Software distribution with Klik

LAUNCH CONTROL

Klik brings convenient two-click installation to the KDE desktop.

BY TIM SCHÜRMANN

Linux systems that run directly from the CD are very popular. These live distributions not only help attract new users to the world of Linux but are useful if you need to repair a broken system. When something breaks, you might find yourself in a situation where the tool you need for the repairs is not available on the CD. If you have been through this before, you will be familiar with unresolved dependencies and missing libraries on read-only filesystems. Even if you manage to install the tool, it might just die despite all your efforts. And users often want to try out new programs on production systems without having to go through a global install.

The Klik project makes all these dreams come true. Klik is short for "KDE-based Live Installer for Knoppix + Kanotix." Its makers have adopted the bundle design made so popular by Apple. Bundles are typically special, compressed archives containing all the files and libraries required by an application along with some metadata. To install, users simply download the archive file off the Internet and drop it on their desktops; the operating system launches the application stored in the archive. The system hides the processes from the user (Figure 1). As there is no need for a complicated and time-consuming install, and no need to spread files all over the filesystem tree, you can even install different versions of the same program. The basic Linux system is untouched. And you can dock applications stored on USB sticks to make them available to a live distribution.

Kernel Matters

Klik does require some preparatory work: Suse Linux 10 (including Open Suse) and Debian-based systems come with the prerequisites (see the "Supported Distributions" box), and let's not forget Knoppix and its derivatives, which triggered the development of Klik. This history also explains why the system was available for the KDE desktop only at first. Thanks to numerous contributors, Klik now also runs on Gnome. If your Linux system is not directly supported, your kernel must at least support the Cramfs filesystem. The kernel sources typically have this support in the form of a module, but you can build in static support. The same applies to the loop device for mounting the image files.

The Client

After fulfilling any prerequisites, you can type the following at the command line: *wget klik.atekon.de/client/install -O - | sh*. This downloads an installation script from the Klik homepage, and the script installs the Klik client. On multi-user systems, this step has to be performed for every user who will be running Klik, as there is no global install at present. If you are running the script for the first time, and are not the root user, you also need the following commands:

```
su
sh klik-cmg-install-root
```

This adds a few lines needed by Klik to */etc/fstab*. After completing the install, a

Supported Distributions

The following distributions support the Klik system: Suse Linux 10 (including Open Suse), Debian, Linspire, Ubuntu, Kubuntu, Kanotix, and Knoppix. Fedora Core 4 is in the pipeline. Gentoo and Mandrake/Mandriva require a new kernel as they do not support Cramfs. The current versions of Knoppix, Kanotix, and the SLICK-enhanced version of Open Suse (see *http://www.opensuse.org/SUPER_KLIK*), and CPX-MINI include the Klik client.

Other Browsers

Klik supports a few other browsers. Opera, the Mozilla suite and its successor Firefox are automatically set up by the installation script. You will need to register the protocol in all other cases, or if you run into difficulty. Most browsers give you a menu item somewhere in the settings or preferences dialog. In Opera, for example, this is the *Preferences... Advanced* tab, following *Programs*. After locating the correct item, select the *klik* protocol and assign the *.klik* application in your home directory.

window appears to tell you that all is well. Following this, Konqueror pops up showing you the Klik homepage with a menu of all Klik-aware applications registered with the project. Of course, you are not restricted to using the Konqueror browser; see the “Other Browsers” box for details.

Klik Me!

The Klik installation script also registers a new protocol with Konqueror in the course of the install. If you then enter a URL such as *klik://xvier*, Konqueror hands your request to the Klik client, which then downloads *xvier* off the Internet and drops it on your desktop, typically using the *$HOME/Desktop* directory to do so. If you check the directory, what you see is a big file. But clicking on the file icon launches the application hidden in the file.

A quick search on the Internet, links on the program homepage, or the catalog on the Klik homepage tell you which URL represents which program. At this time of writing, about 4000 applications are available. Clicking on a link launches a download, which will follow the pattern described earlier.

If you see a message that says *Error while trying to run program*, you can assume that a network error has occurred. Either your Internet connection is down, or the Klik server may be temporarily unavailable. Judging from postings on the Knoppix forum, this happens a lot. If you wish to run your own Klik server, check out the “Klik Server” box for more details.

Under the Hood

With the exception of one-off changes to the */etc/fstab* file, none of these tasks require root privileges. The installation script only affects the home directory of the installing user. But if you check your home directory for signs of a Klik client installation, you might overlook the client at first glance. Klik itself comprises two short scripts named *.klik* and *.zAppRun*. The other changes simply affect the KDE configuration files.

This is why Klik will work without a GUI. When a script is launched, it requests the program from the Klik server. If the program exists, the server just returns another script. This so-called recipe contains details of where the binary version of the requested software package is available and a description of how to create an all-in-one package for the distribution in question from the files contained in the package. This construction plan includes details on resolving dependencies. The finished product is referred to as a bundle. There are examples of recipes for you to follow at [3] and [4]. Any libraries and resources required to run the program are stored in the bundle. To achieve this, Klik scripts smuggle the bundle components past the package manager. This explains why different versions of the same program can be talked into peaceful coexistence.

As you may have gathered, these bundles are not available in a ready-to-run form. Bundles are put together just-in-time; that is, they are individually generated when a user requests a bundle for a specific distribution. This not only saves Klik service providers a lot of webspace, it also means that the developers retain control of their program packages. Unfortunately, there is a drawback. As you might expect, generating bundles for complex software products is anything but trivial. For example, Suse uses the RPM package format, and this means having to convert Debian packages. And different distributions come with different software environments. As a consequence of this, recipes can’t be a perfect match for all distributions. This is why the Klik server has a section for handmade bundles that will run on all supported distributions. There is even a separate bundle archive for Suse Linux at [2], although it has been dormant for many months now.

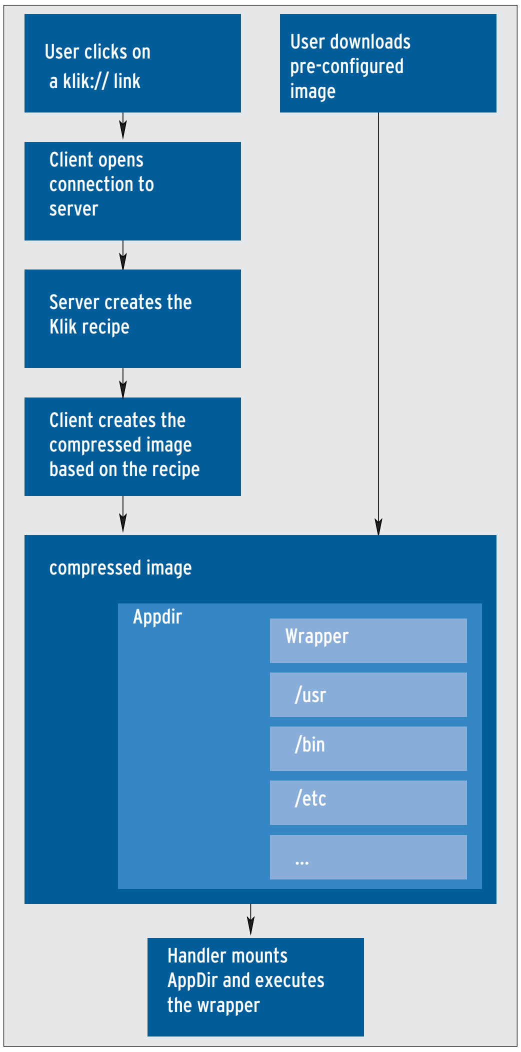
User clicks on a klik:// link
User downloads pre-configured image
Client opens connection to server
Server creates the Klik recipe
Client creates the compressed image based on the recipe
compressed image
Appdir
Wrapper
/usr
/bin
/etc
...
Handler mounts AppDir and executes the wrapper

Figure 1: Overview of Klik architecture: the browser passes the Klik link (1) to the client, which contacts the Klik server (2). The server returns a script, or so-called recipe (3), which the client then uses to create an AppDir (4). The directory is then packed and compressed (5), and finally mounted by the .zApp script (6).

Klik Server

The Klik client always accesses an Internet-based Klik server, but it can make sense to run your own server, especially on closed Intranets, as a good way of making sure that your users get the right versions of the right programs. To point the Klik client at your server, look for the line in the *install* script, which calls *wget* to pick up the recipes. Change the line to reflect your web server’s address, and make sure the web server has the required recipes.

Another issue occurs in this context. As most of the 4000 recipes on the Klik server were automatically generated, the Klik server has a large number of packages that will not give you an independent bundle. In this case, you see the message *This package contains no application. klik can't handle it.* The people who run the Klik server are slowly but surely removing the recipes that cause this issue, but this still leaves many useless recipes at this time of writing.

Packing Bundles

To understand what happens when you launch a Klik application, let's take a closer look at the bundle structure. The bundle contains a compressed filesystem with all the application directories and the files they contain. Klik uses the Cramfs or Zisofs filesystem.

Figure 2 gives you a clearer overview of how Klik works: the package builder first collects all the files and libraries for an application in a single directory, which is referred to as the AppDir (application directory). A special script titled *wrapper* resides at the root of each AppDir.

When a user clicks the AppDir, the operating system does not change to the subdirectory, it launches the script instead. As directories are fairly difficult to distribute, the package builder often just adds an AppDir to a known bundle, which can be a ZIP archive, or in the case of Klik, an image.

There is no clear distinction between the terms AppDir and bundle. In the wild, you are bound to stumble over these terms, which occasionally mean different things.

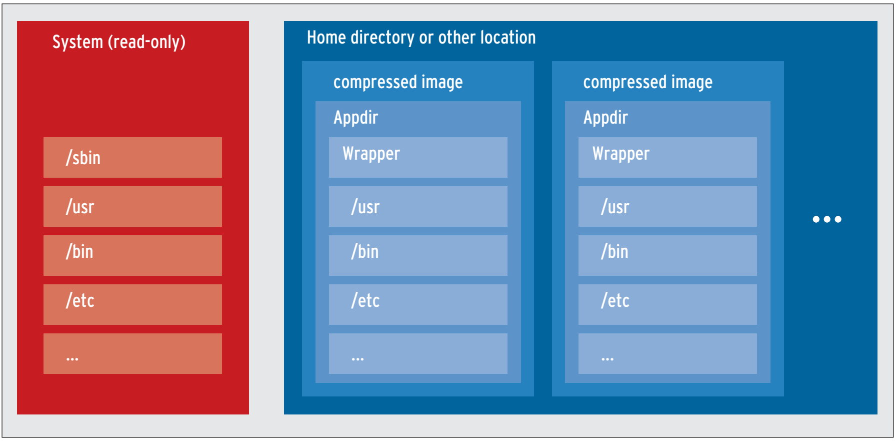

Figure 2: Each bundle comprises its own Linux filesystem with a wrapper script at the top. The directory with the files is referred to as the AppDir. The AppDir directory is compressed for Klik. As the figure shows, multiple bundles can coexist.

The Goodies

The *.klik* script mounts the image in */temp/app/1* via the loop device. If you click the bundle in Konqueror, you can access the content via the directory. If you launch more than one instance of an application, Klik assigns a serial number, instead of *1*. The second program would thus reside in */temp/app/2*.

To allow normal users to benefit from this design, the file needs mountpoints in */etc/fstab*. The Klik installation script automatically creates the required mountpoint entries. If necessary, users can unpack the bundle using filesystem tools: */sbin/fsck.cramfs -x Myprg Myprg.cmg*, and then manually launch the wrapper script: *cd Myprg; ./wrapper*. Incidentally, this means you can modify the package, and then regenerate the bundle by entering */sbin/fsck.cramfs Myprg/Myprg.cmg*.

Eight Times Table

The current Linux kernel restricts the number of simultaneously mounted loop devices. In other words, you can't launch more than eight applications from bundles at the same time. Although you could increase this (to 64 in our example) via a *linux max_loop = 64* boot option or an entry for *options loop max_loop = 64* in */etc/modules.conf*, you might find it hard to run a production system based on bundles, one reason for this being the performance hit.

Incidentally, AppDirs can only house well-behaved programs. If an application needs to spread configuration files all across the system, you run into a brick wall with Klik. Additionally, there is no guarantee that Klik will help you avoid version conflicts. As Klik mounts the bundles in the filesystem tree, conflicts with installed predecessors could occur, depending on the dependencies that need to be resolved.

Conclusions

Klik is lean, elegant, and simply-structured – even under the hood. It makes child's play of single-click installation and distribution of software packages. Developers are well-advised to take a look at the system.

This said, Klik is not suitable for every task. The restriction to eight simultaneous applications, and the lack of a bundle package manager with an automatic update mechanism, have prevented Klik from replacing other package tools. ■

Direct Launch

The *.klick* script always creates a bundle with the *.cmg* suffix and then launches the application the bundle contains. However, there is no need for this, if the bundle already exists on your desktop. In this case, you can simply run *.zApp*, passing the bundle filename to the script as a parameter. If you prefer, you can use Binfmt to register bundles as executable files:

```
mount -t binfmt_misc none ↩
/proc/sys/fs/binfmt_misc
echo ':CMG:E::cmg::↩
/pfad/zu/.zAppRun:' > ↩
/proc/sys/fs/↩
binfmt_misc/register
```

INFO

[1] Klik homepage and software warehouse: *http://klik.atekon.de/*

[2] Archive with ready-to-run bundles for Suse Linux 10: *http://opensuse.linux.co.nz/klik/10.0/*

[3] Sample recipe for Scribus: *http://klik.atekon.de/scribus.recipe.example*

[4] Another sample recipe with annotations: *http://klik.atekon.de/architecture/recipe.php*

[5] Klik forum on the Knoppix pages: *http://www.knoppix.net/forum/viewforum.php?f=17*

[6] Information for developers: *http://klik.atekon.de/docs/?page=A%20note%20to%20application%20developers*