pure:dyne

Aymeric Mansoux and **Antonios Galanopoulos** and **Chun Lee**
Goto10
12, rue Charles Gide
86 000 Poitiers
France
contact@goto10.org

Abstract

pure:dyne is a live GNU/Linux distribution optimized for the purpose of real-time audio and visual performance. As its name suggests, pure:dyne is built upon the dyne:II platform and originally optimized for PureData. However, pure:dyne now also contains several other interesting and useful creative software, and is becoming evermore practical to be used as a complete GNU/Linux distribution for both media art and daily tasks. This paper therefore aims to introduce and discuss several aspects surrounding pure:dyne thus encouraging the usage and feedback of this project.

Keywords

goto10 dyne live-distribution PureData Supercollider Csound media-art FLOSS

1 Introduction

The development of pure:dyne [1] can be traced back to the inclusion of PureData in the dyne:bolic liveCD distribution [2]. As this addition later became increasingly popular, there was suddenly a demand to increase its support for PureData in a more serious production context. Meanwhile, the dyne:II [3] core that Denis Rojo [4] had been developing for the forthcoming version of dyne:bolic provided the necessary development tools needed to make such customized distribution for PureData. As a result, a collaborative effort had begun between dyne.org and Goto10 in early 2005 to work towards a distribution based on the dyne:II core.

After a year of development, pure:dyne started to take shape and began its beta testing. In late 2006, pure:dyne officially left beta to have its first public release. Today, pure:dyne gathers a growing user community and has been used in numerous workshops and performances.

Although many multimedia oriented live GNU/Linux distributions can be found nowadays, many aspects of pure:dyne still remain unique amongst them. This paper hopefully will introduce and demonstrate such features and design, and ultimately encourage its usage.

2 Design principles

Throughout the process of making pure:dyne, several design principles were clearly outlined from the beginning. They can be briefly listed as below:

- pure:dyne is made by practitioners for practitioners
- pure:dyne should be accessible to non technical users
- pure:dyne will be optimized and kept minimal

One of the most important aspects of pure:dyne is that it attempts to offer both "practical" and "portable" solutions for the practitioners in the fields of FLOSS based digital art. Although there are many portable distributions available, they are mostly used for demonstration purposes. pure:dyne, on the other hand, allows artists to build extensive works upon it while keeping the entire system, including artists' works, very portable. This makes it an attractive alternative for artists who wish to develop projects but do not have access to a dedicated environment.

Moreover, accessibility is also an important part of pure:dyne. pure:dyne recognizes artists who intend to take advantage of the innovations in creative FLOSS but do not have the resources and abilities to walk through the lengthy installation, configuration and even compilation of such software. Because of this, pure:dyne aims to provide a working environment that requires a minimal learning curve to be productive with it.

Lastly, pure:dyne follows a minimalist approach in system setup. This enables it to be more streamlined and "clutter free". For example, the default desktop environment is FluxBox as window manager and applications such as Rox-Filer and Xfe can be used for handling the conventional representation and navigation of directories. pure:dyne also includes window managers such as ratpoison, evilwm and dwm. Such an approach enables users to achieve greater productivity when using the system.

[1] http://puredyne.goto10.org

[2] The first inclusion of PureData can be found in dyne:bolic1.4

[3] dyne:II is platform in which a fully functional system can be built upon it. for more detail, please refer to section of this paper.

[4] founder of Rasta Software and the key maintainer of dyne:bolic

3 Usage

Before this paper proceeds further, there are two important concepts in pure:dyne that should be clarified.

- **Dock** - A dock refers to an "installation" of pure:dyne onto the host system. A dock contains all necessary components that are required to boot pure:dyne entirely from the storage device. The process of docking is extremely straightforward, it only requires copying the /dyne directory from the CD or ISO image onto a partition readable [5] by pure:dyne.
- **Nest** - A nest (.nst) is a file that a user can create once pure:dyne has successfully booted. This file contains a user's home directory and configuration files [6]. The nest file can be stored either on the hard disk or on a portable storage device such as a usb key. During the boot process, pure:dyne will look for the nest in any of the partitions it finds and mounts the nest at the appropriate location. Through the integration of UnionFS [7], users can easily save and store any modifications made on the system.

With further development of the dock and the nest in dyne:II [8], pure:dyne can be used with a great deal of flexibility. For example, a system running from a CD or hard disk, in combination with a portable storage device will result in a complete functional system. Once the system is successfully booted, a user can simply write to his or her own home directory and continue working the same way no matter which storage device is being used. One other obvious advantage of the docking system is that pure:dyne can co-exist with other operating systems in a very straight forward manner, as everything is contained in one single directory. Updating to a newer version of pure:dyne only requires to overwrite the content of the dyne directory. Lastly, by simply creating new users following the conventional GNU/Linux method, a nest can also support a multiple user system.

dyne:II also contains a modular system in which applications can be packaged and distributed. Each package is a compressed [9] .dyne file in the /dyne/modules directory. For instance, the applications contained in pure:dyne are simply a pure.dyne module of dyne:II. This means that users and developers can simply package their favorite applications [10] and exchange between them. To include a new module, simply copy the .dyne file into the /dyne/modules directory and either reboot or mount the module directly.

To summarize, pure:dyne can be used/installed in the following ways:

- Used with the CD alone, without saving user data
- Used with the CD in conjunction with a portable storage device that contains the nest
- Used with a dock on the hard disk plus a nest either on the hard disk or portable storage device
- Used with both the dock and the nest on the portable storage device. for example, running pure:dyne entirely from solid state memory.

4 Optimization

As mentioned previously, pure:dyne's main emphasis is in the context of real-time applications. Because of this, pure:dyne consists of several optimizations that are different from dyne:bolic 2.x.

Firstly, the optimization is targeted at the i686 architecture. This is because pure:dyne aims to support more modern hardware, as the real-time audio/visual applications are typically more demanding on cpu cycles. pure:dyne employs the kernel based on Ingo Molnars's real-time patch.

Secondly, pure:dyne makes the installation of necessary drivers to take advantage of the hardware possible and straightforward. For example, one can easily install the ATI and NVIDIA graphics driver to take advantage of the modern graphics cards in order to obtain the acceleration required by video/visual applications. Furthermore, it also includes support for various firewire sound cards through the FreeBoB driver.

Lastly, the gcc compilation flags used in pure:dyne are typically more aggressive than those used in dyne:bolic or usual binary based GNU/Linux distributions. Currently, relevant applications in pure:dyne are compiled with the following flags: -O3 -ffast-math -fomit-frame-pointer -mmmx -msse -pipe.

5 Applications

Applications that are optimized in pure:dyne can be briefly listed below [11] consult the pure:dyne website:

- PureData
  - PureData
  - Gem, PDP, PiDiP, GridFlow of the externals and abstractions from the PureData cvs

[5] current supported filesystems are: fat vfat msdos ntfs ufs befs xfs reiserfs hfsplus ext2 ext3

[6] A nest contains the /home, /root, /var, /tmp and /usr/local

[7] UnionFS allows transparent overlay of files and directory from different filesystems. http://www.unionfs.org

[8] Both dock and nest existed in dyne:I. However, these two elements were significantly further developed in dyne:II

[9] .dyne modules use the squashfs read-only filesystem. http://squashfs.sourceforge.net

[10] currently there are modules for Ardour, network tools, Gimp, OpenOffice, BitTorrent, dvd authoring and more

[11] For the complete listing, please

- Audio
  - SuperCollider
  - Chuck
  - Csound
- Visual
  - Fluxus
  - Packet forth

Besides the pure.dyne module, the pure:dyne distribution also comes with the audio.dyne module from dyne:bolic 2.x which provides the applications for hard disk recording, sequencing, and sound editing. Furthermore, Jack is provided by the dyne:II core. As a result, the combination of pure:dyne with the audio module would result in a fairly comprehensive digital audio workstation.

6 dyne:II and pure:dyne

As mentioned previously, pure:dyne is built using the dyne:II platform. There are, however, some subtle differences that should be pointed out.

dyne:II is a system derived from LFS[12] and initiated by Denis Rojo and Alex Gnoli. It provides the core functionalities such as booting, nesting, docking and the modular system. In other words, dyne:II offers a platform on which a complete live distribution can be built. For example, dyne:bolic 2.x is developped using the dyne:II core system.

pure:dyne, on the other hand, is not only a live distribution built using the generic dyne:II, but also contains several customized core components. Because of this, pure:dyne can be said to be using a customized dyne:II system developed by Goto10. For example, pure:dyne contains its own kernel and has a different optimization policy. Moreover, pure:dyne also provides some developers' tool that are unique to it. In short, pure:dyne not only consists of a pure.dyne module but also its own branch of the dyne:II core system.

The relationship between the two cores is by no means independent of each other. That is to say, the development remains very close between them. As a result, changes can be merged. For example, new features introduced in the modified pure:dyne core could potentially be merged into the generic dyne:II core after they are tested and have proven to be stable. Similarly, new components in generic dyne:II core can be adopted by pure:dyne's customized core. Last but not least, any dyne modules are universal whichs allows applications to be used and shared between the users of these different systems.

7 Which Dyne?

Because of the differences between dyne:bolic 2.x and pure:dyne, it can sometimes be ambiguous as to which of the two should be used. In general, dyne:bolic 2.x offers more stability across a wider range of legacy hardware and pure:dyne is somehow more "bleeding-edge", it adopts the latest drivers and patches to gain performance. Moreover, pure:dyne is created for a very specific context, while dyne:bolic 2.x can be seen as more generic.

	pure:dyne	dyne:bolic 2.x
Type	live-distribution	live-distribution
Core	dyneII customized	dyneII generic
Module policy	.dyne	.dyne
Target hardware	i686	i586
Optimization	aggressive	generic

Table 1: comparison of pure:dyne and dyne:bolic 2.x. note that only the audio related modules are listed in the table

8 Current status

Currently, pure:dyne is at its first official release (version 2.3.6). It has already proved to be usable and stable enough to be employed in real world scenarios. For example, pure:dyne has been used in many workshops where participants are able to learn the software and have a complete and functional system to carry on their learning after workshops ended. Furthermore, pure:dyne has also been seen used in live performances and even installations for a period of weeks without problems.

9 To do

Efforts will be aimed towards the development of the following parts of pure:dyne at this moment.

9.1 Documentation

Apart from the actual live distribution, pure:dyne would also like to provide documentation for the necessary components surrounding its usage and development. For example, there should be a knowledge base for users to learn more about the software it includes and also on more advanced configurations. Developers should also be able to find necessary information to further customize it to suit their needs and ultimately take part in producing the future releases.

9.2 Hardware support

pure:dyne would also like to provide more stable support for hardware that is commonly used by practitioners in the digital art scene. For example, the support for the MacIntel machines is currently being tested and will hopefully be included in the next stable release.

9.3 Software package

pure:dyne is always keen to discover interesting and innovative creative software to include in its package. This is a constant reminder for the pure:dyne developers. After the release of the second stable version,

[12] Linux From Scratch. http://www.linuxfromscratch.org/

a particular attention will be given to a compilation of interesting FLOSS based software art.

10 Conclusion

The collaboration between dyne.org and goto10.org has led to the successful production of pure:dyne. More importantly, many of its current users find pure:dyne useful and it has proven to fulfill its original goals. pure:dyne hopes to continue the fruitful relationship with dyne.org to achieve a higher standard in the future. This would hopefully contribute to raising the awareness of FLOSS culture and tools in the digital arts scene.

11 Acknowledgment

Goto10 would like to thank Denis Rojo and Alex Gnoli for developing dyne:II and their valuable help and advices throughout the making of pure:dyne. Goto10 would also like to thank the digital research unit of Huddersfield in UK for their support during the start of the project. Goto10 would also like to thank BEK in Bergen and Waag Society in Amsterdam for providing the hosting solution for pure:dyne. Lastly, Goto10 would like to thank everyone who contributed to pure:dyne by developing it, using it and disseminating it.