Mozilla Optimization Mini−HOWTO

Salvador J. Peralta

<speralta at willamette dot edu>

January 2003

Revision History

Revision 1.0	2003−01−03	Revised by: SJP
Initial release, reviewed by LDP		

This document discusses how to make configuration and source level customizations of Mozilla to make it more suitable as a primary browser for Linux and X Windows. It is not intended as a guide for programming Mozilla, nor is it a guide to XUL.

The techniques described in this document have been implemented and tested in a kiosk−like public computing environment using thin client workstations running on Redhat Linux, utilizing XFree86.

Table of Contents

1. Copyright

2. Introduction

Mozilla is an Open Source, cross−platform, browser and graphical application environment. It is rapidly becoming a significant component on many Linux−based desktops due to its stability, robust feature−set, large developer−base, and ease of configuration. This Mini HOWTO addresses how to patch and configure Mozilla's source to better optimize it for Linux desktop environments, and is intended to serve as a general set of instructions for the Linux community on how to compile and configure Mozilla.

The most current version of this Mini HOWTO can be obtained in HTML format from www.willamette.edu/~speralta/tldp/mozilla.

3. Get the Source

There are pros and cons to building Mozilla yourself. The biggest drawback is that the source version, compressed, is a 30+ MB file in most releases which is about three times larger than the binary distribution. A second negative is that on a reasonably new machine such as my 1.6 ghz Pentium 4 with 256 MB of memory, compiling Mozilla can take anywhere between one and a half to four hours depending on any other jobs the system might be running. On an older machine, Mozilla can take twenty−four hours or more to compile. The biggest benefit is that there are a large number of compile−time options and patches that can be applied which will dramatically improve the performance and featureset of the browser. These patches are generally necessary if you intend to run Mozilla in a production environment.

If you are not interested in source level modifications of the Mozilla browser, please skip the next few sections and have a look at Section 7, Post−Install Configuartion.

At the time this document was written, the latest version of the Mozilla source code is 1.2.1, which can be obtained via ftp at ftp.mozilla.org/pub/mozilla/releases/mozilla1.2.1. The other stable release, 1.0.1, can be obtained via ftp at ftp.mozilla.org/pub/mozilla/releases/mozilla1.0.1.

For CVS checkout, please review the documentation on the Mozilla website.

4. Patch the Source

Unpatched Mozilla is not a very good X citizen. For example, it does not allow one to specify X and Y geometry position as a command−line option. This is not a big problem on a typical end−user desktop, but in a corporate or public computing environment, it is a killer. In order to fix this, we need to apply a patch to the C++ source code at `embedding/components/windowwatcher/src/nsWindowWatcher.cpp`. The first step is to obtain and apply the patch written by Robert Riches. This patch has been tested in Mozilla versions 1.0 through 1.2.1.

To apply the patch, simply paste it into `embedding/components/windowwatcher/src/nsWindowWatcher.cpp`, in your Mozilla source tree. The notation uses "+" symbols to denote the code that needs insertion. Those symbols need to be removed before saving the `nsWindowWatcher.cpp` text file.

To use the patch, set an environment variable called `MOZILLA_SCREEN_POS` with the proper coordinates. For example, in Bash, type **export MOZILLA_SCREEN_POS='screenx=1,screeny=1'** will set the top left corner of your browser to the top left corner of your screen.

---

5. Configure the Source

Another problem with using Mozilla in a personal or production environment is that it contains code which will override any home page that you set on a timed basis, or whenever a new release of the product is distributed.

Since this information is stored in a configuration file, the easiest thing to do without breaking the browser is to change the configuration file to point default homepage settings to one that is more consistent with the environment that you are using.

Mozilla conveniently stores most of its compile–time configuration files with a *.properties* extension. You can use these files to specify text in message dialogues, fonts, and other options. In this case, assuming that your locale and language settings are `en-US`, the *.property* file that you will need is `region.properties` which is located in the `xpfe/browser/resources/locale/en-US/` directory.

The changes that you need to make to this file are fairly straight forward. As a general rule, I replace the value portion of the name/value pair to one of my own choosing in every instance where the word "homepage" is mentioned in the configuration.

```
# navigator.properties
homePageDefault=http://yoururl.com
shopKeyword=keyword:shop [Product]
quoteKeyword=keyword:quote [Enter symbol here]
localKeyword=keyword:zip [Your zip code]
keywordList=http://home.netscape.com/escapes/keywords
webmailKeyword=http://webmail.netscape.com
careerKeyword=keyword:[Your city] careers
fallbackDefaultSearchURL=http://search.netscape.com/cgi-bin/search?charset=UTF-8&search=
otherSearchURL=http://home.netscape.com/bookmark/6_0/tsearch.html
#
# all.js
#
browser.startup.homepage=http://yoururl.com
browser.throbber.url=http://yoururl.com
browser.search.defaulturl=http://search.netscape.com/cgi-bin/search?search=

wallet.Server=http://www.mozilla.org/wallet/tables/
wallet.Samples=http://www.mozilla.org/wallet/samples/

#config.js
#
startup.homepage_override_url=http://yoururl.com
```

6. Compile the Source

To configure Mozilla's compile−time options, type **./configure** from the root directory of the Mozilla source tree. To get a full list of compile−time options, type **./configure −−help**. Some things to verify before doing a production compilation include verifying that environment preferences (mail vs. no mail, calendar, ldap, etc.) are set and making sure that the crypto package is enabled.

For simplicity's sake, I generally use a simple build script like the following to configure Mozilla.

```
#!/bin/sh

MOZILLA_OFFICIAL=1
export MOZILLA_OFFICIAL
BUILD_OFFICIAL=1
export BUILD_OFFICIAL

./configure --with-x --disable-calendar --enable-crypto --with-system-nspr
--disable-debug --enable-extensions --enable-optimize
--without-system-zlib --without-system-jpeg --without-system-png
--without-system-mng
```

Once you have your patches applied, config files modified, and options set, simply build Mozilla with *gmake* by typing **./gmake** in the root directory of your source tree, and then create your tarball by typing **gmake** in the `xpinstall/packager` directory. This will drop the tarball in the dist subdirectory just beneath the root directory of your Mozilla source tree. Just move that package to `/usr/local`, unpack it, run it once from the command−line as the user who built the package (the command is **/usr/local/mozilla/mozilla**), and you are nearly ready to browse.

7. Post−Install Configuration

There are several configuration choices that you can make after you have installed your browser. You can add plugins for common web technologies such as Java (java.sun.com), PDF printing (www.adobe.com), and Flash animations (www.macromedia.com/software/flash) and slightly less well−known, but very cool plug−ins, such as Enigmail (enigmail.mozdev.org).

To install any plug−in, either copy the plug−in from the source application (e.g. Acrobat5) to the `/plugins` directory in Mozilla, or create a symbolic link from the plug−in to the plug−ins directory in the Mozilla binary tree. For example, to create a Mozilla plug−in for Adobe Acrobat 5 using a symbolic link, simply type **ln −s /path/to/Acrobat5/Browsers/intellinux/nppdf.so /usr/local/mozilla/plugins/**. Similarly, to create a plug−in from your Java runtime environment, type **ln −s /path/to/j2re1.4.0_02/plugin/i386/ns610/libjavaplugin_oji.so /usr/local/mozilla/plugins/**. Please note, it is recommended that you use Sun's JDK 1.4.1 with Mozilla 1.0.1 and above. There are known incompatibilities with Mozilla and earlier versions of Java.

To obtain Mozilla project plug−ins such as Enigmail, which provides an interface for encrypting and decrytping mail, or for Protozilla, which provides full parameterization (including support for multiple multiple protocol and programming languages), visit mozdev.org.