ###

### BEGIN /etc/grub.d/40_custom ###
# This file is an example on how to add custom entries
### END /etc/grub.d/40_custom ###
```

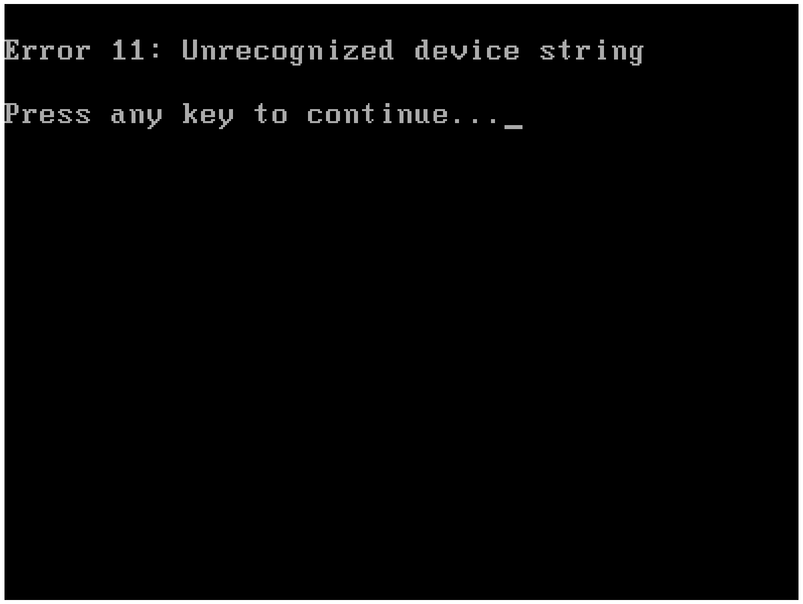
It's time to reboot and test. Your GRUB menu should be changed now and include a Chainload entry for GRUB 2 at the top of the menu.



```
Chainload into GRUB 2

When you have verified GRUB 2 works, you can use this command to
complete the upgrade:  upgrade-from-grub-legacy

Debian GNU/Linux, kernel 2.6.28-11-generic
Debian GNU/Linux, kernel 2.6.28-11-generic (recovery mode)
Debian GNU/Linux, kernel memtest86+
Other operating systems:
Windows Vista (loader)

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, 'e' to edit the
   commands before booting, or 'c' for a command-line.
```

Possible errors

You may encounter GRUB Error 11 or GRUB Error 15 when you try to boot into GRUB 2.

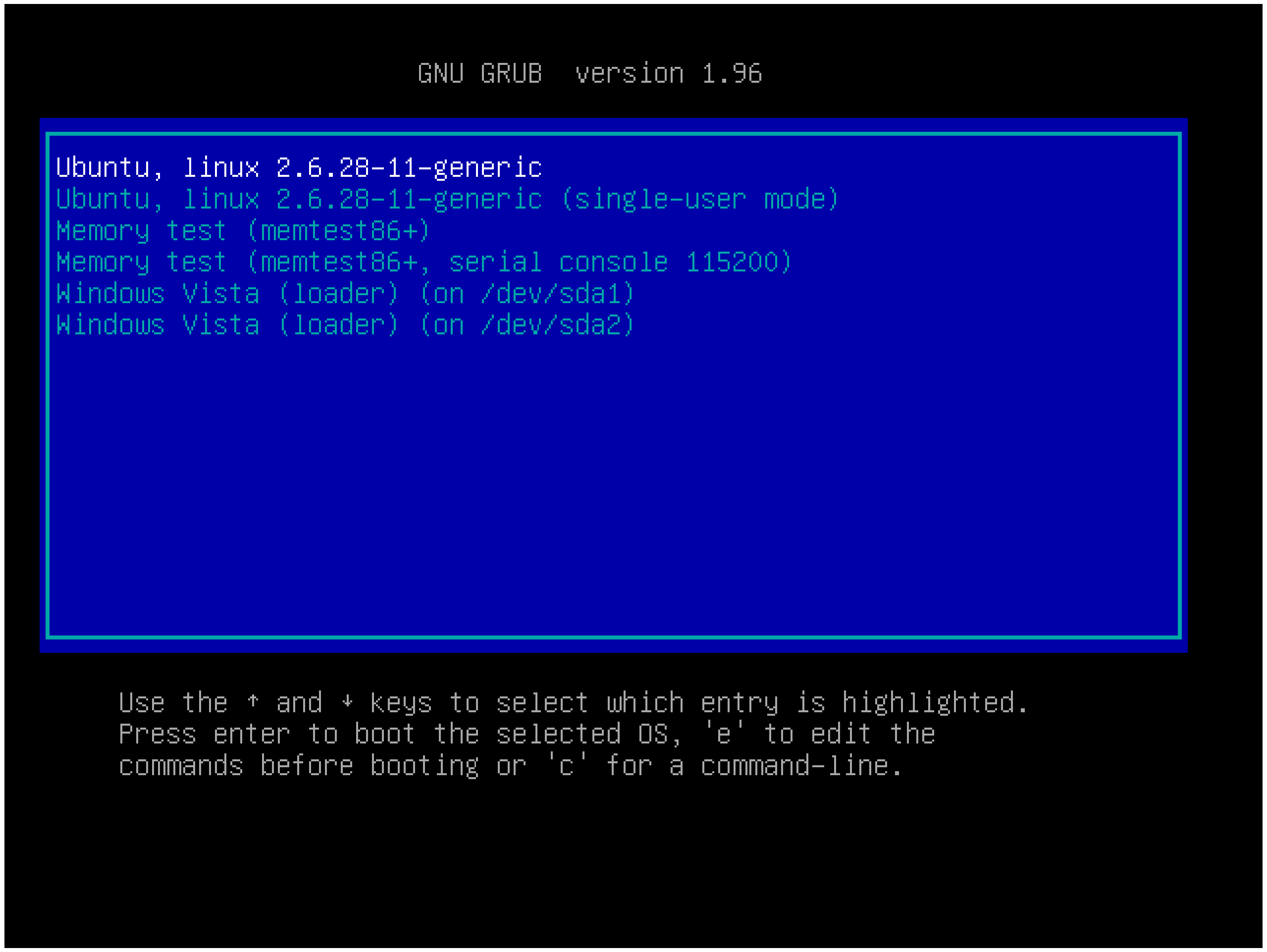


```
Error 11: Unrecognized device string

Press any key to continue..._
```

For instance, GRUB Error 11 means the wrong root device is selected or that you're booting devices by ID rather than numbers, in which case you will have to change one of the strings to make it work.

To remedy the issue, you will have to highlight the Chainload entry, press e to edit, then change the root line to reflect your real selection. Don't forget you're working with GRUB legacy still, so partitions are numbered from 0. In the worst case, cycle through root (hdX,Y) until you nail the right one. Change the root entry, hit Enter, press b to boot.

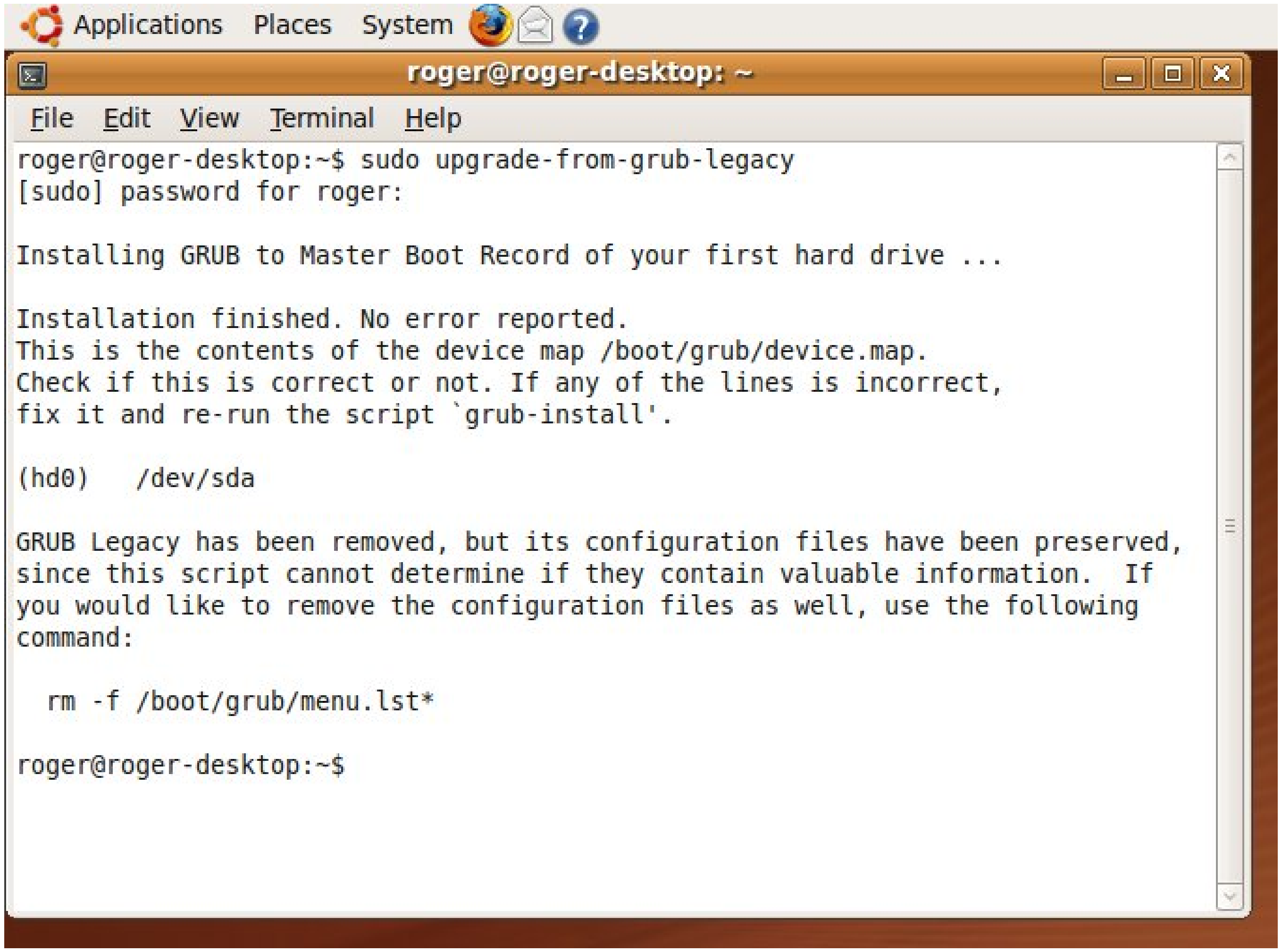
Alternatively, if your GRUB uses strange, long string called device IDs rather than numbers, you will have to replace the string root with uuid and then you should be able to boot your kernel just fine. HowtoForge has a great example, with screenshots. Eventually, you should see the GRUB 2 menu:



```
                    GNU GRUB  version 1.96

 Ubuntu, linux 2.6.28-11-generic
 Ubuntu, linux 2.6.28-11-generic (single-user mode)
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)
 Windows Vista (loader) (on /dev/sda1)
 Windows Vista (loader) (on /dev/sda2)

     Use the ↑ and ↓ keys to select which entry is highlighted.
     Press enter to boot the selected OS, 'e' to edit the
     commands before booting or 'c' for a command-line.
```

As you can see, we have Ubuntu entries and we have Vista loader (Windows 7) entries. All seems well. Now, boot both Ubuntu and Windows 7 to make sure everything works properly. Once you're satisfied, boot into Ubuntu and complete the migration process. You can now write GRUB 2 into MBR and make it the default bootloader.

```
sudo upgrade-from-grub-legacy
```



```
roger@roger-desktop:~$ sudo upgrade-from-grub-legacy
[sudo] password for roger:

Installing GRUB to Master Boot Record of your first hard drive ...

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'.

(hd0)   /dev/sda

GRUB Legacy has been removed, but its configuration files have been preserved,
since this script cannot determine if they contain valuable information.  If
you would like to remove the configuration files as well, use the following
command:

  rm -f /boot/grub/menu.lst*

roger@roger-desktop:~$
```

Pay attention to the device mapping and change accordingly. But that's it. We're done. We have now learned both how to upgrade GRUB, handle some boot errors and successfully boot a dual boot configuration including Windows and Linux.

Additional test case scenarios

Triple boot

From here on, it's damn easy. Use whatever you want. If you're running GRUB legacy as your bootloader, then you will use Windows entries like we did in the original article, plus an entry with /boot/grub/core.img for GRUB 2 entries.

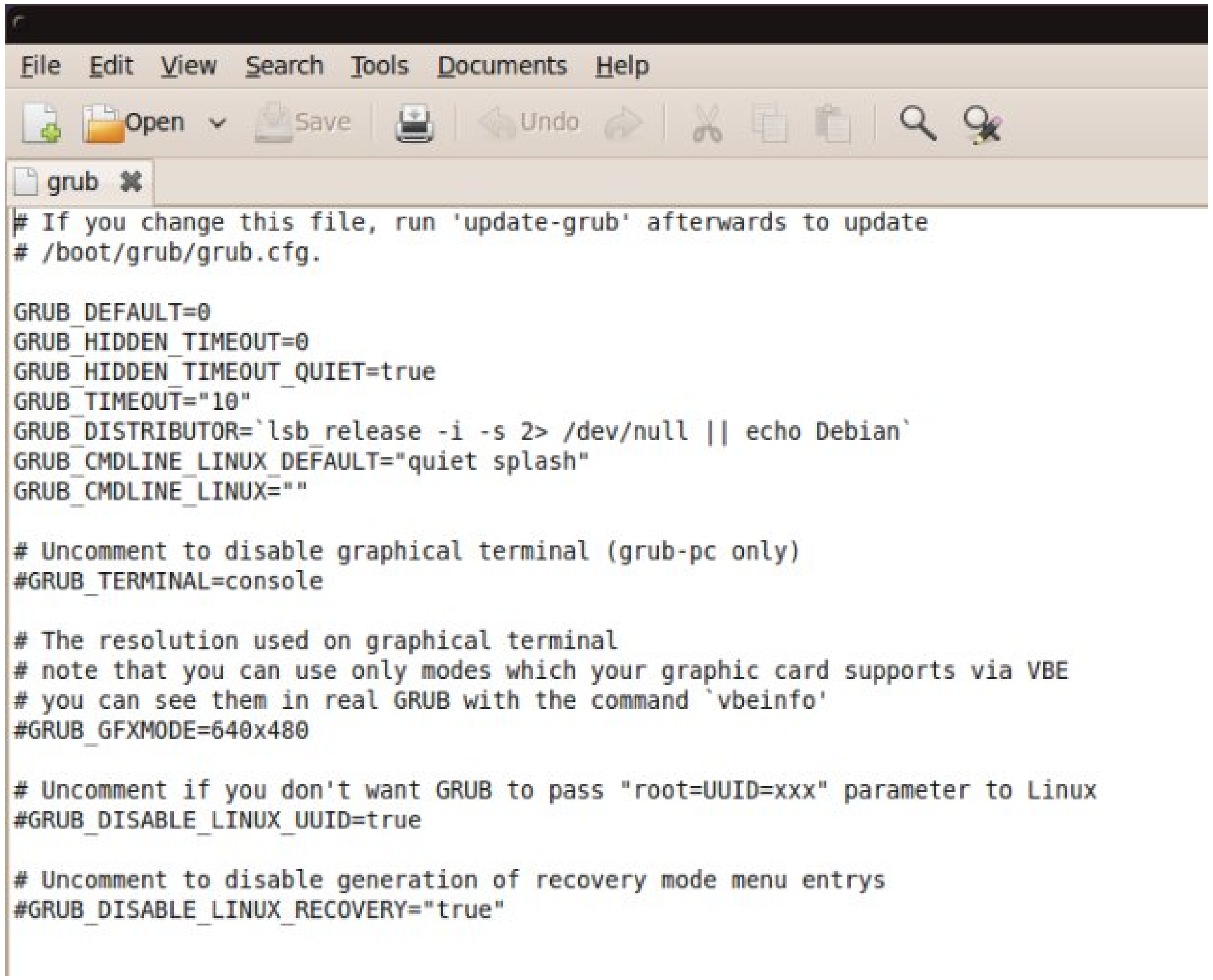
And if you're using GRUB 2, then use OS Prober to find other operating systems. And if you don't like the titles, create custom entries based on the existing selection. The world is your banana, now.

Customize GRUB menu

Let's say we want to edit the GRUB 2 menu a little, including the default selection, the timeout and other options.

Edit /etc/default/grub

This file contains a few interesting options you may want to change, including the default selected entry, the default timeout and additional options.



```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.

GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT="10"
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entrys
#GRUB_DISABLE_LINUX_RECOVERY="true"
```

GRUB_DEFAULT=0 specifies the default entry. It counts from 0, like any geeky menu. Change to anything you like. If you set the entry to GRUB_DEFAULT=saved, it will boot the last selected option from the previous boot.

GRUB_TIMEOUT="10" specifies the default timeout. Change to anything you want. Very small values are not recommended. Setting to -1 will make GRUB wait indefinitely until you manually select an entry and hit Enter.

There are many other options, as you can see, I will not go through all of them. It's important that you remember that they exist and where they can be found. Do not blindly make changes. Consult the official documentation and always backup the file before tampering. One more thing that may interest you is the removal of recovery entries.

Remove recovery entries from the menu

The last line in the /etc/default/grub file allows you to disable the recovery mod menu entries. Simply uncomment the line and update GRUB:

```
GRUB_DISABLE_LINUX_RECOVERY="true"
```

Remove memtest from the menu

If you do not want to have the memtest entries included, simply chmod -x the 20_memtest script. It will no longer be executable and won't be read into the grub.cfg file the next time you update GRUB.

```
sudo chmod -x 20_memtest86+
```

Change titles in menu entries

Instead of using the replacement tweaks I've suggested earlier, by recreating the default Linux entries and removing (chmod-ing -x) the 10_linux script, you can edit the actual script to behave differently and report information in a more human format. This is some really geeky stuff, take at look at Ubuntu install title tweaks.

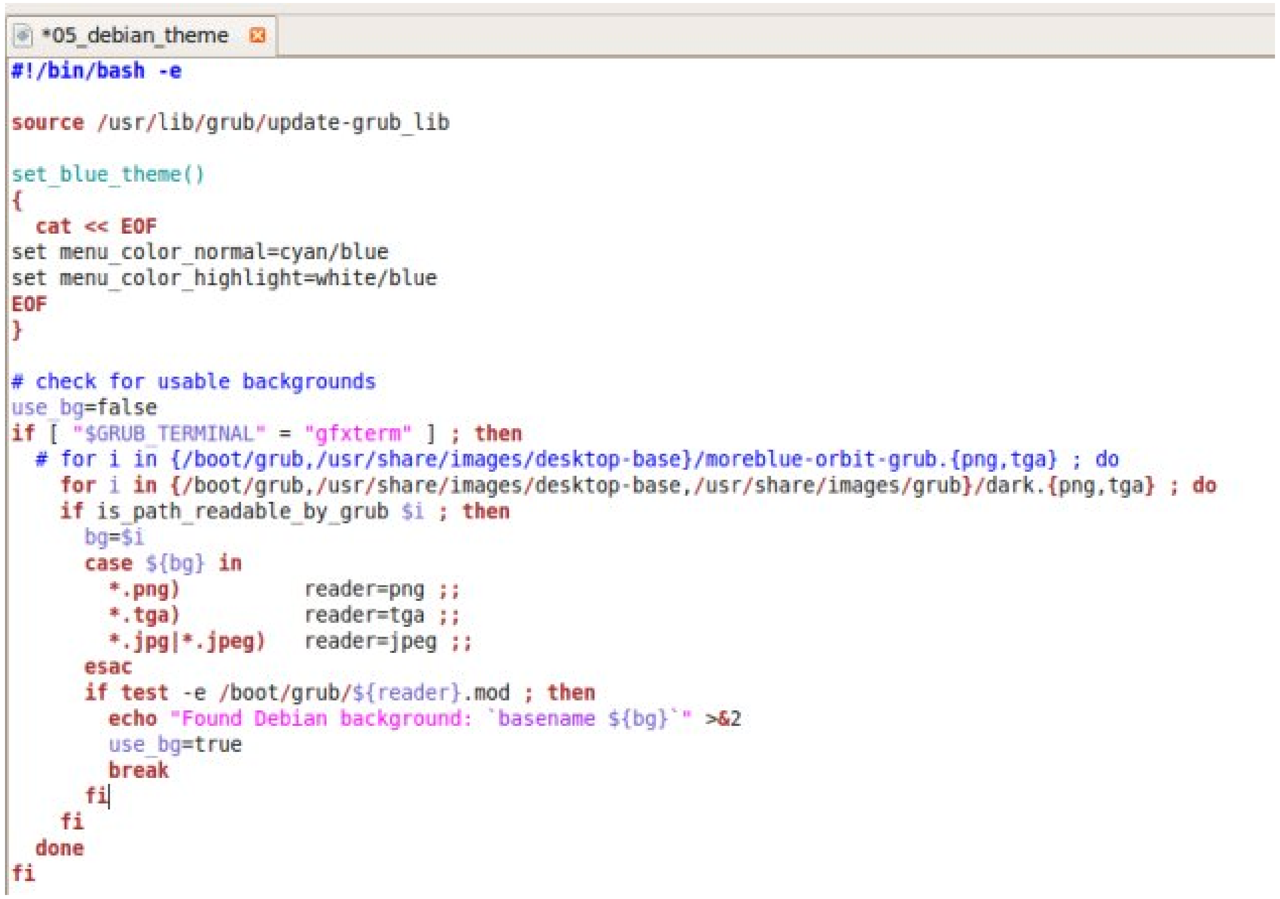
Change theme (boot image)

You may want to use a beautiful image during boot rather than the boring blue one. No worries, this can be easily done, by editing the 05_debian_theme script.

The first thing is to find a number of images you want. Pay attention to image detail, as you want to achieve the right contrast between the entries and the background, so you can still read the menu. I will show you my own example - learn from it.

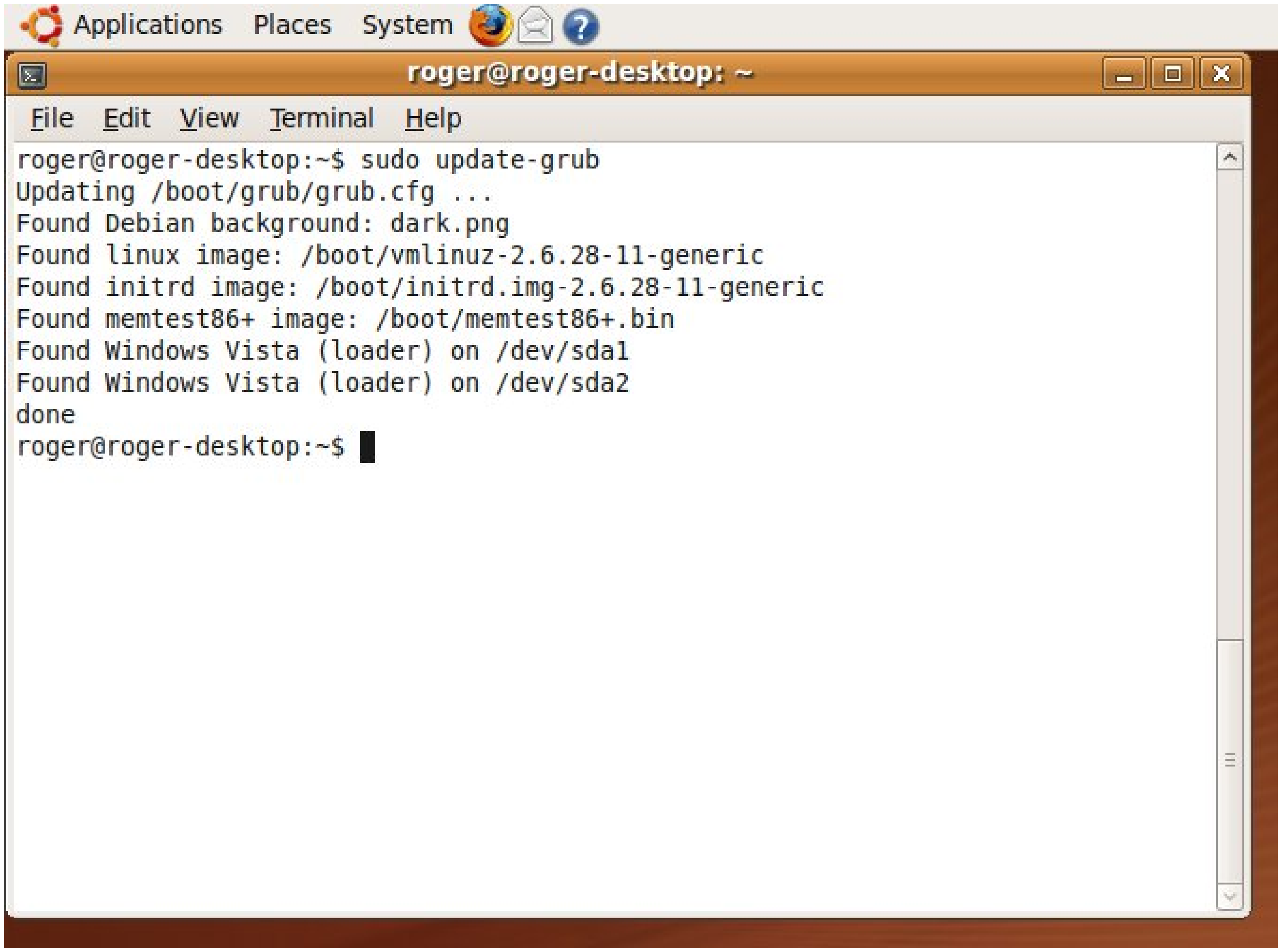
Next, you need a directory to store the images. By default, GRUB 2 takes images from /usr/share/images/desktop-base. This can be a good location - or if you want your own, create one. For instance, /usr/share/images/grub, just as shown on Ubuntu blog.

Copy your images there, mind the extension. By default, GRUB 2 works with .png and .tga files. You can also use .jpg images if you want. The next step is to edit the 05_debian_theme script and change the relevant entry to point to your own:



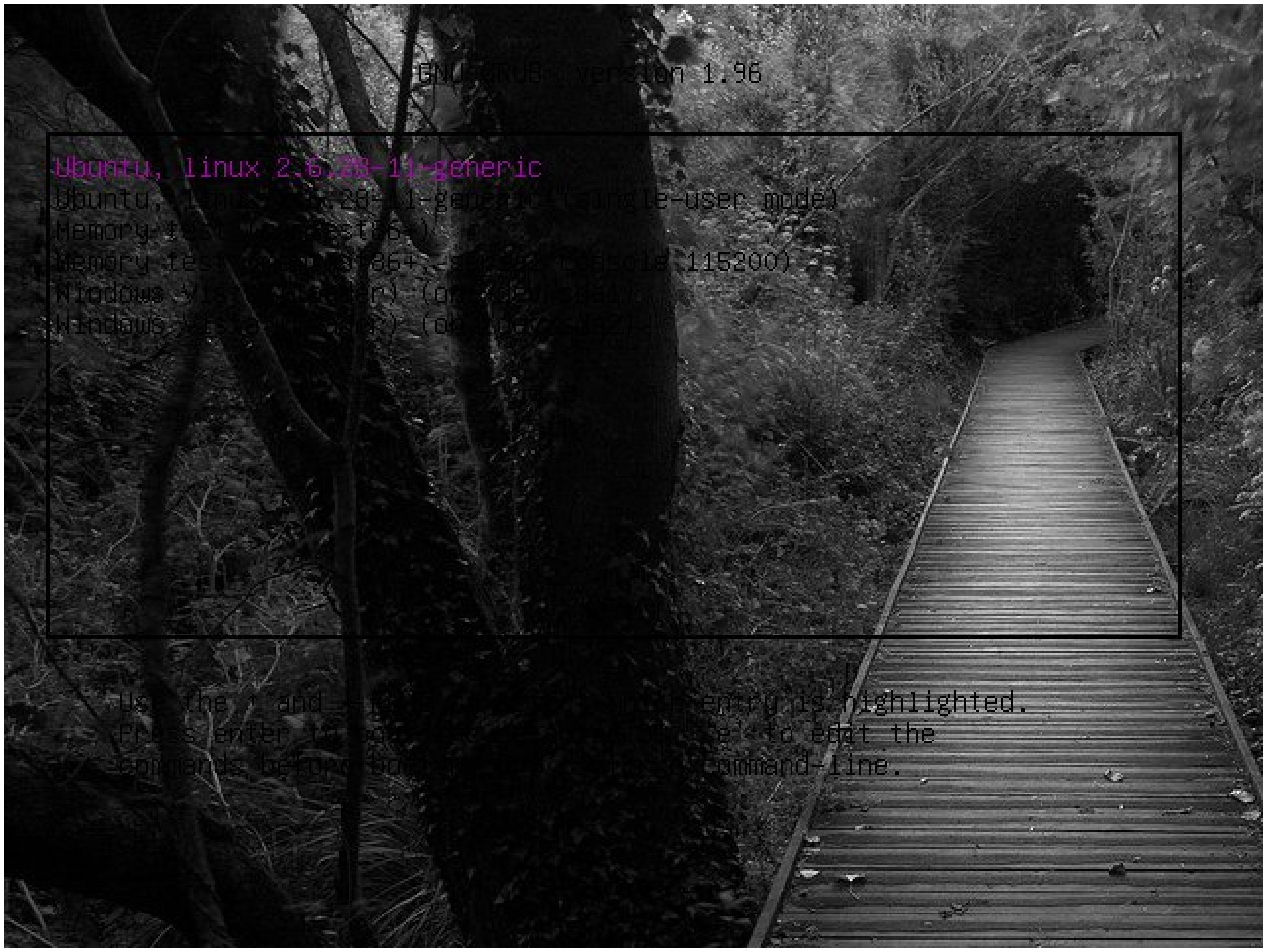
```
#!/bin/bash -e

source /usr/lib/grub/update-grub_lib

set_blue_theme()
{
  cat << EOF
set menu_color_normal=cyan/blue
set menu_color_highlight=white/blue
EOF
}

# check for usable backgrounds
use_bg=false
if [ "$GRUB_TERMINAL" = "gfxterm" ] ; then
  # for i in {/boot/grub,/usr/share/images/desktop-base}/moreblue-orbit-grub.{png,tga} ; do
    for i in {/boot/grub,/usr/share/images/desktop-base,/usr/share/images/grub}/dark.{png,tga} ; do
    if is_path_readable_by_grub $i ; then
      bg=$i
      case ${bg} in
        *.png)          reader=png ;;
        *.tga)          reader=tga ;;
        *.jpg|*.jpeg)   reader=jpeg ;;
      esac
      if test -e /boot/grub/${reader}.mod ; then
        echo "Found Debian background: `basename ${bg}`" >&2
        use_bg=true
        break
      fi
    fi
  done
fi
```

What I did was comment out the original for loop that reads the image(s) and created my own. If you're not really comfortable doing this, then you should probably postpone any theme tweaking until a later date. My file is located: /usr/share/images/grub/dark.png. Once the file has been changed, run update-grub to update the grub.cfg file. If you've done everything correctly, you should see Found Debian background message in the terminal.



```
roger@roger-desktop:~$ sudo update-grub
Updating /boot/grub/grub.cfg ...
Found Debian background: dark.png
Found linux image: /boot/vmlinuz-2.6.28-11-generic
Found initrd image: /boot/initrd.img-2.6.28-11-generic
Found memtest86+ image: /boot/memtest86+.bin
Found Windows Vista (loader) on /dev/sda1
Found Windows Vista (loader) on /dev/sda2
done
roger@roger-desktop:~$
```

Please note that the GRUB menu will not use background images unless these are enabled. To do that, you will have to change the line use_bg=false in theme script and change it to use_bg=true. Reboot and test:



```
                    GNU GRUB  version 1.96

 Ubuntu, linux 2.6.28-11-generic
 Ubuntu, linux [illegible]28-11-generic (single-user mode)
 Memory test [illegible]
 Memory test [illegible]86+, [illegible] 115200)
 Windows [illegible]) ([illegible])
 Windows [illegible]) ([illegible])

   U[illegible] the [illegible] and [illegible] entry is highlighted.
   Press enter to [illegible]e' to edit the
   commands before [illegible] command-line.
```

As you can see, my choice of background image was bad! You cannot see the text! Therefore, choose carefully what kind of background image you want to use. A theme with a few soft gradients seems like the best idea. For more details, take a look at the official Wiki documentation.

GRUB 2 recovery

Recover from failed boots

What happens if you ruin your GRUB 2? There must be a way to reinstall it and save the day? Well, as always, there's the easy way and the hard way.

Easy way: Super Grub Disk

This great tool works with GRUB 2, so no worries, place it into the CD/DVD tray, boot and restore the mangled GRUB. You should keep an image handy and ready, just in case.

Hard way: Manual fix from live CD

You will need a live CD that ships with GRUB 2, like Ubuntu or Kubuntu. Boot into the live session, mount the hard disk and install GRUB 2 to the MBR. This is the sequence of commands you require (assuming disk = /dev/sda). You will need to mount the partition of your installed distribution (e.g. Ubuntu) containing the /boot directory. It may also be a separate partition on your system, depending on your setup. After it is mounted, you will have to rerun the grub-install command.

```
mount /dev/sda1 /mnt/
grub-install --root-directory=/mnt /dev/sda
```

If this does not work, you will have to go through a much longer, more complicated procedure, as explained in the official GRUB 2 Ubuntu Wiki documentation.

Use fdisk to locate the right root device, then mount it as before. For the sake of this section, let's assume that /dev/sda1 is a dedicated /boot partition and /dev/sda2 is the root partition of your distro.

```
sudo mount /dev/sda2 /mnt
```

If you have a separate /boot partition, you will have to mount it too:

```
sudo mount /dev/sda1 /mnt/boot
```

Next, remount the rest of your devices using --bind option. For more details about how the mount command works, please consult the man page.

```
sudo mount --bind /dev /mnt/dev
```

Effectively, you now have a root system under mount. You can now change the root from your real one (/) to /mnt. This is done using the chroot command.

```
sudo chroot /mnt
```

This means that command executed in this terminal will refer to /mnt/. Now, reconfigure the GRUB package:

```
dpkg-reconfigure grub-pc
```

You will have to tell GRUB which device to use. You can change the selection using Spacebar. The devices shown will reflect that in the device.map file, so if you only have one, there should be no issues. Once this step is done, you can exit from the chroot environment. This is done by pressing Ctrl + D keys. After that, unmount the devices, first the /dev, then the rest:

```
sudo umount /mnt/dev
sudo umount /mnt
```

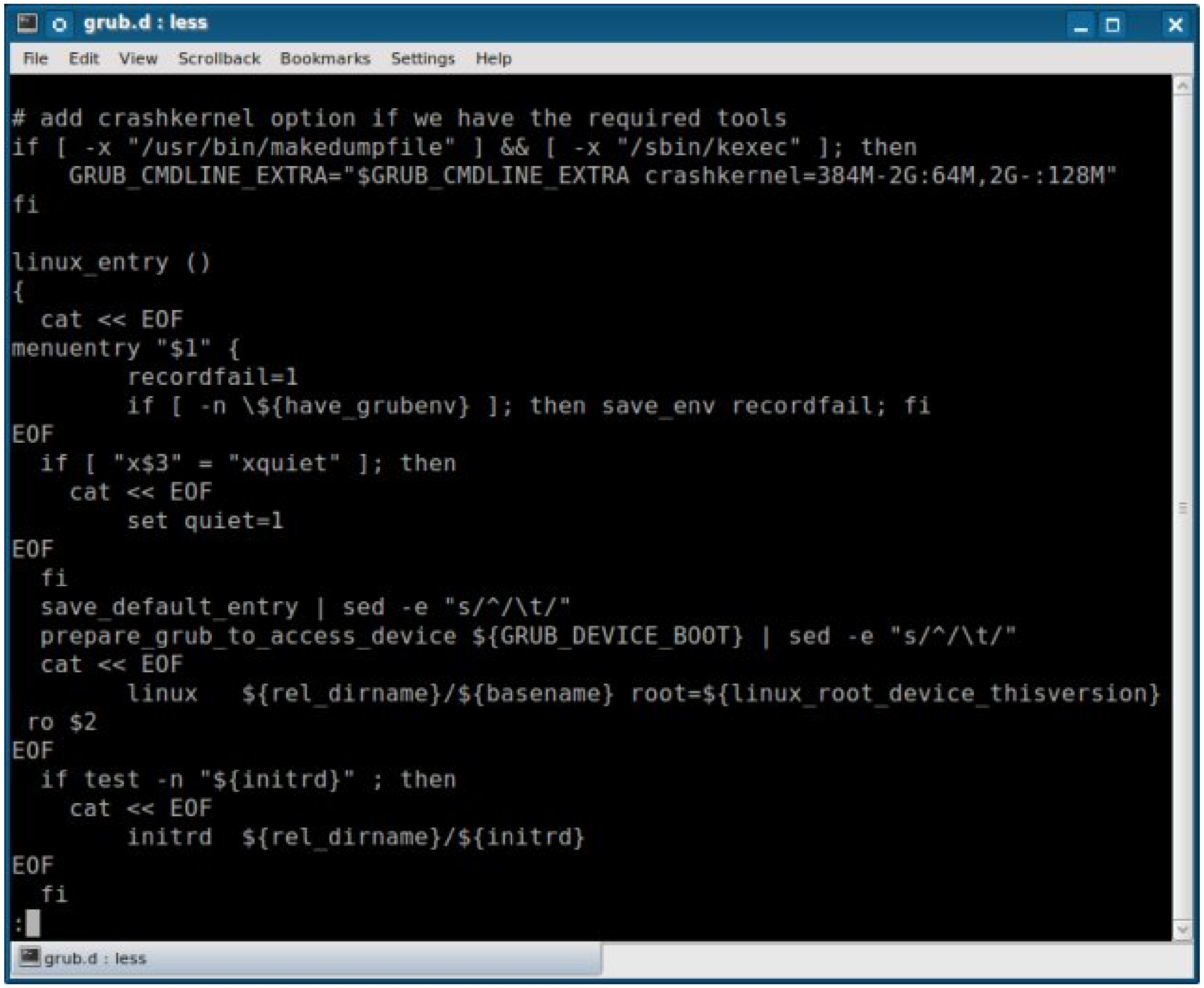
Now you can reboot. GRUB 2 should be in place.

Small problems observed

This might be beta or just new features, but I must comment on a number of items I have observed, which could also help you troubleshoot problems in the future more easily.

Kernel crash dump mechanism

GRUB 2 includes a section that checks if a kernel crash dump mechanism like Kdump is installed configured and appends a section to the kernel line. This is not a good idea, because the script does not check the offset where it can place the crash kernel. Furthermore, the memory allocation should be done by the admin and not arbitrarily, because we do not know how big or small the crash kernel should be.



```
# add crashkernel option if we have the required tools
if [ -x "/usr/bin/makedumpfile" ] && [ -x "/sbin/kexec" ]; then
    GRUB_CMDLINE_EXTRA="$GRUB_CMDLINE_EXTRA crashkernel=384M-2G:64M,2G-:128M"
fi

linux_entry ()
{
  cat << EOF
menuentry "$1" {
        recordfail=1
        if [ -n \${have_grubenv} ]; then save_env recordfail; fi
EOF
  if [ "x$3" = "xquiet" ]; then
    cat << EOF
        set quiet=1
EOF
  fi
  save_default_entry | sed -e "s/^/\t/"
  prepare_grub_to_access_device ${GRUB_DEVICE_BOOT} | sed -e "s/^/\t/"
  cat << EOF
        linux   ${rel_dirname}/${basename} root=${linux_root_device_thisversion}
 ro $2
EOF
  if test -n "${initrd}" ; then
    cat << EOF
        initrd  ${rel_dirname}/${initrd}
EOF
  fi
:
```

Command names

The two major commands, grub-install and update-grub do not align well. Either both should begin with a grub prefix or both should have a grub suffix.

GRUB legacy versus GRUB 2

Now the big question, why one should you (not) use?

GRUB legacy

It is older and no longer developed, but works great and has proven its worth many times over. Editing the GRUB menu is a very simple thing and require little skill on the behalf of the user, save for making sure the right syntax is followed.

GRUB 2

GRUB 2 is beta software at the moment, although version 1.97 could easily become the official one. This makes it a less likely candidate for production systems. Additionally, GRUB 2 is more difficult to work with, because a) it requires shell skill, which is not something everyone can do b) changes to the GRUB menu are more difficult to implement and require three separate steps rather than one as before c) the changes are not automatic and have to be "compiled" into the menu every time, similar to what LILO used to do.

GRUB 2 is currently supported by a very small number of distributions, thus you are less likely to receive support and find answers online, including official and unofficial circles. What more, there's a chance you will encounter problems when using GRUB 2 in a mixed environment.

Verdict

I have shown in both GRUB tutorials that even difficult tasks can be achieved relatively easily. It is possible to enjoy both GRUB legacy and GRUB 2, as my detailed examples clearly show. At the moment, though, you are probably better off with GRUB legacy, especially if you are a less knowledgeable user. If you're running Ubuntu, then you can stick with GRUB 2, as it comes as default, just make sure you follow my instructions for a happy and care-free grubbing.

More reading (documents & links)

You are most advised to take a look at the following articles, reviews and howtos:

Official

GRUB 2 official site

GRUB 2 Manual

GRUB 2 Ubuntu Wiki documentation

GRUB 2 Command List

Other useful GRUB 2 resources

GRUB bootloader - Full tutorial (my article)

Herman's GRUB pages (excellent collection of howtos)

GRUB error messages

How to change GRUB 2 theme

GRUB 2 title tweaks (advanced stuff)

GRUB 2 on archlinux Wiki (advanced stuff)

How to install GRUB 2 on Ubuntu 9.04

Conclusion

That's it. You now have a step-by-step guide for installing, configuring and troubleshooting GRUB 2, including the overview of the layout and basic functions, numerous real-life examples like dual boot scenarios with GRUB legacy, GRUB 2 and Windows mixed together, as well as tools and instructions how to troubleshoot problems. This guide should be very handy for you, whether you're a beginner user, a recent Windows convert or a Linux veteran.

Compared to GRUB legacy, GRUB 2 is somewhat harder to use, as it requires familiarity and confidence working with shell, which is outside the realm of most users. Furthermore, it does not easily permit changes and the boot configuration has to be rebuilt every time one is introduced. The true power of GRUB 2 is yet to be revealed.

I hope you are going to like this tutorial as much as the original. I will update it periodically if certain core functions in GRUB change toward production release, as well as write any important tip or trick that comes up. Have fun and spread the word!

Updates

All and any updates regarding GRUB 2 will be listed here, including new features, bug fixes and additional test cases, as well as any other useful material, tips and tricks you find and recommend. I suggest you occasionally check this section.

December 2009, update:

Several question asked by my readers:

> What if I use Ext3 filesystem? Can my GRUB boot partitions formatted with Ext4?

The answer is: maybe. If your distro supports Ext4, then yes, if it does not, probably not, because it won't be able to read anything from Ext4-formatted partitions. The easiest way to overcome filesystem compatibility issues is to use a dedicated /boot partition formatted with Ext3. This way, all modern Linux distros will support it.

> I have two disks. Ubuntu Karmic is on the second disk. Do I have to use core.img file in my GRUB or can I chainload the usual way?

The answer is: if you have a bootloader installed to the root of the second disk, then you can probably use the traditional chainloading method. But if you're using a single bootloader, then you must resort to core.img file.

> Why there's a difference between the 40_custom and your custom scripts?

The answer is: 40_custom script is already configured to be used within the context of other scripts when building the GRUB configuration file, at the end. Custom scripts that you write can go anywhere. Hence, the `exec tail -n +3 $0` in the script that you need not use in your own scripts.

> Will the custom scripts be updated if my other operating systems get kernel updates?

The answer is: Not automatically, and you'll have to rewrite them to point to relevant kernels. A way of solving this is to create a symbolic link to the latest kernel and the initrd image using vmlinuz and initrd generic names, respectively. This is what openSUSE does, allowing you to keep your GRUB menu static. The only change that needs to be done is to update the symbolic links following kernel updates, without touching the GRUB.

I can't get my custom backgrounds used in the GRUB menu anywhere!

See the use_bg section in the debian_theme script and make sure it's set to true.

June 2010, update:

GRUB 2 is currently at version 1.98 and this is considered a stable release. There are some small changes in the overall functionality, but 99% of what you see above still applies.

GRUB 2 rescue CD

If you're not using Super Grub Disk (SGD) or have a live CD of a Linux distribution supporting GRUB 2 available, then you may want to consider creating a GRUB 2 rescue CD of your own. Another advantage of this procedure is that the rescue CD will contain your custom GRUB menu.

This is done by using the grub-mkrescue command, which has a slightly different usage syntax for Lucid Lynx, which uses GRUB2-1.98-1ubuntu5 and Karmic and earlier, which use beta versions of the bootloader. Now, here's how to do it. Ubuntu 10.04:

```
grub-mkrescue --output=<name>.iso /boot/grub
```

You can use any name you want. Ubuntu 9.10 and earlier:

```
grub-mkrescue --overlay=/boot/grub <name>.iso
```

After creating the ISO image, you should burn it and then test it, preferably on another machine, to make sure the GRUB is booted from the CD and not the local disk. If you use a filename without a full path, the ISO image will be created in the current directory.

On the target (test) host, configured to boot from CD, the rescue CD should boot into GRUB commandline. To make sure that the CD works as expected and contains your GRUB menu, please run the following command:

```
configfile /grub.cfg
```

This should output your menu. Once satisfied with the change, keep the disc in standby for emergencies. Remember to create a new ISO any time you update the GRUB menu.

You should use the rescue CD to boot into desired operating system and then repair the GRUB using grub-install command, as we've seen earlier. The rescue CD will not work if you delete the operating system or partitions containing them. References:

How to make your own GRUB2RESCUE CD-ROM

There are also discussion threads available at Ubuntu forums and wilderssecurity.com, but they most revolve around the usage of these commands and the variations for different Ubuntu releases. Thanks to Ocky for this tip!

December 2010, update:

A few more changes yet. Here's the brief overview and update.

Notation

Once again, the notation has changed. On MS-DOS type partitions, which represents the absolute majority of partition out there, GRUB2 in Maverick uses a new naming convention. Instead of the generic hd(X,Y), now you have hd(X,msdosY), which could be a little confusing. You must take this into account when creating complex multi-boot setups. For more details, please take a look at my Maverick review.

Additional reading

You're also welcome to take a look at the official Ubuntu forums thread on GRUB2.

December 2011, update:

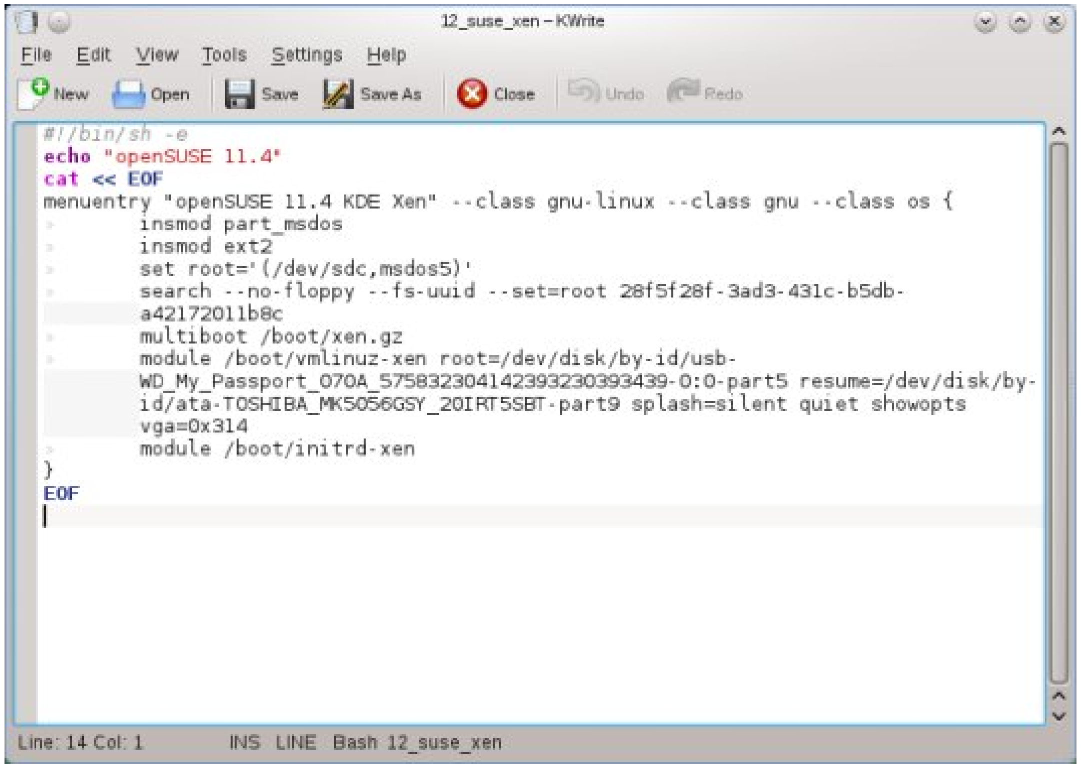
Several more changes. GRUB is officially at version 1.99 and comes with a few more changes. The important cosmetic fix is that additional, older versions of the kernel are all listed under a single directory-like entry, making the menu easier to read. Moreover, Ubuntu comes with a high-resolution menu and smaller font size. Now, let's examine several other things.

Booting Xen kernels in a multi-boot configuration governed by GRUB2

You have seen this example in my Xen intro tutorial. Specifically. Please note that if you're chainloading openSUSE with GRUB2, then you will need to create a special entry for the Xen kernel, which might not be automatically added by the OS probe script. The entry looks different from standard boot stanzas.

```
multiboot /boot/xen.gz
module /boot/vmlinuz-xen <options>
module /boot/initrd-xen
```

Of course, adjust the exact paths to match your installations, devices, etc. Here's a screenshot of what it looks like on my test machine:


12_suse_xen – KWrite
File Edit View Tools Settings Help
New Open Save Save As Close Undo Redo
Line: 14 Col: 1 INS LINE Bash 12_suse_xen

```
#!/bin/sh -e
echo "openSUSE 11.4"
cat << EOF
menuentry "openSUSE 11.4 KDE Xen" --class gnu-linux --class gnu --class os {
        insmod part_msdos
        insmod ext2
        set root='(/dev/sdc,msdos5)'
        search --no-floppy --fs-uuid --set=root 28f5f28f-3ad3-431c-b5db-
        a42172011b8c
        multiboot /boot/xen.gz
        module /boot/vmlinuz-xen root=/dev/disk/by-id/usb-
        WD_My_Passport_070A_575832304142393230393439-0:0-part5 resume=/dev/disk/by-
        id/ata-TOSHIBA_MK5056GSY_20IRT5SBT-part9 splash=silent quiet showopts
        vga=0x314
        module /boot/initrd-xen
}
EOF
```

Fedora 16 Verne support

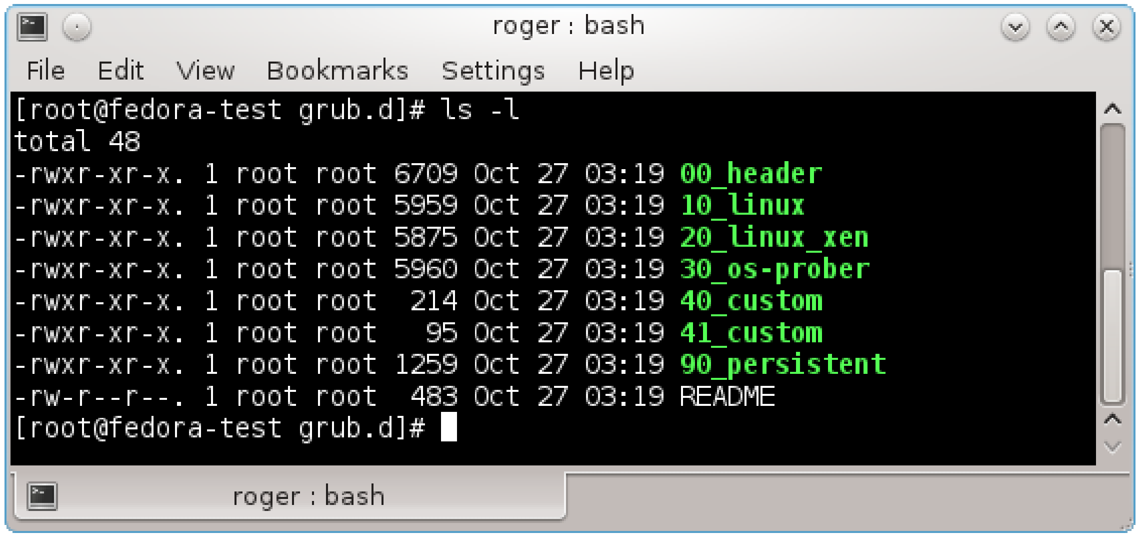
Fedora 16 also comes with GRUB2. Some of the settings are a little different from Debian-based systems. In fact, you may find Fedora a little tricky to navigate after you've used GRUB2 on Ubuntu and family. Fedora GRUB2 support is still very early and rather buggy, with some of the functionality not yet implemented well.

How to update the GRUB menu

The update command is as follows:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

This will recreate the grub.cfg file based on your scripts under /etc/grub.d.

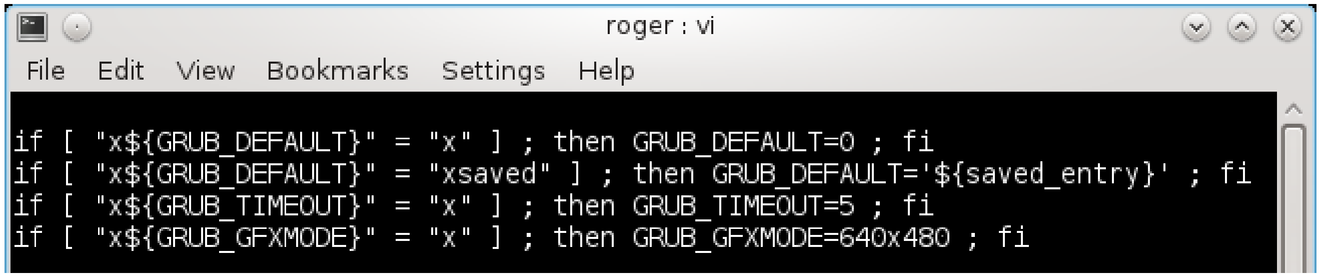


```
[root@fedora-test grub.d]# ls -l
total 48
-rwxr-xr-x. 1 root root 6709 Oct 27 03:19 00_header
-rwxr-xr-x. 1 root root 5959 Oct 27 03:19 10_linux
-rwxr-xr-x. 1 root root 5875 Oct 27 03:19 20_linux_xen
-rwxr-xr-x. 1 root root 5960 Oct 27 03:19 30_os-prober
-rwxr-xr-x. 1 root root  214 Oct 27 03:19 40_custom
-rwxr-xr-x. 1 root root   95 Oct 27 03:19 41_custom
-rwxr-xr-x. 1 root root 1259 Oct 27 03:19 90_persistent
-rw-r--r--. 1 root root  483 Oct 27 03:19 README
[root@fedora-test grub.d]#
```

Edit header defaults (entry and timeout)

Changing the default entry and the timeout in Ubuntu is fairly trivial. This is not so in Fedora. Editing the 00_header file is more complicated. You must actually change the script functionality to get what you need.

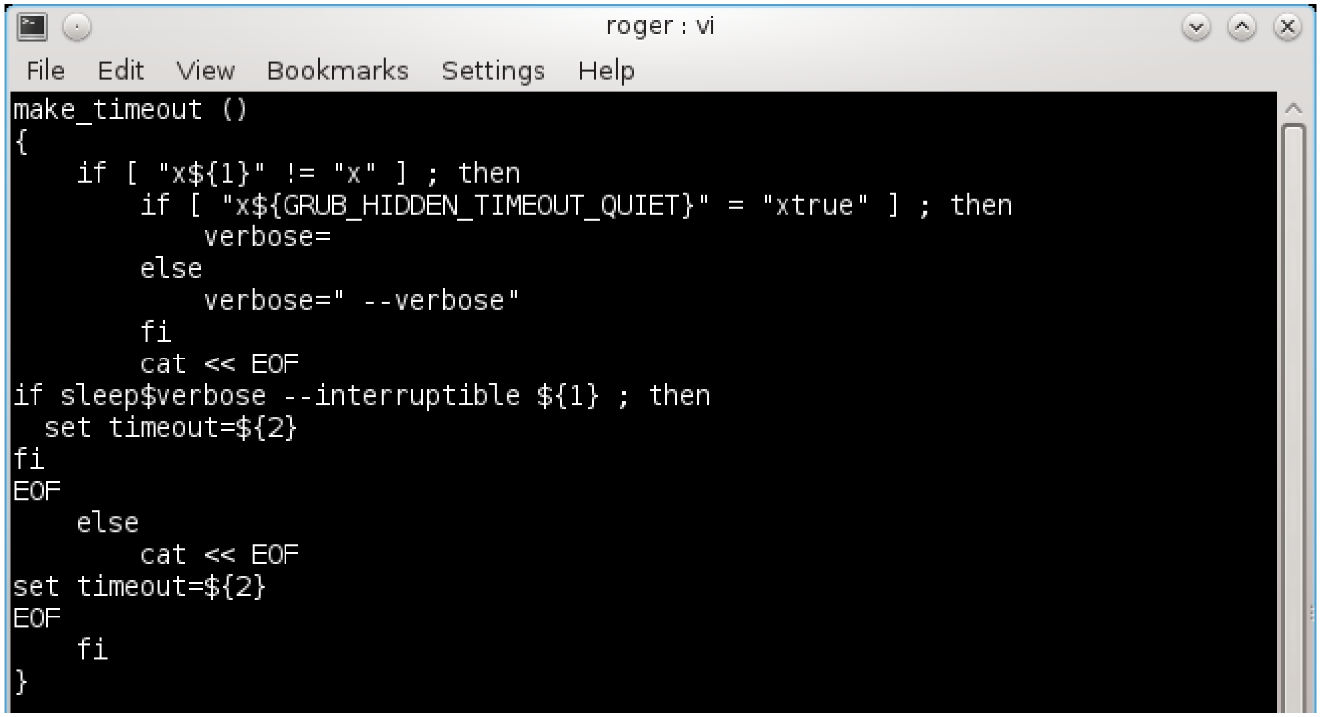
The timeout is defined by make_timeout() function. The function accepts two input parameters, GRUB_TIMEOUT and GRUB_TIMEOUT_BUTTON, which are declared at the beginning of the 00_header script.



```
if [ "x${GRUB_DEFAULT}" = "x" ] ; then GRUB_DEFAULT=0 ; fi
if [ "x${GRUB_DEFAULT}" = "xsaved" ] ; then GRUB_DEFAULT='${saved_entry}' ; fi
if [ "x${GRUB_TIMEOUT}" = "x" ] ; then GRUB_TIMEOUT=5 ; fi
if [ "x${GRUB_GFXMODE}" = "x" ] ; then GRUB_GFXMODE=640x480 ; fi
```

However, the variables will get different values based on a variety of environmental settings and configurations. If you find it too hard tracking down the execution, you may want to brute-force the change.

In the make_timeout() function, you can edit the declaration set timeout=${2} to a static value. The script will then ignore the input parameter and use whatever you choose. For example, you may want to increase the timeout from 5 seconds to 20 seconds.



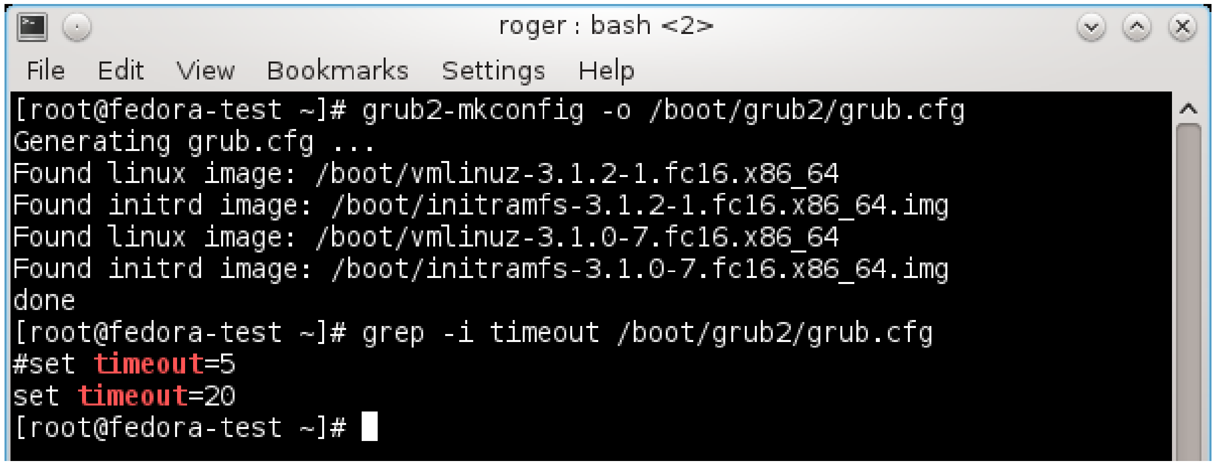
```
make_timeout ()
{
    if [ "x${1}" != "x" ] ; then
        if [ "x${GRUB_HIDDEN_TIMEOUT_QUIET}" = "xtrue" ] ; then
            verbose=
        else
            verbose=" --verbose"
        fi
        cat << EOF
if sleep$verbose --interruptible ${1} ; then
  set timeout=${2}
fi
EOF
    else
        cat << EOF
set timeout=${2}
EOF
    fi
}
```

In that case, change the script as follows:

```
#set timeout=${2}
set timeout=20
```

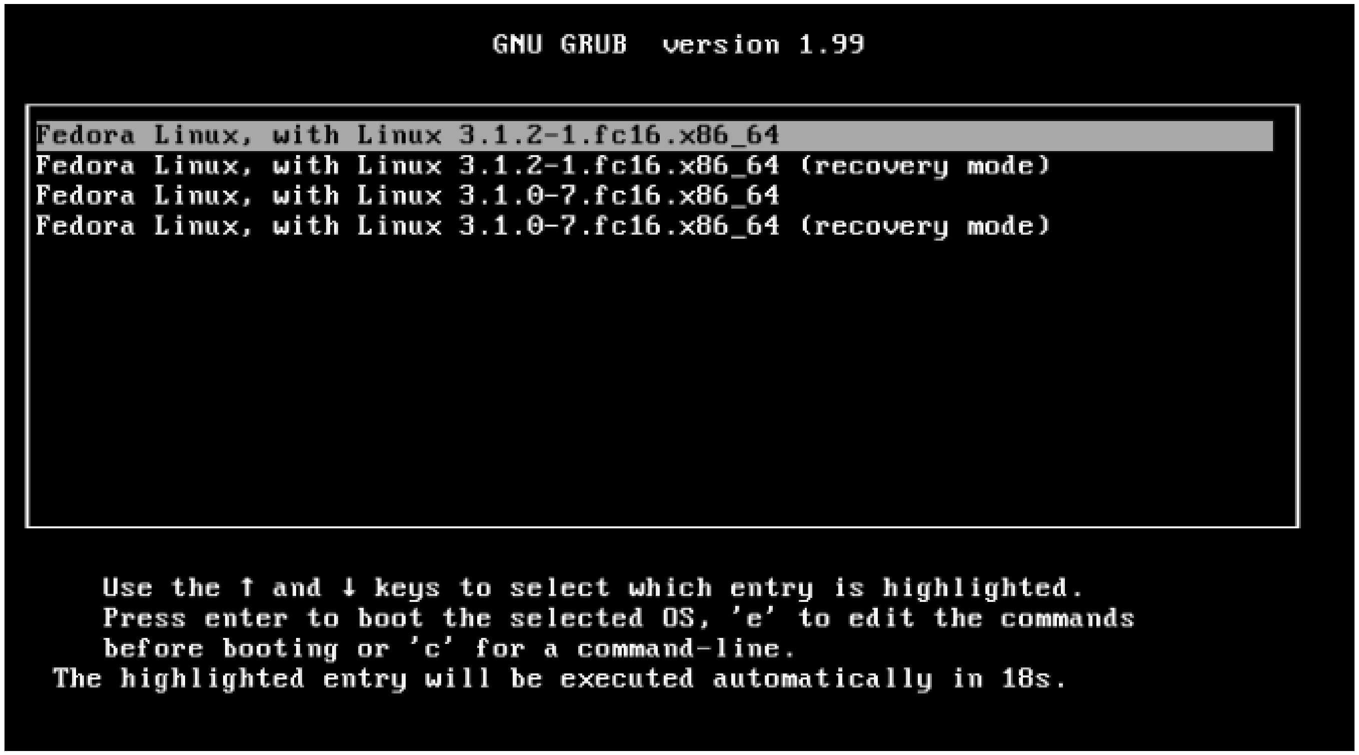
Make necessary backups first! And there's no need to delete anything, just comment out the lines you do not want to use. This way, you will be able to revert back to the original state if needed.

After editing the file, you must recreate the grub.cfg file. Once the file is recreated, you can verify that your change has been configured properly by grepping the timeout value in the configuration file.



```
[root@fedora-test ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-3.1.2-1.fc16.x86_64
Found initrd image: /boot/initramfs-3.1.2-1.fc16.x86_64.img
Found linux image: /boot/vmlinuz-3.1.0-7.fc16.x86_64
Found initrd image: /boot/initramfs-3.1.0-7.fc16.x86_64.img
done
[root@fedora-test ~]# grep -i timeout /boot/grub2/grub.cfg
#set timeout=5
set timeout=20
[root@fedora-test ~]#
```

And here we are booting; notice the timeout counter.



```
GNU GRUB  version 1.99

Fedora Linux, with Linux 3.1.2-1.fc16.x86_64
Fedora Linux, with Linux 3.1.2-1.fc16.x86_64 (recovery mode)
Fedora Linux, with Linux 3.1.0-7.fc16.x86_64
Fedora Linux, with Linux 3.1.0-7.fc16.x86_64 (recovery mode)

    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press enter to boot the selected OS, 'e' to edit the commands
    before booting or 'c' for a command-line.
 The highlighted entry will be executed automatically in 18s.
```

More reading

You may also want to consult the following online pages:

GRUB2 - FedoraProject.org

GRUB2 features - FedoraProject.org

If you feel a separate Fedora & GRUB2 tutorial is needed, we can arrange that too.

June 2012, update:

GRUB 2.00 has been officially released. It comes with all kinds of goodies. We will be seeing GRUB 2 in its major release version coming to various distributions later this year. For the end user, the changes should be transparent. Just use this tutorial and have massive fun.

You might also want to look at my GRUB article in the 117th issue of the Linux User & Developer magazine released in October 2012. It's basically a multi-boot tutorial with focus on the GRUB 2.00 release. Enjoy.

January 2014, update:

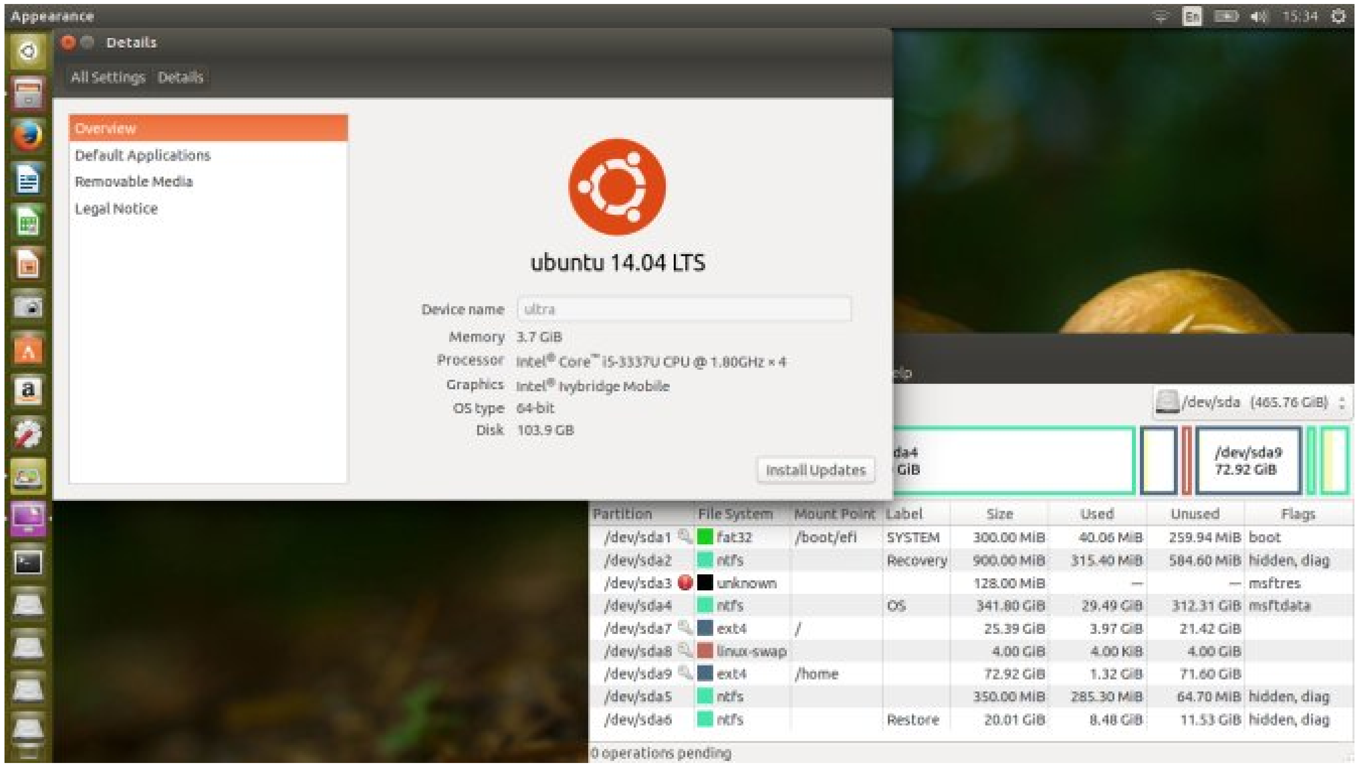
A small update on how to update the GRUB configuration file on openSUSE. This is very similar to the exercise we have conducted on Fedora above. The grub.cfg file needs to be created using the grub2-mkconfig command, with the output file flag.

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

You can then create test files and compare before committing changes.

May 2014, update:

You might also be interested to learn about an Asus Ultrabook setup that includes UEFI, Secure Boot, GPT partition table rather than MS-DOS, and a dual-boot configuration of Windows 8.1 and Ubuntu. The GRUB2 configuration is entirely transparent, because of the built-in support in the Ubuntu image. But as a use case, it is quite interesting, especially given the rather complex default layout of the laptop in question.



ubuntu 14.04 LTS

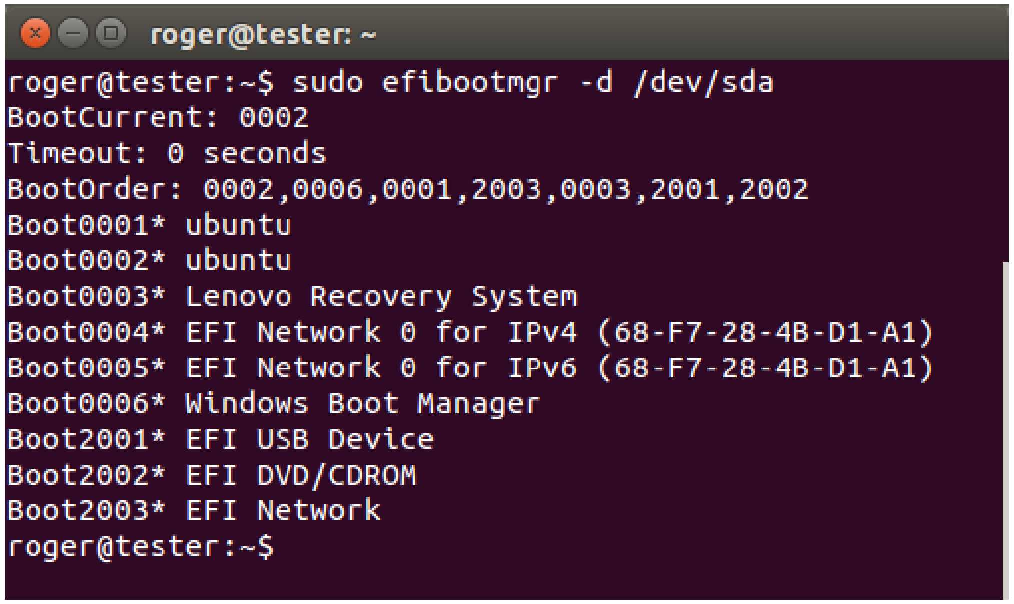
Device name: ultra
Memory: 3.7 GiB
Processor: Intel® Core™ i5-3337U CPU @ 1.80GHz × 4
Graphics: Intel® Ivybridge Mobile
OS type: 64-bit
Disk: 103.9 GB

Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sda1	fat32	/boot/efi	SYSTEM	300.00 MiB	40.06 MiB	259.94 MiB	boot
/dev/sda2	ntfs		Recovery	900.00 MiB	315.40 MiB	584.60 MiB	hidden, diag
/dev/sda3	unknown			128.00 MiB	—	—	msftres
/dev/sda4	ntfs		OS	341.80 GiB	29.49 GiB	312.31 GiB	msftdata
/dev/sda7	ext4	/		25.39 GiB	3.97 GiB	21.42 GiB	
/dev/sda8	linux-swap			4.00 GiB	4.00 KiB	4.00 GiB	
/dev/sda9	ext4	/home		72.92 GiB	1.32 GiB	71.60 GiB	
/dev/sda5	ntfs			350.00 MiB	285.30 MiB	64.70 MiB	hidden, diag
/dev/sda6	ntfs		Restore	20.01 GiB	8.48 GiB	11.53 GiB	hidden, diag

0 operations pending

April 2015, GRUB2 & EFI recovery:

If you need to recover GRUB on systems with UEFI and GPT, then I have a complete separate guide for this. The procedure is a bit long and it warrants its own article, that's why I'm splitting from this already very long and detailed howto.



```
roger@tester:~$ sudo efibootmgr -d /dev/sda
BootCurrent: 0002
Timeout: 0 seconds
BootOrder: 0002,0006,0001,2003,0003,2001,2002
Boot0001* ubuntu
Boot0002* ubuntu
Boot0003* Lenovo Recovery System
Boot0004* EFI Network 0 for IPv4 (68-F7-28-4B-D1-A1)
Boot0005* EFI Network 0 for IPv6 (68-F7-28-4B-D1-A1)
Boot0006* Windows Boot Manager
Boot2001* EFI USB Device
Boot2002* EFI DVD/CDROM
Boot2003* EFI Network
roger@tester:~$
```

June 2015, GRUB2 & UEFI:

Furthermore, a reader named Phil has some additional interesting pointers. Namely, the location of the grub.cfg file is different on systems that use UEFI. You can locate the correct path on your system using the find command:

```
find /boot -name 'grub.cfg'
```

For instance, you may get /boot/efi/EFI/centos/grub.cfg. Therefore, if you need to update the configuration file, and you want to do it explicitly, without using the GRUB update command, you will need to specify the correct file as the output:

```
grub2-mkconfig -o /boot/efi/EFI/<os>/grub.cfg
```

February 2017, GRUB2 & UEFI errors:

I have published two additional tutorials that help address several new boot problem scenarios, including a corrupt EFI partition after an installation of a Linux distribution, and not being able to boot other operating systems in a multi-boot setup governed by a Red Hat based distro bootloader on UEFI systems due to missing EFI modules with linuxefi and initrdefi commands.

```
 error: can't find command 'linux'
 error: can't find command 'initrd'
```

More to come ... P.S. If you find this article useful, please support Dedoimedo.

Cheers for now, stay tuned for updates.

del.icio.us stumble digg reddit slashdot

GCP Get a $300 free trial credit to get started with any GCP product.

Top Home Terms of use Contact me About

W3C HTM 4.01 W3C CSS 2.0 RSS VALID FEEDS