Software & Tools

i-Installer: The evolution of a TEX install on Mac OS X

Gerben Wierda

Abstract

This articles reviews the past, present, and future of the i-Installer program on Mac OS X developed by the author, and its related TEX (re)distribution.

1 Apple, NeXT and Mac OS X

Let's start with some history of Apple, as a foundation for the discussion of the i-Installer program and its related TEX redistribution on Apple systems.

Nowadays, perhaps a few percent of PC users use Apple Macintosh systems. This may not sound like much, but subtract the enormous number of tightly regulated office desktops in large corporations and other institutions and we are talking about a sizeable chunk of the desktop population. These days, Apple's market share is again rising drastically. Apple, in fact, has seen a turnaround not often witnessed in the computer world. From 'beleaguered', the company has become a succesful forefront of innovation.

Most people these days know Apple for its iPod music player and the revolution it has ignited in the music world. But Apple is still above all a computer maker, which a few years ago was in desperate need of a new operating system, having failed at two attempts at producing a modern operating system itself.

To solve this problem, Apple acquired NeXT, the company that Apple founder Steve Jobs started[1] when he left the company after a row about future developments. When Jobs started NeXT, the idea was to build the best desktop possible. As Jobs put it, NeXT would be either the last big success or the first big failure on the desktop. Technologically it was a big success, but it failed in the market. In fact, Jobs was too late. A de facto standard for word processing had arrived (Microsoft Office) which was strengthened by the advent of increased communiation between computer systems, ironically one of NeXTSTEP's key strengths. Microsoft, which saw NeXTSTEP as a possible competitor for OS/2 and later Windows NT, was not interested in strengthening the platform with a version of its office suite.[2]

When Jobs started NeXT, he wanted to rectify some mistakes he had made with the Macintosh while at Apple. As Jobs puts it, when Apple visited Xerox PARC, they had witnessed three revolutions, but they were so mesmerized by one (the graphical user interface) that they completely overlooked the other two (networking and object-orientation). For the robust foundation of the new system he chose the BSD flavour of Unix, but based on the latest of kernel technology: Mach from Carnegie-Mellon University.[3]

When Apple acquired NeXT, it acquired not only a very modern operating system,[4] it also acquired the human talent it desperately needed to turn the company around. Apart from Steve Jobs himself, that included Avie Tevanian for software and (though NeXT had stopped making hardware a few years earlier) Jon Rubinstein for hardware. Within a relatively short time, key positions at Apple had been filled by former NeXT employees, thus prompting the often heard comment that NeXT had taken over Apple from the inside. The years since have seen the dramatic shift in Apple's fortunes, and all that during the IT-bust that followed the dot com boom. As Jobs announced when he became CEO: he wanted to innovate Apple out of the downturn and he actually succeeded in doing so. Of course, the fact that Microsoft has not abandoned Apple has helped considerably.

In many ways, therefore, Mac OS X is a much changed and enhanced version of NeXTSTEP, which in its time was a very smooth TEX environment. The foundation remains a Mach kernel and a BSD layer (both open sourced by Apple), on top of which lives a graphics layer based on PDF.[5] This combination of Unix on the one hand and PDF on the other makes it a very welcome environment for TEX, especially since the emergence of pdfTEX. Having pdfTEX, everything for creating, displaying and printing TEX is available. A Unix-level editor or the standard GUI editor (TextEdit) can be used to write the TEX source. The Unix command line can be used to create the PDF output which can be viewed by the

[1] Besides starting NeXT, Jobs acquired Pixar from George Lucas; Pixar became a huge success in digital movie production.

[2] There was a version of Wordperfect, but though it was the best version of the program available, Wordperfect Corp. had to abandon it because it came under pressure from the emergence of the Microsoft Office monopoly. A very innovative spreadsheet program, Lotus Improv, met the same fate.

[3] Its main designer, Avie Tevanian, chose NeXT above Microsoft for his career and is these days Apple's Vice President in charge of software development.

[4] In principle and in its core, many interfaces, like the BSD Unix interface, had not been kept up to date.

[5] NeXTSTEP was based on Display PostScript.

default PDF previewer (Preview). It is not comfortable, but it is enough.

The step from there to an integrated front end is simple on Mac OS X. With the object oriented frameworks and a nice programming IDE at the disposal of a programmer, a GUI application with an edit window (based on the available text object) and a display window (based on the available view object, which handles PDF) and calling Unix TEX to do the work behind the scenes, a front end is much easier to write than on any other platform. After all, the whole rendering of the result is already taken care of by the operating system. On such a basis the first TEX-IDE (TeXShop) was developed and the developers could spend time on improving the user interface without having to worry about basic technical issues.

1.1 Installing software on Mac OS X

Basically, there are two ways of installing software on Mac OS X:

- Drag and drop
- Installer program

Just dragging something to a folder is of course the simplest metaphor for installing software and is preferred in most situations. Even situations that require configuration and sub-installation in protected locations can use this method if the application itself detects that it has not been installed yet when it is run for the first time. Microsoft Office can be installed this way on Mac OS X.

Using an installer is better for more complicated installs, especially if the location of the install is fixed. After all, installing something at the Unix level often requires something to be installed in a special location in the Unix domain. For these kinds of installs, Mac OS X comes with a program named Installer, that installs "packages" (with the `.pkg` extension). An Installer package contains an archive of software to install, some meta data about location, etc., and may contain scripts which are run just before and just after the archive has been unpacked. The Installer in Mac OS X is a derivative of the Installer.app that was part of NeXTSTEP.

There are also some other commercial installers, such as VISE, but they will not be taken into account here. Mostly they are used by software distributors who used to ship software for the classic Mac operating system.

2 Installing TEX

A front end for TEX on Mac OS X can be a simple application residing anywhere on the system. These are therefore generally installed by means of a simple drag and drop.

The back end is more complex. Though it can in principle be installed anywhere, it does require quite a bit of configuration afterwards, in terms of the paper size, hyphenation patterns, formats, etc. And as Mac OS X is a multi-user environment, having one TEX install to be used by all requires these configurations to be executed such that all can use the results. E.g. the formats need to be generated at install time with administrator privileges because otherwise all users need to be able to create formats, which does not combine well with the strict authorization setup on a Unix environment.

As software installations come, any full featured TEX back end installation is huge. TEXLive itself spans multiple CD's or a DVD. And even a teTEX download weighs in at over 100 MB these days.

2.1 teTEX

It is important to stress that there would not have been an i-Installer.app, a TEX i-Package or any TEX redistribution by myself had it not been for Thomas Esser's teTEX. Without his efforts as a foundation, none of this would have happened. Even today, while I make a lot of use of TEX Live, the main `texmf` tree is still the tree from teTEX, which Thomas maintains. Although I have produced the configuration scripts for the i-Packages (see below), these make heavy use of Thomas Esser's maintenance tools like `fmtutil`, `updmap` and `texconfig`.

2.2 First attempt: from teTEX sources

Initially, my wish to install TEX came from my personal wish to use TEX on the first developer release of Mac OS X on Apple hardware (Rhapsody).[6] So, my first attempt was based on downloading teTEX and compiling and installing that. The first attempt required some real porting, that is, some changes needed to be made to the teTEX code before it would compile and run on Mac OS X. My activities are generally based on the rule that I need to make my installs such that they will not take much time if I have to do them again. Every computer geek knows the frustration of a job that needs to be done once every few months or years, e.g. because one needs to remain up to date or because of a fresh system install. That is not frequent enough to remember what it was that needed to be done. My solution for this has generally been to document and publish, or to write a script. E.g. I will create some sort of

[6] Ironically, I have spent far more time on maintaining and redistributing TEX than on using it.

INSTALL document, a set of patches, and anything else needed to complete the install. Putting that on the net is a guarantee for myself that if I have to reinstall from scratch, I can relatively easily reapply my system changes. For this strategy to work reliably, one needs to have access to all the code in the same version that was used for the actual install.

This first attempt also uncovered a problem with line endings in the TEX code. Mac OS X is a system of mixed heritage: the classic Mac OS uses carriage return as end of line and Unix uses linefeed. TEX is supposed to be ignorant of whitespace, but porting to Mac OS X unearthed a line-ending problem. Other problems generally had to do with compilation only. The result was documentation and patches for the installation of teTEX.

2.3 Second attempt: a binary installer

But compiling TEX from source is not for most users. In the Unix world, there are many more tech-savvy users than not, but the Mac world — the original Mac OS being as closed as it was — is the opposite in the extreme; even the concept of a command line is alien. Instructing such users to compile TEX is more or less impossible, the language barrier is too high. Hence, as soon as the instruction-based distribution became a bit popular, the demand for a binary distribution grew. Given the fact that a TEX install cannot be simply drag-and-drop and that the instructions led to a lot of frustrated Mac users trying to understand the technical gibberish (and asking the author for help), creating an easy installer was mandatory, to keep the support load down.

The primary choice for such an installer, Apple's Installer.app, inherited a major flaw from its archiver pax which had the nasty habit of not honoring symbolic links and if an archive contained a directory 'foo' where the system already contained a symbolic link 'foo', the link was happily replaced by the directory inside the archive and the link was lost.[7] Archivers like (GNU) tar honour the link and follow it to install the contents of the directory in the archive.[8] This change from NeXTSTEP (which uses tar for its installer) has been unfortunate and was the reason for creating a separate installer for TEX.

The initial design for this installer was very simple. It would have one small window with only one button, titled 'Install'. It would probably have been a world record in terms of simplicity for a GUI app, but soon it became apparent that people wanted both install and uninstall, as well as both TEX and Ghostscript. Finally, given that teTEX uses A4 paper for its default setting, but many users require letter-size paper, the selection of paper size became a requirement for any install that does not require many users to use the command line. And finally, the users wanted some sort of idea of progress, the absence of which makes it hard to distinguish a working from a nonworking program. The result was a simple front end to a script that unarchived the archive, and set the paper size, the output of which was captured and displayed. The application was called TeXGSInstaller.app. A screen shot is displayed in figure 1.

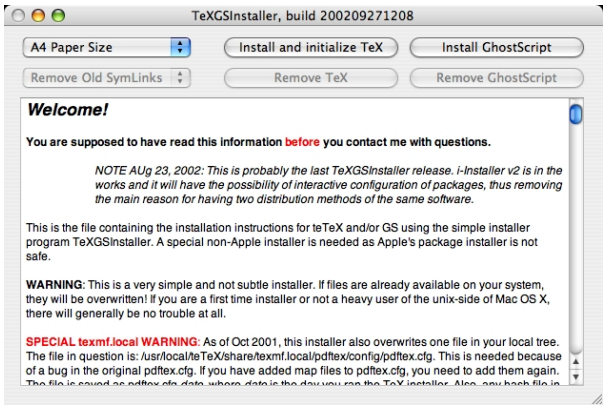

Figure 1: View of TeXGSInstaller application

2.4 Requests and solution

Though liked for its utter simplicity and — together with the TeXShop front end — part of the combo that won the Apple Design Award for "Best Use of Open Source", it soon turned out that more than just setting the paper size was required by many users. Another problem was that the combination of a complete TEX and Ghostscript was large, and had to be downloaded in full every time a small part was updated. In those days, the updates were rather frequent because pdfTEX was still in its initial rapid development. Besides, there were frequent requests for other additions to the redistribution. From TEX-related tools like TEX4ht or XMLTEX to all sorts of converters.

So, I decided to create a more generic installer: i-Installer. This installer was modelled (like Apple's) after the installer from NeXTSTEP, but with some differences:

- It is as transparent as possible. Unknown to most, Apple's Installer.app executes scripts be-

[7] Apple has since made this behavior controllable.

[8] At one time, this was the source for a major goof-up by Apple, where one update actually managed to damage the system it was installed on.

fore or after installation and these scripts may run with administrator privileges. These scripts are a potential source of security problems and I wanted users to be able to inspect them easily.

- It combines the function of downloading with installing, such that it minimizes the download to what is needed for the action selected by the user (hence the 'i' of i-Installer). Downloading is automatic, possibly even without user intervention, as downloading in itself is not a security problem.[9]
- Security is a priority.

Security is an aspect of software that is very difficult to get right. Automation implies moving actions from the conscious to the nonconscious realm. And given that a secure operation is best described as 'conscious risk taking' there is a true conflict of goals.[10] In terms of functionality, security and complexity, the following type of relation might hold:

$$\text{Security} = \frac{\text{Complexity}}{\text{Functionality}}$$

Although systems may combine functionality and security, the result will be complexity; that is, the combined complexity of what the system can do and what the user needs to do. There is no escape from this problem.[11]

2.5 i-Installer v2

The first attempt was expanded with basic functionality for interaction with the user and released as i-Installer v2 (version 2.2.0) in December 2002. A basic description follows.

Installation with i-Installer has at most four phases:

Selection This optional stage[12] allows the user to select which part of an i-Package needs to be installed. E.g. the TEX i-Package comes with various versions of the TEX programs (currently based on one of TL 2003, TL 2004 or TL 2005) and various versions of the texmf trees (based on teTEX 2 or teTEX 3). Also, one can choose to install documentation or not, etc. Selecting which parts to install of course influences what needs to be downloaded, hence, for people on slow Internet connections it pays to do a basic install instead of a full install.

Preparation Sometimes it is a good idea to do some preparation before unarchiving software in a certain location. E.g. if a texmf tree is rearranged it is wise to remove the old one before a new one is unarchived or one would get multiple copies of parts of the tree and it would be unpredictable which one would be used. This is why the TEX i-Package removes old stuff before unarchiving the new stuff. Of course, the removal of parts should react to the selections made in the previous phase.

Unarchiving This is nothing more than unarchiving the compressed tar archives in the correct locations.

Configuration Sometimes software must be configured. For TEX it means setting paper sizes, choosing hyphenation patterns (languages), formats, etc.

The configuration phase, if required, can be run separately.

Most of the phases (everything but unarchiving) may be interactive. Using an Apple technology called 'Distributed Objects', certain predefined settings (e.g. which parts of a package are to be installed) can be communicated between the main i-Installer GUI program and the subprocesses (generally Perl or shell scripts) that make up the phases. Environment variables can also be passed from phase to phase this way. Using this mechanism the install can be reasonably flexible. E.g. a choice for a simple install during selection will result in a simple configuration after unarchiving.

i-Packages have many more settings and options, like 'required' and 'recommended' dependencies, and many more possibly complex behaviours (like semi-automatic updates), which are beyond the scope of this article.

i-Installer has support for (linked) i-Directories (directories of i-Packages), dependencies, automatic background checking for updates to i-Packages and much more. Setting up a repository requires only a working web server. In fact, I run a personal repository in my personal `~/Sites` directory for testing purposes.

[9] Any action with a possible security risk will not be taken without user interventions, some of which are handled by the operating system, such as authentication for administrator access.

[10] This is the main reason why it has been so difficult for Microsoft to make their systems secure. For years they built an empire around automating indiscriminately. But while having an attachment of a mail message open automatically might save a mouse click, it also removes the consciousness from the action.

[11] One of the escapes is establishing trust as a simplification of the user's part in this. In a way, this shifts the burden from the user to the system. i-Installer has been designed to accomodate both so that users have a *choice* of trusting or not. This increases the complexity for the i-Package designer (there is no free lunch).

[12] Available since January 2004.

3 The TEX i-Package

3.1 Layout

The TEX i-Package has a default install location `/usr/local/teTeX`, reflecting the history of starting as a pure teTEX redistribution. This location can be changed in the i-Package's properties. As it follows the teTEX layout, the texmf trees are in a subdirectory called `share`. In that subdirectory four texmf trees are found:

texmf This tree contains whatever a build of TEX from sources produces in a texmf tree.

texmf.tetex This tree contains the basic texmf tree from teTEX, which is well balanced and a reasonable size. Generally, this tree is equivalent to the latest teTEX release.[13]

texmf.gwtex This tree contains additions to the teTEX base, such as additional fonts or macros. This tree differs depending on the install of either teTEX 2 or teTEX 3 in `texmf.tetex`.

texmf.local This is the standard 'local' tree of a TEX install.

The TEX i-Package was the first distribution to have the split between the program-related texmf tree (e.g. with the `.pool` files) and the basic foundation texmf tree (e.g. `texmf-dist` in TEX Live). The reason for this was that in the beginning of i-Installer there were two i-Packages for TEX: one with the programs and one with the foundation. As pdfTEX was rapidly developed, and improvements in pdfTEX were so instrumental for Mac OS X users, it was important to be able to update both separately and independently. These days, i-Packages may have many compressed `tar` archives and have logical sets of those which are presented to the user, so having different i-Packages for different parts is not necessary anymore.

3.2 Construction

Construction of an i-Package on my own systems is managed with `make`. The TEX i-Package contains 35 different compressed `tar` archives.[14] Sets of these archives comprise the 'sets' of the TEX i-Package.[15] This makes it possible to have logical entities, like 'gwTEX' which are built out of different parts and which do not need to be downloaded entirely if only something in one part has changed.

The 35 different compressed `tar` archives are created by other `make` commands, which create the basis from which the archives are built. E.g. the teTEX compressed archive is unpacked in a directory, the Latin Modern fonts are removed and the Latin Modern fonts from the TEX Live repository replace them. The result is placed in a directory `/usr/local/Build/teTeX3`. From this, the `doc`, `fonts`, `tex`, etc. subdirectories end up in compressed `tar` archives in the i-Package and the `fonts` and `tex` compressed archives end up in the `tetex3` set while the `doc` compressed archive ends up in the `tetex3-doc` set.

3.3 Contents

The contents of the TEX i-Package have not changed appreciably over the last years. It currently contains programs (binaries & scripts) based on a patched TEX Live 2003, a patched TEX Live 2004, and TEX Live 2005. TEX Live 2003 programs are combined with a texmf tree based on teTEX 2.0.2, the 2004 and 2005 versions with a texmf tree from teTEX 3.0. Both teTEX trees are updated/extended with a recent version of the Latin Modern fonts. Added to this is a set of macros and fonts in a separate gwTEX texmf tree, based on user requests. These are taken almost exclusively from the TEX Live master tree. Some of these are already available in teTEX 3, which means they are not installed when the 2004 or 2005 setup is chosen. As a result, choosing the 2003 setup with teTEX 2 may actually give you more recent TEX Live versions of certain additional macros.

A recent addition has been support for the immediate use of certain Apple fonts in pdfTEX. This support was created by Adam Lindsay and Thomas Schmitz and has been added to texmf.gwtex. Full use requires unpacking of the Apple fonts into a TEX texmf tree by a program like `fondu`; this is done automatically if `fondu` is available. (An i-Package for `fondu` is available.)

3.4 Configuration

Configuration handles choosing the paper size (for pdfTEX, dvips and dvipdfm), language patterns for LATEX, precompiled formats, font maps and adding TEX to the system-wide PATH settings for the major user command line shells. Since the OpenType versions of Latin Modern give problems in Mac OS X, the Type 1 versions are converted to TrueType by FontForge, if a recent enough FontForge is available. (An i-Package for FontForge is available.)

[13] Only as a last resort will I change anything here. The only current exception is updating the Latin Modern fonts in the teTEX 3 tree or adding those fonts to the teTEX 2 tree.

[14] Having the archive fragmented in many archives was added to i-Installer in September 2003. This meant that one could still have partial updates and less bandwidth requirements while having large combined i-Packages.

[15] Sets and the selector phase were added in January 2004.

4 Other TEX-related i-Packages

There are additional i-Packages for the CM-Super fonts, the CB-Greek fonts and MusixTEX. There is an i-Package for Ghostscript, for those who need to run TEX+dvips (e.g. users of pstricks).

Since ConTEXt is very actively maintained and developed, there is a special ConTEXt updater i-Package. This i-Package installs a ConTEXt (a choice of stable versus beta is offered if the beta is available) in texmf.local. This makes it possible to uninstall the ConTEXt update and revert to whatever is in texmf.tetex. The Pragma ADE website is checked nightly for any changes to ConTEXt; if any changes are found, the ConTEXt Updater i-Package is rebuilt and uploaded automatically. Since i-Installer can inform users via mail whenever an i-Package which they have installed has been updated, this means that ConTEXt users need spend little energy to remain up to date.

5 Looking back

5.1 i-Installer

Looking back, my TEX on Mac OS X project has seen four restarts (source, binary, i-Installer v1 and i-Installer v2). Since the fall of 2002, i-Installer has seen several fundamental additions, including fragmented archives, logical sets of archives, and a substantially improved user interface based on very critical user feedback I received.[16]

The code has withstood the additions so far in that the application is stable enough, but it is now at the point that important additions to the functionality become difficult to add. Luckily, not many more are needed, it seems. i-Installer is heavily based on functionality that comes with Mac OS X: the availability of a Unix layer and a slew of developer frameworks which require time to master but which also make it easy to program functionality.[17]

[16] The best criticism I received was in e-mail that had a flame-like quality, as in "this is the worst user interface I have ever encountered, you produce a worthless program" which after some discussion resulted in such a fundamental critique that it prompted a complete overhaul of the user interface. The program and the i-Packages have been completely user driven.

[17] Though Apple provides rich frameworks, some make life for the developer of an installer pretty difficult. The interface to the authentication mechanism is worse than archaic and inconsistent, for instance, and i-Installer's code in this area is complex as a result. Certain problems can be seen as directly following from Apple-provided functionality (e.g. the loss of the last bytes of the error output of subprocesses). Also, Apple has developed several improved interfaces to certain functionalities (like http traffic) over the years but backwards compatibility with older versions of Mac OS X prevents them from being used in i-Installer.

5.2 The TEX i-Package

The support for TEX on Mac OS X has completely run out of control. There are now 3 releases of the programs combined with 2 releases of the teTEX texmf tree available in the basic TEX i-Package.[18] Configuration is more than 10,000 lines of script code (mostly Perl and FontForge) and though much of the work is automated, keeping an eye on progress of building and uploading the i-Packages takes a considerable amount of time (even with a reasonably fast computer and a reasonable ADSL link). Requests for additions have dwindled to almost nothing, which gives me hope that the current teTEX + gwTEX set is close to ideal.

6 Looking forward

6.1 i-Installer

As of the current version of i-Installer, only one item is on my wish list: replacing the use of outdated Mac OS X frameworks by more recent ones, enabling, for instance, the use of authenticating proxies. Having learnt all that I have in these years provides me with a perspective with which I could design and build an even better installer, one which would support a fine-grained MiKTEX-like fragmentation of what is maintained on local disk (although this would possibly come in conflict with the very simple functionality now required of repository sites) and would be usable from the command line or via web services (authentication would be troublesome).

I hope that Apple will make i-Installer obsolete by producing a world class installer that I can use instead. In the meantime, i-Installer will remain pretty stable.

6.2 The TEX i-Package

For the TEX i-Package, keeping up to date with TEX and providing easy access to users to the different releases and changed functionality is difficult enough. No big changes to the current setup are planned.[19]

7 Availability and acknowledgements

The i-Installer and i-Packages are available from my TEX home page below. I would like to thank the TEX user groups and the TEX Development Fund.

◇ Gerben Wierda
http://www.rna.nl/tex.html

[18] The 2004 release will probably be removed when TL 2006 development starts.

[19] The TEX i-Package contains both PowerPC and Intel binaries so TEX runs natively on both Apple hardware architectures. The other support i-Packages, such as Ghostscript and ImageMagick, have been made Intel-ready.