Ubuntu OEM Recovery framework

Mario Limonciello
Mario_Limonciello@Dell.com

June 23, 2008

Contents

1 Introduction

A common issue for OEMs who want to ship Ubuntu on their machines is developing a sustainable method for creating recovery disks for their customers. The problem is amplified when OEMs attempt to make these recovery disks emulate the process used for creating the factory installation. Multiply this by the large number of variable hardware and software configurations offered at system creation time and the problem becomes unmaintainable.
The logical solution is to create a framework that can be reused and adapted to work with a variety of teams, ensuring that a single team's work will not interfer with or detrimentally affect another team's workflow.

2 Framework overview

This framework provides a solution that allows OEMs to provide hooks into each portion of the installation. These hooks can be workarounds, drivers for hardware enablement, software, documentation, and localization. The same hooks are then replicated on the resultant system allowing users to create a recovery disk with all of the same benefits.

2.1 Basic layout

The framework itself is laid out in the following structure:

- ./grub
- ./docs
- ./preseed
- ./scripts
- ./scripts/chroot-scripts
- ./scripts/chroot-scripts/os-pre
- ./scripts/chroot-scripts/fish
- ./scripts/chroot-scripts/os-post
- ./debs
- ./debs/main
- ./isolinux

This is intended to supplement the existing directory structure of an Ubuntu DVD. Each of these directories (and the scripts in them) will be explained as they are brought up.

3 Factory Installation

When a standard Ubuntu installation is performed, the contents of the DVD are copied onto a system. A few minor modifications are made due to the hardware and input parameters related to localization during installation. For this factory installation, the install is similar to a standard Ubuntu installation, however hooks have been added to perform operations before install begins, modify input variables during install, and add on additional changes to the resultant system.

3.1 Preparing the installation

To get started, checkout a copy of the framework and download a copy of the latest Ubuntu DVD. You will need a blank hard drive to work with as well.

1. Set up the partition structure on the hard drive. It is expected that you have 2 small partitions with the rest of the drive unallocated. The first partition is intended for diagnostic utilities and should be 50-100 mb. The second partition is your recovery partition. You should set it to be 5 GB and either fat32 or ext3.

2. Download an Ubuntu DVD and extract it's contents to the recovery partition. Be sure to extract the hidden .disk directory too This is critical.

3. Download a copy of the OEM framework and extract it "on top" of the extract recovery partition. There are a few files that need to be overwritten.

4. Prepare the drive to boot onto this second partition. This can be achieved using a DOS filesystem and linld, a linux filesystem and kexec, or by installing GRUB to the drive. Do whatever is most conducive to your environment. Use this kernel command line to get started:
   `kernel /casper/vmlinuz boot=casper edd=on preseed/file=/cdrom/preseed/oem.seed noninteractive noprompt`

3.2 Booting the installation

The installation begins by passing a kernel command line to the kernel shipped with the Ubuntu DVD. This command line will generally look something like this:

```
kernel /casper/vmlinuz boot=casper edd=on preseed/file=/cdrom/preseed/oem.seed
noninteractive noprompt
```

Let's break up that command line to explain what each portion is doing.

Option	Purpose
kernel	The option to menu.lst to indicate we are loading a kernel.
/casper/vmlinuz	The kernel image we are loading.
boot=casper	Indicates that this is a live disk booting and to use unionfs.
edd=on	Turns on Enhanced Disk Drive support, very useful in multi disk boxes.
preseed..	The seed file we are passing to the Ubuntu installer.
noninteractive	Indicates that we are using the non-GUI installer.
noprompt	Tells the system to not prompt at the end of install, but just reboot.

The non GUI installer was selected in case the event comes up that video hardware is not supported in X until after postinstall scripts are complete. If you are sure that video hardware is supported by X in all scenarios, you can replace **noninteractive** with **automatic-ubiquity**.

A more complete summary of kernel options available during and related to installation is available at `https://wiki.ubuntu.com/DesktopCDOptions`

3.3 Preboot phase

As soon as the kernel loads, casper will load the contents of the oem.seed file. It will use this to determine what the `early_command` is. The early command starts immediately after the kernel loads but before all of the init scripts start. The current early command will call `scripts/bootstrap.sh`.
This script will then load the OEM framework environment, `environ.sh` and then individually launch all scripts listed in scripts/bootstrap. If a command in one of the scripts has a nonzero return value, `scripts/bootstrap/FAIL-SCRIPT` will be called. If all of the scripts complete successfully, `scripts/bootstrap/SUCCESS-SCRIPT` will be called. Cleanup and recovery options in the fail and success scripts are fully customizable.

> `05-CONFIRM-REINSTALL.sh` checks for the existence of the **REINSTALL** kernel parameter to determine if this is a reinstallation.
>
> `06-CONFIRM-DVD.sh` checks for the existence of the **DVDBOOT** kernel parameter to determine if this is a DVD being booted.
>
> `10-format.sh` formats the disks to start. This ensures that the installation will always look the same when the actual Ubuntu install process gets started. Doing this in all cases (Factory, Reinstall from HD, Reinstall from DVD) will ensure a successful and standard installation.

3.4 Install phase

After the `early_command` completes, the Ubuntu installation will begin. All of the data that was read in oem.seed will be applied to the normal input parameters for the Ubuntu installer, Ubiquity. The exact contents of this file will be explained later. If the installation is successful, the `success_command` from oem.seed will run. If the installation failed, the `failure_command` from oem.seed will launch.

3.5 Post-install phase

The postinstallation script is launched as soon as the installation is complete, whether it succeeds or fails. If the installation failed for any reason, the fail script `scripts/chroot-scripts/FAIL-SCRIPT` will launch. In this script you can perform cleanup or another method to analyze what failed in more detail.

In the event of success, this script will prepare the post installation environment similar to how the preinstall script did. The difference is that all of the scripts launched in postinstall are launched on the resultant system. This means that you can perform package installation, modify configuration files, and tweak anything as necessary in the environment. If any script in here fails with a return code greater than 0, `scripts/chroot-scripts/FAIL-SCRIPT` will launch. The post-installation is broken into three subphases.

3.5.1 os-pre

The *os-pre* phase is the first of the three phases. It is intended to handle files and functions that would break if interfered with by hardware or software enablement teams at inappropriate times.

Currently two scripts are contained in here:

> `10-grub.sh` will add a menu option to the GRUB menu.lst for a hard drive based recovery. If another team decides that additional options must be added to menu.lst, this ensures that the OS requirement is placed first.
>
> `80-install-debs.sh` will install debian archives that provide dependencies for other steps, as well as archives that don't need to be installed in a particular order.

Again, this section is intended to be used by an OS development team for items related to the OS requirements. Hardware and software enablement will happen in the other phases.

3.5.2 fish

The *fish* phase is the second of the three phases. It is intended to handle hardware and software enablement. Teams that need to modify configuration files for default hardware behavior or add drivers should do so here. Also, teams that need to add extra software that did not fall in the *os-pre* step (possibly due to dependency resolution issues) can do so here as well. The framework ships with this directory empty, but some examples are available for things that can go in here.

3.5.3 os-post

The *os-post* phase is the last of the three phases. It is intended for final cleanup steps. The OS team can include scripts here that will be OS wide workarounds as well as final cleanup steps that must happen to prepare the seal on the box.

Shipped here are 4 starting scripts.

> `50-handle-docs.sh` is intended to allow documentation and translation teams to include important documents with the base installation. To prevent having to provide a high bar for them to learn how to package these, this script will simply copy over relevant documentation onto the installation in a standard location.
>
> `60-fix-apt-sources.sh` is used to add additional vendor specific APT sources.
>
> `90-oem-prepare.sh` is used to put the machine into OEM mode for the first boot. This causes the machine to ask for a user name and password on the first boot.
>
> `99-active-partition.sh` sets the active bootable partition on the system. This was intentionally made the last step in case anything failed earlier due to a power outage. If the step is not run, the system will boot up to the recovery partition and the entire process can start over.

4 Recovery installation

During installation, grub is installed to the recovery partition and a default menu.lst is shipped with the OEM framework. If during boot, a user selects the option to reinstall their system, GRUB will chainload to the second partition. This menu.lst is presented to the user giving them the option to recover. The entire process will proceed just like a factory install with the exception of an added boot parameter, **REINSTALL**.

Earlier we skipped over a script, `05-CONFIRM-REINSTALL.sh`. This script checks for the extra boot parameter, and presents the user with a dialog ensuring that they want to wipe the disk. This script comes before `10-format.sh`. If they don't agree to the dialog, the machine will reboot preventing any data loss.

5 DVD installation

Within the framework, a directory isolinux/ is shipped that provides an extra menu option to the standard Ubuntu DVD boot menu. Similar to the HD based recovery, the DVD based recovery provides an extra kernel command line parameter, **REINSTALL**. DVD boots also provides the kernel command line parameter **DVDBOOT** indicating that it is being ran from a DVD.

When `06-CONFIRM-DVD.sh` is ran, the contents of the MBR and the recovery DVD are copied to the user's system. The rest of the installation then proceeds like normal, but instead off of the DVD. This is a little bit slower, but just as functional.
DVD creation will be explained later on.

6 Use case examples

In the Examples Directory you will find a handful of real life examples that have been used with this framework. They have been used for hardware enablement when all the changes could not be easily represented in a debian archive.
If you have a use case that you have found this particularly useful for, please submit it back to this project. Adding it to the examples section will benefit all parties.

6.1 Hardware enablement

There are two exampless of hardware enablements:

70-nvidia-driver.py installs an NVIDIA proprietary graphics driver, removes the warning from Jockey, and then activates it in the xorg.conf. The debs for the NVIDIA graphics driver are not included on the Ubuntu DVD, so they are shipped with this script.

71-thinkfinger.py will install and activate thinkfinger support. The thinkfinger debs come with the Ubuntu DVD so they can be installed from there. The script also turns on the PAM section to allow swiping to login.

6.2 Software enablement

There are also two examples of software enablement:

50-install-DKMS.sh sets up the DKMS package. These debs don't come on the DVD so they are provided independently. Arguably this fits better in os-pre because some hardware enablement relies on it.

90-add-lbm.sh add the linux-backports-modules package to the system. This package provides updated drivers and firmware for certain cards.

6.3 Bug workarounds

85-no-ehci.sh shows how to work around a bug that came with the OS that prevents proper functionality. In the event that it's too late or complex to fix a bug directly in the OS, this is a good example of how to script a workaround into the system. It modifies just the necessary files and in a fashion that would not cause harm later when packages are updated.

7 Debugging

The most important aspect of debugging is identifying which phase of the installation needs to be debugged. The methodology used for each phase can vary depending on the problem you are trying to solve.

7.1 Boot problems

If you have an underlying kernel problem, you will likely get a panic directly upon boot. If you don't, you may be dropped down to a console at which point you can look at the output of `dmesg` for more information. If you are discovering problems during this phase, it's important to file bugs on these issues early as they are most critical and may require a significant amount of time to fix.

7.2 Preboot phase

If the preboot phase is failing, you will be dropped to a console. Debugging this can be done by adding echo statements to the scripts and looking at the output of `casper.log`. Alternatively, you can handle debugging directly in your `FAIL-SCRIPT`.

7.3 Install phase

Debugging the install phase is likely to be the most difficult because problems can either be in the implementation of the preseed file or a bug directly in the installer. The installer will always place debug output in `/var/log/syslog` and possibly `/var/log/installer/debug` depending on the type of crash you encounter.

A useful debug strategy is to boot up a live disk and attempt to simulate an installation environment. You can perform all of the steps normally done in your preboot phase manually. After doing this, you can load your preseed file:

```
cat oem.seed | sudo debconf-set-selections
```

Once the preseed file is loaded, you can launch Ubiquity in **debug** mode.

```
ubiquity -d noninteractive
```

The logging will be much more verbose in `/var/log/installer/debug`. Be sure that if you have any bugs about the installer to file to include your oem.seed in the bug and any of the installation logs that you can provide.

7.4 Post Install phase

The post install phase would be most likely to have problems due to unknown interactions of different team's post installation scripts. Debugging can be performed interactively or retroactively by looking at logs. The post install phase will log all activity to `/var/log/installer/chroot.sh.log` on the resultant system.

You can also interactively prevent each script from launching one by one by creating a file `/tmp/superhalt.flg` on the live filesystem. This will create a flag, `/tmp/halt.flg` before each script runs. When you want the next script to launch, just remove `/tmp/halt.flg`. If you would like all of the scripts to continue on, remove both `/tmp/superhalt.flg` and `/tmp/halt.flg`

8 Preseeding

Each of the items in the preseed file provided with the framework is documented, but if you would like to learn more in depth you can look at the Ubuntu preseeding guide at `https://help.ubuntu.com/8.04/installation-guide/i386/appendix-preseed.html`.

9 DVD Creation

For DVD creation, the current framework expects that you will burn a bootable DVD that loads the standard isolinux setup shipped in the framework.

Additionally, you will have to provide a copy of the system's MBR in a file titled `mbr.bin` in the root of the DVD image and a gzipped copy of the utility partition in a file titled `upimg.bin` in the root of the DVD image. You can choose to remove these portions of the framework as you see fit.

An example application for creating these types of DVDs is available in the Ubuntu archive. Look at the project entitled **dell-recovery**.

10 License

The framework is covered by the GNU General Public License.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

    (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

    (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

    (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly

through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the program's name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty

of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) yyyy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.