

Open Source Desktop Technology Road Map

Jim Gettys, Version 2.0, October 23, 2008

Abstract

Navigating the myriad technologies that comprise the desktop (and palmtop) on open source systems is daunting to say the least, for newcomers of all sorts, open source developers, developers in companies using the technologies internally, and commercial ISV's, and even difficult to navigate for those immersed in open source systems on a day to day basis.

This document attempts to give a sketch of the names and relationships of these technologies and projects, and a glimpse into their status and development. Some technologies have never proved themselves, and/or have been rendered obsolete by later development and are available primarily for legacy code. This document attempts to clarify much of this natural evolution and market selection. Ultimately, some technologies become so rare as to enable their interment into the strata of software history, and it can be important to know which technologies are in such a fossil state, or stuck in the [Labrea Tar Pits](#) and possibly doomed to extinction, if not yet dead. A few may manage to struggle their way out of the tar to safety on dry land again.

Some indication of the licensing terms is made. For commercial software, make sure you understand the differences between licenses. For example, GPL and LGPL'ed libraries have very different consequences; one requires that source code of applications linked against them be made available, and the other does not require such disclosure. It is also possible for software to be available simultaneously under multiple licenses, sometimes allowing the implementer to choose which applies. See [the Open Source Initiative](#) for an explanation of these licenses.

Where known, approximate dates of expected completion are included, but there is no guarantees made. If you would like to ensure the timely completion of technologies under development, you should work with the community to determine if further resources are needed, and if so, to contribute the talent, resources and funding to do so.

Note that this document is still a bit weak in futures and I plan further work in this area. As in a map of a physical area, having information about current areas and how they interrelate was the first goal.

Acknowledgments

This document is the work primarily of its author, and the opinions here are my own; blame me for any errors and biases. Please let me know of any inaccuracies, and in particular, pointers to road maps of projects mentioned here. I would much prefer to have good pointers to similar project road maps than my current (mis) understanding of their time lines and development state, which is, of course, in a constant state of flux. Similarly, if you believe I have overlooked some key piece of open source desktop middleware technology (as opposed to end user applications which are too numerous to list), please let me know.

My thanks to Keith Packard, Jamey Sharp, Kevin Whitwell, Waldo Bastian, and Eric Raymond, Zenaan Harkness, David Alan Gilbert, Maarten Stolte, Maarten Stolte, Kurt Pfeifle, Brenda J. Butler, Zenaan Harkness, Eero Tamminen, Brian Gallaway Sergey V. Oudaltsov, John Smirl, and Vincent for constructive comments and feedback on Version 1 of this document.

Table of Contents

1. [Open Source Desktop Technology Road Map](#)
 1. [Abstract](#)
 2. [Acknowledgments](#)
 3. [Table of Contents](#)
 4. [Introduction](#)
 5. [Specifications](#)
 1. [ICCCM](#)
 2. [Freedesktop Specifications](#)
 6. [X Window System](#)
 1. [Key Protocol Extensions/Libraries](#)
 1. [Xlib - Basic X library](#)
 2. [3D Libraries](#)
 1. [Mesa - The 3D Graphics library](#)
 2. [DRI - Direct Rendering Infrastructure](#)
 3. [XInputExtension](#)
 4. [SHAPE](#)
 5. [XSYNC](#)
 6. [XVideo](#)
 7. [DOUBLEBUFFER](#)
 8. [The X Resize and Rotate Extension - RandR](#)
 9. [Security](#)
 10. [Record](#)
 11. [XTest](#)
 12. [The Render Extension](#)
 13. [Xft2 Library](#)
 14. [Xinerama](#)
 15. [Xnest](#)
 16. [X Extensions under Active Development](#)
 17. [Obsolete X Extensions](#)
 18. [X Libraries under Active Development](#)
 19. [X Toolkits](#)
 20. [GTK+ Toolkit](#)
 21. [Qt Toolkit](#)
 22. [Other Toolkits](#)
 23. [Moribund X Toolkits](#)
 1. [Motif](#)
 2. [TK](#)
 7. [Other Key Libraries](#)
 1. [Fontconfig - Font Configuration Library](#)
 1. [Freetype 2 - Font Rendering](#)
 2. [Cairo: A Vector Graphics Library](#)
 3. [HAL - Hardware Abstraction Layer](#)
 4. [DBUS - Message Bus system](#)
 5. [XML libraries](#)
 6. [Pkgconfig](#)
 7. [Zeroconf](#)
 8. [Multimedia](#)
 1. [Multimedia Frameworks](#)
 1. [Helix Community](#)
 2. [aRts](#)
 3. [Gstreamer](#)

4. [Mplayer](#)
 5. [VideoLAN](#)
 6. [Xine and Xinelib](#)
2. [Audio](#)
 1. [ALSA - Advance Linux Sound Architecture](#)
 2. [Audio Servers](#)
 1. [aRtsd](#)
 2. [ESD - Enlightened Sound Daemon](#)
 3. [Jack](#)
 4. [MAS](#)
9. [Microsoft Interoperability](#)
 1. [SAMBA](#)
 2. [File Systems](#)
 3. [File Formats](#)
 4. [WINE](#)
 5. [Winelib](#)
 6. [DOS emulation](#)
 7. [.Net and Mono](#)
 8. [Displaying Windows Applications on Open Source Systems](#)
 9. [X Implementations for Windows](#)
 1. [Cygwin and Cygwin/X](#)
 2. [Cygwin/X](#)
 3. [Commercial X implementations](#)
10. [Fonts](#)
11. [Printing](#)
 1. [Postscript and PDF](#)
 2. [CUPS Print Spooling System](#)
12. [Thin Clients](#)
 1. [LTSP - Linux Terminal Server Project](#)
 2. [Athena Computing Environment](#)
13. [Java](#)
14. [VNC](#)

Introduction

The most visible desktop projects are the [KDE](#) and [Gnome](#) desktop projects. These projects provide the basic toolkits, window managers, menu systems and control panels found in modern user interfaces along with many end user applications. It is important to note that the work of [freedesktop.org](#) is to ensure that applications and infrastructure can be shared between projects, and to enable this sharing in a way that end users do not know or care what environment these applications may be "native" to. In large part, this goal of freedesktop.org is being met, though there is more work to be done. [The Gnome project's roadmap](#) covers its next few releases.

Other major applications projects, which themselves may be platforms on which other applications are being built include the [Open Office project](#) (Sun's [StarOffice](#) suite is based on OpenOffice), providing a entirely free office suite, and their plans can be found [in their road map](#). Better integration with other applications on the desktop is high on that list; Open Office has used their own toolkit and needs better integration with Gnome and KDE.

[The Mozilla project](#) is also of special mention, who have built a world class free web application suite supporting all the widespread Web technologies (e.g. CSS, Javascript, etc.), including browser, mail client, bug tracking system, and other technology, used not only in their applications but also by other applications

in the open source desktop. [Mozilla's road map](#) covers both its recent history and current plans. Another implementation of web technologies underlies the KHTML Rendering engine of the KDE project and Apple in Mac OS X, and is now called webkit; it may be becoming a viable alternative to the Firefox gecko rendering engine. Firefox has the distinction of having seriously undermined Microsoft's control of the web; it's 20% market share (along with the additional marketshare of webkit, most notably in Mac OSX Safari) has wrested back control to web standards from their proprietary technologies.

Native plugins exist, often many, for most of the commonly used web datatypes (e.g. flash, RealPlayer, PDF). There are a few reasonably common datatypes for which there is no good native plugin available (fewer and fewer as the months go by). Windows plugins can often then be used via [WINE](#). One of the interesting problems is in fact, too many plugins for a given datatype. Better user interfaces to invocation of plugins have helped ameliorate this problem in current desktops, and Linux distributions have matured and reduced the number of options presented to a naive user to a reasonable defaults to help with this embarrassment of riches. A few datatypes remain difficult, but great strides have been made since V1 of this document.

The desktop applications themselves are far too numerous to begin to mention. A (large) subset of open source applications of all sorts numbering in the many thousands can be discovered on the [Freshmeat web site](#), in addition to the KDE and Gnome desktop projects. All of these projects build on the technologies covered in this road map (and sometimes additionally run on Windows and Mac OS X, most particularly the X Window System, but attempting to provide a road map to those projects is outside of the scope of this document.

Specifications

Historically, the X specifications were developed and ratified in the MIT X Consortium, and its successor organization, [X.org](#). X.org has morphed successfully from an industry consortium to an organization in which individuals, both at a personal level and as part of work they do for their companies have voice, working as part of the larger freedesktop.org and free standards community. Current X.org releases form the core of the free desktop.

As discussed below, the X Window System was designed to allow for extension, and many extensions as outlined above have been developed, deployed, and sometimes discarded over the years. Note that an API is just one binding to the specific protocol; there are and have been multiple such API's and implementations at times to the same underlying set of protocols.

Besides the API's and protocols mentioned below, there are a set of other protocols and (sometimes multiple) implementations of API's that are involved in the overall open source desktop. Most of these are primarily of interest to toolkit, window manager, and desktop environment programmers rather than directly to most application programmers. This section attempts to outline the most important of these, and their current status.

ICCCM

The original "Inter-Client Communications Conventions Manual" outlines the original set of conventions required of applications (mostly implemented in toolkits rather than directly by applications) to "play well" in the modular environment of the X architecture, allowing for interchangeable window managers, and other facilities. It was (mostly) sufficient to implement the CDE desktop, but insufficient for more modern environments. These (along with the EWMH (extended window manager hints) are built on top of the X11 core protocol using its general atom and property mechanism.

Freedesktop Specifications

Freedesktop.org was founded to foster the discussions between the Gnome and KDE desktop projects to extend the ICCCM in ways required for more modern environments. It now often hosts core desktop infrastructure projects (e.g. X, dbus, etc.).

Areas needing work to ensure further interoperability of applications build in one toolkit framework to be fully usable in others has included drag-and drop, window manager extensions, desktop entry files that describe information about applications, application embedding, UTF-8 support, bookmark exchange, menus, mime database, desktop settings, to name a few. Descriptions of the status of these specifications along with the specifications themselves [are available](#) and I recommend you look there for more information.

X Window System

The X Window System, Version 11, or X11, or most commonly called X, is the network transparent window system used on Linux, UNIX, and other platforms including Macintosh OS/X, and Microsoft Windows. It provides the basic infrastructure from which graphical user interfaces are built on Linux and UNIX. X11 was first released in 1988, and has an unrivaled reputation for stability; applications running on a MicroVAX of that era will interoperate against the latest X implementations across today's network, unchanged. This stability has been ensured by a careful, extensible protocol design framework, and attention to detail in the addition of new features.

I gave a [USENIX talk](#) on open source software development using the X Window System history that may be of interest.

New X extensions have been defined in recent years to bring X's original capabilities up to (and in some cases well beyond) the proprietary state of the art. Whenever possible, these programmer's API's have been built to allow even downwards compatibility to ease deployment of modern applications to older X server implementations. A good example of this is the Xft2 library, which, while performing best on X implementations where the X Render extension is present, will in fact provide high quality anti-aliased text on old X servers. In some areas X still needs work; much of this work is underway as described below and [in more detail elsewhere](#).

In the X environment GUI's are built using [Toolkit libraries](#), of which the most common at this date are Qt and GTK+. Motif based applications from the earlier generation of development on UNIX are now extremely rare (except as legacy applications inside and of corporate environments).

A component of an X Window System based environment not found as an independent component in other window systems is the external "window manager", which allows users to control the size, location and decoration of application's windows on the screen. They are, in fact, applications like any other application in X11, though you can generally only run one window manager at a time. Window managers are, for the most part, interchangeable components, and the standards defined originally by the X Consortium such as the ICCCM, and its successor X.org, along with [the new specifications](#) developed on freedesktop.org govern the protocols between applications and window managers. Window managers in common use today include KDE's window manager, Compiz Fusion or Metacity used by Gnome, and many, many others. Those that have been kept up to date with the freedesktop.org specifications are generally interchangeable and a matter of personal taste, though both major desktop projects have window managers they prefer, and which may integrate best into that environment. Some of these (e.g. Compiz Fusion) provide amazing visual eye-candy if requested, though mercifully, it's default behavior is now sane.

Other components, such as panels, start buttons, file managers, and many basic applications are provided by

the desktop systems. The largest and most well known of these projects are the [Gnome desktop project](#), the [KDE desktop project](#), and the [CDE desktop](#) previously used on UNIX systems. CDE is dead. A detailed road map of these projects is outside the scope of this document. The projects have a life of their own, and you are best consulting them as to their plans. They encompass many thousands of open source applications at this date.

There are multiple implementations of the X Window System which share code, both open source and provided by commercial vendors. The commonly deployed implementation on open source systems is currently provided by X.org which hosts its development here at [freedesktop.org](#). XFree86, while still existing, has become a relic of the past, triggered by its license change and general disgust with its policies.

There is much mythology of X's size; this is mostly an artifact of how memory usage is reported on systems (the entire frame buffer, off screen memory and any register space is reported against the X server's process's size, even if X itself is only consuming a megabyte or two of space itself). Similarly, some applications request X to save large amounts of pixels on their behalf, when other implementation approaches often easily avoid such memory usage. X is being successfully used on systems from [IBM's Linux watch](#) with 8 megabytes of compressed flash and 8 megabytes of RAM with a tiny 96x120 screen, to current PDA's like [HP's iPAQ](#), to [DMX based projector walls](#) containing tens of millions of pixels. With recent work, the minimal X footprint (X server and cut down Xlib) is currently just over 1 megabyte of code (uncompressed), excluding toolkits that are typically much larger, and could be cut smaller. After all, X11 was developed on VAX 11/750's that had less than one MIP with 2 megabytes of RAM.

Key Protocol Extensions/Libraries

These protocol extensions generally come with a C language library, that is often just a wrapper around the protocol, but sometimes includes additional functionality.

The Freedesktop.org X server and the XFree86 X server along with all base protocol libraries are MIT licensed, Commercial vendors of X technology may have additional restrictive licenses placed on their implementations as allowed by the MIT license .

Xlib - Basic X library

This library provides the basic protocol bindings and client side support for extensions, as well as a number of other facilities, some of which are useful, and some of which do not or have not seen serious use recently. The Motif toolkit uses more of these features than more modern toolkits such as GTK+ or Qt, which, for example, have found Xlib's facilities inadequate in a number of areas (e.g. internationalization).

As the basic library for the X protocol, its API is very stable.

Several sections of the Xlib API are seldom used in modern practice, either because the facilities never achieved widespread acceptance (as in the X Color management part of the API), or because modern toolkits have found that they needed more advanced facilities (as in the Locale section of the Xlib API), where the modern toolkits provide better facilities.

A [replacement for Xlib's protocol bindings](#) called Xcb can offer better performance by making it easier to avoid round trip messages to the window system in some areas. It has recently been deployed underneath the Xlib bindings to allow a migration strategy for applications that use plug-ins. The Xlib interface remains exactly API compatible with the old implementation. Work can now begin to take advantage of this in toolkits. Xcb was also carefully designed for thread safety, which was very difficult indeed in the old Xlib implementation.

[Some work was underway](#) to enable applications to be properly notified of connection failure with the X server (often seen when X is used over wireless networks, and sometimes over the wired internet) and allow for graceful shutdown. It was put on hold in favor of Xcb and would be nice to resurrect now that this work is maturing. This will enable the migration of running applications between X displays and movement of sessions. You would like to be able to go home and (securely) retrieve the applications running on your desktop at work. The GTK+ toolkit already has support for migration of applications, except for proper shutdown in the case of failure, and architecturally, this should be true for Qt as well. We hope that this will become usable during 2004, and widely deployed during 2005.

3D Libraries

3D is provided in the open source environment by industry standard OpenGL. Both closed source commercial and open source implementations of OpenGL are available. With the increasing industry cooperation in providing documentation and programming resources, the open source implementations are rapidly becoming truly competitive, and may reach parity or exceed proprietary implementations during 2009 and 2010. A few vendors (e.g. Nvidia) still do not provide documents or resources for supporting their hardware and should be avoided whenever possible.

Mesa - The 3D Graphics library

[Mesa](#) is an open source 3-D graphics library with an API which is very similar to that of [OpenGL](#). Mesa is used as the core of many hardware OpenGL drivers for XFree86 within the [DRI project](#). Software only implementations of Mesa are generally available even if hardware accelerated versions are not, but such implementations of Mesa will not be sufficient for more than very simple applications.

GLX is the binding of OpenGL to the X protocol, to allow for network transparent OpenGL applications.

Mesa has been a project for more than 10 years,, and various parts of it are available [under a number of open source licenses](#).

DRI - Direct Rendering Infrastructure

DRI is the direct rendering infrastructure for the X server for OpenGL direct rendering, and provides the device driver and coordination with the window system to allow 3D applications direct access to the display hardware. The DRI does not assume or require that the drivers be based on Mesa. Several non-Mesa, closed source drivers have used the DRI.

The DRI provides direct access to graphics hardware in a safe and efficient manner. It includes changes to the X server, to several client libraries, and to the kernel. The first major use for the DRI is to create fast [OpenGL](#) implementations.

It has been in use for a number of years, and is widely deployed with drivers for much of the common 3D hardware. DRI2 has started deployment.

MPX

Peter Hutterer has done amazing work called “MPX”, or Multi-Pointer X, enabling multiple input devices and multiple cursors in the X Window System. Part of this work has just been released in the X.org 7.5 release.

XInputExtension

XInput provides support for "non-core" input devices, such as trackballs, dial boxes, tablets, etc. It provides adequate facilities for these devices, but work continues to extend XInput to support "hot-plug" input devices. Addition of new input devices may still require manual configuration of the X server, but probably not by the end of 2009. Work is also needed to aid use of new facilities provided by the base operating system (e.g. /dev/input) in Linux.

This area needs some serious work. GTK+ application migration can already be demonstrated, and what is there is only adequate for session migration. Shared displays (e.g. a projector being simultaneously used by multiple users) bring to fore another issue: that of network transparent access to input devices. If I have an application I have migrated to a remote display, I may want my keyboard, mouse and other input devices to follow. While X applications like [x2x](#) help, this is really just a bandaid, and a more generic network input event mechanisms are needed, with good integration into the environment. You should be able to use devices anywhere in your environment. With hotplug a reality, building such a network environment is now possible and awaits eager hackers.

Xinput V2 and a revision of the X keyboard extension are planned for the next year.

SHAPE

The Shape extension provides non-rectangular windows in the X environment, and is universally deployed in X implementations.

XSYNC

The X Synchronization Extension provides facilities for applications to synchronize with each other, and with real time. XSYNC is widely deployed and in its current form, very stable.

Facilities to allow XSYNC to synchronize with video vertical retrace, audio sample clocks and other time bases are easy to add (and were the original intent of XSYNC), but while some work has been done to do this on Linux systems, it is not yet available in production X servers. It is also used by toolkits to synchronize repainting with the compositing manager.

XVideo

The XVideo extension provides support for video overlays, and similar facilities desired for video playback. It is commonly available on displays with hardware support for video playback. It provides facilities for the scaling and conversion of HSV data.

The better implementations no longer use chroma keying for display of video data, so that video can be composited as a first class citizen in the new world order.

DOUBLEBUFFER

The Doublebuffer extension provides doublebuffering facilities. It is widely available and supported.

The X Resize and Rotate Extension - RandR

Randr provides facilities for resizing, rotating, and reflecting the screen. It may also report root window size changes to (interested) clients.

RandR's support depends upon the state of driver support for RandR. The XFree86 server only has support for size changes in XFree86 4.3. The Kdrive server provides full support for the extension. It is relatively new and not universally available. Applications can usually be oblivious to its existence, though window managers and toolkits should consider support. Both Gnome/GTK+ and KDE/Qt have good support for RandR at this date.

Security

The Security extension was introduced to address the security issues of mixing trusted and untrusted clients on a single X server, according to compartmented mode workstation thinking of the early 1990's.

The old X security extension does not serve any useful purpose in most of today's environments, however, different facilities are needed, particularly for shared displays. Work done by Eamon Walsh of the NSA has provided X with a generic security framework into which different policies can be plugged (their plan is an SELinux like model; but others may be more usable for other environments); while this work is not quite complete, it is maturing rapidly. Our thanks for the efforts.

Record

The Record extension provides facilities for recording the operation of the X server, and is often used in concert with the XTest extension for window system and application testing.

It is stable and widely deployed.

XTest

The XTest Extension is primarily intended to allow for pretty thorough testing of the X server, by synthesizing input. As such, it is also used in some other situations when synthetic input to the window system is needed (e.g. stroke recognition).

There are two extensions by this name (cool, huh?).

The small XTEST extension was developed by Unisoft as a part of their Xlib test suite development to test pieces of the system which are otherwise unreachable. It is widely deployed and very stable.

The larger XTestExtension1 was developed by HP to help produce automated application testing mechanisms. This is not deployed and an architectural nightmare and can be considered extinct.

Everyone uses XTEST to synthesize input instead of XTestExtension1.

It is widely deployed and stable.

The Render Extension

The largest visible change in X is recent, with the addition of the X Render Extension, which provides Porter-Duff image compositing to the graphics capability of the X Window System. It is now widespread and used by most modern toolkits particularly for text, which is now rendered as alpha-composited, antialiased glyphs cached in the X server, completely supplanting the "core" text facilities of the original X11 protocol. The old "core" text facilities (bitmaps only) are still supported for backwards compatibility, but the Xft2 library and Cairo allows for reasonably painless updating of applications to support full AA text, even on X servers that do not support the Render extension.

The last facility of Render, not yet universally deployed, is support for AA trapezoids. Much device driver work remains to optimize the extension and use the facilities provided by modern graphics accelerators. Even so, the current usually software only implementation of Render, is fast enough for most applications.

Xft2 Library

[Xft2](#) provides a client-side font API for X applications. It uses [Fontconfig](#) to select fonts, FreeType 2 for rendering the fonts and the X protocol for drawing them on the screen. When available, Xft uses the Render extension to accelerate text drawing. When Render is not available, Xft uses the core protocol to draw client-side glyphs. This provides completely compatible support of client-side fonts for all X servers, enabling application developers the benefits of anti-aliased, subpixel decimated text on all X implementations. This backward compatibility is seldom needed now, but was essential for its success.

Xft1 is obsolete and should not be used; it did not provide AA text on old X servers, and combined this functionality with what is now Fontconfig, which has been made a separate library usable by all applications, not just X applications. Both Qt and GTK+ in versions since mid 2002 have used Xft2 for their text rendering.

Xft2 is stable, and now widely available and deployed on open source systems.

Xinerama

Xinerama allows multiple screens to be combined into one larger virtual X screen, mostly invisibly to most applications.

Its major liability is that merging screens of different types force the least common denominator behavior of the capabilities of the screens. In practice, this is not a serious limitation for casual use.

The new extensions under [active development](#) may allow for a more flexible Xinerama like system without its liabilities to be built.

Xinerama is widely available and deployed, though not all X drivers are implemented to allow it.

Xnest and Xephyr

These is not an X extensions, but an X server running in an window. This can be very useful not only for server debugging, but ensuring that your applications work well on different size X displays, for example the small screens found on PDA's.

Xnest is widely available and deployed, though some extensions may not be present. Xephyr is more likely to

be of use on today's systems and supports modern X extensions such as composite, and we recommend using Xephyr under most circumstances.

X Extensions under Active Development

These simple extensions together enable very sophisticated visual effects to be created by [window level image compositing](#), along with other needed facilities such as screen magnifiers, thumbnailers, and integration of applications into 3D immersive environments.

These are under active development and will change before completion.

- [The X Fixes extension](#) "fixes" some problems in the X core protocol, for application embedding, cursor tracking and naming, and introduces regions as a first class resource type for general use, for example, by the X Damage extension.
- [The X Damage Extension](#) allows applications to track modified regions of drawables, useful for compositing or applications such as [VNC](#).
- [The Composite extension](#) causes a entire sub-tree of the window hierarchy to be rendered to an off-screen buffer. Applications can then take the contents of that buffer and do whatever they like. The off-screen buffer can be automatically merged into the parent window or merged by external applications.

Obsolete X Extensions

A number of X extensions have been found to be useless or so difficult to use that they never achieved serious use over the last 15 years. These extensions, while possibly present in an X server implementation, should be considered obsolete and deprecated, and cannot be relied upon being present. Some of these include:

- PEX - the Phigs extension to X. OpenGL won the 3D war; PEX is no longer even built in current X servers.
- XIE - the X Image Extension was an attempt to turn the X server into general purpose image processing. It failed due to excessive complexity and lack of a good implementation. A simple image transport extension may still make sense in the future. The goal of XIE was "image transport and display" with the requirement that it accept arbitrary JPEG images and perform all necessary manipulation (including unsharp masking, color space conversion, gamma correction, affine transformation) in the X server. That slope wasn't just slippery, it was a grease-covered cliff ending in a complete programmable image processing system within the X server. The implementation suffered from being done under contract by the lowest bidder, and being done after the standard was ratified with no feedback loop possible to fix the worst of the problems. Bad spec, Bad process, Bad engineering, Bad standardization process. What's not to hate?
- Multibuffer extension - an early extension providing multibuffering. Included stereo, which the double buffer extension does not have. Some sort of separate stereo extension has been needed by several projects which have ended up using Multibuffer lacking a better mechanism. On the other hand, double buffer is now almost universally implemented by allocating off-screen pixmaps and using bitblt, so it may be all we need is a stereo extension.
- LBX, while often present in current implementations, is almost always a bad idea to use, and may vanish entirely soon. [SSH](#) typically works as well or better while providing the authentication and security that LBX lacks. See ["X Window System Network Performance"](#) for more information and

discussion.

- XPrint, which has seldom been seen to work, tries to turn the X server into a printing device. This is conceptually a bad idea, as, for example, to get document portability, the outlines of glyphs of fonts must be embedded into documents so that they scale properly across the resolution of different printers.

X Libraries under Active Development

Xlib has been the standard C binding for the X Window System protocol for many years now, and is in need of serious rewrite in some areas. It was written before shared libraries and threading were commonly available on UNIX, and several parts of Xlib should have been implemented as fully independent libraries. Work is essentially complete to engineer replacements for the old implementation and allow for better modularity and size.

The new basic protocol library is called XCB that has potentially very good performance properties in that it is much easier to use to avoid round trips to the X server, and is carefully written from scratch.

X Toolkits

In the X architecture, user interface elements used by applications are provided by "toolkits". There have been many X toolkits over the years (and on other systems, such as Microsoft Windows, similarly there are multiple toolkit API's). The most well known are [GTK+](#), [TrollTech's Qt](#), and the older [Motif](#) library. GTK+ is used as the basis of the [Gnome desktop project](#), Qt underlies the [KDE desktop project](#), and Motif is used in the [CDE desktop](#) commonly found on UNIX systems. New open source development generally uses either Qt or GTK+ for toolkits. Additionally, the Mozilla project has a cross platform toolkit used by its applications, though on Linux and UNIX, this work is based on top of the GTK+ library.

At this date, the major projects including [Gnome](#) and [KDE](#) have good facilities for internationalization (I18N) and localization (L10N) of applications, and many languages are already well supported. How comprehensive the localization is will depend upon the desktop and application, and the passion of the translators; complete coverage exists for many languages. Wider and wider localization is occurring, and is a way that motivated people from around the world can make a significant contribution, even for languages not spoken by many people. Even if you are a non-programmer but are a fluent speaker of a language not well covered, you can make a difference, and I encourage your involvement.

Modern UNIX and Linux systems provide facilities for shared libraries that allow multiple versions of a toolkit to be installed simultaneously without introducing problems with applications. For example, applications based on GTK+1.x can coexist comfortably with GTK+-2.x based applications. Both GTK+ and Qt allow customizing much of their look and feel via "themes". There are of themes that provide a largely similar look and feel across both toolkits. Themes can be changed while applications are running.

Also of note is [Winelib](#), which is a major aid in porting programs written to the Win32 API to X. A notable closed source, free as in beer application using these libraries is Google Earth.

GTK+ Toolkit

What is typically called GTK+ in common usage is actually a set of three libraries. Glib is a low level library used by GTK+ and Gnome to provide a portability layer and other common functions. Pango provides for very sophisticated internationalized text layout, for a very wide range of languages including Arabic and Indic languages, generally difficult for most applications to do on their own. ATK is an accessibility toolkit.

By supporting the ATK interfaces, an application or toolkit can be used with such tools as screen readers, magnifiers, and alternative input devices, and with the related applications and facilities, working toward full U.S. Section 508 accessibility conformance.

GTK+-2.x is now widely deployed, has Xft and XRandR support, and is actively maintained and developed. GTK+ is written in C, and there are bindings for GTK+ for just about all commonly (and some uncommonly) used programming languages. For more detail and future plans, see the [GTK+ web site](#). All applications in active use have been updated to GTK2 at this date, and it provides a stable, widely deployed base for new applications development. GTK+-2 is not expected to change in incompatible ways. It has language bindings for all common (and some uncommon) programming languages. While GTK+ development is primarily X based on UNIX and Linux on desktops and [PDA's](#) (e.g. , there has been work to target it at other platforms, including frame buffers and Windows. GTK+ runs on Mac OS X via that platform's support for X, now standard in the latest version of OS X.

GTK+-1.x is obsolete, and no longer being developed, having inadequate I18N facilities.

GTK+ is LGPL licenced.

Qt Toolkit

The [Qt library](#), actively developed and maintained by Trolltech, has Xft and XRandR support, and is the toolkit used by the [KDE desktop project](#). It is a modern toolkit primarily targeted towards C++ development, although bindings have been developed by the KDE project and others including support for Perl, Python, Ruby, JavaScript, Java, C#, Objective-C, and C.

A main feature of Qt is its strong cross-platform support (X11, Windows, Mac OS X, Linux frame buffer on [PDAs](#)). In addition to user interface APIs, Qt provides APIs for various platform-specific tasks such as file-handling, networking, process-handling, threading, database access, etc. Qt 3.2 also has strong support for internationalization including support for Arabic, Asian, Cyrillic, Greek, Hebrew, Indic, Latin and Persian languages as well as general Unicode support. Qt 3 supports accessibility features on Windows/Mac with the rudiments of support existing for X11. Full accessibility support under X11 will be available in Qt 4.

Qt 3 is the current version used on the desktop, although QtE 2 is still widespread in Qtopia PDA applications.

Qt is available [under the GPL and under a variety of licenses under different terms for commercial and non-commercial use](#).

Other Toolkits

Other toolkits seeing significant development that may be interesting for particular purposes include [FLTK](#), [GNUstep](#), and [wxWindows](#). Fltk is cross platform and quite small and fast, (hence its name), GNUstep is Objective C based and implements much of the original OpenStep specifications and may be useful in porting Apple Macintosh applications, and wxWindows is an open source, cross platform C++ GUI framework. On Linux, wxWindows is GTK+ based, and will be able integrate quite well into the current desktop (when its port to GTK+ 2 is complete).

Moribund X Toolkits

Motif

Motif (now called [OpenMotif](#)) is the toolkit used in the [CDE desktop](#) found on most UNIX systems, though KDE and Gnome are often also available for those UNIX systems. A clone of it, called [Lesstif](#), was developed while Motif remained fully proprietary. No new open source applications are now being developed by open source developers using Motif, though Motif may be used in corporations. As Motif currently lacks theming support, Motif applications often look out of place on current desktops, but will continue to work into the indefinite future, and is still seen used in some older commercial UNIX and Linux applications.

I believe that Motif/Lesstif are obsolete, but it is still used extensively in commercial settings, even if relatively little new code is written to it. It is based on a library called Xt, which attempted to define a policy free toolkit framework. In retrospect, this was a mistake. Other toolkits widget sets were built on Xt, including Xaw (the Athena Widgets), Xaw3d, and the like. I don't know if it has been ported to Windows or not (possibly via cygwin), but would still be running under X.

Lesstif is LGPL licensed. Motif has a [unique license](#), with different terms for commercial and non-commercial use on open source or proprietary platforms.

TK

[The TK toolkit](#) distributed as part of the cross platform TCL/TK system has some very substantial ongoing applications built with it, particularly some commercial CAD applications (e.g. Magic). It preserved some of the original ideas of Joel Bartlett's ezd's program canvas to have been adopted by other toolkits, including GTK+. I would not currently recommend it for new applications, as TK applications do not integrate very well into today's desktops (e.g. lack of theme support, and modern window manager hint support). Similar scripting language/toolkits that do integrate well are available, for example, in PyGTK (Python bindings to the GTK toolkit), to name one of several recent options. TK's most widely visible application has been the "make xconfig" program used for configuring Linux prior to Linux 2.6, and some git applications.

[There are some signs](#) that the Tcl/TK community understands that it has fallen behind the times and some steps are underway to rectify the issues, but these plans are not yet far enough along for me to believe TK will extricate itself from the tar. If you value TK, then now would be the time to aid that effort.

Other Key Libraries

Fontconfig - Font Configuration Library

[Fontconfig](#) will discover new fonts when installed automatically, removing a common source of configuration problems, perform font name substitution, so that appropriate alternative fonts can be selected if fonts are missing. Fontconfig will identify the set of fonts required to completely cover a set of languages. It will efficiently and quickly find the fonts you need among the set of fonts you have installed, even if you have installed thousands of fonts, while minimizing memory usage. Fontconfig can be used in concert with the X Render Extension and [FreeType](#) to implement high quality, anti-aliased and subpixel rendered text on a display. It is believed to be "best in class" of equivalent facilities on other operating systems.

Fontconfig does not render the fonts themselves (this is left to FreeType or other rendering mechanisms) or

depend on the X Window System in any fashion, so that printer only applications do not have such dependencies.

Fontconfig is stable, and available on all current Linux distributions. Note that the obsolete X core font mechanism does not (yet) use fontconfig for its configuration mechanism, which is a source of configuration confusion we hope to see remedied during 2004-2005.

Fontconfig is MIT licensed.

Freetype 2 - Font Rendering

FreeType2 is a software font engine that is designed to be small, efficient, highly customizable and portable while capable of producing high-quality output (glyph images). It can be used in graphics libraries, display servers, font conversion tools, text image generation tools, and many other products as well.

Note that FreeType 2 is a *font service* and doesn't provide APIs to perform higher-level features, like text layout or graphics processing (e.g. colored text rendering, "hollowing", etc..). However, it greatly simplifies these tasks by providing a simple, easy to use and uniform interface to access the content of font files.

Freetype's [features list is very long](#): in short, Freetype supports most types of fonts used to date.

Freetype 2 is stable, actively developed and maintained, and available on all current systems.

Freetype 1 is obsolete and should no longer be used.

FreeType 2 is released under two open-source licenses: its own BSD-like [FreeType License](#) and the [GPL](#).

Cairo: A Vector Graphics Library

[Cairo](#) is a cross platform vector graphics library with cross-device output support. Cairo is designed to produce identical output on all output media while taking advantage of display hardware acceleration when available (e.g. through the X Render Extension or the platform's native interfaces on Windows and Mac OSX). Cairo fills a missing piece of the unification of graphics between the window system and printing system.

Cairo provides a statefull user-level API with capabilities similar to the PDF 1.4 imaging model. Cairo provides operations including stroking and filling Bezier cubic splines, transforming and compositing translucent images, and antialiased text rendering. It is believed to be second to none in its intended purpose.

Currently supported output targets include the X Window System, in-memory image buffers and Postscript, PDF and SVG (structured vector graphics). Similarly as with Xft2, the X back end for Cairo is engineered to be able to draw on X servers that do not support the X Render extension, to enable applications to use its API everywhere. Bindings for many languages are already available, and Cairo is being used for applications. It underlies the GTK+ library, and is used by Firefox 3 on all platforms (Linux, Mac, Windows)..

Cairo is a free software library available under the MIT license.

HAL - Hardware Abstraction Layer

[HAL](#) is the infrastructure that deals with how to enable the good integration of devices into the desktop

environment, so that when you plug in a new device, it "just works" in sensible ways for users. Linux 2.6 [has extensive support for "hotplug" devices](#) of all forms, not just PCMCIA and USB, and this provides the desktop integration to go with the base operating system support.

DBUS - Message Bus system

[D-BUS](#) is a message bus system, a simple way for applications to talk to one another. There are bindings for multiple toolkits (e.g. Qt, GTK+). It is in common use and widespread deployment.

XML libraries

XML is increasingly used in free software, not only for document and information markup but other related uses such as configuration files (e.g. fontconfig). [Libxml2](#) is the XML C parser and toolkit; . It and a several other XML libraries (e.g. expat) are commonly used as the basis of XML related applications. LibXML2 is available under the MIT License, stable, very fast, and widely deployed in many applications.

Pkgconfig

Not directly part of desktop software, but being increasingly used as part of the build environment of desktop software is the [pkgconfig system](#).

Zeroconf

The [zeroconf work in the IETF](#) is key to good user experience in network configuration. First to support zeroconf well is Apple Rendezvous on the MacIntosh; [UNIX and Linux support](#) is well underway and beginning to ship in vendor's Linux distributions.

Multimedia

Multimedia is still in a great state of flux on open source systems. It is, however, to the point that serious applications exist and work well; the impetus is more around how to better integrate this to improve the user experience. As you will see, there is much available, and the community and users will decide which are of greatest importance, and it is probably the area of greatest ferment in desktop development.

Multimedia Frameworks

Helix Community

RealNetworks has made their [Helix community](#) multimedia framework that underlies their product suite and under [multiple licenses, depending upon commercial or non-commercial use](#). Not all codecs are available open source.

aRts

[ARts](#) is the multimedia framework used by the KDE project, is stable and is maintained by the KDE project, but has not seen much recent development.

Gstreamer

The multimedia framework used by the Gnome project is [Gstreamer](#). It is still working toward its first 1.0 release, but is in use in some applications.

It is LGPL licensed.

Mplayer

[Mplayer](#) is a movie player for LINUX and runs on many other UNIXs, and non-x86 CPUs and is being ported to Windows. It has support for a very wide variety of codecs and output devices. Multiple user interfaces have been built using Mplayer.

It is now very stable and very capable and at its 1.0 stable release.

Mplayer is available under the GPL.

VideoLAN

The [VideoLAN](#) project targets multimedia streaming of **MPEG-1**, **MPEG-2**, **MPEG-4** and **DivX** files, **DVDs**, digital satellite channels, digital terrestrial television channels and live videos on a high-bandwidth IPv4 or **IPv6** network in unicast or **multicast** under many OSes. VideoLAN also features a cross-plaform multimedia player, [VLC](#), which can be used to read the stream from the network or display video read locally on the computer under all GNU/Linux flavours, all BSD flavours, Windows, Mac OS X, BeOS, Solaris, QNX, Familiar Linux...

It is available under the GPL.

Xine and Xinelib

Xinelib is the library underlying the [Xine](#) multimedia player. It has been so successful that many user interfaces have been built on top, and has support for a huge variety of audio and video codecs natively and provisions for use of binary commercial codecs from Windows on x86 platforms.

It is very stable and very capable, and runs on Linux and most UNIX platforms and is being ported to other platforms as well.

Xine is available under the GPL.

Totem

Audio

Audio in open source systems is in a funny state. Amazing tools, applications, and codecs have been and continue to be developed, primarily on Linux systems, and you will find substantial parts of the computer music and audio research community using Linux. On the other hand, the tools and applications aren't all well plugged together for casual desktop use. So the good news is you'll probably find [more software](#) than you can make heads or tails of, and the bad news is you'll have to fuss with it to get it running, and may have

problems when different applications need to be used simultaneously. This is not a good situation, and needs serious improvement.

One issue for some audio applications has been latency, and particularly variance of latency. Low latency and good synchronization facilities are required by both professional audio work but even more for telephony and teleconferencing applications. Much work has gone on over the last several years to work on the Linux scheduler and get a good upper bound on latency in the system, in support of audio and other real time applications. Latency in Linux 2.4 is greatly improved over previous Linux releases, and is now quite good, and Linux 2.6 is outstanding.

Linux has transitioned from the old OSS interface to the new, much improved ALSA driver infrastructure and implementation.

ALSA - Advance Linux Sound Architecture

The Advanced Linux Sound Architecture ([ALSA](#)), provides audio and MIDI functionality to the Linux operating system. It boasts: efficient support for all types of audio interfaces, from consumer soundcards to professional multichannel audio interfaces, fully modularized sound drivers, SMP and thread-safe design, user space library (alsa-lib) to simplify application programming and provide higher level functionality, support for the older OSS API, providing binary compatibility for most OSS programs. This has put the Linux audio driver interface on a very firm footing.

Audio Servers

But X is a network transparent window system, and, particularly in LTSP environments, applications need network transparent access to audio services. As yet, open source systems have not settled on a standard sound server, which with luck will happen over 2004; several earlier sound servers such as AF or NAS are no longer widely used. The [AF audio server](#) was notable in that it shows that it is possible to design network transparent audio servers with no inherent latency and with tight synchronization facilities. As AF lacked sample rate conversion (machines weren't fast enough in 1992), it is inadequate on that and other grounds for today's use.

aRtsd

The aRts sound daemon used and maintained by the KDE project. It is also stable and not undergoing further development.

ESD - Enlightened Sound Daemon

ESD is widely available, maintained as part of the Gnome project, and is stable but is not undergoing further development. While adequate for routine use, it is very insufficient for advanced applications or those with demanding latency or synchronization requirements (e.g. video playback).

Jack

[JACK](#), the Jack Audio Connection Kit, provides low latency, high efficiency inter-process audio connectivity, as well as audio device sharing, suitable for pro-audio applications. While an audio server, it is not a network transparent server at this date, and therefore unsuitable for LTSP or other distributed X deployments. On single systems it has much to offer.

It is under active development, not yet API stable (though may become so during 2004), and available under the GPL (server) and LGPL licenses (libraries).

MAS

[MAS](#) - the Media Application Server, by Shiman and Associates, is a very ambitious audio server and multimedia framework. If it can be made to meet the needed latency (required by telephony and teleconference applications) and synchronization requirements (of video playback), it may be a viable contender to replace the current relatively poor audio servers.

MAS is under active development and evaluation.

The core of the MAS system recently became available under the MIT license.

Microsoft Interoperability

SAMBA

[Samba](#) enables UNIX and Linux systems to provide high performance file and print services to Microsoft Windows clients, and to gateway these services to open source desktops (via [CUPS](#)).

File Systems

Linux is able to read and write FAT file systems of all types, and read NTFS file systems. Read/write of NTFS file systems is in an experimental stage.

Linux systems can mount Microsoft SMB file shares and read and write them. The desktop file browsers for Gnome and KDE have improving support for enabling browsing of a Microsoft file share network and browsing of file shares.

Further polish of this work can be expected during 2004.

File Formats

The Microsoft file formats are a challenge, but open source programmers appear to be up to the challenge. The latest version of StarOffice/OpenOffice can import and export Microsoft office file formats (along with many others) most of the time with great success. Generally, open source applications are moving toward XML based representations of documents, such as that used by OpenOffice.

For perfect interoperability, many releases of Microsoft Office are now running very successfully using the [Wine](#) technology.

There will be continued incremental improvements; of course, since Microsoft is a moving target this will continue to challenge the community.

WINE

[Wine](#) provides a Microsoft Windows execution environment on Linux and x86 based UNIX. Many key "shrink wrap" applications written for Microsoft Windows including Microsoft Office, Lotus Notes, Adobe Photoshop, Intuit Quicken can run on x86 based Linux systems using Wine. Commercial versions of Wine that provide support and easy installation and configuration are available from [Codeweavers](#) for commercial applications and [Transgaming](#) for games. The applications in general are well integrated into the open source desktop, though in this environment, certain Windows peculiarities (e.g. MSDOS drive letters and file name paths) show through.

Your exact mileage will vary, but many of these applications (e.g. Microsoft Office) now run extremely well and stably. More and more applications will become usable with time.

Wine is distributed under the LGPL license.

WineLib

[WineLib](#) provides Microsoft Windows compatible API's as an aid to porting applications, and is the underpinnings of the Wine execution environment. As such, it can be viewed as yet another toolkit. Applications for which source is available may be able to be recompiled and relinked against WineLib as a way of porting the application to become a native application on UNIX and Linux systems.

While tracking the evolution of Microsoft API's is a perpetual, thankless task, a higher and higher fraction of applications will find that WineLib covers their needs with time. The fact that many major Microsoft Windows applications run well under Wine shows that the coverage is actually quite good over the API's actually used by Windows applications.

As a porting tool of applications built to Win32 API's, it can't be beat.

WineLib is distributed under the LGPL license.

DOS emulation

An easy, portable way to run old Microsoft DOS software is to use [Dosbox](#). It emulates enough of DOS to be able to run most of scripts and can automatically mount a directory as it were a DOS drive. The current 0.60 version can run e.g. most of the older DOS games, just extract program to a directory and use 'dosbox <directory>'.

The harder, x86 and Linux specific way is to use Dosemu and install a real MS-DOS on a disk image. [Dosemu](#) is faster than Dosbox. (The Dosemu page claims that MS-DOS is not needed, but in practice FreeDos is not compatible enough.)

.Net and Mono

[The Mono project](#) is an effort to create an open source implementation of the .NET Development Framework, some of which involve desktop technologies.

Mono includes: a compiler for the C# language, a runtime for the Common Language Infrastructure (also referred as the CLR) and a set of class libraries. The runtime can be embedded into your application. It

implements of both ADO.NET and ASP.NET. It is adopting [Cairo](#) for its 2D graphics.

Mono's licenses are GPL for tools and LGPL for Mono libraries.

Mono's expected 1.0 release is Q2 2004, according to its [road map](#).

Displaying Windows Applications on Open Source Systems

There are both [commercial](#) and [open source](#) RDP clients for Linux available, that enable remote Windows applications using Windows Terminal Server to be displayed on open source desktops.

X Implementations for Windows

There are X Window System implementations for Windows. X applications can run on Windows and use X servers on any operating system (UNIX, Linux, Mac OS X, etc.) for their display, and if an X server is running on Windows, X applications on UNIX, Linux, MacOS X can be used on those Windows machines. Tools like Cygwin make it very easy to develop cross platform tools that can be used on both Windows and UNIX/Linux/MacOSX.

There are both commercial and non-commercial X implementations for Windows.

Cygwin and Cygwin/X

[Cygwin](#) is a Linux-like environment for Windows, allowing for use of much Linux software on Windows. It consists of two parts: 1) A DLL (cygwin1.dll) which acts as a Linux emulation layer providing substantial Linux API functionality, 2) A collection of tools, which provide Linux look and feel. A large collection [packages](#) of software available.

The Cygwin DLL works with all non-beta, non "release candidate", ix86 versions of Windows since Windows 95, with the exception of Windows CE.

Cygwin is actively maintained and developed, and has been making regular releases,. Various packages are, of course under their own license terms.

Cygwin/X

Cygwin/X is a port of the [X Window System](#) to the Microsoft Windows family of operating systems. Cygwin/X runs on all recent consumer and business versions of Windows; as of 2002-05-12 those versions are specifically Windows 95, Windows 98, Windows Me, Windows NT 4.0, Windows 2000, and Windows XP.

Cygwin/X consists of an X Server, Xlib, and nearly all of the standard X clients, such as xterm, xhost, xdpinfo, xclock, and xeyes. Cygwin/X, as the name implies, uses the [Cygwin](#) project which provides a UNIX-like API to Xlib and X clients, thereby minimizing the amount of porting required.

Cygwin/X is licensed under an X style license; Cygwin is licensed under a modified GNU General Public License that specifically allows libcygwin1.a to be linked to programs that are licensed under an Open Source compliant license without such linking requiring that those Open Source programs be licensed under the GNU General Public License (see the [Cygwin licensing page](#) for more information). Source code and

binaries for both projects are freely available.

Cygwin/X is [actively maintained](#), developed and enhanced and has been making regular releases.

Commercial X implementations

There are a number of commercial X Window System implementations for Windows, for example, [Hummingbird's Exceed](#).

Fonts

Freetype 2, Fontconfig, and client side fonts have taken open source desktops from worst in class to best in class in basic font and internationalization at the core technology level, and at the toolkit level, GTK+'s Pango and recent work in Qt have finally brought X desktops from the 1980's to the modern world, with good Internationalization support, and first class linguistic layout and font rendering.

Essentially any TrueType or OpenType fonts are usable. Additionally, the Postscript 35 Type1 fonts are now also available universally, courtesy a donation from [URW](#) four years ago.

A further issue that most people are not familiar with is that of "metric compatibility". Unless fonts with very similar metrics are available, documents will not lay out identically across platforms. So without compatible fonts, typical documents in Microsoft format will not appear the same on open source platforms. This is a serious inhibitor to complete interoperability with Microsoft.

The Microsoft Web fonts are available on the Internet, but their license does not permit bundling for commercial sale, which means that users have to find and install them. This has meant that when open source systems are first turned on, and until and unless the users understand how to locate and install them or other fonts, we have had a major "out of the box" problem.

[Bitstream's](#) wonderful donation in April 2003 of the [Vera font family](#) for open source use has helped greatly in making open source systems presentable "out of the box". But the international coverage of these fonts is insufficient for the global nature of open source, and the fonts are not metric compatible with Microsoft fonts.

So, while the open source community has spawned decent graphics artists and GUI designers, to date we have failed to establish a serious culture of font design and hinting of fonts, so fonts, particularly high quality hinted fonts for screen use, remain in short supply that can be "preinstalled" so that things "just work" in routine use. The projects building fonts seem to be are pretty fragmented and uncoordinated at the moment. [Tools are available](#) for font design and editing on Linux (and commercial tools are available on other platforms), but hinting of fonts for screen use is just slow, painful, repetitive work. Work by [John Hobby](#) a decade ago shows it may be possible to do good automatic hinting of fonts now that systems are so fast, but implementations of those algorithms are not currently available (another potential project for mathematically inclined open source programmers). Fonts require ongoing maintenance and enhancement, as language coverage and/or entirely new characters need to be added to fonts. In principle, this effort should be a lot of fun for the right person/people, involving as it does so many languages and cultures over the world. Further donations of fonts and/or funding for further font development would be greatly appreciated along with someone with the passion and ability to lead such efforts.

Availability of high quality fonts, particularly for screen use, is therefore a significant issue. A second issue has been that Apple asserts two patents on how certain hints are implemented in TrueType; these patents are only valid in the U.S. and Great Britain.

As a result, commercial Linux or application distributions for desktop use now often bundle other high quality proprietary fonts with them as well, so this is less an inhibitor than one might first expect. But as this means that applications have not been able to rely on the availability of a metric compatible set of fonts, we may end up paying multiple times for the same fonts (from both application and base Linux system vendors). Better would be for the commercial open source vendor community to pool their efforts and "buy out" a set of such fonts.

Printing

Printing has been a difficult problem on UNIX and Linux systems, but a number of changes during 2002 and 2003 are turning this from poor to quite usable. It is a very large topic, requires a document of this size to sketch, so this is just a 20,000 foot overview. Much additional information about printing on open source systems can be found on the [Linux Printing web site](#). Open source printer drivers for almost all HP and Epson printers are available, with more spotty coverage for other vendors, and commercial support and more drivers available and tools from [Easy Software Products](#), CUP's creator.

UNIX/Linux printing in the past been badly hobbled by the lack of a good spooling system, which is now on its way to solution with the increasing deployment of CUPS and other developments. Berkeley lpr/lpd was really lame, System V lp similarly lame. Lprng was an improvement, but at this date, all Linux distributions, at least, are using CUPS as their default spooling system

Desktop applications were badly hobbled by the original X Window System font design (for which I am partially to blame), which denied easy access to font files needed by applications which did anything sophisticated with text. Keith Packard's font related work has been outstanding, and the Fontconfig/Xft2/Freetype/Render combination takes us from poorest to best in class, both in quality of rendering and in internationalization on the screen, but some more work is needed in applications for printing to catch up. The widespread deployment of client side fonts using Fontconfig, Xft2, Freetype the X Render extension during 2003 marks a sea change in enabling good desktop display and printer integration.

Additional work to complete integration of printer selection and management of small to moderate CUPS installations into the desktop environments is well underway. KDE's support has been complete for a while and Gnome projects and will be substantially complete in 2003, with StarOffice and Mozilla the remaining major application suites needing update.

The expected uptake of [Cairo](#) in these projects during 2004 should significantly improve the generation of high quality 2D graphics for screen and printing, and allow for embedding of font information into print files that has been lacking (with the notable exception of StarOffice/OpenOffice, which has internal PDF generation already).

Postscript and PDF

The fundamental "Lingua Franca" of printing on open source and UNIX systems has always been postscript, and there are open source and closed source PDF utilities.

CUPS Print Spooling System

[CUPS](#) is now "standard" on all major currently shipping Linux distributions, Mac OS X, and available for all UNIX systems. It provides an [IETF IPP \(Internet Printing Protocol\) print spooling system](#) with provisions for fail-over, printing of many MIME types (though PDF and Postscript are by far the most common).

Microsoft Windows XP has IPP support, but as it involves running IIS currently not typically used.

CUPS makes all printers postscript capable that are not already Postscript capable, by using either [Aladdin or the free versions of ghostscript](#), as a "Raster Image Processor".

CUPS uses industry standard PPD files for printer descriptions, for all printers, not just Postscript printers, providing a uniform printer capabilities mechanism. CUPS provides for dynamic discovery of printers, and fail-over between printers.

One of CUPS's other major strengths is that it has back ends for Microsoft SMB printers via [SAMBA](#), Berkeley LPD protocols, JetDirect from HP, and so on, providing a fully uniform access method for applications to use for printing. Additionally SAMBA can be configured so that CUPS printers are available to Microsoft Windows desktop users.

CUPS will continue to spread rapidly into the remaining environments. More work is needed in the management of CUPS environments, but the basic needs are now met in the latest versions of KDE and Gnome applications and those provided by Linux distributions. Commercial large enterprise management software is available from Easy Software. Integration of CUPS into desktop applications is well underway, and should be substantially complete by the end of 2003, and entirely complete by mid 2004.

CUPS is GPL (the CUPS server itself) and LGPL licensed (the CUPS libraries).

Thin Clients

The X Window System, arguably, invented thin client computing. In their heyday, there were a significant number of hardware companies offering what were called "X Terminals" (e.g. NCD). Most/all of these are defunct, as they became less economic than commodity PC hardware, but the concept still has serious validity to reduce system management costs. Some thin client hardware for sale today can be used with X and X based applications as well, particularly since they make machines interchangeable, have lower CPU and memory requirements, and can easily eliminate the disk and fans of conventional desktop machines at a somewhat lower price point. The combination can result in a much lower TCO, particularly using Linux, since Microsoft's pricing of its software licenses actively discriminates against this computing model.

Also note the availability of RDP clients for open source systems as mentioned above, which allow Windows applications to display on X desktops via Windows Terminal Server.

LTSP - Linux Terminal Server Project

[The Linux Terminal Server Project](#) is building "thin client" versions of Linux. Most applications are run remotely on servers (via X's network transparency), but there are also provisions for locally run clients. This provides much better manageability of desktops and is very cost effective in many environments, making machines truly interchangeable and obviating the need for "sneaker-net" system management. One view is that this is a reinvention of X terminals, but as there is provision for local clients it goes beyond that, and can take advantage of commodity PC hardware, thin client hardware, and old systems you thought were just junk. The [K12LTSP](#) makes a version specifically targeted toward use in schools.

LTSP is in widespread use, has been through four major releases, and has an active developer and user community, and supports multiple Linux distributions.

Athena Computing Environment

A similar style of not-quite-so thin computing is typified by the Athena computing environment at MIT (where both the X Window System and Kerberos have their roots). In this model, machines are also interchangeable, but rely on distributed file systems and good authentication to keep any permanent data off the local machine. Local disks are used as caches for files and for swap, but never for long term data storage. As an integrated whole, this is not seen much outside of MIT, though X11 and Kerberos are reasonably widespread. As a interesting historical note, Athena's "Zephyr" system was arguably the first instant message system.

Java

[Java is available from Sun](#) for Linux, and there are several static Java compilers (e.g. [GCI](#), part of the GCC compiler suite and [Jikes from IBM](#)), and may be preinstalled. The [Blackdown](#) project provides community source distributions of Sun's Java for additional platforms that Sun may not. Java release 1.4.2 introduces Swing support based on GTK2 look and feel, which aids in the natural integration of GUI applications built with Java on the open source desktop. As this deploys widely over 2004, Java applications using Sun's VM will share the look and feel of Gnome desktops.

VNC

[VNC](#) stands for Virtual Network Computing. It is remote control software which allows you to view and interact with one computer (the "server") using a simple program (the "viewer") on another computer anywhere on the Internet. The two computers don't even have to be the same type, so for example you can use VNC to view an office Linux machine on your Windows PC at home. VNC is freely and publicly available and is in widespread active use by millions throughout industry, academia and privately.

Note that VNC can be trivially replaced with a simple X application along with "ssh -X -C" in concert with the Damage extension.

[*Jim Gettys*](#)

