A tool to support the introduction of GNU/Linux desktop system in a professional environment

Francesco Di Cerbo[1], Daniele Favara[1], Marco Scotto[2],

Alberto Sillitti[2], Giancarlo Succi[2], Tullio Vernazza[1]

1 DIST – Università di Genova
Via Opera Pia, 13I-16145 Genova
{Francesco.DiCerbo, Tullio.Vernazza}@unige.it,
Daniele.Favara@gmail.com
WWW home page: http://www.lips.dist.unige.it

2 Libera Università di Bolzano/Bozen
Piazza Domenicani, 3
I-39100 Bolzano-Bozen,
{Marco.Scotto, Alberto.Sillitti, Giancarlo.Succi@unibz.it
WWW home page: http://www.unibz.it

Abstract. The introduction of a GNU/Linux-based desktop system in a large company is often problematic, in terms of technical issues but especially for employees' training costs. Mainly, these obstacles are represented by different hardware configurations that might require several ad-hoc activities to adapt a standard release to the specific environment, including company's application profile. On the other hand, GNU/Linux live distributions provide to the users' community new and interesting capabilities, as self-configuration and better usability, but loosing compatibility with original distributions, that is unaffordable in professionals scenarios. DSS (Debased Scripts Set) is an answer to both questions. It is a live distribution that includes an unmodified Debian-based Linux release and a modular-designed file system.

Keywords: GNU/Linux, live distributions, meta-distribution, early user-space, usability, scalability, large environments application deployment

1 Introduction

When dealing with massive installation of desktop computers in a professional scenario, usually the choice falls on proprietary solutions for both the operating system and deployment tools. This happens thanks to their capability to lower total costs in many aspects, first of all simplifying overall complexity and time required for deployment operations.

Please use the following format when citing this chapter:

Di Cerbo, F., Favara, D., Scotto, M., Sillitti, A., Succi, G., and Vernazza, T., 2006, in IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston: Springer), pp. 253-260

The introduction of GNU/Linux[1] desktop systems in large companies is often problematic for both startup and successive maintenance operations. These obstacles are often represented by different hardware configurations that may require several ad-hoc activities to adapt a standard release to the a specific environment. On the other hand, GNU/Linux live distributions provides to the users' community new and interesting capabilities, such as self-configuration and better usability, but the trade-off is represented by some relevant differences with the distributions from which they derive. Such differences make them useless in a professional scenario.

DSS (Debased Scripts Set) is an answer to previous issues. It is a live distribution based on an unmodified Debian[2]-based Linux release (Ubuntu[3]), including a pure "stock" kernel, i. e. a standard distribution-provided precompiled Linux kernel[4]. DSS includes innovative hardware detection and configuration techniques, even if based on sound and largely adopted software (such as hotplug daemon), that is loaded since the very first boot operations. Combining these aspects with a modular software package approach, made it possible by using a specialunification file system (Unionfs [5]), DSS is also able to deploy, in a single package, a customized company-specific release containing both the operating system and all the desired applications.
To summarize, DSS is a framework that allows an easy customization of a 100% Debian-based GNU/Linux live-cd distribution. It provides tools to repackage all the modifications into a derived Linux distribution. Morevover, thanks to its smart file system design, completely constituted by modular parts loaded at runtime, it may be easily repackaged again into a live distribution.

State of the art – Knoppix live distribution

Knoppix[6] may be considered the pioneer of GNU/Linux live distributions, both for diffusion, also demonstrated by a large number of works based on it, and historical reasons.
However, its approach to make a Debian GNU/Linux distribution bootable from a CD / DVD / USB pen-drive, makes its use, in a professional setting, practically impossible, except for data recovery or hardware testing. Its severe modifications to the standard Debian distribution, cross-combining unstable and testing versions, makes new application's distribution and upgrade quite difficult, requiring a great effort to bring to stability a new hypothetical desktop installation based on Knoppix.
Moreover, "exotic" hardware suffers about Knoppix deep-kernel specificity, fit to its hardware detection requirements. Uncommon or not completely supported hardware often comes with drivers usually not contained in standard kernels, which may be provided with commercial license, incompatible with GPL (Generale Public License) statements and so undeliverable inside Debian. In this cases, the adoption of Knoppix can be a great deal. Last but not least, hardware detection and configuration

techniques come with special boot applications (knoppix-autoconfig, hwsetup, kudzu), that require a constant maintenance process to be able to recognize new or uncommon hardware. Moreover, their approach, based on kernel-space routines, forces successive setups (e.g. file systems configuration) to be unfit to use user-space libraries and applications, to give a user flexibility in data and device access, especially in case of plug-and-play USB hardware. Such features use ad-hoc scripts running with maximum privileges, which may lead to security problems, particularly critic in an industrial environment.

DSS main features

DSS adopts a completely new approach to live distributions based on a "early user-space"[7] mode. It is a set of libraries and programs (that are available even without a running Linux kernel) which provide various functionalities required while a Linux kernel is coming up .
The "Early user space" mode allows DSS to use hotplug, a daemon program normally used for hardware discovery and configuration in standard non-live GNU/Linux distributions, since the first boot. This is a great advantage, as the booting kernel relies on already detected hardware, and using its 2.6 series features, may automatically load needed kernel modules to use just discovered hardware, quite like in a common installed GNU/Linux system. Due to this feature, DSS does not require developing and maintaining an ad-hoc kernel, but it may use a stock one, exactly like any other Debian release.
"Early user-space" mode is based on initramfs, a chunk of code that unpacks a compressed file system image (in cpio format) midway through the kernel boot process. It replaces the old initrd file system format, which contained a set of kernel modules stated to be available at boot time, before mounting root file system and so before having all kernel resources available. The main advantage of initramfs is its capability to be used with ramfs, a file system designed to work on physical RAM, scalable in size, instead of usual initrd. This allows DSS, in conjunction with unionfs, to save time in the boot phase: instead of setting up a boot environment for hardware detection/configuration operations, DSS directly sets up a final working environment, and when the kernel finishes its startup operations, the boot process is over, with a simple environment update. This because RAM allocated since boot start for required boot operations does not need to be freed/removed, and running klibc environment is not used anymore except for boot process. Eventually, it is possible to allocate all available RAM on system to improve overall performances, reducing physical medium access delays. Moreover, DSS adopts unionfs, a file system designed to merge different devices, allows to group physical devices with ramfs devices to set up final root filesystem.

In this way, except for a small set of scripts which effectively coordinates boot process, no ad-hoc component is used to bring a Debian GNU/Linux release to be live bootable and completely able to fulfill hardware detection/configuration for Linux's supported peripherals.
DSS is also designed to be a meta-distribution framework, allowing creation of derivative distribution, both live or in standard package, built up upon a pure Debian release, in a very simple way. This feature is provided thanks to a special modular file system design, made possible by adoption of Unionfs[8].
DSS root filesystem is split into modules, which are added together via Unionfs.
All modules in DSS are compressed archives, which can be mounted at runtime, as filesystem. These modules contains programs and libraries, which are merged together into a unique filesystem, thanks to Unionfs; additivity in modules management permits to create a final filesystem layout which may be different from distribution to distribution, allowing different installation profiles, e. g. a server one, without graphical server, or a customized GNU/Linux desktop distribution, containing a specific corporative environment ready-to-use.
Moreover, as compressed modules are merged in an ordered way, a single installation may be multi-purpouse, including or excluding any of them from boot loader parameters. This feature is very important to contain different installation profiles in a single location, and it's extremely useful in a network installation, or in a DVD release, for example.
Module creation process is also very simple, and it may be created in two way: non-interactive, which relies on "debconf" program, just producing a list of desired debian packages to include in outcoming module, and a script would download packages and compress them into a cpio archive, or in an interactive way, booting DSS, using "synaptic" program and then executing another script. Resulting archives may be redistributed inside a standard DSS release without any further modifications to original status.

Key technology: UnionFS

Unionfs is a stackable file system that operates on multiple underlying file systems. It merges the updated contents of multiple directories but keeps their original physical content separated. The Dsslive implementation of UnionFS merges the Dsslive RAMdisk with the read-only file systems on the boot CD so it's possible to modify any read-only file as if it was writeable. UnionFS is part of FiST, File System Translator project. Its goal is to address the problem of file system development, a critical area of operating-system engineering. The FiST lab notes that even small changes to existing file systems require deep understanding of kernel internals, making the barrier to entry for new developers high. Moreover, porting file system code from one operating system to another is almost as difficult as the first port.

FiST, developed by Erez Zadok and Jason Nieh in the computer science department at Columbia University, combines two methods to solve the above problems in an innovative way: a set of stackable file system templates for each operating system, and a high-level language that can describe stackable file systems in a cross-platform portable fashion. The key idea is that with FiST, a stackable file system would need to be described only once. Then FiST's code-generation tool would compile one system description into loadable kernel modules for different operating systems (currently Solaris, Linux and FreeBSD are supported).

DSS inside UnionFS

Dsslive within the "pre-USS" script mount different compressed file systems in different mount points and uses a read-writable directory as last layer, with a outcome to have just one final mount point (the root directory). UnionFS allows DSS to virtually merge- (or unify-) different directories (recursively) in a way that they appear to be one tree; this is done without physically merging the directories content. Such namespace unification has a benefit in allowing the files to remain physically separate, even if they appear as belonging in one unique location. The collection of merged directories is called a union, and each physical directory is called a branch. When creating the union, each branch is assigned a precedence and access permissions (i.e., read-only or read-writable). Unionfs is a namespace-unification file system that addresses all of the known complexities of maintaining Unix semantic without compromising versatility and the features it offers. It supports two file deletion modes that manage even partial failures. It allows efficient insertion and deletion of arbitrary read-only or read-writable directories into the union. Unionfs includes in-kernel handling of files with identical names; a careful design that minimizes data movement across branches; several modes for permission inheritance; and support for snapshots and sandboxing.
Unionfs has an n-way fan-out architecture [5,6]. The benefit of this approach is that Unionfs has direct access to all underlying directories or branches, in any order.
Even if the concept of virtual namespace unification appears simple, there are three key problems that arise when using it as root file system of Dsslive.
The first is that two or more unified directories can contain files with the same name. If such directories are unified, duplicate names must not be returned to user-space for obvious reasons. Unionfs solves this point defining a priority ordering of the individual directories being unified. When several files have the same name, files from the directory with higer priority take precedence.

The second problem relates to file deletion. Files with same name could appear in the directories been merged or files to be deleted reside on a read-only branch. Unionfs handles this sitruation inserting a without, a special high-priority entry that marks the file as deleted.

When file system code finds a without for a file, it simply behaves as the file doesn't exists.
The third problem is relegated to the previous one and it involves mixing read-only and read-write directories in the union. When users want to modify a file that resides in a read-only branch, Unionfs performs a "copyup", the file is copied to the higher priority directory and modified there.

Unionfs and The Upstream Salmon Struct (USS)

The power of Dsslive resides on its design, offering high modularity and allowing the customization as easy as possible. This has been achieved by designing the USS and using Unionfs as background.

The unified root file system is made of the content of different modules, each module is a squashfs compressed file system:

1.base: console mode module, it contains a basic bootstrapped debian system;
2.kernel: it contains the /lib/modules/ directory plus kernel related utilities;
3.xserver: graphical mode modules, (in case of file names clash, the priority in the unified directory is defined by sorting the modules name);
4.deliver: it contains the runlevel scripts needed to reconfigure "debconf" database and the environment reading the user configuration from /proc/cmdline passed to kernel at boot from boot loader (e.g.: locales information, force screen resolution);
5.overall: the read-writable branch, it can reside in ram or even be an external hd;
Base, kernel and xserver use is self-explaining enough, but the packages inside those modules are stored using a "noninteractive" debconf frontend, and so they maintain their own default configurations, that's why Dsslive can be considered a pure debian system booting from a cdrom. Anyway to allow the user to use his own locales settings (i. e. language, keyboard) and video card optimized drivers, some packages need to be reconfigured: and this is made using the runlevel scripts in deliver.

Deliver

The scripts in "yuch-bottom", the directory within the initramfs, write the environment variables in the file /etc/deliver.conf, parsing command line parameters from boot loader, as lang(uage), username, hostname etc. Deliver uses those variables to reconfigure some packages, upgrading at the same time the debconf database.

The scripts in deliver are plain text bash scripts, this allows DSS use not only for a i386 livecd distribution, but even for powerpc or sparc computers, and all the other 11 architectures that debian supports, making DSS fully architecture-independent.
Thanks to its scripts, DSS, to be ported from an architecture to another, just needs a right initramfs and the deliver module, without caring about kernel customization, as it is sufficient a pure debian stock kernel.
Dsslive, differently from knoppix, uses debconf to configure the system, which provides a consistent interface for configuring packages, allowing to choose from several user interface frontends. It supports even a special "pre-configuration" of software packages before they are actually installed, which allows massive installation or upgrade sessions demanding all necessary configuration informations up front, without user interactions (frontend "noninteractive"). It allows to skip over less important questions and informations while installing a package, giving anyway a chance to revise them later.
It is also interesting to remark that debconf itself is completely a Debian supported tool, and its use is not customized at all: another key point into 100% Debian compatibility.

Conclusion

DSS is a 100% Debian live distribution, and may be proficiently used to install a pure Debian system on a desktop pc. Thanks to its features, it's very simple to customize starting base version, in a way to meet, for example, large-scale installations with specific requirements, such as in large companies networks. Its maintenance is not effort-prone, due to adoption of standardized technologies, but their use in a live environment, thanks to DSS innovative design, represents a unicum in current scenario. Moreover, there are no limitations to port DSS into any of Debian supported architectures, of to use it in embedded systems.

References

[1], Stallman, R. et al., Free Software, Free Society: Selected Essays of Richard M. Stallman, , www.gnu.org
[2], Ian Murdock, "Overview of the Debian GNU/Linux System", Linux Journal, Volume 1994 Issue 6es
[3], Ubuntu group, Ubuntu philosophy, , http://www.ubuntu.com/ubuntu/philosophy
[4], D. Rusling, The Linux Kernel, , http://www.tldp.org/LDP/tlk/tlk.html
[5], E. Zadok and J. Nieh, FiST: A Language for Stackable File Systems, 2000
[6], Knopper, K. "Building a self-contained auto-configuring Linux system on an iso9660 filesystem", Usenix 2000 Conference

[7] Petullo, M., "Encrypt your root filesystem", Linux Journal,Volume 2005 , Issue 129 (January 2005) Page: 4, 2005,ISSN:1075-3583
[8], CP Wright, J Dave, P Gupta, H Krishnan, E Zadok, Versatility and Unix Semantics in a Fan-Out Unification File System, , http://www.fsl.cs.sunysb.edu/docs/unionfs-tr/