The
Definitive Guide
To
Theming GRUB 2

2nd Edition

Towheed Mohammed

Contents

The Definitive Guide to Theming GRUB2

DISCLAIMER

The procedures outlined below have been tested using the release of GRUB2 from the following distributions:

- Ubuntu 10.04 (Lucid) using GRUB 1.98-1ubuntu6 to GRUB 1.98-1ubuntu10
- Ubuntu 10.10 (Maverick) using GRUB 1.98+20100804-5ubuntu3
- Debian 6.0 (Squeeze) using GRUB 1.98+20100804-14
- Fedora 14 (Laughlin) using GNU GRUB 1.98
- openSUSE 11.3 (Teal) using GNU GRUB 1.98
- Sabayon 5.5 (Fermi) using GNU GRUB 1.98

Any differences between distributions will be noted as necessary.

There is no guarantee that the procedures in this guide will work on other versions of GRUB2. If you decide to test the procedures on other versions, YOU DO SO AT YOUR OWN RISK.

Introduction

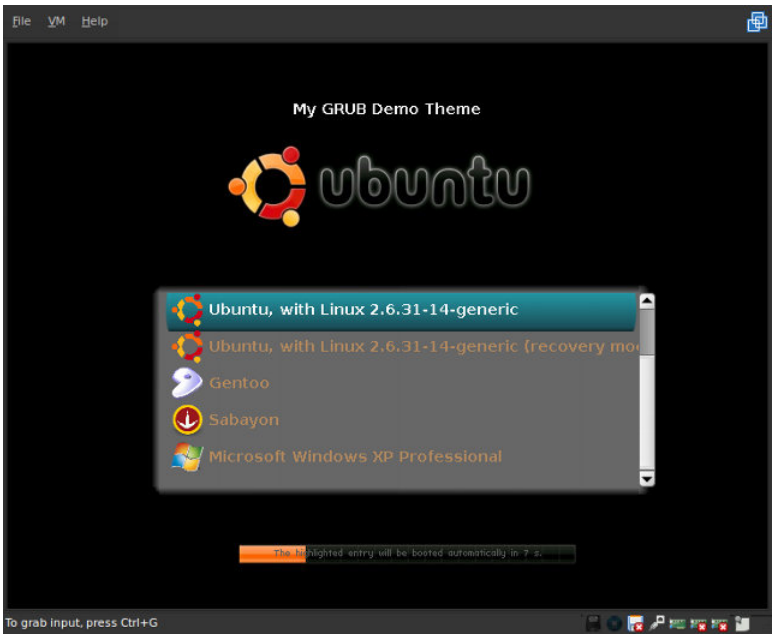
This is the Definitive Guide to Theming GRUB, the Grand Unified Bootloader. In this guide, GRUB refers to GRUB2 and not GRUB Legacy. For this guide, we will be using the latest release of GRUB from the repositories of the Linux distributions listed on the previous page.

When we say 'Theming GRUB", we are not referring to simply adding a background image and changing the text's colors, but instead, we are referring to adding a graphical menu using Colin D. Bennett's ***gfxmenu*** module. This is his Google Summer of Code 2008 project. The graphical menu uses a simple text file for defining the theme which allows for the use of images, text, fonts, scrollbar, progress bar and various styled boxes.

In previous versions of GRUB, including GRUB Legacy, a themed GRUB menu may have looked like this:

Screenshot 1: GRUB's Old Theme

Using the gfxmenu theming engine, it will look like this:



Screenshot 2: GRUB's Graphical Menu

Terminal

Most of our work will be done from the command line interface (CLI). This requires the use of a terminal. To open a terminal window, go to the main menu and click on:

Applications → Accessories → Terminal

Text Editor

In this guide, gedit will be used as the default text editor. You may use your editor of choice. I recommend that you turn on line numbering in the text editor. To do this in gedit, click:

Edit → Preferences → Display Line Numbers

Issuing a command with root privileges

Most of the commands used in this guide must be run with *root privileges*. There are two methods of issuing a command with *root privileges* which depends on the distribution you use. The two methods involve the use of either sudo or su. Check the documentation of your distribution to determine which one applies to you.

Using sudo and gksudo

The sudo command allows you to run other commands with *root privileges*. To issue a command with *root privileges* using sudo, enter sudo followed by the command (enter your password when prompted):

```
sudo <command>
```

The gksudo command allows you to open a GUI based application with *root privileges*. To open the text editor (gedit) with *root privileges*, enter this command in the terminal window:

```
gksudo gedit
```

Using su and gksu

The su command also runs a command with *root privileges,* but it's syntax is different from the sudo command. To issue a command with *root privileges* using su, use the following (enter root's password when prompted):

```
su -c "<command options parameters>"
```

To process files from your ~ directory (*/home*) or any of it's sub-directories using su, use the following command:

```
su -mc "<command options parameters>"
```

If you use only the -c option, you must specify an absolute path for your home directory, i.e.: use */home/your_username/Documents* instead of *~/Documents*.

The gksu command allows you to open a GUI based application with *root privileges*. To open the text editor (gedit) with *root privileges*, enter this command in the terminal window:

```
gksu gedit
```

For Fedora 14 Users

The beesu command allows Fedora 14 users to open a GUI based application with *root privileges*. Install the beesu wrapper using the following command:

```
su -c "yum install beesu"
```

To open the text editor (gedit) with *root privileges*, enter this command in the terminal window:

```
beesu gedit
```

For openSUSE 11.3 Users

For openSUSE 11.3 users, you must use the gnomesu command to open a GUI application with *root privileges*. To open the text editor (gedit) with *root privileges*, enter this command in the terminal window:

```
gnomesu gedit
```

For the remainder of this guide, sudo and gksudo will be the commands used to gain *root privileges*. Please substitute these with the appropriate commands listed above for your distribution.

Installing GRUB

To check the installed version of GRUB, enter this command in the terminal window:

```
grub-install --version
```

For non-Debian based distributions, enter this command (some distributions may require you to run it with *root privileges*):

```
grub2-install --version
```

Ubuntu Maverick

For users of Ubuntu Maverick, the required version of GRUB is already installed on your system and you may continue on to the next chapter.

Ubuntu Lucid

For users of Ubuntu Lucid who have been performing regular updates, version *1.98-1ubuntu10* of GRUB is already installed on your system. If you would like to install the Maverick release then follow the procedure listed below for Karmic users. If you intend to keep your installed version then continue on to the next chapter.

Ubuntu Karmic

For users of Ubuntu Karmic, we will install the version of GRUB available from the Maverick main repository.

First, backup your sources.list file by entering the following command in the terminal window:

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list_orig
```

Now add the Maverick main repository to your sources.list file by entering the following command in the terminal window:

```
sudo add-apt-repository "deb http://archive.ubuntu.com/ubuntu/ maverick main"
```

Update the packages using this command:

```
sudo apt-get update
```

Install GRUB using this command:

```
sudo apt-get install grub-pc grub-common
```

During the installation, GRUB may ask whether or not to replace it's script files with the package maintainer's version. Select 'Y' for all such questions.

Verify the correct version of GRUB is installed using this command:

```
grub-install --version
```

grub-install (GRUB) 1.98+20100804-5ubuntu3 should be returned.

To prevent other Maverick updates into Karmic or Lucid, restore your original sources.list file and update the packages using these commands:

```
sudo mv /etc/apt/sources.list_orig /etc/apt/sources.list

sudo apt-get update
```

At this point, perform a reboot to verify that GRUB is working properly.

Fedora 14

A broken dependency in the GRUB2 package requires Fedora 14 users to install the *gettext* package first. Install the *gettext* package using this command:

```
su -c "yum install gettext"
```

Before you decide to install GRUB2 please consider the following warning from the GRUB2 package:

PLEASE NOTE: This is a development snapshot, and as such will not replace grub if you install it, but will be merely added as another kernel to your existing GRUB menu. Do not replace GRUB (grub package) with it unless you know what are you doing. Refer to README.Fedora file that is part of this package's documentation for more information.

Install GRUB2 using this command:

```
su -c "yum install grub2"
```

Install the GRUB2 bootloader using this command:

```
su -c "grub2-install /dev/sdx"
```

where x represents the drives letter, **a** is Drive 0, **b** is Drive 1 etc.

Perform a reboot and ***GNU GRUB version 1.98*** should be displayed at the top of the menu items.

openSUSE 11.3

For openSUSE 11.3 users, use the following procedure to install GRUB2.

Open the Install/Remove Software application and enter ***grub2*** in the search box. Click on the *checkbox* next to the grub2 package and then click on the *Apply* button. When the installation is completed, go to the terminal and do the following:

Install the GRUB2 bootloader using this command:

```
su -c "grub2-install /dev/sdx"
```

where x represents the drives letter, **a** is Drive 0, **b** is Drive 1 etc.

Generate GRUB's configuration file using this command:

```
su -c "grub2-mkconfig -o /boot/grub2/grub.cfg"
```

Perform a reboot and ***GNU GRUB version 1.98*** should be displayed at the top of the menu items.

Debian 6.0, LinuxMint 10 and Sabayon 5.5

For users of the above distributions, GRUB2 is installed by default. You can continue on to the next chapter.

Backup Files

Before any files are edited, we will create a backup copy of them. If something goes wrong, you can easily restore them.

We will backup the */etc/default/grub*, */usr/sbin/grub-mkconfig* script files, and the */etc/grub.d* directory.

First, create a directory to hold the backup files using the following command:

```
mkdir ~/Documents/grub_orig
```

Backup

Backup the files using the these commands:

```
sudo cp -r /etc/grub.d/ ~/Documents/grub_orig/
sudo cp {/etc/default/grub,/usr/sbin/grub-mkconfig} ~/Documents/grub_orig/
```

Sabayon 5.5

For Sabayon users, the *grub-mkconfig* script file is located in */sbin* instead of */usr/sbin*. Replace */usr/sbin* with */sbin* in the second command above.

Debian 6.0, Fedora 14 and openSUSE 11.3

For Debian, Fedora and openSUSE users (or any distributions that uses su), backup the files using these commands:

```
su -mc "cp -r /etc/grub.d/ ~/Documents/grub_orig/"
su -mc "cp {/etc/default/grub,/usr/sbin/grub-mkconfig} ~/Documents/grub_orig/"
```

Fedora and openSUSE users must replace *grub-mkconfig* with *grub2-mkconfig*.

Restore

If you need to restore the files, use these commands:

```
sudo cp -r ~/Documents/grub_orig/grub.d/ /etc/
sudo cp ~/Documents/grub_orig/grub /etc/default/
```

```
sudo cp ~/Documents/grub_orig/grub-mkconfig /usr/sbin/
```

Fedora and openSUSE users must replace *grub-mkconfig* with g*rub2-mkconfig* in the third command above and run the commands with root privileges using:

```
su -mc "<command>"
```

Sabayon 5.5

Sabayon users must replace */usr/sbin* with */sbin* in the third command above.

GRUB's Configuration Scripts

GRUB's configuration scripts must be modified to support the graphical menu. The first script that we will modify is */etc/default/grub.*

For users of the following distributions:

- Ubuntu Lucid using GRUB version 1.98-1ubuntu6 to 1.98-1ubuntu10
- Fedora 14
- openSUSE 11.3, and
- Sabayon 5.5

we will also modify the */etc/grub.d/00_header* script file and add a new script file to */etc/grub.d*.

For testing purposes only, we will also add a few menu entries to */etc/grub.d/40_custom.*

These are not bootable entries, but are only there to add some additional menu items to the boot menu.

Modify /etc/default/grub

Open your */etc/default/grub* script file with *root privileges*:

```
gksudo gedit /etc/default/grub &
```

Remember to use the command appropriate for your distribution, ie: gksu, beesu or gnomesu.

Debian 6.0, Ubuntu (All), LinuxMint 10 and Sabayon 5.5

Comment out the contents of the lines beginning with:

- GRUB_HIDDEN_TIMEOUT
- GRUB_HIDDEN_TIMEOUT_QUIET
- GRUB_GFXMODE

by adding a hash (#) to the beginning to each line.

If the GRUB_TIMEOUT variable is commented, uncomment it by removing the hash and setting a timeout value of 30 seconds.

The beginning of the file should now look like this (the numbers at the start of each line are the line

numbers shown in gedit and may vary slightly depending on your distribution):

```
1  # If you change this file, run 'update-grub' afterwards to update
2  # /boot/grub/grub.cfg.
3
4  GRUB_DEFAULT=0
5  #GRUB_HIDDEN_TIMEOUT=0
6  #GRUB_HIDDEN_TIMEOUT_QUIET=true
7  GRUB_TIMEOUT=30
```

Save your changes but do not close gedit.

Fedora 14 and openSUSE 11.3

Add the following lines to the beginning of your */etc/default/grub* script file (do not delete the existing GRUB_CMDLINE_LINUX line):

```
1  # If you change this file, run 'grub2-mkconfig -o /boot/grub2/grub.cfg'
2  # afterwards to update /boot/grub2/grub.cfg.
3
4  GRUB_DEFAULT=0
5  #GRUB_HIDDEN_TIMEOUT=0
6  #GRUB_HIDDEN_TIMEOUT_QUIET=true
7  GRUB_TIMEOUT=30
8
```

For Fedora 14

Add *vga=0x311* to the existing value of GRUB_CMDLINE_LINUX at line 9:

```
9  GRUB_CMDLINE_LINUX="vga=0x311 quiet rhgb"
```

Copy the unicode.pf2 font to your */usr/share/grub* directory using this command:

```
su -mc "cp ~/Documents/grub_guide/unicode.pf2 /usr/share/grub/"
```

Save your changes but do not close gedit.

For openSUSE 11.3

Add *vga=0x311* to the existing value of GRUB_CMDLINE_LINUX at line 9:

```
9  GRUB_CMDLINE_LINUX="vga=0x311 quiet"
```

Save your changes but do not close gedit.

Create a */usr/share/grub* directory and copy the unicode.pf2 font to it using these commands:

```
su -c "mkdir /usr/share/grub"
su -mc "cp ~/Documents/grub_guide/unicode.pf2 /usr/share/grub/"
```

Modify /etc/grub.d/00_header

If you are using GRUB version **1.98+20100804** please skip this section and continue on to '*Adding More Menu Items*'.

Open your */etc/grub.d/00_header* script file and delete the following lines (line numbers will vary considerably across distributions):

```
106 EOF
107 if [ x$GRUB_THEME != x ] && [ -f $GRUB_THEME ] \
108     && is_path_readable_by_grub $GRUB_THEME; then
109     echo "Found theme: $GRUB_THEME" >&2
110     prepare_grub_to_access_device `${grub_probe} --target=device
    $GRUB_THEME` | sed -e "s/^/  /"
111     cat << EOF
112   insmod gfxmenu
113   set theme=(\$root)`make_system_path_relative_to_its_root $GRUB_THEME`
114   set menuviewer=gfxmenu
115 EOF
116 fi
117     cat << EOF
```

Save your changes but do not close gedit.

Copy 08_gfxmenu_theme

Copy the provided *08_gfxmenu_theme* script file to your */etc/grub.d* directory using this command:

```
sudo cp ~/Documents/grub_guide/08_gfxmenu_theme /etc/grub.d/
```

Sabayon 5.5

Open the *08_gfxmenu_theme* script file and change line 3 to:

```
3  prefix=        # Sabayon 5.5 users must delete /usr from this line.
```

Save the changes.

Adding More Menu Items

Open your */etc/grub.d/40_custom* and add the following lines to the end of the file:

```
6
7  menuentry 'Gentoo' --class gentoo {
8    set root='(hd0,1)'
9  }
10
11 menuentry 'Sabayon' --class sabayon {
12   set root='(hd0,1)'
13 }
14
15 menuentry 'Microsoft Windows XP Professional' --class windows {
16   set root='(hd0,1)'
17 }
```

Save the changes and close gedit.

This is not absolutely necessary. It is for testing purposes only and should be deleted after all testing is completed. If you have multiple OS's on your system, you can skip this step.

Updating GRUB's configuration file

Whenever you change any of GRUB's script files, you must update it's configuration script file for the changes to take effect. This is done using the following command:

```
sudo update-grub
```

Fedora 14 and openSUSE 11.3

Fedora and openSUSE users must use this command to update GRUB's configuration script file:

```
su -c "grub2-mkconfig -o /boot/grub2/grub.cfg"
```

Sabayon 5.5

Sabayon users must use this command to update GRUB's configuration script file:

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

Theme

GRUB's graphical menu customizes the boot menu using themes. A theme may include various components such as images, labels, various styled boxes, the boot menu and a progress bar. Each component can have one or more properties assigned to it which controls the appearance of the component. The components and their respective properties are defined in a plain text configuration file, which in this guide we will refer to as the *theme definition file*.

Theme Definition File Format

A *theme definition file* has two types of statements, the global property statement which appears only once, and the component statement which must be prefixed with the **+** sign. There must be one or more component statement in the *theme definition file*. A component's properties and their respective values must be enclosed within braces. They can be specified either on a single line thus:

+ component_name {property 1 = value property 2 = value . . . . . }

or indented on multiple lines:

+ component_name {

property 1 = value

property 2 = value

.

.

.

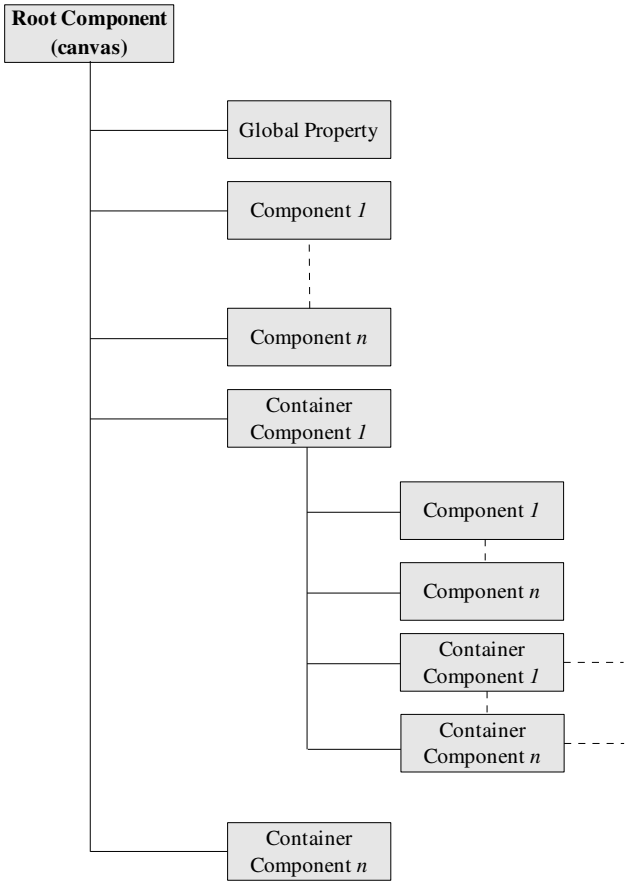
property n = value

}

There is one special type of component referred to as a *container* component. This special component is used as a container for other components. The top-level (root) component is an instance of the *canvas* container component. A container component may hold other container components.

Hierarchical Structure of a Theme

The following diagram shows a simple hierarchical structure of a theme. The root component which is the *canvas* container component, holds all the other components of the theme.



Hierarchical Structure of a Theme

Global Property

The following table describes each item of the global property statement:

Item Name	Description	Value Type	Default Value
title-text	Sets a title for the menu	String	GRUB Boot Menu
title-font	Font used for the title text	Font name	Unknown Regular 16
title-color	Color of the title text	Color value	Black
message-font	Font used for GRUB messages	Font name	Unknown Regular 16
message-color	Color used for GRUB messages	Color value	White
message-bg-color	Background color for GRUB messages	Color value	Black
desktop-image	Background image for the menu	Image file	
desktop-color	Color of the background if a background image is not specified	Color value	White

terminal-box	Image used for the terminal box	Styled Box	Plain black box
terminal-font	Font used for the terminal box.	Font name	Fixed 10

The default value is applied if an item name is not specified.

Components

Boot Menu

Component name: boot_menu

The boot menu component sets the visual style of the boot menu. This component must be included in the *theme definition file*. The table below describes each item of the boot menu's property list:

Property Name	Description	Value Type	Default Value
left	Sets the left position of the menu	Numeric	
top	Sets the top position of the menu	Numeric	
width	Sets the width of the menu	Numeric	
height	Sets the height of the menu	Numeric	
item_font	Font used for the menu items	Font name	Unknown Regular 16
selected_item_font	Font used for the selected menu items. Valid values are any font name or inherit. When set to inherit, it uses the same font set by the item_font property	Font name	inherit
item_color	Color of the menu items	Color value	Black
selected_item_color	Color of the selected menu items. Valid values are any color value or inherit. When set to inherit, it uses the same color set by the item_color property	Color value	Black
item_height	Height of each menu item	Numeric	42 pixels
item_padding	Amount of space to leave on either side of the menu items	Numeric	14 pixels
item_spacing	Amount of space between menu items	Numeric	16 pixels
icon_width	Width of the menu item icons	Numeric	32 pixels
icon_height	Height of the menu item icons	Numeric	32 pixels
item_icon_space	Space to leave between a menu item's icon and it's text	Numeric	4 pixels

menu_pixmap_style	Image used for the menu background	Styled Box	
selected_item_pixmap_style	Image used to highlight the selected item	Styled Box	
scrollbar	Show or hide the scrollbar	Boolean	false
scrollbar_frame	Image used for the scrollbar's frame	Styled Box	
scrollbar_thumb	Image used for the scrollbar's thumb	Styled Box	
id	Identifier for the boot menu	String	
visible	Show or hide the boot menu	Boolean	true

The boot menu items will not be shown if the boot menu component is not included in the theme definition file.

Label

Component name: label

The label component displays a single line of text on the screen. The table below describes each item of the label's property list:

Property Name	Description	Value Type	Default Value
left	Sets the left position of the label	Numeric	
top	Sets the top position of the label	Numeric	
width	Sets the width of the label	Numeric	
height	Sets the height of the label	Numeric	
text	Text to display	String	
align	Horizontal alignment of the text. Possible values are left, center or right	String	left
font	Font used for the text	Font name	Unknown Regular 16
color	Color of the text	Color value	black
id	Identifier for the label	String	
visible	Show or hide the text	Boolean	true

Image

Component name: image

The image component displays an image on the screen. The image is scaled to fit the size specified by

the width and height properties. The table below describes each item of the image's property list:

Property Name	Description	Value Type	Default Value
left	Sets the left position of the image	Numeric	
top	Sets the top position of the image	Numeric	
width	Sets the width of the image	Numeric	
height	Sets the height of the image	Numeric	
file	Absolute path to the image	String	
id	Identifier for the image	String	

Progress Bar

Component name: progress_bar

Displays a solid color or styled horizontal progress bar that indicates the time remaining until the highlighted menu item is booted. The table below describes each item of the progress bar's property list:

Property Name	Description	Value Type	Default Value
left	Sets the left position of the progress bar	Numeric	
top	Sets the top position of the progress bar	Numeric	
width	Sets the width of the progress bar	Numeric	
height	Sets the height of the progress bar	Numeric	
id	Identifier for the progress bar. This must be set to **__timeout__**	String	
bg_color	Background color for a solid color progress bar	Color value	gray
fg_color	Foreground color for a solid color progress bar	Color value	light gray
border_color	Border color for a solid color progress bar	Color value	black
show_text	Show or hide the countdown text	Boolean	
text	Text to show on the progress bar. Possible values are: @TIMEOUT_NOTIFICATION_SHORT@ @TIMEOUT_NOTIFICATION_MIDDLE@ @TIMEOUT_NOTIFICATION_LONG@, or any valid string	String	
text_color	Color of the text	Color value	black
font	Font used for the text	Font name	Unknown Regular 16
bar_style	Background image for the styled progress bar. If this is not specified, a solid color progress bar is displayed	Styled box	
highlight_style	Foreground image for the styled progress bar. If this is not specified, a solid color progress bar is displayed	Styled box	

start	Start value of the progress bar. Should not be set manually	Numeric	
end	End value of the progress bar. Should not be set manually	Numeric	
value	Current value of the progress bar. Should not be set manually	Numeric	
visible	Show or hide the progress bar	Boolean	true

Circular Progress Bar

Component name: circular_progress

Displays a circular progress bar that indicates the time remaining until the highlighted menu item is booted. The table below describes each item of the progress bar's property list:

Property Name	Description	Value Type	Default Value
left	Sets the left position of the progress bar	Numeric	
top	Sets the top position of the progress bar	Numeric	
width	Sets the width of the progress bar	Numeric	
height	Sets the height of the progress bar	Numeric	
id	Identifier for the progress bar. This must be set to **__timeout__**	String	
center_bitmap	Image to show at the center of the progress bar	Image file	
tick_bitmap	Image to use for the tick marks	Image file	
num_ticks	Number of ticks for a full circle	Numeric	64
start_angle	Position of first tick mark to disappear or appear	Numeric	-64
ticks_disappear	Sets whether ticks should disappear or appear. Set to false for ticks to disappear or true for ticks to appear	Boolean	
start	Start value of the progress bar. Should not be set manually	Numeric	
end	End value of the progress bar. Should not be set manually	Numeric	
value	Current value of the progress bar. Should not be set manually	Numeric	
visible	Show or hide the progress bar	Boolean	true

Vbox

Component name: vbox

The vbox component is a *container* component which lays out other components within it vertically starting from the top. It sets the width of each component to the width of the widest component. The components retain their height. The table below describes each item of the vbox's property list:

Property Name	Description	Value Type	Default Value
left	Sets the left position of the vbox	Numeric	
top	Sets the top position of the vbox	Numeric	
id	Identifier for the vbox	String	

Hbox

Component name: hbox

The hbox component is a *container* component which lays out other components within it horizontally starting from the left. It sets the height of each component to the height of the tallest component. The table below describes each item of the hbox's property list:

Property Name	Description	Value Type	Default Value
left	Sets the left position of the hbox	Numeric	
top	Sets the top position of the hbox	Numeric	
id	Identifier for the hbox	String	

Canvas

Component name: canvas

The canvas component is a *container* component which allows for the placement of other components within it according to their individual positions and sizes. Unlike the vbox and hbox components, it does not change the components sizes. The table below describes each item of the canvas' property list:

Property Name	Description	Value Type	Default Value
left	Sets the left position of the canvas	Numeric	
top	Sets the top position of the canvas	Numeric	
width	Sets the width of the canvas	Numeric	
height	Sets the height of the canvas	Numeric	
id	Identifier for the canvas	String	

A component property's default value is applied only if the respective property is excluded from the component statement line.

Specifying values for different value type

Property values may be specified differently for certain properties. Understanding the different ways of specifying these values is key to making your theme display properly.

Numeric Values

There are three ways to specify a numerical value:

- Absolute values – 100, 150, 1280, etc
- Relative values - 15%, 25%, 60% etc.
- A combination of both absolute and relative values - 25%-154, 256+15%

String Values

A string is any bit of text enclosed within double-quotes, eg:

- "Some bit of text"
- "My GRUB Theme"

Font name

GRUB uses a new font format called **PFF2** which is a bitmap font used only by GRUB. Since these fonts are not readily available, any required fonts must be created using the *grub-mkfont* utility. This utility will convert **BDF**, **PCF** and **TTF** fonts to the new **PFF2** font format. The file extension for **PFF2** fonts is **pf2**.

In this guide, we will use this command to create our grub fonts:

```
sudo grub-mkfont --verbose --range=0x0-0x7F --size=SIZE --
output=OUTPUT_FILENAME FONT_FILE
```

where:

- --verbose – show verbose messages
- --size – set the point size of the converted font

- --output – set the output filename for the created font
- --range=0x0-0x7F – create only ASCII characters
- FONT_FILE – font to be converted to PFF2

Fedora 14 and openSUSE 11.3 users must use the *grub2-mkfont* utility.

The *font name* specified in your *theme definition file* is the value of the **Font Name** property returned by the *grub-mkfont* utility and not the filename of the created font. The name is specified as a string value.

Boolean

A boolean value is specified as either *true* or *false*.

Color value

There are three ways to specify a color value:

- HTML Style value - "#RRGGBB" where RR, GG and BB are hexadecimal values (00-FF) representing the levels of red, green and blue respectively.
- Comma separated decimal RGB value - "R, G, B" where R, G and B are decimal values (0-255) representing the levels of red, green and blue respectively.
- SVG 1.0 color names, eg: "brown". Valid color names are listed in Appendix A.

Image File

An absolute path to a valid image file. Valid image formats are; PNG, TGA or JPG. Transparency is supported for image formats that support it.

Styled Box

A styled box contains 9 rectangular regions or slices that's used to draw the image on the screen. The illustration below shows the regions or slices of the styled box. The slices must be saved using the following format:

imagename_slicename.ext

*The slice's name **(slicename)**, are shown in brackets in the illustration below. The imagename must be the same for all slices.*

Northwest Slice (nw)	North Slice (n)	Northeast Slice (ne)
West Slice (w)	Center Slice (c)	East Slice (e)
Southwest Slice (sw)	South Slice (s)	Southeast Slice (se)

In order to take full advantage of the styled box, you must understand how the slices are rendered to produce the image on the screen.

First, the dimensions of the styled box are set using the values set for the width and height property. The slices are then scaled to fit these dimensions using the following rules:

- The corner slices (nw, ne, sw and se) are not scaled. They are rendered using the image's native dimensions.
- The East and West slices are scaled vertically to fit exactly between their respective corner slices. The images retain their native widths.
- The North and South slices are scaled horizontally to fit exactly between their respective corner slices. The images retain their native heights.
- The center slice is scaled to fill the empty area between the edge and corner slices.
- Any missing slice, or slices are left empty.

Demo Theme

In this chapter, we will start by creating a very basic theme. We will then advance the theme to include all of the components of the gfxmenu theming engine.

This chapter calls for rebooting your computer quite often. This can become very tedious and time-consuming. I would recommend that you install a virtual machine (VM) and then install the guess OS that you would like to use for testing (provided of course that you have the available resources). The actual theme creation can be done on the host machine and then the files simply copied to the guest OS in the VM. This way, you only need to reboot the guest OS in the VM as opposed to rebooting your physical machine.

A Basic Theme

First create a directory for the theme's files. We will name this directory *ubuntu* and it will be located in the */boot/grub/themes* or the */boot/grub2/themes* directory. Create the directory using this command:

```
sudo mkdir -p /boot/grub/themes/ubuntu
```

Copy the desktop and terminal images to the newly created *ubuntu* directory using this command:

```
sudo cp ~/Documents/grub_guide/demo/{desktop,terminal_*}.png
/boot/grub/themes/ubuntu/
```

Create a *theme definition file* in this directory:

```
gksudo gedit /boot/grub/themes/ubuntu/theme.txt &
```

Fedora 14 and openSUSE 11.3 users must replace */boot/grub* with */boot/grub2* in the above commands.

Add the following lines to the *theme definition file*:

```
1  # GRUB2 gfxmenu Ubuntu Demo theme
2  # The Definitive Guide to Theming GRUB2,
3  # using the ubuntu1 demo theme from http://www.gibibit.com
4  # Designed for 640x480 resolution
5
6  # Global Property
7  title-text: "GRUB2 Demo Theme"
8  title-font: "DejaVu Sans Regular 12"
9  title-color: "#B08458"
10 message-font: "DejaVu Sans Regular 12"
```

```
11  message-color: "#FFFFFF"
12  message-bg-color: "#020202"
13  desktop-image: "desktop.png"
14  desktop-color: "#000000"
15  terminal-box: "terminal_*.png"
16  terminal-font: "Fixed Regular 13"
17
18  # Show the boot menu
19  + boot_menu {
20      left = 96
21      top = 178
22      width = 544
23      height = 202
24      item_font = "DejaVu Sans Bold 14"
25      item_color = "#B08458"
26      selected_item_color = "#FFFFFF"
27      item_height = 32
28      item_padding = 0
29      item_spacing = 3
30  }
```

openSUSE 11.3 users must replace line 16 with:

```
16  terminal-font: "Misc Fixed Regular 13"
```

Save the file but do not close gedit.

We must now tell GRUB the location of our theme definition file.

Open the */etc/default/grub* file and add the following lines:

```
3
4   GRUB_THEME=/boot/grub/themes/ubuntu/theme.txt
5
```

Fedora 14 and openSUSE 11.3 users must replace */boot/grub* with */boot/grub2*.

This tells GRUB to use the theme definition file '**theme.txt**' located in the */boot/grub/themes/ubuntu* directory. To change the theme, change this line to point to the new theme definition file.

Create the required fonts

The DejaVu Sans fonts will be created from the DejaVuSans truetype font which is located in a directory under the */usr/share/fonts* directory. The actual location will depend on your distribution.

The Fixed Regular 13 font will be created from the 7x13.pcf.gz font which is located in the */usr/share/fonts/X11/misc* directory. Again, the actual location will depend on your distribution.

We will save the converted fonts in the */boot/grub/fonts* directory.

Installing the 7x13.pcf.gz font on Fedora 14

On Fedora 14, the 7x13.pcf.gz font is not installed by default. Users must install it using this command:

```
su -c "yum install xorg-x11-fonts-misc"
```

Creating the PFF2 fonts

We must first create a *fonts* directory for our font files. This directory will be located under the */boot/grub* directory. For Fedora 14 and openSUSE 11.3, it will be located under the */boot/grub2* directory.

Create the */boot/grub/fonts* directory with this command:

```
sudo mkdir /boot/grub/fonts
```

Create the PFF2 fonts using these commands:

```
sudo grub-mkfont --verbose --range=0x0-0x7F --size=10
--output=/boot/grub/fonts/dejavu-sans-10.pf2 $(locate DejaVuSans.ttf)
```

```
sudo grub-mkfont --verbose --range=0x0-0x7F --size=12
--output=/boot/grub/fonts/dejavu-sans-12.pf2 $(locate DejaVuSans.ttf)
```

```
sudo grub-mkfont --verbose --range=0x0-0x7F --size=14
--output=/boot/grub/fonts/dejavu-sans-bold-14.pf2 $(locate DejaVuSans-
Bold.ttf)
```

```
sudo grub-mkfont --verbose --size=13 --output=/boot/grub/fonts/7x13.pf2 $
(locate 7x13.pcf.gz)
```

The above commands uses 'command substitution' to locate the required font files regardless of their locations across distributions. If the 'locate' command fails because it's not installed, then you must replace $(locate...) with the absolute path to the font.

Using the grub-mkfont utility in GRUB version 1.98+20100804 may generate errors when converting truetype fonts. As a workaround, you may use the grub-mkfont utility from GRUB version 1.98.

Fedora 14 and openSUSE 11.3 users must make the appropriate substitutions in the above commands.

GRUB expects the fonts specified in the theme definition file to be located in the theme's directory. We can simply copy the required fonts from the */boot/grub/fonts* directory to the theme's directory, or we can create a symbolic link to the required fonts. The latter allows us to have a central location for all our fonts and prevents duplicate files. We will use this method in this guide.

To create a link to the required fonts, use these commands:

```
sudo ln -s /boot/grub/fonts/dejavu-sans-10.pf2 /boot/grub/themes/ubuntu/
```

```
sudo ln -s /boot/grub/fonts/dejavu-sans-12.pf2 /boot/grub/themes/ubuntu/
```

```
sudo ln -s /boot/grub/fonts/dejavu-sans-bold-14.pf2 /boot/grub/themes/ubuntu/
```

```
sudo ln -s /boot/grub/fonts/7x13.pf2 /boot/grub/themes/ubuntu/
```

Finally, update GRUB's configuration script using this command:

```
sudo update-grub
```

Sabayon users must use this command:

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

Fedora and openSUSE users must use this command:

```
su -c "grub2-mkconfig -o /boot/grub2/grub.cfg"
```

Reboot and GRUB's new graphical menu should look like this:

Screenshot 3: A Basic GRUB Graphical Menu

Theme Resolution

Our theme was designed for a screen resolution of 640x480 pixels. How would our theme display at a different resolution?

GRUB's resolution is set by the GRUB_GFXMODE variable in the */etc/default/grub* configuration file. Set the resolution to 1024x768 by changing this line to:

```
GRUB_GFXMODE=1024x768
```

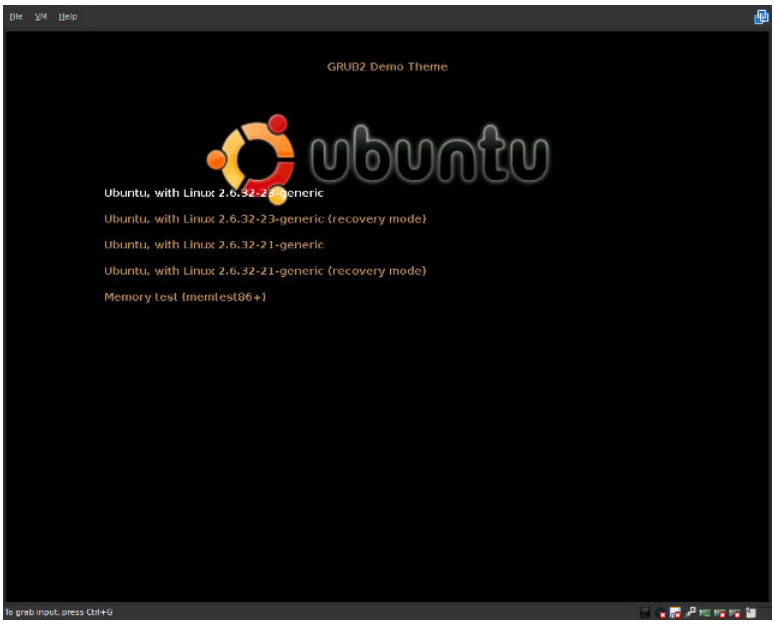
Fedora 14 and openSUSE 11.3

Fedora and openSUSE must add these lines to the end of their */etc/default/grub* script file:

```
14
15 # The resolution used on graphical terminal
16 # note that you can use only modes which your graphic card supports via VBE
17 # you can see them in real GRUB with the command `vbeinfo'
18 GRUB_GFXMODE=1024x768
```

Update GRUB's configuration script and reboot.

The graphical menu should look like this:



Screenshot 4: GRUB's Graphical Menu at 1024x768

The basic demo theme was designed for a screen resolution of 640x480 pixels. Notice that the boot menu is not centered anymore. That's because the values specified for the left, top, width and height properties were absolute values given in pixels.

If relative values are specified for these properties, the theme becomes *resolution independent*. These properties are given relative values by setting them as a percentage of GRUB's resolution.

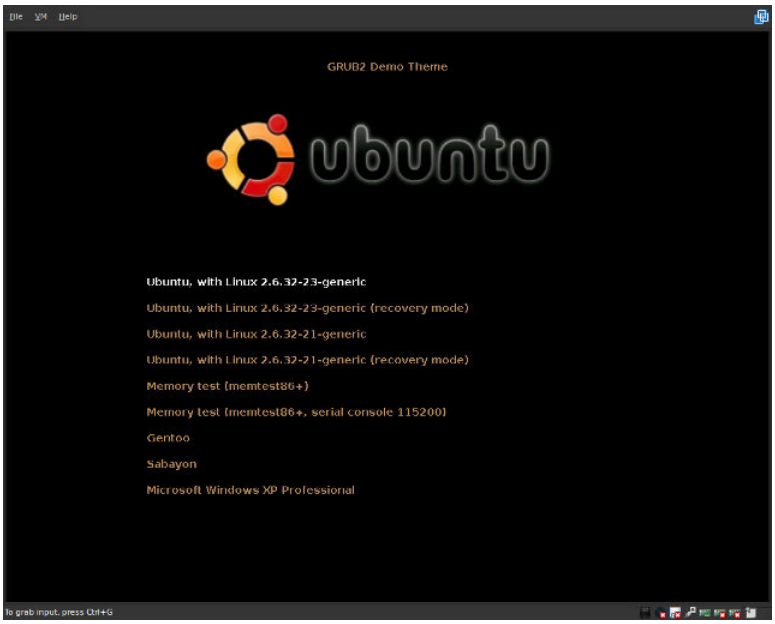
Open the theme definition file:

```
gksudo gedit /boot/grub/themes/ubuntu/theme.txt &
```

Change the following lines:

```
4   # Designed for any resolution
            .
            .
            .
20      left = 15%
21      top = 37%
22      width = 70%
23      height = 42%
```

Save the changes and reboot. The menu now looks like this at a screen resolution of 1024x768:



Screenshot 5: Resolution Independent Graphical Menu

This is all that is required for creating a basic theme. In the next section, we will advance the theme further by adding more properties to the boot menu and more components to the theme.

Reset GRUB's resolution to 640x480 and remember to update GRUB's configuration script file.

An Advanced Theme

Adding Icons to the boot menu

We can set the graphical menu to display an icon at the beginning of the line of each menu item. How does GRUB know which icon to use? It looks at the value of the *class* parameter on the *menuentry* line of GRUB's configuration script file:

```
menuentry 'Ubuntu Karmic' --class ubuntu --class gnu-linux --class gnu
--class os {
```

It then looks for an icon with that name in the icons directory and uses that icon for the menu item. If multiple *class* parameters are specified as shown above, it iterates the list and uses the first icon found whose name matches the *class* parameter.

The icons directory must be located under the theme's directory.

We will further customize the boot menu by setting:

- the icon's width and height to 32 pixels
- vertical spacing between each icon to 10 pixels
- the horizontal spacing between the icon and the menu item to 25 pixels.

Open the *theme definition file* and change the item spacing:

```
29      item_spacing = 10
```

Add the following lines to set the icon's width, height and horizontal spacing:

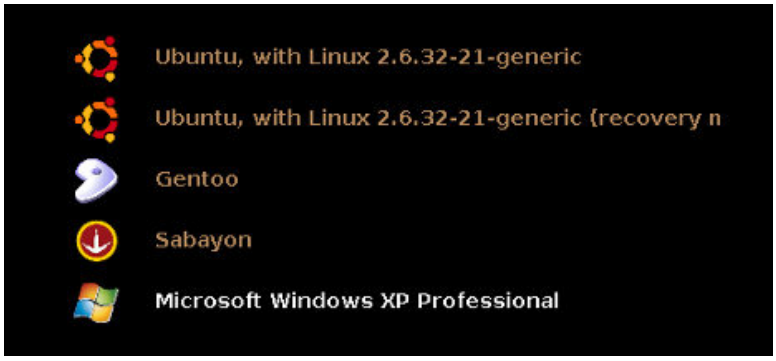
```
30      icon_width = 32
31      icon_height = 32
32      icon_spacing = 25
33  }
```

Save the file.

Copy the icons directory to the */boot/grub/themes/ubuntu* directory.

```
sudo cp -r ~/Documents/grub_guide/icons /boot/grub/themes/ubuntu/
```

Reboot and the menu should look like this with the icons:



Screenshot 6: Icons for each menu items

Fedora 14 users may not see the Fedora icons. Please see Appendix C to correct this. Also, users with other OS's installed, you may not see their icons. We will correct this in the final chapter.

Adding an image for the selected item of the boot menu

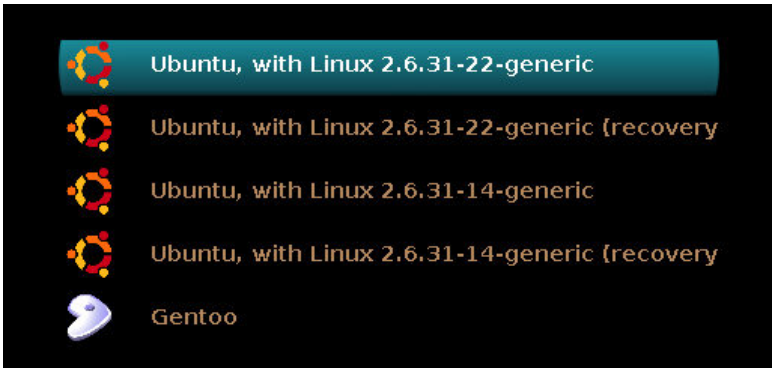
Let's add an image to highlight the selected menu item. Copy the required images to the theme's directory:

```
sudo cp ~/Documents/grub_guide/demo/select_*.png /boot/grub/themes/ubuntu/
```

Open the theme definition file and add the following lines:

```
33      selected_item_pixmap_style = "select_*.png"
34  }
```

Save the file and reboot. The menu should look like this:



Screenshot 7: Highlight image for selected menu item

Adding a background image to the boot menu

We will now add a background image to the boot menu. Copy the required images to the theme's directory:

```
sudo cp ~/Documents/grub_guide/demo/menu_bkg_*.png /boot/grub/themes/ubuntu/
```

Open the *theme definition file* and add the following lines:

```
34      menu_pixmap_style = "menu_bkg_*.png"
35  }
```

Save the file and reboot. The menu should look like this:

Screenshot 8: Background image for the boot menu

Adding a Scrollbar to the boot menu

We will now add the final property to the boot menu, the scrollbar. The scrollbar is shown only if the number of menu items cannot fit into the height set for the boot menu, provided of course that the *scrollbar* property is set to true.

The scrollbar has two parts:

- the frame, which include the borders, track and arrows, and
- the thumb.

Copy the required images to the theme's directory:

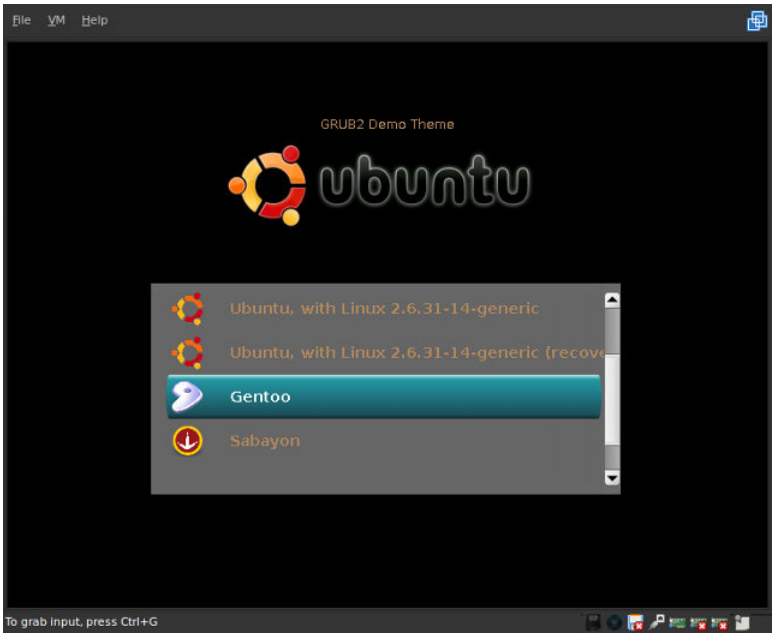
```
sudo cp ~/Documents/grub_guide/demo/sb_*.png /boot/grub/themes/ubuntu/
```

Open the *theme definition file* and add the following lines:

```
35      scrollbar = true
36      scrollbar_width = 16
37      scrollbar_thumb = "sb_th_*.png"
38      scrollbar_frame = "sb_fr_*.png"
39  }
```

*An important point to remember about the scrollbar is that it maps directly on top of the east slice **only** of the boot menu background image, occupying the entire slice (width and height). Therefore, the width of the scrollbar should be the exact width of the east slice of the boot menu background image. If it is narrower, it is scaled up to fit the width, but if it is wider, it is cropped.*

Save the changes and reboot. The menu should look like this:



Screenshot 9: GRUB's Boot Menu with Scrollbar

Progress Bar

The progress bar indicates the time remaining before the highlighted menu item is booted. It does this by providing a graphical indication via a bar that increases in size as the time decreases and optionally, via a countdown timer.

The graphical menu supports three types of progress bar:

- a solid color horizontal progress bar
- a styled horizontal progress bar, and
- a circular progress bar. This progress bar does not support the display of the countdown timer.

The format of the countdown timer is set by the value of the text property. It can take one of three formats:

Text Property Value	Countdown Timer Format
@TIMEOUT_NOTIFICATION_LONG@	The highlighted entry will be executed automatically in xxs.
@TIMEOUT_NOTIFICATION_MIDDLE@	xxs remaining.
@TIMEOUT_NOTIFICATION_SHORT@	xxs

where xx is the time remaining in seconds.

The progress bar is the only component whose *id* property **must** be set to a predefined value:

```
    id = "__timeout__"
```

Circular Progress Bar

The circular progress bar requires two images:

- the center image (the image placed at the center of the progress bar), and
- the tick image (the image used for the tick marks).

The center image is scaled to the dimensions set by the *width* and *height* properties while the tick image is shown at it's native size. For the tick marks to be rendered properly, set the width and height properties to the same value.

The ticks are placed along the circumference of a circle whose diameter is set by the width property. The center of this circle is the center of the center image.

We will create a circular progress bar with the following properties:

- Set the width and height to 100 pixels
- Make the ticks disappear at the rate of one tick per second. Remember our timeout value was set to 30 seconds (GRUB_TIMEOUT=30).

Copy the required images to the theme's directory:

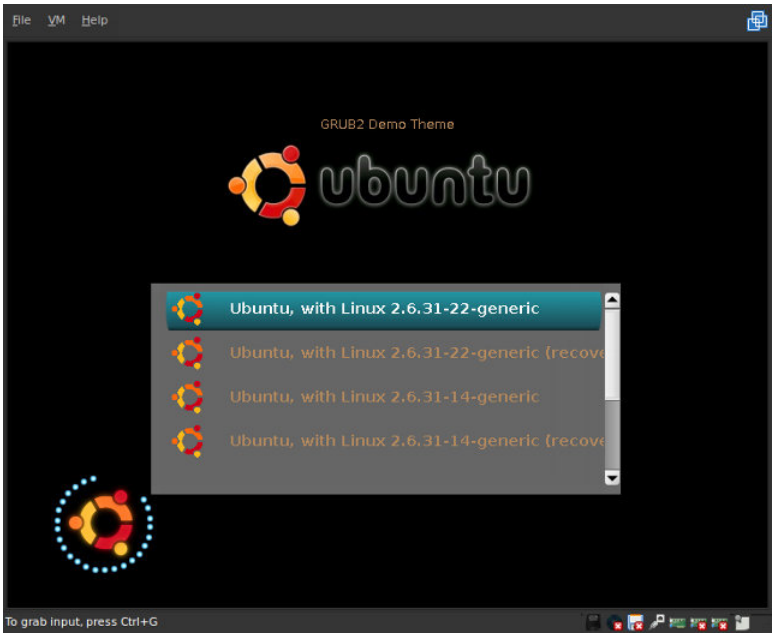
```
sudo cp ~/Documents/grub_guide/demo/{center,tick}.png
/boot/grub/themes/ubuntu/
```

Open the *theme definition file* and add the following lines:

```
40
41  # Show a circular progress bar
42  + circular_progress {
43      id = "__timeout__"
44      left = 0
45      top = 380
46      width = 100
47      height = 100
48      num_ticks = 30
```

```
49      start_angle = -56
50      ticks_disappear = true
51      center_bitmap = "center.png"
52      tick_bitmap = "tick.png"
53  }
```

Save the changes and reboot. The circular progress bar should look like this:



Screenshot 10: GRUB's Circular Progress Bar

Solid Color Horizontal Progress Bar

A solid color horizontal progress bar is a very basic progress bar with a foreground color which is the part of the progress bar that increases in size as the time decreases, a background color and border color which is a 1 pixel wide line that encloses the background color.

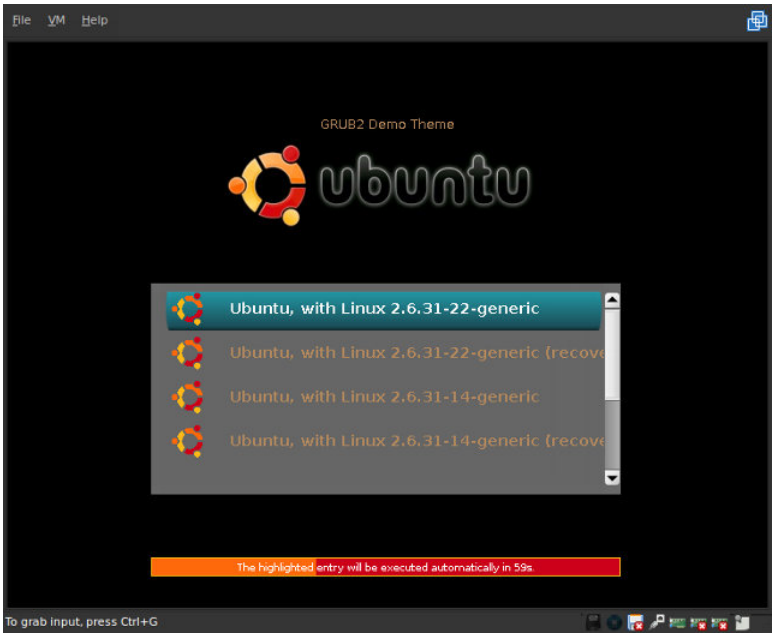
Let's create a solid color horizontal progress bar with the following properties:

- Resolution independent
- Align the left edge with, and make it as wide as the boot menu
- 18 pixels high
- Always at the lower edge of the screen
- Show a countdown timer on it using the DejaVu Sans font set to 10 points.

Open the *theme definition file*. Delete lines 41 to 53 and add the following lines:

```
41  # Show a solid color horizontal progress bar
42  + progress_bar {
43      id = "__timeout__"
44      left = 15%
45      top = 100%-18
46      width = 70%
47      height = 18
48      bg_color = "201, 0, 22"
49      fg_color = "255, 99,9"
50      border_color = "255, 181, 21"
51      show_text = true
52      font = "DejaVu Sans Regular 10"
53      text_color = "white"
54      text = "@TIMEOUT_NOTIFICATION_LONG@"
55  }
```

Save the changes and reboot. The boot screen should now look like this:



Screenshot 11: GRUB's Solid Color Progress Bar

Styled Horizontal Progress Bar

A styled horizontal progress bar requires two images:

- the background image, and
- the highlight image (the part of the progress bar that grows as the timer counts down).

We will create a styled horizontal progress bar with the same properties that was set for the solid color horizontal progress bar.

Copy the required images to the theme's directory:

```
sudo cp ~/Documents/grub_guide/demo/progress_*.png /boot/grub/themes/ubuntu/
```

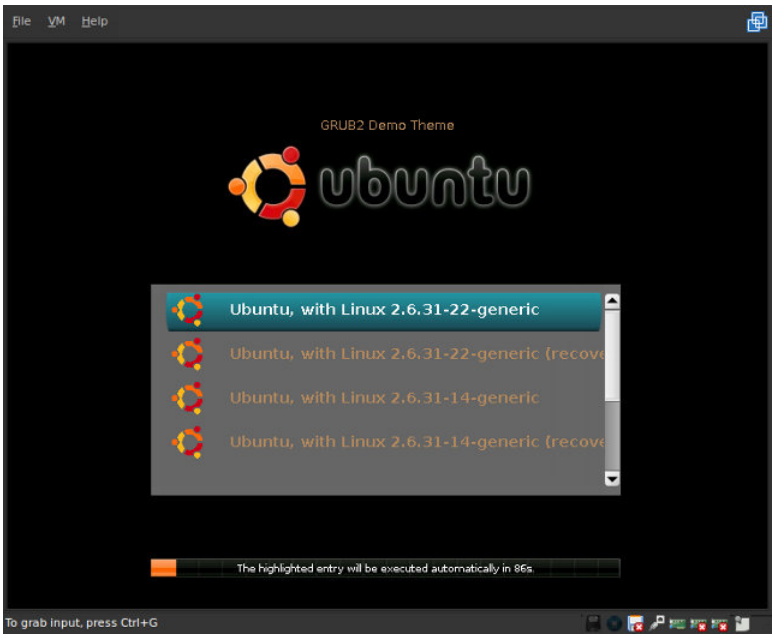
Open the *theme definition file* and change line 41 to:

```
41  # Show a styled horizontal progress bar
```

And add the following lines:

```
55      bar_style = "progress_bar_*.png"
56      highlight_style = "progress_highlight_*.png"
57  }
```

Save the changes and reboot. The boot screen should now look like this:



Screenshot 12: GRUB's Styled Progress Bar

Label

The label component displays a single line of text on the screen. GRUB will calculate the values for either the width or height properties if they are not explicitly set.

Let's create a label with the following properties:

- Make it resolution independent
- Center the text at the top of the screen
- Use the DejaVu Sans Bold font set to 14 point to show the text, and
- Set the color of the text to blue.

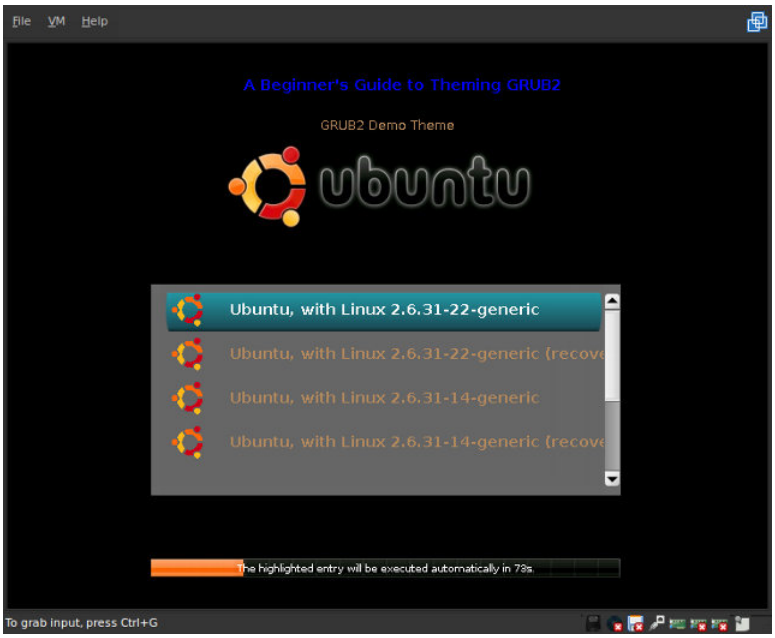
Open the *theme definition file* and add the following lines:

```
58
59  # Show some text at the top of the screen
60  + label {
61      top = 0
62      left = 5%
63      width = 95%
64      text = "The Definitive Guide to Theming GRUB2"
65      color = "blue"
66      font = "DejaVu Sans Bold 14"
67      align = "center"
68  }
```

If the width is manually set and the text does not fit within it, and the align property is set to:

- ***left**: the end of the line is truncated to fit within the set width*
- ***right**: the beginning of the line is truncated to fit within the set width*
- ***center**: the entire line is truncated (it does not show)*

Save the changes and reboot. The boot screen should now look like this:



Screenshot 13: Displaying text using the Label component

Image

The image component displays a bitmap image on the screen. The image is scaled to fit the dimensions set by the *width* and *height* properties. If these are not set, the image is displayed at it's native size.

Let's display the kubuntu icon (from the icons directory) with the following properties:

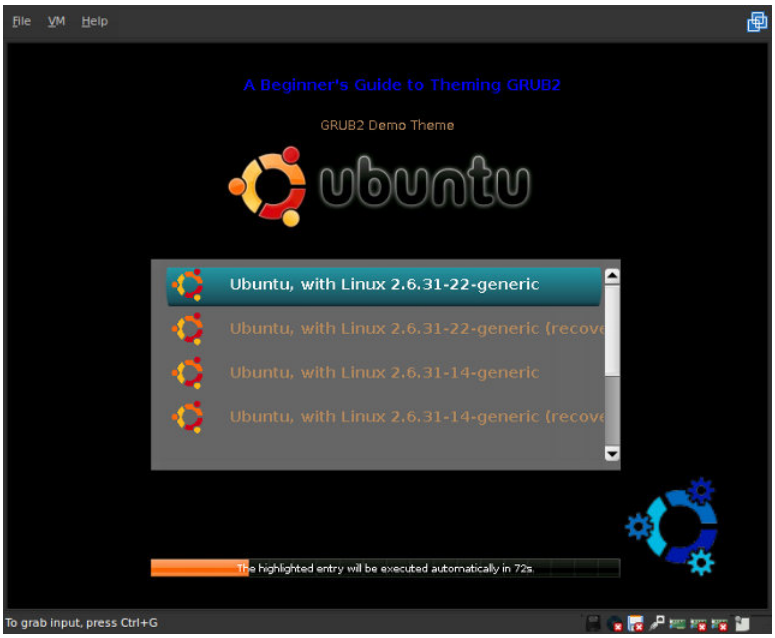
- Place it at the lower right corner of the screen, and
- Set it's size to 96x96 pixels.

Open the *theme definition file* and add the following lines:

```
69
70  # Show the kubuntu icon from the /boot/grub/themes/ubuntu/icons/ directory
71  + image {
72      top = 100%-96
73      left = 100%-96
74      width = 96
75      height = 96
76      file = "/boot/grub/themes/ubuntu/icons/kubuntu.png"
77  }
```

Fedora 14 and openSUSE 11.3 users must replace grub with grub2 on line 76.

Save the file and reboot. The boot screen should now look like this:



Screenshot 14: Adding an image using the Image component

Container Components

The graphical menu has a special component referred to as a *container* component. Think of it as a parent component which contains one or more child components.

The graphical menu has three container components:

1. The vbox, which arranges it's children vertically
2. The hbox, which arranges it's children horizontally
3. The canvas, which arranges it's children according to their individual preferences.

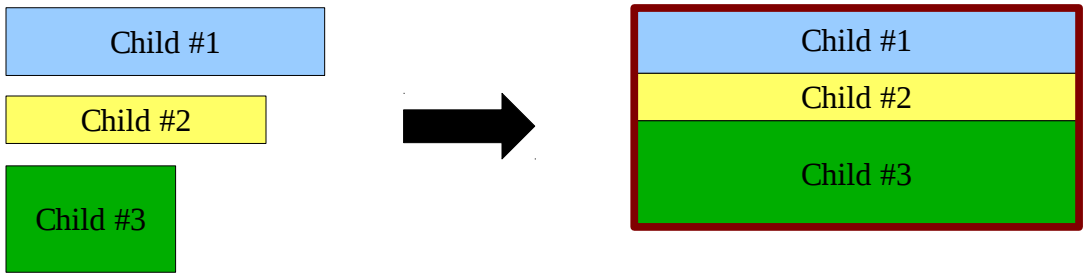
Vbox

The vbox container component lays out it's children from top to bottom. The width of it's children is set to the width of the widest child while each child maintains it's individual height.

If they are not explicitly set, the *width* and *height* properties of the vbox is calculated thus:

- the width is set to the width of the widest child
- the height is set to the sum of the heights of it's children

This is shown in the following illustration:

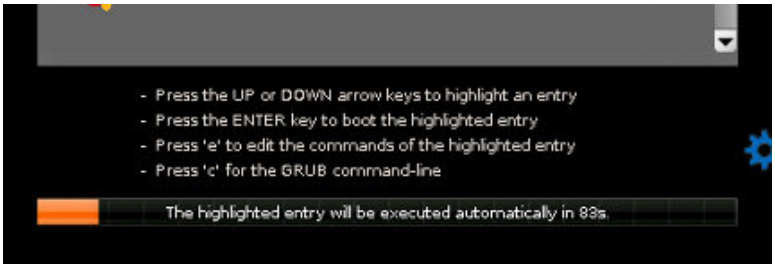


Children **Children arranged in vbox**

Let's use the vbox to show four lines of text below the boot menu. Open the *theme definition file* and add the following lines:

```
78
79  # Show an informational message below the boot menu
80  + vbox {
81      left = 25%
82      top = 81%
83      + label { text = "- Press the UP and DOWN arrow keys to highlight an
    entry" color = "#E6DDD5" font = "DejaVu Sans Regular 10" }
84      + label { text = "- Press the ENTER key to boot the highlighted entry"
    color = "#E6DDD5" font = "DejaVu Sans Regular 10" }
85      + label { text = "- Press 'e' to edit the commands of the highlighted
    entry" color = "#E6DDD5" font = "DejaVu Sans Regular 10" }
86      + label { text = "- Press 'c' for the GRUB command-line" color =
    "#E6DDD5" font = "DejaVu Sans Regular 10" }
87  }
```

Save the changes and reboot. The boot screen should now look like this:



Screenshot 15: Displaying a message using the vbox component

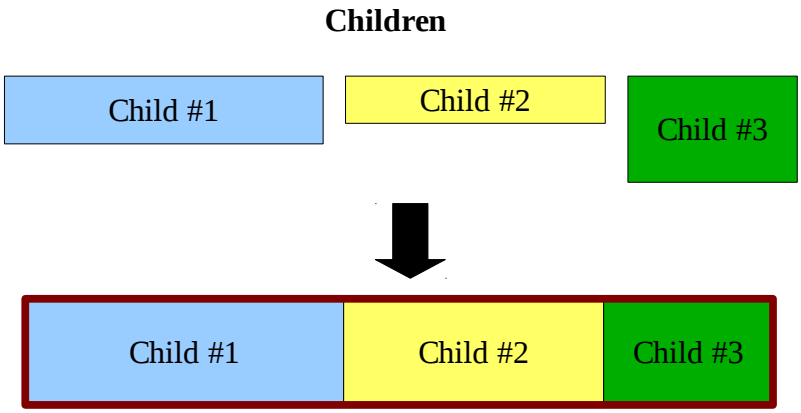
Hbox

The hbox container component lays out it's children from left to right. The height of it's children is set to the height of the tallest child while each child maintains it's individual width.

If they are not explicitly set, the *width* and *height* properties of the hbox is calculated thus:

- the height is set to the height of the highest child
- the width is set to the sum of the widths of it's children

This is shown in the following illustration:



Children arranged in hbox

Let's use the hbox to display an image with some text next to it.

Open the *~/Documents/grub_guide/demo/center.png* image in your image editor and scale it down to 24x24 pixels. Save it as *ubuntu-glow-24.png* in your *~/Documents/grub_guide/demo* directory.

Copy the *ubuntu-glow-24.png image* to the theme's directory:

```
sudo cp ~/Documents/grub_guide/demo/ubuntu-glow-24.png
/boot/grub/themes/ubuntu/
```

Open the *theme definition file* and add the following lines:

```
88
89  # Show the ubuntu logo with some text next to it
90  + hbox {
91      left = 48%
92      top = 28%
93      + image { width = 32 height = 32 file = "ubuntu-glow-24.png" }
94      + label { text = "Linux " color = "#FCB31D" font = "DejaVu Sans Regular
```

```
    10" }
95      + label { text = "for " color = "#FC6000" font = "DejaVu Sans Regular
    10" }
96      + label { text = "human beings" color = "#CD0C0F" font = "DejaVu Sans
    Regular 10" }
97  }
```

Save the changes and reboot. The boot screen should now look like this:

Screenshot 16: Using the hbox to show an image with text next to it

Canvas

The canvas container component is probably the most important component in the graphical menu because the top-level component of any theme is indeed, a canvas component.

The canvas component assigns each of it's child their respective positions and sizes. Each child is placed relative to the upper left corner of the canvas component. Unlike the vbox and hbox container components, the width and height properties must be explicitly set for the canvas component.

The components are placed along an imaginary z-axis with the first child component being the foremost and each successive child component placed one level beneath the preceding one.

Let's use the canvas component to place some images and text on the screen.

Open the following images from your *~/Documents/grub_guide/icons* directory in your image editor:

- debian.png
- gentoo.png
- kubuntu.png
- linuxmint.png
- opensuse.png
- sabayon.png
- ubuntu.png
- windows.png

Scale each one down to 24x24 pixels and save them to your *~/Documents/grub_guide/demo* directory using this format; *original_filename**-24**.png*.

Copy them to the theme's directory:

```
sudo cp ~/Documents/grub_guide/demo/*-24.png /boot/grub/themes/ubuntu/
```

Open the *theme definition file* and add the following lines:

```
98
99  # Show some images and text using the canvas component
100 + canvas {
101     left = 1
102     top = 1
103     width = 150
104     height = 150
105     + vbox {
106         top = 5
107         left = 5
108         + hbox {
109             + label { text = "GR" color = "green" font = "DejaVu Sans
    Regular 12" }
110             + label { text = "and" color = "#E6DDD5" font = "DejaVu Sans
    Regular 10" }
111         }
112         + hbox {
113             + label { text = "U" color = "green" font = "DejaVu Sans
    Regular 12" }
114             + label { text = "nified" color = "#E6DDD5" font = "DejaVu Sans
    Regular 10" }
115         }
116         + hbox {
117             + label { text = "B" color = "green" font = "DejaVu Sans
    Regular 12" }
118             + label { text = "ootloader" color = "#E6DDD5" font = "DejaVu
    Sans Regular 10" }
119         }
120     }
121     + image { left = 0 top = 100%-57 file = "windows-24.png" }
122     + image { left = 0 top = 100%-28 width = 28 height = 28 file =
    "opensuse-24.png" }
123     + image { left = 33 top = 100%-30 width = 30 height = 30 file =
```

```
    "linuxmint-24.png" }
124     + image { left = 68 top = 100%-32 width = 32 height = 32 file =
    "sabayon-24.png" }
125     + image { left = 105 top = 100%-34 width = 34 height = 34 file =
    "gentoo-24.png" }
126     + image { left = 100%-36 top = 100%-75 width = 36 height = 36 file =
    "kubuntu-24.png" }
127     + image { left = 100%-38 top = 100%-118 width = 38 height = 38 file =
    "ubuntu-24.png" }
128     + image { left = 25 top = 25 width = 100 height = 100 file = "debian-
    24.png" }
129 }
```

Save the changes and reboot. The boot screen now looks like this:

Screenshot 17: Using the canvas component

Distributing Your Theme

This chapter has been included to help the theme designer in distributing their themes. One of the more important aspect of distributing any theme is it's ease of installation for the end user.

A common practice is to include a small program that automates the installation process. This program should check the user's system to ensure that all the prerequisites for the successful execution of the theme are present. Another common practice is to include a text file with detailed instructions on installing the theme. This file is usually named README.

The packaged theme should be distributed in a compressed archive to reduce download size.

Let's package our demo theme for distribution. If you were following the guide from the beginning, all the files for the demo theme will be in the */boot/grub/themes/ubuntu* directory with the exception of the font files, which were symlinked to the actual files in the */boot/grub/fonts* directory.

Copy the */boot/grub/themes/ubuntu* directory to your *~/Documents* directory:

```
cp -rL --no-preserve=ownership /boot/grub/themes/ubuntu ~/Documents/
```

At this point you can simply distribute your theme and leave it to the end user to copy the files to their appropriate locations, specify the location of the *theme definition file* in their */etc/default/grub* configuration file and update GRUB's configuration script. But, this is not what we want. We want to make the installation of the theme as simple as possible to the end user.

An Installation Script

I have included a BASH script that does exactly this. You can distribute this script with your theme. The user will then execute the script to install the theme.

Copy the script to your *~/Documents/ubuntu* directory:

```
cp ~/Documents/grub_guide/install.sh ~/Documents/ubuntu/
```

Before we test the script, let's have a look at what it does. Open the script file for viewing in gedit:

```
gedit ~/Documents/ubuntu/install.sh &
```

Lines 21- 23:

```
21  Theme_Name="ubuntu"    # The theme will be installed in a dir with this
    name. Avoid spaces.
22  Theme_Definition_File="theme.txt"    # Filename of theme definition file.
```

```
23  Theme_Resolution="any"    # The resolution the theme was designed to show
    best at, 640x480, 1024x768 etc,
                              # or "any" for any resolution (resolution
    independent).
```

- Line 21 sets the theme's name. The script will install the theme to a directory with this name.
- Line 22 sets the name of the theme definition file.
- Line 23 sets the theme's resolution. If the theme is resolution independent set this to **'any'**, otherwise specify the resolution you would like to use for the theme.

These are the only parameters that you will have to change in the script.

Lines 33 - 36:

```
33  if [[ $(id -u) != 0 ]]; then
34    echo "Please run this script with root privileges."
35    exit 0
36  fi
```

The script must be run with root privileges. These lines check that the script is being run with root privileges and exits if it's not.

Lines 39 - 49:

```
39  for i in $Grub_Dist_Dirs; do
40    if [[ -d $i ]]; then
41      Grub_Dir=$i
42    fi
43  done
44
45  # Exit this script if we could not locate GRUB's installation directory.
46  if [[ -z $Grub_Dir ]]; then
47    echo "Could not locate GRUB's installation directory."
48    exit 0
49  fi
```

These lines attempt to locate GRUB's installation directory, which can be either /grub, /boot/grub or /boot/grub2, depending on the distribution. The script exits if it cannot locate either of these directories.

Lines 52 - 65:

```
52  if [[ -f $(which grub2-install) ]]; then
53    Grub_Version_Long=$(grub2-install --version)
54  elif [[ -f $(which grub-install) ]]; then
55    Grub_Version_Long=$(grub-install --version)
56  else
57    echo 'Could not locate grub-install or grub2-install in your path.'
58    exit 0
59  fi
60  Grub_Version=$(echo $Grub_Version_Long | sed 's,[[:alpha:][:punct:]
    [:blank:]],,g')
61  if (( ${Grub_Version:0:3} < Grub_Min_Version )); then
62    echo "GRUB must be at least version ${Grub_Min_Version:0:1}.$
    {Grub_Min_Version:1:2}."
63    echo "The installed version is ${Grub_Version:0:1}.${Grub_Version:1:2}."
64    exit 0
65  fi
```

These lines verify that the installed version of GRUB is at least 1.98 and exits if not.

Lines 68 - 71:

```
68  if [[ ! -f $Grub_File ]]; then
69    echo "Could not find $Grub_File"
70    exit 0
71  fi
```

These lines verify that the */etc/default/grub* configuration file exists. We need this file to tell GRUB where it can find the theme definition file. The script exits if it cannot locate the file.

Lines 74 - 75:

```
74  mkConfig_File=$(which ${Grub_Dir##*/}-mkconfig) || \
75  (echo "GRUB's mkconfig script file was not found in your path." && exit 0)
```

These lines attempt to find the grub-mkconfig (or grub2-mkconfig) script file. These files are located in different directories depending on the distribution. The script exits if it cannot locate the file.

These are the prerequisite checks the script does to ensure that the theme will install and display as intended.

The rest of the script installs the theme, set's the theme's resolution if specified and generates the new *grub.cfg* configuration script file if the user so wishes.

Testing The Script

Let's test the script and check that it works as intended.

First delete the existing /boot/grub/themes/ubuntu directory:

```
sudo rm -r /boot/grub/themes/ubuntu
```

Change your working directory to *~/Documents/ubuntu* and run the script:

```
cd ~/Documents/ubuntu
sudo ./install.sh
```

On successful completion, a new *grub.cfg* configuration script file is created and a reboot will show the new theme.

Packaging Your New Theme

There are several ways of packaging your theme. The best way is to create a DEB or RPM package which the user can use to install the theme. This method allows the user to automatically update the theme should you make any changes to it. However, such an exercise is beyond the scope of this guide.

We will look at packaging the new theme using a much simpler method, ie: creating a compressed archive. We will use the tar archive utility to create the compressed archive. Of course, you can also do this using the GUI archive manager, *File-Roller* for the Gnome desktop environment or *Ark* for KDE. Similar managers may be available for other desktop environments.

Create the archive with this command (from within your *~/Documents/ubuntu* directory):

```
tar acvf ~/Documents/ubuntu_grub_theme.tar.gz ../ubuntu/*
```

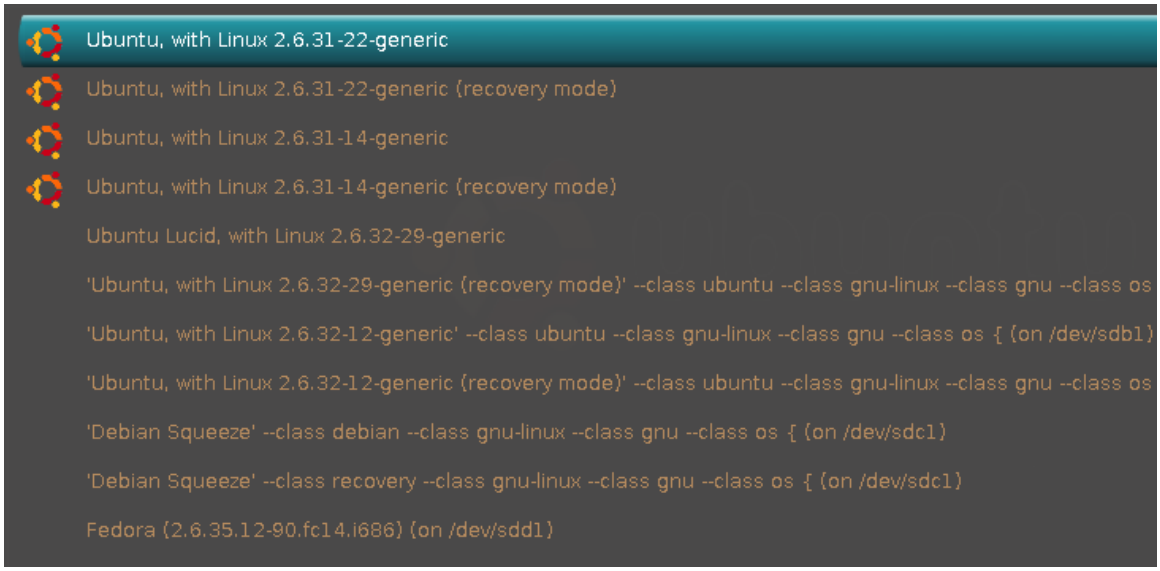
The files in the archive can be restored with:

```
tar xvf ~/Documents/ubuntu_grub_theme.tar.gz
```

Customizing The Menu Items

The final chapter in this guide will attempt to customize the text shown for the menu items. This should only be attempted if you are comfortable with editing the various GRUB script files as any mistakes here may prevent you from booting normally from GRUB's menu.

Why would we want to customize the menu items? Well, here is a screenshot of my GRUB menu:



Screenshot 18: GRUB's Menu Items

The menu includes kernels to boot Ubuntu Karmic, Lucid, Maverick and Kubuntu. There are also entries for Debian, Fedora, openSUSE, Sabayon and Windows XP (not shown). From the screenshot, how can I tell which entry is Lucid or Maverick, or even Kubuntu? Well I would need to remember exactly which kernel the respective releases use or, on which device the releases are installed. I do not find this very convenient. You will also notice that the icons are only shown for my host OS (the first four entries).

There are also entries for each kernel found on a specific distribution or release together with recovery mode entries.

The customizations in this chapter will give the user the following additional options:

- Show or hide the boot device in the menu items
- Show all kernel versions or only the latest one in the menu items
- Show or hide the kernel version in the menu items

Add New Variables To GRUB

Let's start by opening the *grub-mkconfig* script with root privileges:

```
gksudo gedit /usr/sbin/grub-mkconfig &
```

Fedora 14 and openSUSE 11.3 users must open the /usr/sbin/grub2-mkconfig script instead. On Sabayon 5.5, the grub-mkconfig script file is located in the /sbin directory.

Some commands use the back-tick (`) character, not to be confused with the single-quote (') character. The back-tick is shown using a cyan highlight.

Locate the lines that export GRUB's environment variables (line numbers may vary considerably across distributions):

```
236 # These are optional, user-defined variables.
237 export GRUB_DEFAULT \
238   GRUB_HIDDEN_TIMEOUT \
239   GRUB_HIDDEN_TIMEOUT_QUIET \
```

Add a line continuation character (\ ) to the last variable at line 266 in the list. The lines in green are shown for reference only and should not be changed (the last variable will vary across distribution):

```
266   GRUB_BADRAM \
267
268 if test "x${grub_cfg}" != "x"; then
269   rm -f ${grub_cfg}.new
```

Now add the following new variables to the end of the list:

```
267   GRUB_DISTRIBUTOR_CODENAME \
268   GRUB_INCLUDE_BOOT_DEVICE \
269   GRUB_ADD_ONLY_LATEST_KERNEL \
270   GRUB_INCLUDE_KERNEL_VERSION
```

Save the changes and open the */etc/default/grub* configuration file.

Add the GRUB_DISTRIBUTOR_CODENAME variable beneath the GRUB_DISTRIBUTOR variable. The lines in green are shown for reference only and should not be changed (line numbers may vary considerably across distributions):

```
12  GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || Debian`
13  GRUB_DISTRIBUTOR_CODENAME=`lsb_release -cs 2> /dev/null`
```

Add the following lines to the end of the file (line numbers may vary considerably across distributions):

```
38
39 # Uncomment to include the boot device in menu entries
40 #GRUB_INCLUDE_BOOT_DEVICE=true
41
42 # Uncomment to add only the latest kernel to the menu entries
43 #GRUB_ADD_ONLY_LATEST_KERNEL=true
44
45 # Uncomment to include the kernel version in the menu entries
46 #GRUB_INCLUDE_KERNEL_VERSION=true
```

Save the changes but do not close gedit.

Ubuntu Variants

For users of Ubuntu variants (Kubuntu, Lubuntu, etc.), replace the value of the GRUB_DISTRIBUTOR variable with the name of the variant (Kubuntu, Lubuntu etc.):

```
12 GRUB_DISTRIBUTOR="Kubuntu"
```

Sabayon 5.5

Add the following line beneath the GRUB_DISTRIBUTOR variable:

```
9  GRUB_DISTRIBUTOR_CODENAME="Fermi"
```

Change the value of the GRUB_DISABLE_LINUX_RECOVERY variable to **true**:

```
35 #GRUB_DISABLE_LINUX_RECOVERY="true"
```

Add lines 38 to 46 shown above to the end of the file. Save the changes but do not close gedit.

Fedora 14

Add the following lines to the end of the file:

```
19
21 # Uncomment to disable generation of recovery mode menu entries.
22 #GRUB_DISABLE_LINUX_RECOVERY="true"
```

Add lines 38 to 46 shown above to the end of the file. Save the changes but do not close gedit.

openSUSE 11.3

Add the following line beneath the GRUB_DISTRIBUTOR variable:

```
13 GRUB_DISTRIBUTOR_CODENAME="Teal"
```

Add the following lines to the end of the file:

```
19
20 # Uncomment to disable generation of recovery mode menu entries.
21 #GRUB_DISABLE_LINUX_RECOVERY="true"
```

Add lines 38 to 46 shown above to the end of the file. Save the changes but do not close gedit.

Customizing The Menu Items For The Host OS

Before any of the new options can take effect, we must modify the */etc/grub.d/10_linux* script file. This file generates the menu items for the host OS.

Open the file in gedit with root privileges. The lines in green are shown for reference only and should not be changed (line numbers may vary considerably across distributions).

Replace lines 34 and 35 with:

```
31 if [ "x${GRUB_DISTRIBUTOR}" = "x" ] ; then
32   OS=GNU/Linux
33 else
34   OS="`echo ${GRUB_DISTRIBUTOR} | cut -d' ' -f1` `echo $
   {GRUB_DISTRIBUTOR_CODENAME} | sed -e 's/^./\u&/'`"
35   CLASS1="--class $(echo ${GRUB_DISTRIBUTOR} | tr '[A-Z]' '[a-z]' | cut -d'
   ' -f1) ${CLASS}"
36 fi
```

Delete lines 70 to 74:

```
70   if ${recovery} ; then
71     title="$(gettext_quoted "%s, with Linux %s (recovery mode)")"
72   else
73     title="$(gettext_quoted "%s, with Linux %s")"
74   fi
```

Add the following lines:

```
69    args="$4"
70
71  # If we are including all kernels, then always show the kernel version to
    differentiate between them.
72    if [ "x$GRUB_ADD_ONLY_LATEST_KERNEL" != "xtrue" ] ||
    [ "x$GRUB_INCLUDE_KERNEL_VERSION" = "xtrue" ]; then
73      title="$(gettext_quoted "%s with Linux %s")"
74    else
75      title=$1
76   fi
77
78    if ${recovery}; then
79      CLASS="--class recovery ${CLASS1}"
80    else
81      CLASS="${CLASS1}"
82    fi
83
84    if [ "x$GRUB_INCLUDE_BOOT_DEVICE" = "xtrue" ]; then
85      title="${title} on $GRUB_DEVICE"
86    fi
87
88    printf "menuentry '${title}' ${CLASS} {\n" "${os}" "${version}"
```

For all distributions (except Debian based distributions), replace **gettext_quoted** with **gettext** on line 73.

Finally, add these lines:

```
164   if [ "x${GRUB_DISABLE_LINUX_RECOVERY}" != "xtrue" ] ; then
165     linux_entry "${OS}" "${version}" true \
166     "single ${GRUB_CMDLINE_LINUX}"
167   fi
168
169   if [ "x$GRUB_ADD_ONLY_LATEST_KERNEL" = "xtrue" ]; then
170     list=
171   else
172     list=`echo $list | tr ' ' '\n' | grep -vx $linux | tr '\n' ' '`
173   fi
```

```
174 done
```

Save the changes and update GRUB's configuration script. Reboot to see the changes.

openSUSE 11.3

Add the following lines to remove any archives of the kernel's development files from the kernel list. These files have an extension of **.gz**.

```
113 list=`for i in /boot/vmlinu[xz]-* /vmlinu[xz]-* ; do
114         if grub_file_is_not_garbage "$i" ; then echo -n "$i " ; fi
115       done`
116
117 # Remove all archives of kernel development files.
118 list=`echo $list | tr ' ' '\n' | sed '/gz$/ d' | tr '\n' ' '`
119
120 prepare_boot_cache=
```

Save the changes and update GRUB's configuration script. Reboot to see the changes.

The distribution's codename should now be shown on the menu items. Try uncommenting the newly added variables and rebooting to see the effect on the menu items. Remember to update GRUB's configuration script after any changes.

Customizing The Menu Items For Other OS's

The */etc/grub.d/30_os-prober* script file searches for any additional OS's on the system and adds them to the menu items. We must modify this file to include our newly added variables and show both the distribution's codename and icon in the menu items.

Open the file in gedit with root privileges.

```
gksudo gedit /etc/grub.d/30_os-prober &
```

Add the following lines. The lines in green are shown for reference only and should not be changed (line numbers may vary considerably across distributions).

Add lines 23 and 24:

```
20 prefix=/usr
21 exec_prefix=${prefix}
22 libdir=${exec_prefix}/lib
```

```
23  sysconfdir=/etc
24  GRUBFILE=${sysconfdir}/default/grub
25
26  . ${libdir}/grub/grub-mkconfig_lib
```

Add the following new function starting at line 30:

```
26  . ${libdir}/grub/grub-mkconfig_lib
27
28  found_other_os=
29
30  unmount_tmpdir () {
31    if grep -q ${tmpdir} /proc/self/mountinfo; then
32      umount ${tmpdir} 2> /dev/null
33    fi
34    if [ -d ${tmpdir} ]; then
35      rm -r ${tmpdir}
36    fi
37  }
38
```

Add the following new function starting at line 91:

```
87      make_timeout "${GRUB_HIDDEN_TIMEOUT}" "${GRUB_TIMEOUT}"
88    fi
89  }
90
91  get_distro_and_codename () {
92
93    mount ${DEVICE} ${tmpdir}
94
95    GRUB_DISTRO="`grep -s ^GRUB_DISTRIBUTOR= ${tmpdir}${GRUBFILE} | cut -d'='
    -f2 \
96      | tr -d '"\`' | sed -e 's,2>.*$,,'`"
97    GRUB_CODENAME="`grep -s ^GRUB_DISTRIBUTOR_CODENAME= ${tmpdir}${GRUBFILE}
    | cut -d'=' -f2 \
98      | tr -d '"\`' | sed -e 's,2>.*$,,'`"
99
100   if [ "x${GRUB_DISTRO}" != "x" ]; then
101     cmd=${GRUB_DISTRO}
```

```
102   else
103     cmd='lsb_release -is'
104   fi
105
106   set +e
107   rval_CHROOT=`chroot ${tmpdir} ${cmd} 2> /dev/null`
108   err_CHROOT=$?
109   DISTRIBUTOR=`echo ${rval_CHROOT} | cut -d' ' -f1 | tr -d '"'`
110   set -e
111
112   if [ "${err_CHROOT}" = "127" ]; then
113     if [ "x${GRUB_DISTRO}" = "x" ]; then
114       DISTRIBUTOR=${LONGNAME}
115     else
116       DISTRIBUTOR=${GRUB_DISTRO}
117     fi
118   fi
119   if [ "${err_CHROOT}" = "1" ]; then
120     DISTRIBUTOR=${LONGNAME}
121   fi
122
123   if [ "x${GRUB_CODENAME}" != "x" ]; then
124     cmd=${GRUB_CODENAME}
125   else
126     cmd='lsb_release -cs'
127   fi
128
129   set +e
130   rval_CHROOT=`chroot ${tmpdir} ${cmd} 2> /dev/null`
131   err_CHROOT=$?
132   CODENAME=`echo ${rval_CHROOT} | cut -d' ' -f1 | tr -d '"' | sed -e
    's,^.,\u&,'`
133   set -e
134
135   if [ "${err_CHROOT}" = "127" ]; then
136     CODENAME=${GRUB_CODENAME}
137   fi
138   if [ "${err_CHROOT}" = "1" ]; then
139     CODENAME=
```

```
140   fi
141
142   umount ${tmpdir}
143 }
144 if [ "x${GRUB_DISABLE_OS_PROBER}" = "xtrue" ]; then
144
```

Change the Mac OSX menuentry line at line 165 to:

```
165 menuentry "${LONGNAME} (${2}-bit) ${SHOWBOOTDEVICE}" --class macosx {
```

Add lines 214 and 215:

```
211 EOF
212 }
213
214 tmpdir="`mktemp -d --tmpdir mounted.XXXXXXXXXX`"
215 trap "unmount_tmpdir" EXIT HUP INT QUIT TERM
216
217 for OS in ${OSPROBED} ; do
```

Add the following lines starting at line 230:

```
227   echo "Found ${LONGNAME} on ${DEVICE}" >&2
228   found_other_os=1
229
230   if [ "x${GRUB_INCLUDE_BOOT_DEVICE}" != "x" ] ; then
231     SHOWBOOTDEVICE="on ${DEVICE}"
232   fi
233   CLASS="--class `echo ${LABEL%%[0-9]*} | tr [A-Z] [a-z]`"
234
235   case ${BOOT} in
```

Change line 239 to:

```
239 menuentry "${LONGNAME} ${SHOWBOOTDEVICE}" ${CLASS} {
```

At the following lines starting at line 263:

```
260       LINUXPROBED="`linux-boot-prober ${DEVICE} 2> /dev/null | tr ' ' '^' |
paste -s -d ' '`"
```

```
261       prepare_boot_cache=
262
263       if [ "x$GRUB_ADD_ONLY_LATEST_KERNEL" = "xtrue" ] ; then
264         echo "Adding only the latest kernel" >&2
265         LINUXPROBED="`echo ${LINUXPROBED} | cut -d' ' -f1`"
266       fi
267
268       get_distro_and_codename
269       CLASS1="--class `echo ${DISTRIBUTOR} | cut -d' ' -f1 | tr [A-Z] [a-
z]`"
270
271       for LINUX in ${LINUXPROBED} ; do
```

Add the following lines starting at line 279:

```
277         LPARAMS="`echo ${LINUX} | cut -d ':' -f 6- | tr '^' ' '`"
278
279         if `echo ${LLABEL} | tr [A-Z] [a-z] | grep -qE "(recovery|
failsafe)"`; then
280           CLASS="--class recovery ${CLASS1}"
281           if [ "x${GRUB_DISABLE_LINUX_RECOVERY}" = "xtrue" ]; then
282             continue
283           fi
284         else
285           CLASS=${CLASS1}
286         fi
287
288         if [ -z "${LLABEL}" ] ; then
```

Add the following lines starting at line 297:

```
294           LINITRD="${LINITRD#/boot}"
295         fi
296
297         TITLE="${DISTRIBUTOR} ${CODENAME}"
298
299         if [ "x$GRUB_ADD_ONLY_LATEST_KERNEL" != "xtrue" ] \
300           || [ "x$GRUB_INCLUDE_KERNEL_VERSION" = "xtrue" ] ; then
301           TITLE="${TITLE} with Linux `echo ${LKERNEL} | sed -e "s,^[^0-
9]*-,,g"`"
302         fi
```

```
303
```

Change line 305 to:

```
305 menuentry "${TITLE} ${SHOWBOOTDEVICE}" ${CLASS} {
```

Finally, change line 332 to:

```
332 menuentry "${LONGNAME} ${SHOWBOOTDEVICE}" --class hurd {
```

Save the changes and update GRUB's configuration script. Reboot to see the changes.

openSUSE 11.3

Users of openSUSE 11.3 must install the os-prober package as described in Appendix D.

Fedora 14

Fedora 14 users must install the os-prober package using this command:

```
su -c "yum install os-prober"
```

The distribution's codename should now be shown on the menu items. Try uncommenting the newly added variables and rebooting to see the effect on the menu items. Remember to update GRUB's configuration script after any changes.

These modifications depend heavily on the distribution being LSB compliant. For non-compliant distributions such as Sabayon 5.5 you must specify the distribution's name and codename via the GRUB_DISTRIBUTOR and GRUB_DISTRIBUTOR_CODENAME variables respectively.

The same can be done for variants of certain distributions such as Kubuntu, Lubuntu, etc. as stated previously. This ensures that the variant's name is shown on the menu items.

Here is a screenshot of my GRUB menu items after applying the above modifications:

Screenshot 19: Final GRUB Menu Items

Appendix A – Valid Color Names

The following table lists the SVG 1.0 color names together with their equivalent hexadecimal and decimal RGB values that GRUB recognizes:

Color Name	Hex RGB	Dec RGB	Color
aliceblue	#f0f8ff	240, 248, 255	
antiquewhite	#faebd7	250, 235, 215	
aqua	#00ffff	0, 255, 255	
aquamarine	#7fffd4	127, 255, 212	
azure	#f0ffff	240, 255, 255	
beige	#f5f5dc	245, 245, 220	
bisque	#ffe4c4	255, 228, 196	
black	#000000	0, 0, 0	
blanchedalmond	#ffebcd	255, 235, 205	
blue	#0000ff	0, 0, 255	
blueviolet	#8a2be2	138, 43, 226	
brown	#a52a2a	165, 42, 42	
burlywood	#deb887	222, 184, 135	
cadetblue	#5f9ea0	95, 158, 160	
chartreuse	#7fff00	127, 255, 0	
chocolate	#d2691e	210, 105, 30	
coral	#ff7f50	255, 127, 80	
cornflowerblue	#6495ed	100, 149, 237	
cornsilk	#fff8dc	255, 248, 220	
crimson	#dc143c	220, 20, 60	
cyan	#00ffff	0, 255, 255	
darkblue	#00008b	0, 0, 139	
darkcyan	#008b8b	0, 139, 139	
darkgoldenrod	#b8860b	184, 134, 11	
darkgray	#a9a9a9	169, 169, 169	
darkgreen	#006400	0, 100, 0	
darkgrey	#a9a9a9	169, 169, 169	
darkkhaki	#bdb76b	189, 183, 107	
darkmagenta	#8b008b	139, 0, 139	
darkolivegreen	#556b2f	85, 107, 47	
darkorange	#ff8c00	255, 140, 0	
darkorchid	#9932cc	153, 50, 204	
darkred	#8b0000	139, 0, 0	
darksalmon	#e9967a	233, 150, 122	
darkseagreen	#8fbc8f	143, 188, 143	

Color Name	Hex RGB	Dec RGB	Color
darkslateblue	#483d8b	72, 61, 139	
darkslategray	#2f4f4f	47, 79, 79	
darkslategrey	#2f4f4f	47, 79, 79	
darkturquoise	#00ced1	0, 206, 209	
darkviolet	#9400d3	148, 0, 211	
deeppink	#ff1493	255, 20, 147	
deepskyblue	#00bfff	0, 191, 255	
dimgray	#696969	105, 105, 105	
dimgrey	#696969	105, 105, 105	
dodgerblue	#1e90ff	30, 144, 255	
firebrick	#b22222	178, 34, 34	
floralwhite	#fffaf0	255, 250, 240	
forestgreen	#228b22	34, 139, 34	
fuchsia	#ff00ff	255, 0, 255	
gainsboro	#dcdcdc	220, 220, 220	
ghostwhite	#f8f8ff	248, 248, 255	
gold	#ffd700	255, 215, 0	
goldenrod	#daa520	218, 165, 32	
gray	#808080	128, 128, 128	
green	#008000	0, 128, 0	
greenyellow	#adff2f	173, 255, 47	
grey	#808080	128, 128, 128	
honeydew	#f0fff0	240, 255, 240	
hotpink	#ff69b4	255, 105, 180	
indianred	#cd5c5c	205, 92, 92	
indigo	#4b0082	75, 0, 130	
ivory	#fffff0	255, 255, 240	
khaki	#f0e68c	240, 230, 140	
lavender	#e6e6fa	230, 230, 250	
lavenderblush	#fff0f5	255, 240, 245	
lawngreen	#7cfc00	124, 252, 0	
lemonchiffon	#fffacd	255, 250, 205	
lightblue	#add8e6	173, 216, 230	
lightcoral	#f08080	240, 128, 128	
lightcyan	#e0ffff	224, 255, 255	

Color Name	Hex RGB	Dec RGB	Color
lightgoldenrodyellow	#fafad2	250, 250, 210	
lightgray	#d3d3d3	211, 211, 211	
lightgreen	#90ee90	144, 238, 144	
lightgrey	#d3d3d3	211, 211, 211	
lightpink	#ffb6c1	255, 182, 193	
lightsalmon	#ffa07a	255, 160, 122	
lightseagreen	#20b2aa	32, 178, 170	
lightskyblue	#87cefa	135, 206, 250	
lightslategray	#778899	119, 136, 153	
lightslategrey	#778899	119, 136, 153	
lightsteelblue	#b0c4de	176, 196, 222	
lightyellow	#ffffe0	255, 255, 224	
lime	#00ff00	0, 255, 0	
limegreen	#32cd32	50, 205, 50	
linen	#faf0e6	250, 240, 230	
magenta	#ff00ff	255, 0, 255	
maroon	#800000	128, 0, 0	
mediumaquamarine	#66cdaa	102, 205, 170	
mediumblue	#0000cd	0, 0, 205	
mediumorchid	#ba55d3	186, 85, 211	
mediumpurple	#9370db	147, 112, 219	
mediumseagreen	#3cb371	60, 179, 113	
mediumslateblue	#7b68ee	123, 104, 238	
mediumspringgreen	#00fa9a	0, 250, 154	
mediumturquoise	#48d1cc	72, 209, 204	
mediumvioletred	#c71585	199, 21, 133	
midnightblue	#191970	25, 25, 112	
mintcream	#f5fffa	245, 255, 250	
mistyrose	#ffe4e1	255, 228, 225	
moccasin	#ffe4b5	255, 228, 181	
navajowhite	#ffdead	255, 222, 173	
navy	#000080	0, 0, 128	
oldlace	#fdf5e6	253, 245, 230	
olive	#808000	128, 128, 0	
olivedrab	#6b8e23	107, 142, 35	
orange	#ffa500	255, 165, 0	
orangered	#ff4500	255, 69, 0	
orchid	#da70d6	218, 112, 214	
palegoldenrod	#eee8aa	238, 232, 170	
palegreen	#98fb98	152, 251, 152	

Color Name	Hex RGB	Dec RGB	Color
paleturquoise	#afeeee	175, 238, 238	
palevioletred	#db7093	219, 112, 147	
papayawhip	#ffefd5	255, 239, 213	
peachpuff	#ffdab9	255, 218, 185	
peru	#cd853f	205, 133, 63	
pink	#ffc0cb	255, 192, 203	
plum	#dda0dd	221, 160, 221	
powderblue	#b0e0e6	176, 224, 230	
purple	#800080	128, 0, 128	
red	#ff0000	255, 0, 0	
rosybrown	#bc8f8f	188, 143, 143	
royalblue	#4169e1	65, 105, 225	
saddlebrown	#8b4513	139, 69, 19	
salmon	#fa8072	250, 128, 114	
sandybrown	#f4a460	244, 164, 96	
seagreen	#2e8b57	46, 139, 87	
seashell	#fff5ee	255, 245, 238	
sienna	#a0522d	160, 82, 45	
silver	#c0c0c0	192, 192, 192	
skyblue	#87ceeb	135, 206, 235	
slateblue	#6a5acd	106, 90, 205	
slategray	#708090	112, 128, 144	
slategrey	#708090	112, 128, 144	
snow	#fffafa	255, 250, 250	
springgreen	#00ff7f	0, 255, 127	
steelblue	#4682b4	70, 130, 180	
tan	#d2b48c	210, 180, 140	
teal	#008080	0, 128, 128	
thistle	#d8bfd8	216, 191, 216	
tomato	#ff6347	255, 99, 71	
turquoise	#40e0d0	64, 224, 208	
violet	#ee82ee	238, 130, 238	
wheat	#f5deb3	245, 222, 179	
white	#ffffff	255, 255, 255	
whitesmoke	#f5f5f5	245, 245, 245	
yellow	#ffff00	255, 255, 0	
yellowgreen	#9acd32	154, 205, 50	

Appendix B – Linux Kernel Video Modes Values

The table below list the values for some of the more common video modes.

	640x480	800x600	1024x768	1280x1024
16 colors		770	772	774
256 colors	769	771	773	775
64K colors	785	788	791	794
16M colors	786	789	792	795

Appendix C - Setting Fedora 14 To Show It's Icon

To correct this issue, first install the redhat-lsb package with this command:

```
su -c "yum install redhat-lsb"
```

Open the */etc/default/grub* configuration file:

```
beesu gedit /etc/default/grub
```

Add the following lines (the highlighted characters are back-ticks (`) and not single quotes (') :

```
10
11 GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null`
12
```

Update GRUB's configuration script file with this command:

```
su -c "grub2-mkconfig -o /boot/grub2/grub.cfg"
```

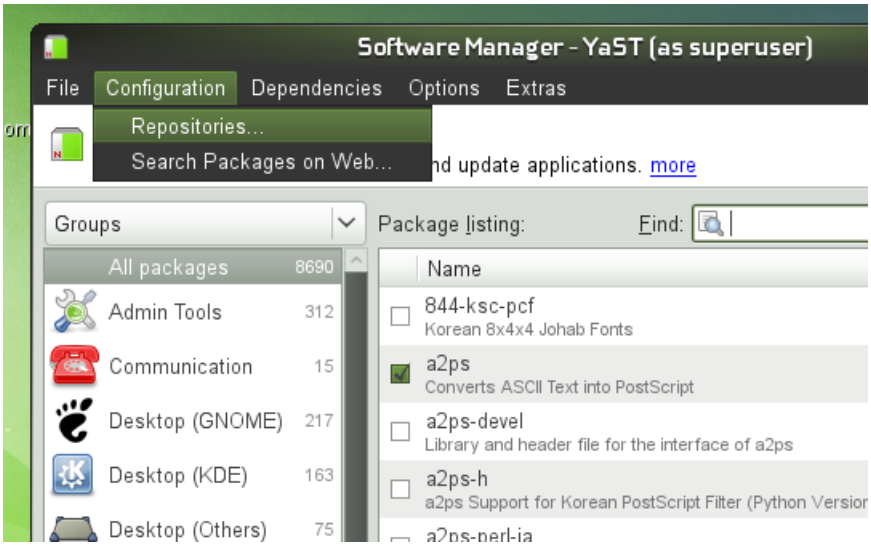
Reboot and the Fedora icon should now be shown.

Appendix D - Installing The os-prober Package In openSUSE 11.3

The os-prober package probes the system for additional OS's. This package is not installed by default and must be installed from the 'testing' repository located at:

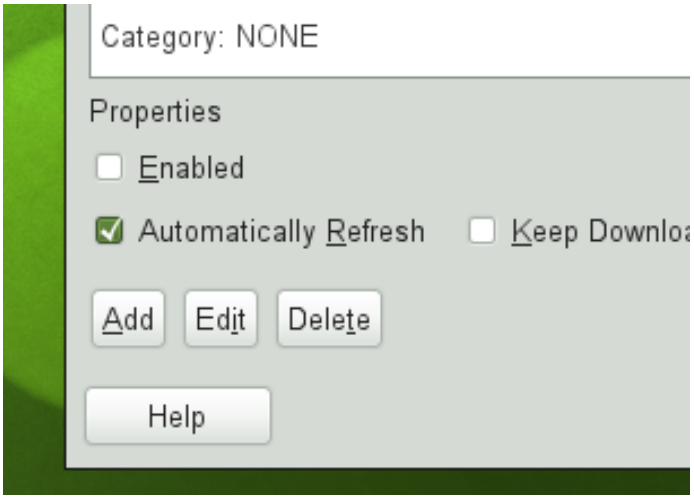
http://download.opensuse.org/repositories/home:/please_try_again/openSUSE_11.3/

Start by opening the Install/Remove Software application. Click on *Configuration* → *Repositories* from the main menu.



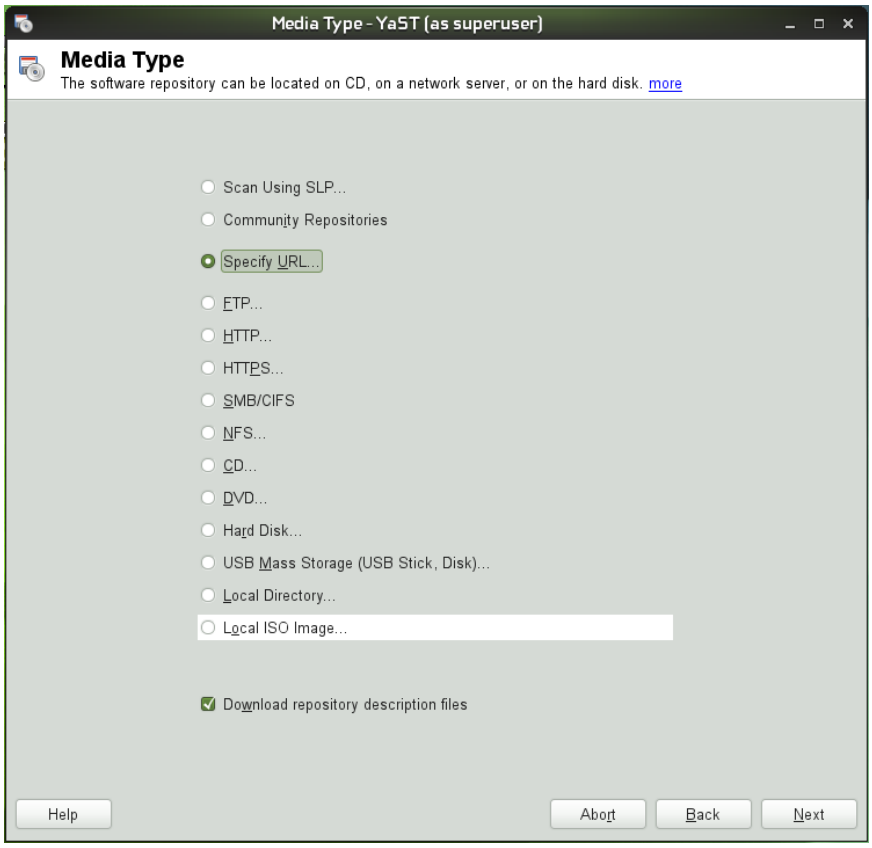
YaST: Configuration → Repositories

When the 'Configure Software Repositories' dialog window appears, ensure that the '*Automatically Refresh*' option button is checked and click on '*Add*'.



YaST: Add Repository

From the 'Media Type' dialog window, select the '*Specify URL...*' option and click on '*Next*'.

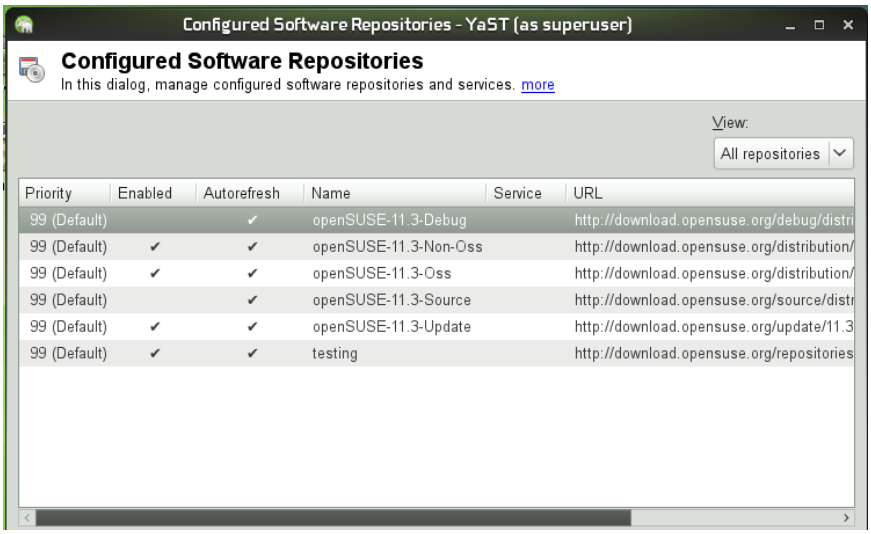


YaST: Specify URL

In the '*Repository URL*' dialog window, enter '**testing**' for the '*Repository Name*' and '**http://download.opensuse.org/repositories/home:/please_try_again/openSUSE_11.3/**' for the '*URL*' and click on '*Next*'.

YaST: Repository URL

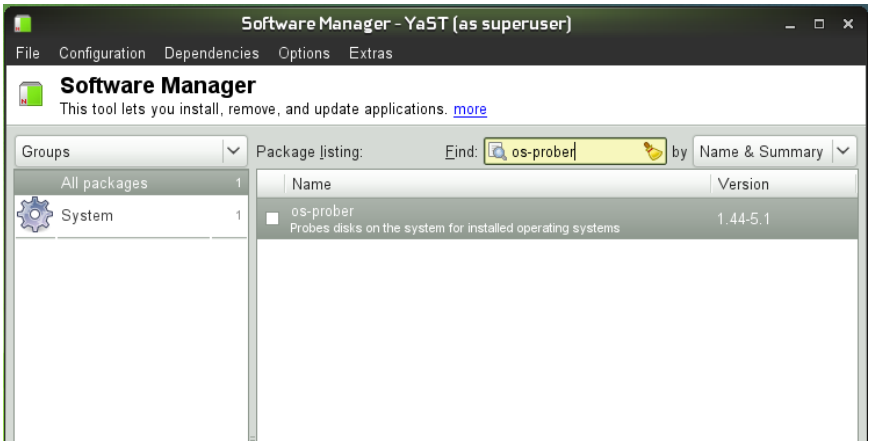
YaST will now update the list of configured repositories and a new repository name 'testing' will be added to the list of repositories in the 'Configure Software Repositories' dialog window. Click on '*OK*' when this window re-appears.



YaST: New Repository Added

It is now your decision whether or not you want to accept the untrusted GnuPG key. Click on the appropriate button.

When the 'Software Manager' window re-appears, enter '**os-prober**' in the 'Find' box and select it from the package listing. Click on '*Apply*' to install the os-prober package.



YaST: Select os-prober Package

Appendix E - Modified GRUB Configuration File

This is the contents of my modified /etc/default/grub file (changes are shown in red):

```
1   # If you change this file, run 'update-grub' afterwards to update
2   # /boot/grub/grub.cfg.
3
4   GRUB_THEME=/boot/grub/themes/ubuntu1/theme.txt
5
6   GRUB_FONT=/boot/grub/fonts/7x13.pf2
7
8   GRUB_DEFAULT=0
9   #GRUB_HIDDEN_TIMEOUT=0
10  #GRUB_HIDDEN_TIMEOUT_QUIET=true
11  GRUB_TIMEOUT=30
12  GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
13  GRUB_DISTRIBUTOR_CODENAME=`lsb_release -cs 2> /dev/null`
14  GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
15  GRUB_CMDLINE_LINUX=""
16
17  # Uncomment to enable BadRAM filtering, modify to suit your needs
18  # This works with Linux (no patch required) and with any kernel that
    obtains
19  # the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
20  #GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"
21
22  # Uncomment to disable graphical terminal (grub-pc only)
23  #GRUB_TERMINAL=console
24
25  # The resolution used on graphical terminal
26  # note that you can use only modes which your graphic card supports via VBE
27  # you can see them in real GRUB with the command `vbeinfo'
28  GRUB_GFXMODE=1024x768
29
30  # Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to
    Linux
31  #GRUB_DISABLE_LINUX_UUID=true
32
33  # Uncomment to disable generation of recovery mode menu entries
34  GRUB_DISABLE_LINUX_RECOVERY=true
```

```
35
36  # Uncomment to get a beep at grub start
37  #GRUB_INIT_TUNE="480 440 1"
38
39  # Uncomment to include boot device in menu entries
40  #GRUB_INCLUDE_BOOT_DEVICE=true
41
42  # Uncomment to add only the latest kernel in the menu entries
43  GRUB_ADD_ONLY_LATEST_KERNEL=true
44
45  # Uncomment to include the kernel version in the menu entries
46  #GRUB_INCLUDE_KERNEL_VERSION=true
```

Appendix F - Modified 00_header Script File

This the content of my modified */etc/grub.d/00_header* script file (deleted lines are shown in red):

```
#! /bin/sh -e

# grub-mkconfig helper script.
# Copyright (C) 2006,2007,2008,2009,2010  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}`
locale_dir=`echo /boot/grub/locale | sed ${transform}`
grub_lang=`echo $LANG | cut -d _ -f 1`

. ${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

```

```
36 if [ "x${GRUB_DEFAULT}" = "x" ] ; then GRUB_DEFAULT=0 ; fi
37 if [ "x${GRUB_DEFAULT}" = "xsaved" ] ; then GRUB_DEFAULT='${saved_entry}' ;
   fi
38 if [ "x${GRUB_TIMEOUT}" = "x" ] ; then GRUB_TIMEOUT=5 ; fi
39 if [ "x${GRUB_GFXMODE}" = "x" ] ; then GRUB_GFXMODE=640x480 ; fi
40
41 cat << EOF
42 if [ -s \$prefix/grubenv ]; then
43   load_env
44 fi
45 set default="${GRUB_DEFAULT}"
46 if [ \${prev_saved_entry} ]; then
47   set saved_entry=\${prev_saved_entry}
48   save_env saved_entry
49   set prev_saved_entry=
50   save_env prev_saved_entry
51   set boot_once=true
52 fi
53
54 function savedefault {
55   if [ -z \${boot_once} ]; then
56     saved_entry=\${chosen}
57     save_env saved_entry
58   fi
59 }
60
61 function recordfail {
62   set recordfail=1
63   if [ -n \${have_grubenv} ]; then if [ -z \${boot_once} ]; then save_env
   recordfail; fi; fi
64 }
65 EOF
66
67 case ${GRUB_TERMINAL_INPUT}:${GRUB_TERMINAL_OUTPUT} in
68   serial:* | *:serial)
69     if ! test -e ${grub_prefix}/serial.mod ; then
70       echo "Serial terminal not available on this platform." >&2 ; exit 1
71     fi
72
73     if [ "x${GRUB_SERIAL_COMMAND}" = "x" ] ; then
```

```
74        grub_warn "Requested serial terminal but GRUB_SERIAL_COMMAND is
    unspecified. Default parameters will be used."
75        GRUB_SERIAL_COMMAND=serial
76      fi
77      echo "${GRUB_SERIAL_COMMAND}"
78    ;;
79  esac
80
81  case x${GRUB_TERMINAL_INPUT} in
82    x)
83      # Just use the native terminal
84    ;;
85    x*)
86      cat << EOF
87  if terminal_input ${GRUB_TERMINAL_INPUT} ; then true ; else
88    # For backward compatibility with versions of terminal.mod that don't
89    # understand terminal_input
90    terminal ${GRUB_TERMINAL_INPUT}
91  fi
92  EOF
93    ;;
94  esac
95
96  case x${GRUB_TERMINAL_OUTPUT} in
97   xgfxterm)
98      # Make the font accessible
99      prepare_grub_to_access_device `${grub_probe} --target=device ${
    GRUB_FONT_PATH}`
100
101     cat << EOF
102 if loadfont `make_system_path_relative_to_its_root ${GRUB_FONT_PATH}` ;
    then
103   set gfxmode=${GRUB_GFXMODE}
104   insmod gfxterm
105   insmod ${GRUB_VIDEO_BACKEND}
106   if terminal_output gfxterm ; then true ; else
107     # For backward compatibility with versions of terminal.mod that don't
108     # understand terminal_output
109     terminal gfxterm
110   fi
```

```
111 EOF
112 if [ x$GRUB_THEME != x ] && [ -f $GRUB_THEME ] \
113       && is_path_readable_by_grub $GRUB_THEME; then
114     echo "Found theme: $GRUB_THEME" >&2
115     prepare_grub_to_access_device `${grub_probe} --target=device
    $GRUB_THEME` | sed -e "s/^/  /"
116     cat << EOF
117   insmod gfxmenu
118   set theme=(\$root)`make_system_path_relative_to_its_root $GRUB_THEME`
119   set menuviewer=gfxmenu
120 EOF
121 fi
122     cat << EOF
123 fi
124 EOF
125   ;;
126   x)
127     # Just use the native terminal
128   ;;
129   x*)
130     cat << EOF
131 if terminal_output ${GRUB_TERMINAL_OUTPUT} ; then true ; else
132   # For backward compatibility with versions of terminal.mod that don't
133   # understand terminal_output
134   terminal ${GRUB_TERMINAL_OUTPUT}
135 fi
136 EOF
137   ;;
138 esac
139
140 # Gettext variables and module
141 if [ "x${LANG}" != "xC" ] ; then
142     prepare_grub_to_access_device $(${grub_probe} --target=device $
    {locale_dir})
143   cat << EOF
144 set locale_dir=(\$root)$(make_system_path_relative_to_its_root $
    {locale_dir})
145 set lang=${grub_lang}
146 insmod gettext
147 EOF
```

```
148 fi
149
150 cat << EOF
151 if [ \${recordfail} = 1 ]; then
152   set timeout=-1
153 else
154   set timeout=${GRUB_TIMEOUT}
155 fi
156 EOF
157
158 # Play an initial tune
159 if [ "x${GRUB_INIT_TUNE}" != "x" ] ; then
160   cat << EOF
161 insmod play
162 play ${GRUB_INIT_TUNE}
163 EOF
164 fi
```

Appendix G - 08_gfxmenu_theme Script File

This is the contents of my new *08_gfxmenu_theme* file:

```
#! /bin/sh -e

prefix=/usr       # Sabayon 5.5 users must delete /usr from this line.
exec_prefix=${prefix}
libdir=${exec_prefix}/lib

. ${libdir}/grub/grub-mkconfig_lib

if [ "$GRUB_TERMINAL_OUTPUT" = "gfxterm" ] && [ "x$GRUB_THEME" != x ] && [ -f "$GRUB_THEME" ] \
    && is_path_readable_by_grub "$GRUB_THEME" ; then
    echo "Found theme: $GRUB_THEME" >&2
    prepare_grub_to_access_device `${grub_probe} --target=device "$GRUB_THEME"`
    cat << EOF
insmod gfxmenu
EOF
    themedir="`dirname "$GRUB_THEME"`"
    for x in "$themedir"/*.pf2; do
        if [ -f "$x" ]; then
        cat << EOF
loadfont (\$root)`make_system_path_relative_to_its_root $x`
EOF
        fi
    done
    if [ x"`echo "$themedir"/*.jpg`" != x"$themedir/*.jpg" ] || [ x"`echo "$themedir"/*.jpeg`" != x"$themedir/*.jpeg" ]; then
        cat << EOF
insmod jpeg
EOF
    fi
    if [ x"`echo "$themedir"/*.png`" != x"$themedir/*.png" ]; then
        cat << EOF
insmod png
EOF
    fi
```

```
34      if [ x"`echo "$themedir"/*.tga`" != x"$themedir/*.tga" ]; then
35          cat << EOF
36  insmod tga
37  EOF
38      fi
39      cat << EOF
40  set theme=(\$root)`make_system_path_relative_to_its_root $GRUB_THEME`
41  EOF
42  fi
```

Appendix H - Modified 10_linux Script File

This is the content of my modified */etc/grub.d/10_linux* script file (changes are shown in red):

```
1   #! /bin/sh
2   set -e
3
4   # grub-mkconfig helper script.
5   # Copyright (C) 2006,2007,2008,2009,2010  Free Software Foundation, Inc.
6   #
7   # GRUB is free software: you can redistribute it and/or modify
8   # it under the terms of the GNU General Public License as published by
9   # the Free Software Foundation, either version 3 of the License, or
10  # (at your option) any later version.
11  #
12  # GRUB is distributed in the hope that it will be useful,
13  # but WITHOUT ANY WARRANTY; without even the implied warranty of
14  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15  # GNU General Public License for more details.
16  #
17  # You should have received a copy of the GNU General Public License
18  # along with GRUB.  If not, see <http://www.gnu.org/licenses/>.
19
20  prefix=/usr
21  exec_prefix=${prefix}
22  bindir=${exec_prefix}/bin
23  libdir=${exec_prefix}/lib
24  . ${libdir}/grub/grub-mkconfig_lib
25
26  export TEXTDOMAIN=grub
27  export TEXTDOMAINDIR=${prefix}/share/locale
28
29  CLASS="--class gnu-linux --class gnu --class os"
30
31  if [ "x${GRUB_DISTRIBUTOR}" = "x" ] ; then
32    OS=GNU/Linux
33  else
34    OS="`echo ${GRUB_DISTRIBUTOR} | cut -d' ' -f1` `echo $
    {GRUB_DISTRIBUTOR_CODENAME} | sed -e 's/^./\u&/'`"
35    CLASS1="--class $(echo ${GRUB_DISTRIBUTOR} | tr '[A-Z]' '[a-z]' | cut -d'
```

```
' -f1) ${CLASS}"
fi

# loop-AES arranges things so that /dev/loop/X can be our root device, but
# the initrds that Linux uses don't like that.
case ${GRUB_DEVICE} in
  /dev/loop/*|/dev/loop[0-9])
    GRUB_DEVICE=`losetup ${GRUB_DEVICE} | sed -e "s/^[^(]*(\([^)]\
+\)).*/\1/"`
    # We can't cope with devices loop-mounted from files here.
    case ${GRUB_DEVICE} in
      /dev/*) ;;
      *) exit 0 ;;
    esac
  ;;
esac

if [ "x${GRUB_DEVICE_UUID}" = "x" ] || [ "x${GRUB_DISABLE_LINUX_UUID}" =
"xtrue" ] \
    || ! test -e "/dev/disk/by-uuid/${GRUB_DEVICE_UUID}" \
    || uses_abstraction "${GRUB_DEVICE}" lvm; then
  LINUX_ROOT_DEVICE=${GRUB_DEVICE}
else
  LINUX_ROOT_DEVICE=UUID=${GRUB_DEVICE_UUID}
fi

# 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 ()
{
  os="$1"
  version="$2"
  recovery="$3"
  args="$4"

# If we are including all kernels, then always show the kernel version to
```

```
differentiate between them.
  if [ "x$GRUB_ADD_ONLY_LATEST_KERNEL" != "xtrue" ] ||
[ "x$GRUB_INCLUDE_KERNEL_VERSION" = "xtrue" ]; then
    title="$(gettext_quoted "%s with Linux %s")"
  else
    title=$1
  fi

  if ${recovery}; then
    CLASS="--class recovery ${CLASS1}"
  else
    CLASS="${CLASS1}"
  fi

  if [ "x$GRUB_INCLUDE_BOOT_DEVICE" = "xtrue" ]; then
    title="${title} on $GRUB_DEVICE"
  fi

  printf "menuentry '${title}' ${CLASS} {\n" "${os}" "${version}"
  cat << EOF
      recordfail
EOF
  save_default_entry | sed -e "s/^/\t/"

  # Use ELILO's generic "efifb" when it's known to be available.
  # FIXME: We need an interface to select vesafb in case efifb can't be
used.
  if [ "x$GRUB_GFXPAYLOAD_LINUX" != x ]; then
        cat << EOF
      set gfxpayload=$GRUB_GFXPAYLOAD_LINUX
EOF
  fi

  if [ -z "${prepare_boot_cache}" ]; then
    prepare_boot_cache="$(prepare_grub_to_access_device ${GRUB_DEVICE_BOOT}
| sed -e "s/^/\t/")"
  fi
  printf '%s\n' "${prepare_boot_cache}"
  if [ "x$5" != "xquiet" ]; then
    cat << EOF
```

```
108       echo  '$(printf "$(gettext_quoted "Loading Linux %s ...")" $
          {version})'
109 EOF
110   fi
111   cat << EOF
112       linux ${rel_dirname}/${basename} root=${linux_root_device_thisversion}
          ro ${args}
113 EOF
114   if [ "x$5" != "xquiet" ]; then
115     cat << EOF
116       echo  '$(gettext_quoted "Loading initial ramdisk ...")'
117 EOF
118   fi
119   if test -n "${initrd}" ; then
120     cat << EOF
121       initrd      ${rel_dirname}/${initrd}
122 EOF
123   fi
124   cat << EOF
125 }
126 EOF
127 }
128
129 list=`for i in /boot/vmlinu[zx]-* /vmlinu[zx]-* ; do
130         if grub_file_is_not_garbage "$i" ; then echo -n "$i " ; fi
131       done`
132 prepare_boot_cache=
133
134 while [ "x$list" != "x" ] ; do
135   linux=`version_find_latest $list`
136   echo "Found linux image: $linux" >&2
137   basename=`basename $linux`
138   dirname=`dirname $linux`
139   rel_dirname=`make_system_path_relative_to_its_root $dirname`
140   version=`echo $basename | sed -e "s,^[^0-9]*-,,g"`
141   alt_version=`echo $version | sed -e "s,\.old$,,g"`
142   linux_root_device_thisversion="${LINUX_ROOT_DEVICE}"
143
144   initrd=
145   for i in "initrd.img-${version}" "initrd-${version}.img" \
```

```
146         "initrd-${version}" "initramfs-${version}.img" \
147         "initrd.img-${alt_version}" "initrd-${alt_version}.img" \
148         "initrd-${alt_version}" "initramfs-${alt_version}.img"; do
149     if test -e "${dirname}/${i}" ; then
150       initrd="$i"
151       break
152     fi
153   done
154   if test -n "${initrd}" ; then
155     echo "Found initrd image: ${dirname}/${initrd}" >&2
156   else
157     # "UUID=" magic is parsed by initrds.  Since there's no initrd, it can't
    work here.
158     linux_root_device_thisversion=${GRUB_DEVICE}
159   fi
160
161   linux_entry "${OS}" "${version}" false \
162       "${GRUB_CMDLINE_LINUX} ${GRUB_CMDLINE_EXTRA} $
    {GRUB_CMDLINE_LINUX_DEFAULT}"
163
164   if [ "x${GRUB_DISABLE_LINUX_RECOVERY}" != "xtrue" ] ; then
165     linux_entry "${OS}" "${version}" true \
166       "single ${GRUB_CMDLINE_LINUX}"
167   fi
168
169   if [ "x$GRUB_ADD_ONLY_LATEST_KERNEL" = "xtrue" ] ; then
170     list=
171   else
172     list=`echo $list | tr ' ' '\n' | grep -vx $linux | tr '\n' ' '`
173   fi
174 done
```

Appendix I - Modified 30_os-prober Script File

This is the content of my modified */etc/grub.d/30_os-prober* script file (changes are shown in red):

```
#! /bin/sh
set -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/>.

prefix=/usr
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
sysconfdir=/etc
GRUBFILE=${sysconfdir}/default/grub

. ${libdir}/grub/grub-mkconfig_lib

found_other_os=

unmount_tmpdir () {
  if grep -q ${tmpdir} /proc/self/mountinfo; then
    umount ${tmpdir} 2> /dev/null
  fi
  if [ -d ${tmpdir} ]; then
    rm -r ${tmpdir}
```

```
36    fi
37  }
38
39  make_timeout () {
40    if [ "x${found_other_os}" = "x" ] ; then
41      if [ "x${1}" != "x" ] ; then
42        if [ "x${GRUB_HIDDEN_TIMEOUT_QUIET}" = "xtrue" ] ; then
43        verbose=
44        else
45        verbose=" --verbose"
46        fi
47
48        if [ "x${1}" = "x0" ] ; then
49        cat <<EOF
50  if [ "x\${timeout}" != "x-1" ]; then
51    if keystatus; then
52      if keystatus --shift; then
53        set timeout=-1
54      else
55        set timeout=0
56      fi
57    else
58      if sleep$verbose --interruptible 3 ; then
59        set timeout=0
60      fi
61    fi
62  fi
63  EOF
64        else
65        cat << EOF
66  if [ "x\${timeout}" != "x-1" ]; then
67    if sleep$verbose --interruptible ${GRUB_HIDDEN_TIMEOUT} ; then
68      set timeout=0
69    fi
70  fi
71  EOF
72        fi
73      fi
74    fi
```

```
}

adjust_timeout () {
  if [ "x$GRUB_BUTTON_CMOS_ADDRESS" != "x" ]; then
    cat <<EOF
if cmostest $GRUB_BUTTON_CMOS_ADDRESS ; then
EOF
    make_timeout "${GRUB_HIDDEN_TIMEOUT_BUTTON}" "${GRUB_TIMEOUT_BUTTON}"
    echo else
    make_timeout "${GRUB_HIDDEN_TIMEOUT}" "${GRUB_TIMEOUT}"
    echo fi
  else
    make_timeout "${GRUB_HIDDEN_TIMEOUT}" "${GRUB_TIMEOUT}"
  fi
}

get_distro_and_codename () {

  mount ${DEVICE} ${tmpdir}

  GRUB_DISTRO="`grep -s ^GRUB_DISTRIBUTOR= ${tmpdir}${GRUBFILE} | cut -d'=' -f2 \
    | tr -d '"\`' | sed -e 's,2>.*$,,'`"
  GRUB_CODENAME="`grep -s ^GRUB_DISTRIBUTOR_CODENAME= ${tmpdir}${GRUBFILE} | cut -d'=' -f2 \
    | tr -d '"\`' | sed -e 's,2>.*$,,'`"

  if [ "x${GRUB_DISTRO}" != "x" ]; then
    cmd=${GRUB_DISTRO}
  else
    cmd='lsb_release -is'
  fi

  set +e
  rval_CHROOT=`chroot ${tmpdir} ${cmd} 2> /dev/null`
  err_CHROOT=$?
  DISTRIBUTOR=`echo ${rval_CHROOT} | cut -d' ' -f1 | tr -d '"'`
  set -e

  if [ "${err_CHROOT}" = "127" ]; then
```

```
113     if [ "x${GRUB_DISTRO}" = "x" ]; then
114       DISTRIBUTOR=${LONGNAME}
115     else
116       DISTRIBUTOR=${GRUB_DISTRO}
117     fi
118   fi
119   if [ "${err_CHROOT}" = "1" ]; then
120     DISTRIBUTOR=${LONGNAME}
121   fi
122
123   if [ "x${GRUB_CODENAME}" != "x" ]; then
124     cmd=${GRUB_CODENAME}
125   else
126     cmd='lsb_release -cs'
127   fi
128
129   set +e
130   rval_CHROOT=`chroot ${tmpdir} ${cmd} 2> /dev/null`
131   err_CHROOT=$?
132   CODENAME=`echo ${rval_CHROOT} | cut -d' ' -f1 | tr -d '"' | sed -e
    's,^.,\u&,'`
133   set -e
134
135   if [ "${err_CHROOT}" = "127" ]; then
136     CODENAME=${GRUB_CODENAME}
137   fi
138   if [ "${err_CHROOT}" = "1" ]; then
139     CODENAME=
140   fi
141
142   umount ${tmpdir}
143 }
144
145 if [ "x${GRUB_DISABLE_OS_PROBER}" = "xtrue" ]; then
146   adjust_timeout
147   exit 0
148 fi
149
150 if [ -z "`which os-prober 2> /dev/null`" -o -z "`which linux-boot-prober 2>
    /dev/null`" ] ; then
```

```
151   # missing os-prober and/or linux-boot-prober
152   adjust_timeout
153   exit 0
154 fi
155
156 OSPROBED="`os-prober | tr ' ' '^' | paste -s -d ' ' `"
157 if [ -z "${OSPROBED}" ] ; then
158   # empty os-prober output, nothing doing
159   adjust_timeout
160   exit 0
161 fi
162
163 osx_entry() {
164         cat << EOF
165 menuentry "${LONGNAME} (${2}-bit) ${SHOWBOOTDEVICE}" --class macosx {
166 EOF
167       save_default_entry | sed -e "s/^/\t/"
168       prepare_grub_to_access_device ${DEVICE} | sed -e "s/^/\t/"
169       cat << EOF
170         load_video
171         set do_resume=0
172         if [ /var/vm/sleepimage -nt10 / ]; then
173            if xnu_resume /var/vm/sleepimage; then
174              set do_resume=1
175            fi
176         fi
177         if [ \$do_resume = 0 ]; then
178            xnu_uuid ${OSXUUID} uuid
179            if [ -f /Extra/DSDT.aml ]; then
180               acpi -e /Extra/DSDT.aml
181            fi
182            $1 /mach_kernel boot-uuid=\${uuid} rd=*uuid
183            if [ /System/Library/Extensions.mkext -nt
/System/Library/Extensions ]; then
184               xnu_mkext /System/Library/Extensions.mkext
185            else
186               xnu_kextdir /System/Library/Extensions
187            fi
188            if [ -f /Extra/Extensions.mkext ]; then
```

```
189             xnu_mkext /Extra/Extensions.mkext
190           fi
191           if [ -d /Extra/Extensions ]; then
192             xnu_kextdir /Extra/Extensions
193           fi
194           if [ -f /Extra/devprop.bin ]; then
195             xnu_devprop_load /Extra/devprop.bin
196           fi
197           if [ -f /Extra/splash.jpg ]; then
198             insmod jpeg
199             xnu_splash /Extra/splash.jpg
200           fi
201           if [ -f /Extra/splash.png ]; then
202             insmod png
203             xnu_splash /Extra/splash.png
204           fi
205           if [ -f /Extra/splash.tga ]; then
206             insmod tga
207             xnu_splash /Extra/splash.tga
208           fi
209         fi
210 }
211 EOF
212 }
213
214 tmpdir="`mktemp -d --tmpdir mounted.XXXXXXXXXX`"
215 trap "unmount_tmpdir" EXIT HUP INT QUIT TERM
216
217 for OS in ${OSPROBED} ; do
218   DEVICE="`echo ${OS} | cut -d ':' -f 1`"
219   LONGNAME="`echo ${OS} | cut -d ':' -f 2 | tr '^' ' '`"
220   LABEL="`echo ${OS} | cut -d ':' -f 3 | tr '^' ' '`"
221   BOOT="`echo ${OS} | cut -d ':' -f 4`"
222
223   if [ -z "${LONGNAME}" ] ; then
224     LONGNAME="${LABEL}"
225   fi
226
227   echo "Found ${LONGNAME} on ${DEVICE}" >&2
```

```
228   found_other_os=1
229
230   if [ "x${GRUB_INCLUDE_BOOT_DEVICE}" != "x" ] ; then
231     SHOWBOOTDEVICE="on ${DEVICE}"
232   fi
233   CLASS="--class `echo ${LABEL%%[0-9]*} | tr [A-Z] [a-z]`"
234
235   case ${BOOT} in
236     chain)
237
238       cat << EOF
239 menuentry "${LONGNAME} ${SHOWBOOTDEVICE}" ${CLASS} {
240 EOF
241       save_default_entry | sed -e "s/^/\t/"
242       prepare_grub_to_access_device ${DEVICE} | sed -e "s/^/\t/"
243
244       case ${LONGNAME} in
245             Windows\ Vista*|Windows\ 7*)
246             ;;
247             *)
248         cat << EOF
249       drivemap -s (hd0) \${root}
250 EOF
251             ;;
252       esac
253
254       cat <<EOF
255       chainloader +1
256 }
257 EOF
258     ;;
259     linux)
260       LINUXPROBED="`linux-boot-prober ${DEVICE} 2> /dev/null | tr ' ' '^' |
    paste -s -d ' '`"
261       prepare_boot_cache=
262
263       if [ "x$GRUB_ADD_ONLY_LATEST_KERNEL" = "xtrue" ] ; then
264         echo "Adding only the latest kernel" >&2
265         LINUXPROBED="`echo ${LINUXPROBED} | cut -d' ' -f1`"
```

```
266       fi
267
268       get_distro_and_codename
269       CLASS1="--class `echo ${DISTRIBUTOR} | cut -d' ' -f1 | tr [A-Z] [a-
z]`"
270
271       for LINUX in ${LINUXPROBED} ; do
272         LROOT="`echo ${LINUX} | cut -d ':' -f 1`"
273         LBOOT="`echo ${LINUX} | cut -d ':' -f 2`"
274         LLABEL="`echo ${LINUX} | cut -d ':' -f 3 | tr '^' ' '`"
275         LKERNEL="`echo ${LINUX} | cut -d ':' -f 4`"
276         LINITRD="`echo ${LINUX} | cut -d ':' -f 5`"
277         LPARAMS="`echo ${LINUX} | cut -d ':' -f 6- | tr '^' ' '`"
278
279         if `echo ${LLABEL} | tr [A-Z] [a-z] | grep -qE "(recovery|
failsafe)"`; then
280           CLASS="--class recovery ${CLASS1}"
281           if [ "x${GRUB_DISABLE_LINUX_RECOVERY}" = "xtrue" ]; then
282             continue
283           fi
284         else
285           CLASS=${CLASS1}
286         fi
287
288         if [ -z "${LLABEL}" ] ; then
289           LLABEL="${LONGNAME}"
290         fi
291
292         if [ "${LROOT}" != "${LBOOT}" ]; then
293           LKERNEL="${LKERNEL#/boot}"
294           LINITRD="${LINITRD#/boot}"
295         fi
296
297         TITLE="${DISTRIBUTOR} ${CODENAME}"
298
299         if [ "x$GRUB_ADD_ONLY_LATEST_KERNEL" != "xtrue" ] \
300           || [ "x$GRUB_INCLUDE_KERNEL_VERSION" = "xtrue" ] ; then
301           TITLE="${TITLE} with Linux `echo ${LKERNEL} | sed -e "s,^[^0-
9]*-,,g"`"
302         fi
```

```
303
304         cat << EOF
305 menuentry "${TITLE} ${SHOWBOOTDEVICE}" ${CLASS} {
306 EOF
307       save_default_entry | sed -e "s/^/\t/"
308       if [ -z "${prepare_boot_cache}" ]; then
309         prepare_boot_cache="$(prepare_grub_to_access_device ${LBOOT} | sed
    -e "s/^/\t/")"
310       fi
311       printf '%s\n' "${prepare_boot_cache}"
312       cat <<  EOF
313       linux ${LKERNEL} ${LPARAMS}
314 EOF
315         if [ -n "${LINITRD}" ] ; then
316           cat << EOF
317       initrd ${LINITRD}
318 EOF
319         fi
320         cat << EOF
321 }
322 EOF
323       done
324     ;;
325     macosx)
326       OSXUUID="`grub-probe --target=fs_uuid --device ${DEVICE} 2>
    /dev/null`"
327       osx_entry xnu_kernel 32
328       osx_entry xnu_kernel64 64
329     ;;
330     hurd)
331       cat << EOF
332 menuentry "${LONGNAME} ${SHOWBOOTDEVICE}" {
333 EOF
334       save_default_entry | sed -e "s/^/\t/"
335       prepare_grub_to_access_device ${DEVICE} | sed -e "s/^/\t/"
336       grub_device="`${grub_probe} --device ${DEVICE} --target=drive`"
337       mach_device="`echo "${grub_device}" | tr -d '()' | tr , s`"
338       grub_fs="`${grub_probe} --device ${DEVICE} --target=fs`"
339       case "${grub_fs}" in
340       *fs)  hurd_fs="${grub_fs}" ;;
```

```
341       *)    hurd_fs="${grub_fs}fs" ;;
342       esac
343       cat << EOF
344       multiboot /boot/gnumach.gz root=device:${mach_device}
345       module /hurd/${hurd_fs}.static ${hurd_fs} --readonly \\
346                   --multiboot-command-line='\${kernel-command-line}' \\
347                   --host-priv-port='\${host-port}' \\
348                   --device-master-port='\${device-port}' \\
349                   --exec-server-task='\${exec-task}' -T typed '\${root}' \\
350                   '\$(task-create)' '\$(task-resume)'
351       module /lib/ld.so.1 exec /hurd/exec '\$(exec-task=task-create)'
352 }
353 EOF
354     ;;
355     *)
356       echo "${LONGNAME} is not yet supported by grub-mkconfig." >&2
357     ;;
358   esac
359 done
360
361 adjust_timeout
```

Appendix J - Theme Installation Script File

This is the content of my theme installation script file:

```
1   #! /bin/bash
2   set -e
3
4   # This script installs the GRUB2 theme in /boot/grub/themes/,
    /boot/grub2/themes/ or /grub/themes/
5   # depending on the distribution.
6   #
7   # Copyright (C) 2011 Towheed Mohammed
8   # who just started learning bash scripting, sed and regex's.
9   #
10  # This is free software: you can redistribute it and/or modify
11  # it under the terms of the GNU General Public License as published by
12  # the Free Software Foundation, either version 3 of the License, or
13  # (at your option) any later version.
14  #
15  # This software is distributed in the hope that it will be useful,
16  # but WITHOUT ANY WARRANTY; without even the implied warranty of
17  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
18  # GNU General Public License for more details at
    <http://www.gnu.org/licenses/>.
19
20  # Set variables
21  Theme_Name="ubuntu"           # The theme will be installed in a dir with
    this name. Avoid spaces.
22  Theme_Definition_File="theme.txt"   # Filename of theme definition file.
23  Theme_Resolution="any"        # The resolution the theme was designed to show
    best at, 640x480, 1024x768 etc,
24                                # or "any" for any resolution (resolution
    independent).
25  Inst_Dir=$(dirname $0)
26  Grub_Dist_Dirs="/grub /boot/grub /boot/grub2"   # Directories must be in this
    order.
27  let Grub_Min_Version=198      # Do not change this.
28  Grub_File="/etc/default/grub"
29  Grub_Dir=
30  mkConfig_File=
31
```

```
32  # Check that the script is being run as root.
33  if [[ $(id -u) != 0 ]]; then
34    echo "Please run this script with root privileges."
35    exit 0
36  fi
37
38  # Get GRUB's installation directory.
39  for i in $Grub_Dist_Dirs; do
40    if [[ -d $i ]]; then
41      Grub_Dir=$i
42    fi
43  done
44
45  # Exit this script if we could not locate GRUB's installation directory.
46  if [[ -z $Grub_Dir ]]; then
47    echo "Could not locate GRUB's installation directory."
48    exit 0
49  fi
50
51  # Exit the script if GRUB's version is < 1.98
52  if [[ -f $(which grub2-install) ]]; then
53    Grub_Version_Long=$(grub2-install --version)
54  elif [[ -f $(which grub-install) ]]; then
55    Grub_Version_Long=$(grub-install --version)
56  else
57    echo 'Could not locate grub-install or grub2-install in your path.'
58    exit 0
59  fi
60  Grub_Version=$(echo $Grub_Version_Long | sed 's,[[:alpha:][:punct:]
    [:blank:]],,g')
61  if (( ${Grub_Version:0:3} < Grub_Min_Version )); then
62    echo "GRUB must be at least version ${Grub_Min_Version:0:1}.$
    {Grub_Min_Version:1:2}."
63    echo "The installed version is ${Grub_Version:0:1}.${Grub_Version:1:2}."
64    exit 0
65  fi
66
67  # Check that /etc/default/grub exists.
68  if [[ ! -f $Grub_File ]]; then
69    echo "Could not find $Grub_File"
```

```
70    exit 0
71  fi
72
73  # Check that GRUB's mkconfig script file exists.
74  mkConfig_File=$(which ${Grub_Dir##*/}-mkconfig) || \
75  (echo "GRUB's mkconfig script file was not found in your path." && exit 0)
76
77  # Create theme directory.  If directory already exists, ask the user if they
    would like
78  # to overwrite the contents with the new theme or create a new theme
    directory.
79  Theme_Dir=$Grub_Dir/themes/$Theme_Name
80  while [[ -d $Theme_Dir ]]; do
81    echo "Directory $Theme_Dir already exists!"
82    echo -n "Would you like to overwrite it's contents or create a new
    directory? [(o)verwrite (c)reate] "
83    read Response
84    case $Response in
85      c|create)
86        echo -n "Please enter a new name for the theme's directory: "
87        read Response
88        Theme_Dir=$Grub_Dir/themes/$Response;;
89      o|overwrite)
90        echo -n "This will delete all files in $Theme_Dir.  Are you sure?
    [(y)es (n)o] "
91        read Response
92        case $Response in
93          y|yes)
94          rm -r $Theme_Dir;;
95          *)
96            exit 0;;
97        esac;;
98      *)
99        exit 0;;
100   esac
101 done
102 mkdir -p $Theme_Dir
103
104 # Copy the theme's files to the theme's directory.
105 for i in $Inst_Dir/*; do
```

```
106   cp -r $i $Theme_Dir/$(basename $i)
107 done
108
109 # Check whether an icons directory exists.  If icons are not included in this
    theme,
110 # check if one exists in ..../themes/icons.  If it exists, ask the user if
    they would like to use it.
111 if [[ ! -d $Theme_Dir/icons && -d $Grub_Dir/themes/icons ]]; then
112   echo "An icons directory was not included in this theme."
113   echo "However, one was found in $Grub_Dir/themes/icons containing these
    files:"
114   echo $(ls $Grub_Dir/themes/icons)
115   echo -n "Would you like to use these icons? [(y)es (n)o] "
116   read Response
117   case $Response in
118     y|yes)
119       ln -s $Grub_Dir/themes/icons $Theme_Dir/;;
120     *)
121       echo "This theme will not show any icons.";;
122   esac
123 elif [[ ! $Theme_Dir/icons && ! -d $Grub_Dir/themes/icons ]]; then
124   echo "Could not find an icons directory.  This theme will not show any
    icons."
125 fi
126
127 # Change GRUB's resolution to match that of the theme.
128 if [[ $Theme_Resolution != "any" ]]; then
129   i=$(sed -n 's,^#\?GRUB_GFXMODE=,&,p' $Grub_File)
130   if [[ -z $i ]]; then
131     echo -e "\nGRUB_GFXMODE=$Theme_Resolution" >>$Grub_File
132   else
133     sed "s,^#\?GRUB_GFXMODE=.*,GRUB_GFXMODE=$Theme_Resolution," <$Grub_File
    >$Grub_File.~
134     mv $Grub_File.~ $Grub_File
135   fi
136 fi
137
138 # Ask the user if they would like to set the theme as their new theme.
139 echo -n "Would you like to set this as your new theme? [(y)es (n)o] "
140 read Response
141 if [[ $Response = yes || $Response = y ]]; then
```

```
142   i=$(sed -n 's,^#\?GRUB_THEME=,&,p' $Grub_File)
143   if [[ -z $i ]]; then
144     echo -e "\nGRUB_THEME=$Theme_Dir/$Theme_Definition_File" >>$Grub_File
145   else
146     sed "s,^#\?GRUB_THEME=.*,GRUB_THEME=$Theme_Dir/$Theme_Definition_File,"
    <$Grub_File >$Grub_File.~
147     mv $Grub_File.~ $Grub_File
148   fi
149   $($mkConfig_File -o $Grub_Dir/grub.cfg) # Generate new grub.cfg
150 fi
151 exit 0
```