目录

第 1 章. 介绍

目录

1.1. Wine 是什么?

编写： John R. Sheets < jsheets@codeweavers.com >;

1.1.1. Windows 和 Linux

许多人面临着拥有的软件不能在他的计算机上运行的困扰。随着 Linux 近来的流行，因为操作系统的不同而发生的更加频繁。你的 Windows 软件不能在 Linux 上运行，而你的 Linux 软件不能在 Windows 上运行。

对这个问题的一个常见的解决方法是在一个计算机上同时安装这两个操作系统，成为一个 "双引导" 系统。如果你想在 MS Word 中写一个文档，你可以启动 Windows；如果你想运行 GnuCash，一个 GNOME 财务应用程序，你可以关闭你 Windows 任务 (session)并重启动到 Linux。问题是你不能同时使用它们。每次在 MS Word 和 GnuCash 之间前后切换，你必须再次重启动。很快就会厌倦的。

如果你可以在同一个系统上运行所有你的程序，而不管它们是为 Windows 还是 Linux 写的，生活就容易多了。在 Windows 上，这实际上不可能。[1] 但是，Wine 使在 Linux (或 Solaris)上同时运行本地 Windows 应用程序和本地 Linux 应用程序成为可能。你可以在 MS Word 和 GnuCash 之间共享桌面空间，交叠窗口，图标化它们，甚至从同一个启动项(launcher)运行它们。

1.1.2. 模拟与本地连接

Wine 是 win32 库的一个 UNIX 实现，由上百个志愿开发者从头 (from scratch)写成并在一个开放源代码许可之下发行。任何人都可以下载并阅读源代码，并修理存在的缺陷。Wine 社区众多有才干的程序员在增进 Wine 上花费了上千个小时的个人时间，所以它可以很好的与 win32 应用编程接口 (API)一起工作，并保持与 Microsoft 相同的开发步调。

Wine 可以用两种不同的方式运行应用程序：作为预编译的 Windows 二进制程序，或作为本地编译的 X11 (X Window 系统) 应用程序。前面的方法使用模拟把一个 Windows 应用程序和连接到 Wine 库上。通过 Wine 安装或简单的把 Windows 可执行文件复制到你的 Linux 系统上，你可以用模拟器直接运行你的 Windows 应用程序。

用 Wine 运行 Windows 应用程序的另一种方法需要拥有这个应用的源代码。不用本地 Windows 编译器编译它，象对其他 UNIX 应用程序所做的那样，你要用本地 Linux 编译器编译它 -- 例如 gcc -- 并连接上 Wine 库。这些本地连接的应用程序被称为 Winelib 应用程序。

Wine 用户指南将关注于使用 Wine 模拟器运行预编译的 Windows 应用程序。Winelib 用户指南将覆盖 Winelib 应用程序。

注释

[1]从技术的角度上，如果你有两个连网的计算机，一个运行 Windows 而另一个运行 Linux，并且你在这个 Windows 系统上运行了某种 X 服务器软件，你可以把 Linux 应用程序导出到 Windows 系统上。不幸的是，多数正规的 win32 X 服务器是商业产品，它们通常都很贵。如果你只有一个计算机那么这种方案就解决不了问题。

1.2. Wine 要求和特征

编写：Andreas Mohr < am ohr@ codeweavers.com >；

1.2.1. 系统要求

要运行 Wine 需要满足下列条件：

一台计算机 Wine：目前只支持 PCs >= i386。Winelib：可支持其他平台，但有点困难。

一个 UNIX 类的操作系统如 Linux、*BSD、Solaris x86。

>= 16MB 的 RAM。在此之下所有东西都是不可用的。"良好"的执行需要 >= 64MB。

一个 X11 window 系统(XFree86 等)。Wine 为其他图形显示驱动器做了准备，但写支持不是很容易的。文本控制台显示驱动器基本上是可用的。

1.2.2. Wine 的功能

希望你已经设法完全满足了上述要求。现在我们告诉你 Wine 能做/支持的：

支持执行 DOS、Win 3.x 和 Win9x/NT/Win2000 程序(支持多数 Win32 的控件)

选择使用外部厂商的 DLL(比如，原始的 Windows DLL)

基于 X11 的图形显示器(到任何可能的 X 终端的远程显示器)，文本模式控制台

Desktop-in-a-box 或可混合的窗口

对游戏的非常高级的 DirectX 支持

对声音的良好支持，可替换(alternative)的输入设备

打印机：支持固有 Win16 打印机驱动程序，内部 PostScript 驱动程序

支持调制解调器、串行设备

Winsock TCP/IP 连网

ASPI 接口 (SCSI) 支持扫描仪、CD 刻录机 ...

Unicode 支持，相对高级的语言支持

Wine 调试器和可配置的跟踪日志消息

第 2 章. 获得 Wine

目录

2.1. Wine 的多种形式

标准 Wine 发布包括许多不同的可执行、库、和配置文件。要使 Wine 良好的工作必须正确的设置所有这些文件。本章将指导你通过必要的步骤把 Wine 安装到你的系统上。

如果你运行的 Linux 的发行使用包来跟踪安装的软件，你可能走运了:Wine 的一个预编译的版本可能已经存在于你的系统中了。前三节将告诉你如何找到最新的 Wine 包并安装它们。你应该小心在不同的发布之间混用包，即使是同一个发布的不同版本之间。经常是一个包只能在它所编译的发布上运行。我们将覆盖 Debian、Redhat 和其他发布。

如果你不够幸运的获得给你的操作系统的一个包，或者你偏好一个新版本的 Wine，它比现存的包要新，你必须下载 Wine 源代码并在你自己的机器上亲自编译它。不要担心，这不是很难，特别是与 Wine 一起的还有许多有用的工具。你无须有任何编程经验来编译和安装 Wine，但有一些 UNIX 管理经验就更好了。我们将覆盖如何从 FTP 归档取回并编译官方源代码，还有如何从 CVS (并发版本系统)获得最新 (cutting edge up-to-the-minute fresh)的 Wine 源代码。二者的源代码安装的过程是类似的，并且你一旦掌握了其中一个，你处理另一个应当没有任何问题。

最后，你可能有一天需要知道如何向你的 Wine 版本提供补丁。可能你找到了 Wine 中的一个未发现的缺陷，请向 Wine 邮件列表报告，并从某个开发者接受一个补丁来修理这个缺陷。本章的最后一段将告诉你如何安全的提供补丁以及如果补丁不工作如何复原。[/color]

2.2. 为 Debian 系统获得 Wine

在 Debian 系统的多数情况下，你可以用一个简单的命令安装 Wine，作为 root:

```
# apt-get install wine
```

apt-get 将通过 Internet 连接到一个 Debian 归档(所以，你必须在线)，接着下载 Wine 包并安装到你的系统上。完事了。

当然，Debian 的 Wine 预打包的版本可能不是最新的发行。如果你运行 Debian 的稳定版本，你可以通过从不稳定发布获取包来得到一个稍微新点的 Wine 版本，但这可能有点冒险，依赖于不稳定发布从稳定发布分叉多远。你可以在 www.debian.org 使用包查找引擎找到给各种 Debian 发行的 Wine 二进制包的一个列表。

如果要安装的一个包不是你的发布的一部分，你需要使用 dpkg 而不是 apt-get。因为 dpkg 不为你下载文件，你必须自己下载。依从在包查找引擎上的链接找到所需的包，接着点击 Go To Download 页面按钮并依从指示。保存文件到你的硬盘，接着在其上运行 dpkg。例如，如果你把文件保存到你的主目录中，你可以进行下列动作来安装它:

```
$ su -
<键入 root 口令>;
# cd /home/user
# dpkg -iwine_0.0.20000109-3.deb
```

你可能还需要安装 wine-doc 包，如果你使用 Wine 的 2.3 发布(Woody)，则还需要 wine-utils 包。[/color]

2.3. 为 Redhat 系统 获得 Wine

Redhat/RPM 用户可以使用 rpm find.net 来搜寻可获得的 Wine RPM 二进制包。 这个页面包含以字母"W"开头的所有 rpm 包的一个列表，包括一些 Wine 包。[/color]

2.4. 为其他发布获得 Wine

如果你的系统不是 Debian 或 Redhat，第一步是看 WineHQ 下载页。这个页面列出了许多 Wine 的分类混合的(assorted)二进制(预编译)归档文件。

Lycos FTPSearch 是搜寻杂类发布包的另一个有用的资源。

2.5. 从 FTP 归档获得 Wine 源代码

如果你想要的 Wine 的版本没有现存的包，你可以自己下载源代码并在你的机器上编译它。如果以前从未做过，第一次时可能感觉有点恐怖，你将发现这通常是非常顺畅的，特别是在新近的 Linux 发布上。

获得源代码的最安全的方式是从官方 FTP 归档获取。在 Wine 发布中的 ANNOUNCE 文件中(如果你已经下载了它的话)有一个最新列表。下面是(可能过时了)承载 Wine 的 FTP 服务器的一个列表:

ftp://metalab.unc.edu/pub/Linux/ALPHA/wine/development/

ftp://tsx-11.mit.edu/pub/linux/ALPHA/Wine/development/

ftp://ftp.infomagic.com/pub/mirrors/lin ...velopment/

ftp://orcus.progsoc.uts.edu.au/pub/Wine/development/

官方发布加上了"Wine-YYYYMMDD.tar.gz"格式的日期标注。你最好的赌注是获得最新的发布。

FIXME: 解释如何 un-tar、从一个 tarball 编译和安装 Wine。[/color]

2.6. 从 CVS 获得 Wine 源代码

Wine CVS 的官方 Web 页是 http://www.winehq.com/dev.html。

首先，你需要使用 CVS 得到最新的 Wine 源代码的一个复件。你可以通过设置 CVSROOT 环境变量来告诉它到那里去找到源代码树。你还必须匿名登录到 wine CVS 服务器上。在 bash 中，过程可能如下:

```
$ export CVSROOT=:pserver:cvs@cvs.winehq.com:/home/wine
$ cvs login
Password:cvs
$ cvs checkout wine
```

这将从 winehq.com 取回整个 Wine 源代码树并把它放置到当前目录中(实际上在'wine'子目录中)。CVS 有大量命令行参数，所以有多种方式从修订历史中的某个地方取回文件。此后，你可以只获取更新的文件:

```
$ cvs -dP update
```

cvs update 从源代码树的内部工作。你不需要 CVSROOT 环境变量来运行它。你只需要在代码树中。-d 和 -P 选项确保你的本地 Wine 树目录结构与远程仓库相同步。

在你做了变动之后，你可以通过 cvs diff -u 建立一个补丁，它把输出发送的 stdout(-u 控制这个补丁的格式)。所以，要建立一个 my_patch.diff 文件，你可以这样做:

```
$ cvs diff -u >;my_patch.diff
```

你可以从树中的任何地方调用 cvs diff(如同 cvs update)，并且它将总是从这一点上递归的获取文件。你还可以指定一个单一的文件或子目录:

```
$ cvs diff -u dlls/winaspi>;my_aspi_patch.diff
```

做点实验，这是非常凭直觉的。

2.7. 用补丁更新 Wine

如果你有 Wine 源代码，与二进制发布相反，你可以选择向源代码树提供补丁来修理缺陷并增加实验性的特征。可能你已经发现了一个缺陷，请向 Wine 邮

件列表报告，并接收一个补丁来修理这个缺陷。你可以用 patch 命令运用补丁，它从 stdin 接受补丁:

```
$ cd wine
$ patch -p0 < ../patch_to_apply.diff
```

要删除补丁使用 -R 选项:

```
$ patch -p0 -R < ../patch_to_apply.diff
```

如果你想要测试一下是否成功的提供了补丁(例如，如果这个补丁是从这个树的(比当前)老或新的版本建立的)，你可以使用 --dry-run 参数来运行补丁而不写任何文件:

```
$ patch -p0 --dry-run < ../patch_to_apply.diff
```

patch 是从一个文件中提取补丁的聪明的方法，所以如果你保存了一封邮件，其中包含了给你的硬驱动器上的一个文件的内置(inline)的补丁，你可以在其上调用补丁而不须剥除邮件头部和其他文本。 patch 忽略看起来不象补丁的所有东西。

FIXME: 进一步解释 -p0 选项...

第 3 章. 安装/编译 Wine

目录

3.1. WWN #52 特征: 替代 Windows

编写： Ove Ken <ovek@winehq.com>;

3.1.1. 安装概述

一个 Windows 安装由许多不同部分组成。

注册表。即使在一个新安装的 Windows 系统中，都假定存在许多键并包含着有意义的数据。

目录结构。应用程序期望在指定的预先决定的位置找到和/或安装东西。期望多数这些目录存在。但与 Unix 目录结构不同，多数这些位置是硬编码的(hardcode)，并通过 Windows API 和注册表来查找。在 Wine 安装上这提出了额外的要求。

系统 DLL。在 Windows 中，它们通常驻留在 system (或 system32)目录中。一些 Windows 应用程序在尝试装载它们之前在这些目录中检查它们的存在。当应用程序要求一个 DLL 而 Wine 不能装载它们自己的内部 DLL(.so 文件)时，Wine 不模拟不存在文件的存在。

尽管用户可以自由的自己设置所有的东西，Wine 小组仍将制作自动 Wine 安装脚本 tools/wineinstall，做我们认为必要的所有事情；除非你知道正在做什么，否则不推荐运行常规的 configure && make depend && make && make install 周期。此时，可使用 tools/wineinstall 来建立一个配置文件、安装注册表、并建立自己的目录结构。

3.1.2. 注册表
缺省的注册表在文件 winedefault.reg 中。它包含目录路径、类 ID、及其他东西；在多数 INSTALL.EXE 或 SETUP.EXE 运行之前必须安装它。在以前的文章中有对注册表非常详细的介绍。

3.1.3. 目录结构
下面是 Windows 应用程序和安装器所期望的基本格局。没有它，这些程序不能正确操作。

C:\ 主磁盘驱动器的根目录
Windows\Windows 目录，包含 .INI 文件、附件等
System\Win3.x/95/98/ME 给公用 DLL 的目录
WinNT/2000 给公用 16-bit DLL 的目录

System32\WinNT/2000 给公用 32-bit DLL 的目录
Start Menu\ 程序启动项目录结构
Programs\ 到应用程序的程序启动项连接(.LNK 文件)
Program Files\ 应用程序二进制文件(.EXE 和 .DLL 文件)

Wine 通过把它们的虚拟驱动器根放置到 Unix 文件系统中用户可配置点上来模拟驱动器，所以你自己选择 C：的根应该在哪 (tools/wineinstall 会问你的)。如果你选择了，比如 /var/wine 作为你的虚拟驱动器 C 的根，则你应该把它放置到你的 ~/.wine/config 文件中:

```
[Drive C]
"Path" = "/var/wine"
"Type" = "hd"
"Label" = "MS-DOS"
"Filesystem" = "win95"
```

使用这个配置，被 windows 应用程序当作为 "c:\windows\system " 的目录将被映射为 UNIX 文件系统中的 /var/wine/windows/system 。注意你必须指定 "Filesystem " = "win95" 而不是 "Filesystem " = "unix",来使 Wine 模拟 Windows-相容的(大小写不敏感)文件系统，否则多数应用程序将不能工作。

3.1.4. 系统 DLL
Wine 小组决定需要建立伪 DLL 文件来欺骗那些通过检查文件存在来确定是否可获得一个特征(比如 Winsock 和它的 TCP/IP 连网)的应用程序。如果你也有这个问题，你可以在 system 目录中建立一个空文件来让应用程序认为它存在，而在应用实际要求它的时候 Wine 将装载它的内置的 DLL。(不幸的是，tools/wineinstall 自身不建立这样的空文件。)

应用程序有时还尝试从物理文件中检查资源的版本(例如，要确定 DirectX 版本)。在这种情况下空文件就不起作用了，必须有完整版本资源的安装文件。当前正在处理这个问题。目前，你仍需获取一些真实的 DLL 来欺骗这些应用程序。

对于那些 wine 目前未很好实现(或根本未实现)的 DLL。如果不拥有一个真实的 Windows，则有的人将会窃取所需的 DLL，总是可以从象 http://solo.abac.com/dllarchive/ 这样的地方得到一个 DLL 归档。

3.2. 安装 Wine 而无须 Windows
编写：James Juran <juran@cse.psu.edu>;

(提取自 wine/documentation/no-windows)

Wine 的一个主要目标是允许用户运行 Windows 程序而无须在它们的机器上安装 Windows。Wine 实现了通常由 Windows 提供的主要 DLL 的功能。所以，一旦完成了 Wine，使用 Wine 将不需要拥有一个安装好的 windows。

Wine 已经进行到足够的程度，它已经可以运行你的目标程序而无须安装好的 Windows。如果你要尝试，请依从下列步骤:

建立空 C:\windows、C:\windows\system 、C:\windows\Start Menu、和 C:\windows\Start Menu\Programs 目录。不要把 Wine 指向充满了旧有安装和不干净的注册表的 Windows 目录。(Wine 在你的 home 目录中建立一个特殊的注册表，在 $HOME/.wine/*.reg 中。你可能需要删除这些文件)。

把 ~/.wine/config 中的 [Drive C] 指向你希望 C: 所在的地方。参照 Wine 手册页来的得到更详细的信息。记住使用 "Filesystem " = "win95"!

使用 tools/wineinstall 来编译 Wine 并安装缺省注册表。或者如果你偏好自己做，编译 programs/regapi，并运行:programs/regapi/regapisetValue

< winedefault.reg

运行和/或安装你的应用程序。

因为 Wine 仍未最终完成，一些应用程序与固有 Windows DLL 一起运行要比与 Wine 的替代品一起运行更好。Wine 被设计为使之可能。Juergen Schmied (和其他人)关于如何进行有一些提示。这里假定在配置文件中你的 C:\windows 目录不指向一个固有 Windows 安装而是指向一个独立的 Unix 文件系统。(例如， "C:\windows" 是位于 "/home/ego/wine/drives/c" 下的一个真实的子目录 "windows")。

运行应用程序并加上 --debugmsg +module,+file 参数来找出所须的文件。把所须的文件逐个复制到 C:\windows\system 目录中。但不要复制 KERNEL/KERNEL32、GDI/GDI32、或 USER/USER32。它们实现了 Windows API 的核心功能，而必须使用 Wine 的内部版本。

编辑 ~/.wine/config 中的 "[DllOverrides]" 段，为你要使用的 Windows DLL 在 "builtin" 之前指定 "native"。详情请参见 Wine 手册页。

注意尽管 Wine 寻找一些网络 DLL 但不需要它们。Windows MPR.DLL 目前不能工作；你必须使用内部实现。

把 SHELL/SHELL32 和 COMDLG/COMDLG32、COMMCTRL/COMCTL32 成对的复制到你的 Wine 目录中 (使用这些 DLL 是"干净的")。确保在 ~/.wine/config 的"[DllPairs]"段中指定了这些。

要一致: 只一起使用同一个 Windows 版本的 DLL。

把 regedit.exe 放置到 C:\windows 目录中。(Office 95 与一个空注册表一起工作时导入一个 *.reg 文件，对 Office 97 不清楚)。

如果你要浏览程序的帮助功能还要添加 winhelp.exe 和 winhlp32.exe 。
[/color]

3.3.处理 FAT/VFAT 分区
编写：Steven Elliott < elliotsl@mindspring.com > ;

(提取自 wine/documentation/linux-fat-permissions)

本文描述工作在 Linux 中对 FAT 和 VFAT 文件系统的权限，焦点是为 Wine 配置它们。

3.3.1. 介绍
Linux 可以使用 FAT (老的 8.3 DOS 文件系统) 或 VFAT (新的 Windows

95 或后来的长文件名文件系统）模块访问 DOS 和 Windows 文件系统。在双引导(Linux + Windows)系统上，要通过 Wine 访问现存的应用程序和它们的数据，挂装 FAT 或 VFAT 文件系统是提供的主要的方式。

按照 ~/.wine/config 文件的指示，Wine 把挂装的 FAT 文件系统比如 /c，映射成驱动器字母比如 "c:"。下列摘录自一个 ~/.wine/config 文件:

```
[Drive C]
"Path" = "/c"
"Type" = "hd"
```

尽管在长文件名支持上 VFAT 文件系统优于 FAT 文件系统，在本文剩余部分中使用术语 "FAT" 来参照 FAT 文件系统和它的衍生品。还有，本文通篇使用 "/c" 作为 FAT 挂装点的例子。

多数现代 Linux 发布要么检测现存的 FAT 文件系统要么允许配置现存的 FAT 文件系统，这样，可以要么持久的(在引导时)要么在需要的时候，在一个位置上如 /c 挂装它们。在任何一种情况下，缺省的，权限可能被配置成:

```
~>;cd /c
/c>;ls -l
-rwxr-xr-x 1 root root 91 Oct 10 17:58 autoexec.bat
-rwxr-xr-x 1 root root 245 Oct 10 17:58 config.sys
drwxr-xr-x 41 root root 16384 Dec 30 1998 windows
```

这里所有的文件都属于 "root"，都在 "root" 组中并只能被 "root" 写(755 权限)。这限制了应用程序要想写文件系统的任何部分，只能以 root 运行 Wine。

有三种主要的途径来克服上面段落提及的受限制的权限:

以 root 运行 Wine

用更少受限制的权限来挂装 FAT 文件系统

通过完全或部分复制它来隐蔽 FAT 文件系统

在下面的小节中将讨论每种方式。

3.3.2. 以 root 运行 Wine

要给予 Wine 所运行的应用程序无限制的对 FAT 文件系统访问，以 root 运行 Wine 是最简单和最直接的方式。以 root 运行 wine 还允许应用程序做与 FAT 文件系统无关的事情，比如监听小于 1024 的端口。以 root 运行 Wine 是危险的，原因是对应用程序在系统上的所作所为没有限制。

3.3.3. 挂装 FAT 文件系统
可以用比缺省更少受限制的权限挂装 FAT 文件系统。要么变更挂装 FAT 文件系统的用户，要么显式的变更挂装的 FAT 文件系统的权限。 FAT 文件系统从挂装 FAT 文件系统的进程继承权限。因为挂装 FAT 文件系统的进程通常是以 root 运行的一个启动脚本，FAT 文件系统继承 root 的权限。这导致在 FAT 文件系统上的文件有类似于用 root 建立的文件的权限。例如:

```
~> ;whoami
root
~> ;touch root_file
~> ;ls -l root_file
-rw-r--r-- 1 root root 0 Dec 10 00:20 root_file
```

它匹配属主、组和除了缺少 'x' 之外在 FAT 文件系统上见到的文件权限。在 FAT 文件系统上的权限可以通过改变 root 的 umask (unset permissions bits) 来变更。例如:

```
~> ;umount /c
~> ;umask
022
~> ;umask 073
~> ;mount /c
~> ;cd /c
/c> ;ls -l
-rwx---r-- 1 root root 91 Oct 10 17:58 autoexec.bat
-rwx---r-- 1 root root 245 Oct 10 17:58 config.sys
drwx---r-- 41 root root 16384 Dec 30 1998 windows
```

用 umask 码 000 挂装 FAT 文件系统给予所有用户对它的完全控制。在挂装的时候提供额外的控制，显式的指定 FAT 文件系统的权限。与 FAT 权限相关的有三个挂装选项: uid、gid 和 umask。在手动挂装文件系统的时候可以指定它们。例如:

```
~> ;umount /c
~> ;mount -o uid=500 -o gid=500 -o umask=002 /c
~> ;cd /c
/c> ;ls -l
-rwxrwxr-x 1 sle sle 91 Oct 10 17:58 autoexec.bat
-rwxrwxr-x 1 sle sle 245 Oct 10 17:58 config.sys
drwxrwxr-x 41 sle sle 16384 Dec 30 1998 windows
```

给予 "sle" 在 /c 上完全控制权限。可以通过在上面列出的选项添加到 /

etc/fstab 文件中而使之成为永久:

```
~>;grep /c /etc/fstab
/dev/hda1 /c vfat uid=500,gid=500,umask=002,exec,dev,suid,rw 1 1
```

注意 umask 码 002 一般用于用户私有组文件权限中。在 FAT 文件系统上这个 umask 确保在指定组(gid)中的所有用户可以完全访问所有文件。

3.3.4. 隐蔽 FAT 文件系统
隐蔽提供了精细的控制粒度。通过复制部分最初的 FAT 文件系统，应用程序可以安全的在复制的这部分上工作，而继续直接读其余的部分。这是通过符号连接完成的。例如，考虑一个叫 AnApp 的应用程序必须能读写 c:\windows 和 c:\AnApp 目录而可读整个 FAT 文件系统的的一个系统。在这个系统中 FAT 文件系统有缺省的权限，出于安全的原因或缺乏 root 访问权限而不应该被改变。在这个系统上可以用下列方式设置一个隐蔽目录:

```
~>;cd /
/>;mkdir c_shadow
/>;cd c_shadow
/c_shadow>;ln -s /c_/*.
/c_shadow>;rm windows AnApp
/c_shadow>;cp -R /c_/{windows,AnApp}.
/c_shadow>;chmod -R 777 windows AnApp
/c_shadow>;perl -p -i -e 's|/c$|/c_shadow|g'
/usr/local/etc/wine.conf
```

上述命令给予所有用户对 windows 和 AnApp 目录完全的读写访问，而只有 root 可以写访问其他目录。[/color]

3.4. SCSI 支持
编写： Bruce Milner; 补充：Andreas Mohr<
amohr@codeweavers.com>;

(提取自 wine/documentation/aspi)

这个文件描述设置 Windows ASPI 接口。

警告/警告/警告!!!!!!
　如果不正确使用可能会使你的系统报废
如果正确使用可能会使你的系统报废

我已经说过，ASPI 是从 windows 程序到 SCSI 设备的直接连接。ASPI 只转发程序发送到 SCSI 总线上 SCSI 命令。

如果在你的设置文件中使用了错误的 SCSI 命令,你可以向不合适的设备发送完全伪造的命令 - 例如格式化你的硬设备(假定这个设备给予你权限 - 如果你以 root 运行，所有赌注都将失去)。

所以请确保把程序不需要的所有 SCSI 设备的权限设置为尽可能的受限制!

设置扫描仪的指导:(至少让扫描仪工作起来)(同样适用于其他设备如 CD 刻录机， MO 驱动器，...，诸如此类)

3.4.1.Windows 要求
扫描仪软件需要使用 "Adaptec" 兼容驱动器(ASPI)。至少与 Mustek 一起，它们允许你选择使用内置卡或 "Adaptec (AHA)" 兼容驱动器。任何其他方式都不能工作。支持通过 DOS ASPI 驱动器(例如， ASPI2DOS) 访问扫描仪的软件。[AM]

你可能需要这个软件的一个真实的 windows 安装来正确设置 LUN's/SCSI id。我也不是非常确定。

3.4.2.LINUX 要求
在 linux 下必须支持你的 SCSI 卡。对未知 SCSI 卡将不能工作。甚至对廉价的垃圾 "扫描仪专用" 控制器在网上都有特定的 Linux 驱动程序。如果你要使用你的 IDE 设备，你需要使用 ide-scsi 模拟器。阅读 http://www.linuxdoc.org/HOWTO/CD-Writing-HOWTO.html 来获得 ide-scsi 设置指导。把 SCSI 驱动程序编译到你的内核中。

对于最新的(2.2.x)内核好象不要求别的什么了:Linux 缺省使用比 Windows 小的 SCSI 缓冲区。定义了 SG_BIG_BUFF(在 sg.h 中)的内核建造缺省设置太低。SANE 计划推荐 130560 好象就工作的很好。这要求重新建造系统内核。

为扫描仪建造一个设备(一般 SCSI 设备)- 关于设备编号请参见 http://www.linuxdoc.org/HOWTO/SCSI-Prog...HOWTO.html 的 SCSI 编程 HOWTO。

我建议让扫描仪设备对一个组可写。我建立了一个叫 scanner 的组并添加上了我自己。以 root 运行会增加向不适当的设备发送错误 SCSI 命令的危险。使用常规用户，你将受到更好的保护。

对于 Win32 软件(WNASPI32)，Wine 在适当的位置进行自动检测。对于 Win16 软件(WINASPI)，你需要在 ~/.wine/config 中为你的特定扫描仪添加一个 SCSI 设备条目。格式是 [scsicCtTdD],这里的 "C" = "controller"、"T" = "target"、D=LUN

例如，我设置扫描仪为 controller0、Target6、LUN 0。 [scsic0t6d0] "Device" = "/dev/sgi"

对你的特定的 SCSI 设置可能不同。

3.4.3. 一般信息

我使用一个包 "ipplus" 承载 mustek 扫描仪。这个程序使用 TWAIN 驱动器规定来访问扫描仪。

(TWAIN 管理器)

ipplus.exe <---> (TWAIN INTERFACE) <---> (TWAIN DATA SOURCE.ASPI) -> WINASPI

3.4.4. 注意/缺陷
最大的缺陷是目前只能在 linux 下工作。

ASPI 代码只在下列扫描仪上进行了测试:

在 Linux 下一个 Mustek 800SP 加上在一个 Buslogic 控制器[BM]

在 Linux 下通过 DOSASPI 访问的一个 Siemens Nixdorf 9036 加上 Adaptec AVA-1505。注意我有颜色问题，通过(少见可读的结果)[AM]

一个 Fujitsu M2513A MO 驱动器 (640MB) 使用一般 SCSI 驱动器。格式化和弹出(eject)工作良好。为访问硬件而感谢 Uwe Bonnes![AM]

我不担保 ASPI 代码。它可以使我的扫描仪工作。但可能使你的驱动器爆炸。我无法确定。我承担零责任![/color]

HonestQiao 回复于：2005-09-13 14:11:34

第 4 章. 配置 Wine
目录
4.1. 一般配置
4.2. Win95/98 外观
4.3. 配置 x11drv 驱动器
4.4. 注册表
4.5. Wine 的驱动器标签和系列号
4.6. Dll 加载
4.7. 键盘
4.8. 处理字体
4.9. 在 Wine 中打印
设置 config 文件等。

4.1. 一般配置
Copyright 1999 Adam Sacarny <magicbox@bestweb.net>;

(提取自 wine/documentation/config)

4.1.1. Wine 配置文件

Wine config 文件存储各种 Wine 的设置。包括:

驱动器和关于它们的信息

目录设置

端口设置

Wine 外观和感觉

Wine 的 DLL 用法

4.1.2. 我如何做一个?
本节将带领你经过制作一个 config 文件的(全部)过程。看一下文件 <dirs to wine>;/documentation/samples/config。它是由段组织起来的。

段名 需要? 用途
[Drive X] 是 设置 wine 识别的驱动器
[wine] 是 设置 wine 目录
[DllDefaults] 推荐 缺省装载的 DLL
[DllPairs] 推荐 对 DLL 的健全检查
[DllOverrides] 推荐 屏弃缺省的 DLL 装载
[options] 否 好象没人知道
[fonts] 是 字体外观和识别
[serialports] 否 wine 见到的 COM 端口
[parallelports] 否 wine 见到的 LPT 端口
[spooler] 否 打印缓冲池
[ports] 否 直接访问端口
[spy] 否 怎样处理特定的调试信息
[Registry] 否 指定 windows 注册表文件的位置
[tweak.layout] 推荐 wine 的外观
[programs] 否 自动运行的程序
[Console] 否 控制台设置

4.1.2.1. [Drive X] 段
这是自明的,下面是深入的指导。对于 Wine 中的每个驱动器最多有 6 行。

```
[Drive X]
```

上面的行开始了一个给字母是 X 的那个驱动器的段落。

```
Path=/dir/to/path
```

这个路径是这个驱动器开始的地方。当 Wine 在驱动器 X 中浏览的时候，它将见到目录 /dir/to/path 中的文件。不要忘记去掉尾随的斜杠！

"Type" = "floppy|hd|cdrom|network" <--- 这个 | 的意思是 "Type = "<选项之一>;"

设置 Wine 将见到的驱动器类型。类型必须等于下列四者之一 floppy、hd、cdrom 、或 network。它们是自明的。

"Label" = "blah"

定义驱动器标签。一般只有查找一个特定的 CD-ROM 的应用程序需要它。关于查找标签的信息请参见 <dirs to wine>;/documentation/cdrom-labels。标签最多是 11 个字符。

"Serial" = "deadbeef"

告诉 Wine 这个驱动器的系列号。一些意图防止剽窃的应用程序可能需要它，否则不要使用它。最多 8 个字符和十六进制数。

"Filesystem" = "msdos|win95|unix"

设置 Wine 查看这个驱动器上的文件的方式。

msdos
大小写不敏感文件系统。类似于 DOS 和 Windows 3.x。 8.3 是文件名的最大长度(eightdot.123)- 更长的部分将被截掉。(注意：如果你想运行使用长文件名的程序，这将是非常糟的选择。win95 可以很好的与设计在 msdos 系统下运行的应用程序一起工作。换句话来说，你可能不需要使用它。)

win95
大小写不敏感。类似于 Windows 9x/NT4。这可能是你用来工作的长文件名文件系统。是给 wine 下运行的多数应用程序选择的文件系统。可能正是你需要的!

unix
大小写敏感。这个文件系统基本不用(Windows 应用程序需要大小写不敏感文件名)。如果你敢就试一下，win95 是更好的选择。

"Device" = "/dev/xx"

只用于软盘和光盘设备。在 Extended2 分区上使用它将有灾难性的结果(在一个 windows 应用程序尝试做一次低层写的时候，他们以在 FAT 下的方式去做

-- FAT 不能与 Extended2 混同使用)。

注意:这个设置不是十分重要;如果保持未指定,几乎所有的应用程序都没有问题。对于 CD-ROM 你可能希望添加它来获得自动的标签检测。如果你不能确定指定的设备名字,只须为你的设备空缺这个设置。

下面是 Drive X 的一个设置,它是一个通用的硬盘驱动器:[Drive X]
"Path"= "/dos-a"
"Type"= "hd"
"Label"= "Hard Drive"
"Filesystem"= "win95"
下面是 Drive X 的一个设置,它是一个通用的 CD-ROM 驱动器:
[Drive X]
"Path"= "/dos-d"
"Type"= "cdrom"
"Label"= "TotalAnnihilation"
"Filesystem"= "win95"
"Device"= "/dev/hdc"
下面是 Drive X 的一个设置,它是一个通用的软盘驱动器:
[Drive X]
"Type"= "floppy"
"Path"= "/mnt/floppy"
"Label"= "Floppy Drive"
"Serial"= "87654321"
"Filesystem"= "win95"
"Device"= "/dev/fd0"

4.1.2.2.[wine]段
配置文件的 [wine]段包含 wine 使用的目录的信息。在给这些设置指定目录的时候,按它们在 wine 中出现的那样设置它们。如果你的驱动器 C 有一个路径 /dos,并且你的 windows 目录位于 /dos/windows,则使用:"
Windows"= "c:\\windows"

它设置 windows 目录。如果你未曾有这个目录则建一个。没有尾随的斜杠(不能是 C:\\windows\)!

"System"= "c:\\windows\\system"

它设置 windows 系统文件所在的地方。这个目录应当驻留在用于 Windows 设置的目录中。如果你没有 windows 则它应是系统文件所在的地方。再次强调,没有尾随的斜杠!

"Temp"= "c:\\temp"

这应该是你打算把临时文件存储到其中的目录。你必须有到它的写访问权限。

"Path" = "c:\\windows;c:\\windows\\system;c:\\blanco"

行为好象是 UNIX 的 PATH 环境变量设置。在 wine 运行如 wine sol.exe，如果 sol.exe 驻留在 Path 设置中指定的一个目录中，wine 将会运行它(当然，如果 sol.exe 驻留在当前目录中，wine 也可以运行它)。确保它总包含你的 windows 目录和系统目录(对于这个设置，它必须包含 "c:\\windows;c:\\windows\\system")。

"SymbolTableFile" = "wine.sym"

为 wine 调试器设置符号表。你可能不需要摆弄它。如果你的 wine 出了问题(stripped)可能用到它。

"printer" = "off|on"

告诉 wine 是否允许打印机驱动程序和打印工作。这些东西仍处在 alpha 阶段，所以使用它你要当心。但一些人可能发现它很有用。如果你不打算进行打印工作，则不要把它添加到你的 ~/.wine/config 中(它可能不在其中)。还要检查[spooler]和 [parallelports]段。

4.1.2.3. 介绍 DLL 段
在 wine 配置文件中设置 DLL 段之前，你需要知道一些事情。

4.1.2.3.1. Windows DLL 对
多数 windows DLL 有 win16 (Windows 3.x)和 win32 (Windows 9x/NT)两种形式。win16 和 win32 DLL 版本的组合叫做"DLL 对"。下面是最常见的 DLL 对:

Win16	Win32	固有 [a]
KERNEL	KERNEL32	否!
USER	USER32	否!
SHELL	SHELL32	是
GDI	GDI32	否!
COMMDLG	COMMDLG32	是
VER	VERSION	否

注释:a.
是 wine 可以使用固有的 dll?(参见下节)

4.1.2.3.2. DLL 的不同形式

wine 可以装载的 DLL 有多种形式:

native(本地，固有)
DLL 包含在 windows 中。许多 windows DLL 可以用它们固有的形式来装载。许多时候这些固有版本比它们的非 Microsoft 替代品要好一些，但不总是。

elfdll
用 ELF 封装的 windows DLL。当前还是实验性的(仍不能工作)。

so
本地 ELF 库。仍不能工作。

builtin(内置)
DLL 装载的最通常形式。如果 DLL 用固有形式装载是错误的(error-prone)(例如，KERNEL)，你没有固有的 DLL，或你想自由于 Microsoft，则可以使用它们。

4.1.2.4.[DllDefaults]段
这些设置提供了 wine 的缺省 DLL 装载处理。

"DefaultLoadOrder"="native,so,builtin"

这个设施是一个逗号分界的列表，按它的次序尝试装载 DLL。如果第一种选项失败了，它将尝试第二种，以次类推。上面的次序在多数情况下是最好的。

4.1.2.5.[DllPairs]段
有时，在缺省配置文件中有一个叫做 [DllPairs] 的段，它已经被废弃了，原因是组对信息已经被嵌入到 Wine 自身中。(本段的目的只不过是如果用户尝试组对(paircodependent)不同类型的 16-bit/32-bitDLL 则发起警告。) 如果你的 wine.conf 或 ~/.wine/config 中仍然有它，你删除它是安全的。

4.1.2.6.[DllOverrides]段
本段的格式对于每行都是相同的:<DLL>;{<DLL>;,<DLL>;...}=
<FORM>;{<FORM>;,<FORM>;...}

例如，要装载内置的 KERNEL 对(这里大小写都行):"kemel,kemel32"="builtin"

要装载固有 COMMDLG 对，但如果它们不工作则尝试内置的:"

comm dlg,comdlg32"= "native,builtin"

要装载 COMCTL32:"comctl32"= "native"

下面是一个很好的通用设置(在你 wine 包中的 config 文件中定义的):

```
[DllOverrides]
"commdlg"= "builtin,native"
"comdlg32"= "builtin,native"
"ver"= "builtin,native"
"version"= "builtin,native"
"shell"= "builtin,native"
"shell32"= "builtin,native"
"lzexpand"= "builtin,native"
"lz32"= "builtin,native"
"comctl32"= "builtin,native"
"commctrl"= "builtin,native"
"wsock32"= "builtin"
"winsock"= "builtin"
"advapi32"= "builtin,native"
"crtdll"= "builtin,native"
"mpr"= "builtin,native"
"winspool.drv"= "builtin,native"
"ddraw"= "builtin,native"
"dinput"= "builtin,native"
"dsound"= "builtin,native"
"mmsystem"= "builtin"
"winmm"= "builtin"
"msvcrt"= "native,builtin"
"msvideo"= "builtin,native"
"msvfw32"= "builtin,native"
"mcicda.drv"= "builtin,native"
"mciseq.drv"= "builtin,native"
"mciwave.drv"= "builtin,native"
"mciavi.drv"= "native,builtin"
"mcianim.drv"= "native,builtin"
"msacm.drv"= "builtin,native"
"msacm"= "builtin,native"
"msacm32"= "builtin,native"
"midimap.drv"= "builtin,native"
"wnaspi32"= "builtin"
"icmp"= "builtin"
```

注意:你见到此中的一些 dll 的第一选项是 elfdll 或 so。对于你这个尝试将是失败的，但你不用管它，它会使用第二个或第三个选项。

4.1.2.7.[options]段
好象没人知道这段是干什么用的...

"AllocSystemColors"="100"

分配的系统颜色?保持它为 100。

4.1.2.8.[fonts]段
本段设置 wine 的字体处理。

"Resolution"="96"

因为 X 处理字体的方式与 Windows 的方式不同，wine 使用一个特殊的机制来处理它们。它必须使用在"Resolution"设置中的数来缩放字体。60-120 是合理的值，96 是此间的一个很好的值。如果你能获得真实的 windows 字体(参见 <dirs to wine>/documentation/ttfserver 和 fonts)，这个选项就不重要了。当然，总是可以使你的 X 字体在 wine 中工作的很好。

"Default"="-adobe-times-"

wine 使用的缺省字体。随你的喜好去设置(foolaround with)它。

可选的:

Alias 设置允许你把一个 X 字体映射成在 wine 中使用的一个字体。如果应用程序需要使用你没有的特定字体,而存在一个很好的替代字体,可使用这个设置。语法如下:"AliasX"="[伪装的 windows 名字],[真实的 X 名字]"<,可选的"屏蔽"标志 >;

非常直接，把 "AliasX" 替换为 "Alias0", 接着 "Alias1" 并以此类推。伪装的 windows 名字是在 wine 中的 windows 应用程序将见到的字体名字。而真实的 X 名字是 X 见到的字体名字(运行 "xfontsel"可以查看)。可选的"屏蔽"段允许你利用你定义的伪装 windows 名字。如果不使用它，则 wine 将只是尝试提取伪装 windows 名字自身而不使用你输入的值。

下面是没有屏蔽的别名的例子。字体在 windows 应用程序中将表现为"Google"。在一个 config 文件中定义一个别名的时候，请忘记我的注释文本("<-- blah" 材料)"Alias0"="Foo,--google-"

下面是启用屏蔽的例子。在 windows 中字体将表现为 "Foo"。 "Alias1" = "Foo,--google-,subst"

详情参见 <dirs to wine>;/documentation/fonts

4.1.2.9.[serialports]、[parallelports]、[spooler]、和 [ports]段
尽管看起来好象是很多段,它们是紧密关联的。它们都是关于通信和并行端口的。

[serialports]段告诉 wine 那些串行端口是允许使用的。 "Com X" = "/dev/cuaY"

把 X 替换为 Windows 中 COM 端口号(1-8)而 Y 是它在 X 的编号(通常是在 Windows 中端口号减 1)。 Com X 实际上可以是任何设备(/dev/modem 是可接受的)。不总是需要定义任何 COM 端口(一个可选的设置)。下面是一个例子:"Com 1" = "/dev/cua0"

你想设置多少就设置多少。定义你需要的所有 COM 端口。

[parallelports]段设置在 wine 下可以访问的任何并行端口。 "LptX" = "/dev/lpY"

很熟悉? 语法很象 COM 端口设置。把 X 替代为 1-4 的一个值如同在 Windows 中那样,把 Y 替代为 0-3 的一个值 (同 COM 端口一样, Y 通常是在 windows 中的值减 1)。你不总是需要定义一个并行端口(AKA,它是可选的)。象其他段一样, LptX 可以等于任何设备(可能是 /dev/printer)。下面是一个例子:"Lpt1" = "/dev/lp0"

[spooler]段将通知 wine 在那里缓冲(spool)打印作业。如果你想打印的话就要使用它。Wine docs 声称现在的缓冲池是 "非常原始的",所以它不能很好的工作。这是可选的。在本段中你唯一的设置工作是把一个端口(例如,LPT1)映射到一个文件或一个命令上。下面是一个例 子,把 LPT1 映射到一个文件 out.ps 上:"LPT1:" = "out.ps"

下列命令把到 LPT1 的打印作业映射到命令 lpr。注意这个 |:"LPT1:"= "|lpr"

[ports]段只在需要直接端口访问的时候是有用的，例如用户的程序需要 dongle(加密/解密器)或扫描仪。如果不需要，就不要用它!

"read"= "0x779,0x379,0x280-0x2a0"

给予到这些 IO 的直接读访问。

"write"= "0x779,0x379,0x280-0x2a0"

给予到这些 IO 的直接写访问。这是保持 read 和 write 的设置相同可能是个好主意。只有你以 root 运行的时候这些材料才能工作。

4.1.2.10.[spy]、[Registry]、[tweak.layout]、和 [programs]段
使用 [spy]来包含或排除调试信息，并把它们输出到一个文件中。后者是很少使用的。这些都是可选的，你可能不需要向你的 config 中的这个段增加或删除任何东西。

"File"= "/blanco"

为 wine 设置日志文件。设置为 CON 来记录到标准输出。这很少使用。

"Exclude"= "WM_SIZE;WM_TIMER;"

在日志文件中排除关于 WM_SIZE 和 WM_TIMER 的调试信息。

"Include"= "WM_SIZE;WM_TIMER;"

在日志文件中包含关于 WM_SIZE 和 WM_TIMER 的调试信息。

使用 [Registry]来告诉 wine 你的旧有的 windows 注册表文件存在于什么地方。这个段是完全可选的，而且对没有现存 windows 安装的人是没用的。

"UserFileName"= "/dirs/to/user.reg"

你旧有的 user.reg 文件的位置。

[tweak.layout]决定 wine 的外观。它只有一个设置。

"WineLook"= "win31|win95|win98"

可以把 wine 的外观改变为 Windows 3.1 和 Windows 95。win98 设置的行为在多数情况下类似于 win95。

使用 [programs]来说明在特定条件下运行什么程序。

"Default"= "/program /to/execute.exe"

设置启动 wine 而未指定一个程序的时候运行的程序。

"Startup"= "/program /to/execute.exe"

设置在每次启动的时候自动运行的程序。

4.1.3. 我把它放在哪里?
wine config 文件可以放到两个地方。

/usr/local/etc/wine.conf
系统范围的 config 文件，用于没有自己的配置文件的任何人。注意：这个文件当前未使用，原因是一个新的全局配置机制现在仍未准备好。

$HOME/.wine/config
你自己的 config 文件，只用于你的用户。

把你的 wine.conf 版本文件复制到 /usr/local/etc/wine.conf 或 $HOME/.wine/config 来让 wine 识别它。

4.1.4. 如果它不工作怎么办?
事情总是可能出错。如果不可想象的事情发生了，尝试一下新闻组，comp.emulators.ms-windows.wine，或者在 irc.stealth.net:6668 或连接的服务器上找寻 IRCnet 频道 #WineHQ。确保你完全的查阅了本文档，并且还读了:

README

http://www.la-sorciere.de/wine/index.html (可选的，推荐的)

如果看起来你真的已经作了研究，等着收到有帮助的建议好了。如果你没有，作好受到指责的准备。[/color]

4.2. Win95/98 外观
编写：David A. Cuthbert <dacut@ece.cmu.edu>;

(提取自 wine/documentation/win95look)

介绍 Win95/Win98 界面设置。

不再使用 #define 开关为 Win3.1 和 Win95 外观来编译 Wine，现在在 ~/.wine/config 中一个特殊的 [Tweak.Layout] 段中编码，其中有 "WineLook"= "Win95" 或 "WineLook"= "Win98" 条目。

已经向 ~/.wine/config 文件增加了一些新的段和一些条目 -- 它们只是用来

调试 Win95 tweak 并且可能在将来的发行中删除掉!这些条目/段是:

```
[Tweak.Fonts]
"System.Height"= "<pointsize>;"# 设置系统字型的高度
"System.Bold"= "[true|false]"# 系统字体是否应该是粗体
"System.Italic"= "[true|false]"# 系统字体是否应该是斜体
"System.Underline"= "[true|false]"# 系统字体是否应该有下划线
"System.StrikeOut"= "[true|false]"# 系统字体是否应该有删除线
"OEMFixed.xxx"# 给 OEM fixed 字型的一些参数
"AnsiFixed.xxx"# 给 Ansifixed 字型的一些参数
"AnsiVar.xxx"# 给 Ansivariable 字型的一些参数
"System Fixed.xxx"# 给系统 fixed 字型的一些参数

[Tweak.Layout]
"WineLook"= "[Win31|Win95|Win98]"# 改变 Wine 的感观
```

4.3. 配置 x11drv 驱动器
编写: Ove Ken <ovek@winehq.com>;

(提取自 wine/documentation/cdrom-labels)

多数 Wine 用户在一个叫做 X11 的窗口系统下运行 Wine。在 Wine 历史上的多数时期,这是唯一的可获得的显示驱动器,但是在近几年,已经重新组织了部分 Wine 来允许其他显示驱动器(当前只能获得一个可替代的显示驱动器是 Patrik Stridvall 的基于 ncurses 的 ttydrv,他声称用它来显示 calc.exe)。在~/.wine/config 的 [wine] 段中用 GraphicsDriver 选项来选择显示驱动器,但在本文中我只覆盖 x11drv 驱动器。

4.3.1. x11drv 操作模式
x11drv 驱动器由两个概念上的不同部分组成,图形驱动器(GDI 部分),和窗口驱动器(USER 部分)。但它们二者都被连接到 libx11drv.so 模块中(你用 GraphicsDriver 选项装载它)。在 Wine 中,运行在 X11 上,图形驱动器必须在窗口驱动器提供的 drawable (window interior)上绘图。这与 Windows 模型有一点区别, 在 X11 中,窗口系统建立和配置由图形驱动器控制的设备上下文,而允许应用程序在它们喜欢的任何地方用挂钩连接起(hook into)这个联系。所以,为了要在兼容性和可用性之间提供有道理的任何折中,x11drv 有三种不同的操作模式。

Unmanaged/Normal
缺省的。窗口管理器无关(完全忽略任何运行的窗口管理器)。 窗口装饰(decoration)(标题条,边界,等)由 Wine 绘制来使感观如同真实的 Windows。这兼容于依靠可以计算出这些装饰的精确大小的应用程序,或想自己画它们的程序。

Managed

使用 --managed 命令行选项或 Managed wine.conf 选项(见后)指定。普通的顶层框架窗口，有粗边界，标题条，和由你的窗口管理器管理的系统菜单。这允许这些应用可以与你的桌面的其余部分更好的集成，但可 能不总是工作的很好。(非常需要一个重写的这种操作模式，使它更加健壮和更少瑕疵(patchy)，并计划在 Wine 1.0 发行之前完成。)

Desktop-in-a-Box
使用 --desktop 命令行选项(加上一个几何，比如给一个 800x600 这么大的桌面 --desktop 800x600， 或者在显示器的左上角自动的定位桌面 --desktop 800x600+0+0)。这是与 Windows 模型最兼容的模式。所有应用程序窗口都是在提供给 Wine 的桌面窗口内的 Wine 绘制的窗口(桌面窗口自身由你的窗口管理器来管理)，而 Windows 应用可以随心所欲的在这个虚拟工作空间操作并认为完全拥有它，不受你的其他 X 应用程序的打扰。

4.3.2.[x11drv]段

AllocSystemColors
只在你拥有一个基于调色板(palette)的显示器时使用，例如，如果你的 X 服务器被设置为 8bpp 的色深，并且你没有想要的一个私有颜色映射。它指定 Wine 所占有的共享颜色映射(colormap)单元(palette entries)的最大数目。这个值越高，其他应用程序可获得的颜色就越少。

PrivateColorMap
在你拥有一个基于调色板的显示器时使用，例如，如果你的 X 服务器被设置为 8bpp 的色深。它指定你不想使用共享的颜色映射，而是使用一个私有的颜色映射，这这里可以获得全部的 256 种颜色。不足是这个 Wine 的私有颜色映射只在鼠标指针在这个 Wine 窗口期间可以见到，所以如果你频繁使用鼠标就会经常见到迷幻的闪烁和滑稽的颜色。

PerfectGraphics
它确定在 blit 操作中对于特定的 ROP 代码使用快速 X11 例程还是严格的 Wine 例程。多数用户不能找出任何区别。

ScreenDepth
只在多色深显示器上使用。它指定 Wine 应当使用(并告诉 Windows 应用程序)那种可获得的色深。

Display
它指定使用那个 X11 显示器，并且如果被指定了，它将屏弃 DISPLAY 环境变量和 --display 命令行选项二者。

Managed
Wine 可以让框架窗口由你的窗口管理器来管理。这个选项指定你是否想使之成为缺省的。

UseDGA
它指定你是否想让 DirectDraw 使用 XFree86 的直接图形体系 (Direct Graphics Architecture -- DGA)，这可以接管整个显示器并在全屏幕下以最大

速度运行游戏。(使用 DGA1 (XFree86 3.x),你仍需要首先把 X 服务器配置成游戏所需的 bpp，而使用 DGA2 (XFree86 4.x)，依赖于你的驱动器的能力，有可能使用运行时颜色选择。)但要当心如果 Wine 在 DGA 模式下崩溃，要想收复对你的计算机的控制就只能重启动了。 DGA 通常要求要么特权要么对 /dev/mem 的读/写访问权限。

UseXShm
如果你不希望 DirectX 使用 DGA，你至少可以使用 X 共享内存扩展(X Shared Memory extensions -- XShm)。它比 DGA 慢很多，原因是应用程序不直接访问物理的桢缓冲区，但是使用共享内存来绘制桢至少比使用标准 X11 套接口要快，即使通过 Wine 的 XShm 支持有时仍会崩溃。

DXGrab
如果你不使用 DGA，你可能想要一种替代的方式来确保鼠标光标滞留在游戏窗口中。这个选项就是干这个的。当然，象使用 DGA 一样，如果你的 Wine 崩溃了，你就有麻烦了(但不象 DGA 情况下那么糟，因为你仍然可以使用键盘来退出 X)。

DesktopDoubleBuffered
只在你使用了 --desktop 命令行选项来在一个桌面窗口中运行时使用。指定建立的桌面窗口是否有双缓冲区的 visual，是多数 OpenGL 游戏正确运行所需的东西。

4.4. 注册表
编写：Ove Ken

(提取自 wine/documentation/registry)

在 Win3.x 之后，注册表成为 Windows 的一个基本部分。Windows 自身，和所有遵循 Win95/98/NT/2000/whatever 的应用程序，在其中存贮配置和状态数据。尽管多数有理智的系统管理员(和 Wine 开发者)恶毒的诅咒 Windows 注册表的扭曲天性(twisted nature)。Wine 以某种方式支持它仍是必须的。

4.4.1. 注册表结构
Windows 注册表是一个复杂的树结构,而且多数 Windows 程序员不完全知道注册表是怎样布置的，加上它的不同的 "蜂窝"和它们之间的大量连接；一个完整个论述(coverage)超出了本文档的范围。下面是你需要知道的基本的注册键。

HKEY_LOCAL_MACHINE
这个基本根键(在 win9x 中，存储在隐藏文件 system.dat 中)包含与当前 Windows 有关的所有东西。

HKEY_USERS
这个基本根键(在 win9x 中，存储在隐藏文件 user.dat 中)包含这个安装的每个用户的配置数据。

HKEY_CLASSES_ROOT
这是到 HKEY_LOCAL_MACHINE\Software\Classes 的一个连接。它包含描述文件关联、OLE 文档处理器、和 COM 类的数据。

HKEY_CURRENT_USER
这是到 HKEY_USERS\your_username 的一个连接，比如，你的个人配置。

4.4.2. 使用 Windows 注册表
如果你把 Wine 指向一个现存的 MS 安装(通过在 ~/.wine/config 中设置适当的目录，则 Wine 能从中装载注册表数据。但是，Wine 不会把任何东西保存到真实的 Windows 注册表中，而是保存到它自己的注册表文件中(见后)。当然,如果在 Windows 注册表和 Wine 注册表二者中都存在一个特定的注册值，则 Wine 将使用后者。

偶尔的，Wine 装载 Windows 注册表可能有麻烦。通常，这是因为注册表是不一致或以某种方式被破坏的。如果出现了这个问题，你可以从 MS 的网站下载 regclean.exe 并使用它来清理注册表。作为替代，你总是可以使用 regedit.exe 来把你需要的数据导出到一个文本文件中，并接着导入到 Wine 中。

4.4.3.Wine 注册表数据文件
在用户的主目录中，有一个叫 .wine 的子目录，Wine 缺省的将尝试在这里保存它的注册表。它保存到四个文件中，它们是:

system .reg
这个文件包含 HKEY_LOCAL_MACHINE。

user.reg
这个文件包含 HKEY_CURRENT_USER。

userdef.reg
这个文件包含 HKEY_USERS\.Default (比如，缺省用户设置)。

wine.userreg
Wine 把 HKEY_USERS 保存到这个文件中(当前和缺省二者)，但不从中装载，除非 userdef.reg 丢失的。

所有这些文件是常人可读的文本文件，所以不象 Windows，如果需要的话，你实际上可以用一个普通的文本编辑器来编辑它。

除了这些文件之外，Wine 还可以有选择的从全局注册表文件中装载，它驻留在与全局 wine.conf 相同的目录中(比如，如果你从源代码编译的话则是 /usr/local/etc)。它们是:

wine.system.reg
包含 HKEY_LOCAL_MACHINE。

wine.userreg
包含 HKEY_USERS。

4.4.4. 系统管理
一个系统管理员可以使用上面的文件结构配置系统，这样一个系统 Wine 安装(和应用程序)可以被所有用户共享，而仍旧让用户拥有它自己的个人化配置。一个管理员可以，在安装了 Wine 和用户需要访问的所有 Windows 应用软件之后，把结果的 system.reg 和 wine.userreg 复制成全局注册表文件(我们假定它驻留在 /usr/local/etc)，使用：

```
cd ~/.wine
cp system.reg /usr/local/etc/wine.system.reg
cp wine.userreg /usr/local/etc/wine.userreg
```

并且甚至可以把它们符号连接回管理员的帐户上，以便易于以后安装系统范围的应用程序：

```
ln -sf /usr/local/etc/wine.system.reg system.reg
ln -sf /usr/local/etc/wine.userreg wine.userreg
```

注意如果你以 root 安装 Wine，tools/wineinstall 脚本已经为你作好了这一切。如果你接着在 root 登录期间安装 Windows 应用程序，你的所有用户就自动的可以使用它们了。应用程序的设置将接受全局注册表，而用户的个人化配置将保存在他们自己的主目录中。

但是要注意对管理员帐户的操作 - 如果你把管理员的注册表复制或连接成全局注册表，任何用户就都可以读到管理员的偏好，如果在其中存储了敏感信息(口令、个人信息，等)这就不好了。只使用管理员帐户安装软件，而不是进行日常工作；平时使用一个普通用户帐户。

4.4.5. 缺省的注册表
一个 Windows 注册表缺省的包含许多键，而其中的一些对于安装程序进行正确的操作是必须的。在叫 winedefault.reg 的文件中包含了 Wine 开发者找到的对安装应用程序是必须的键。如果你使用 tools/wineinstall 脚本则已经为你自动安装了它，如果你想手动安装它，你可以使用 regapi 工具做这件事。你可在 Wine 发布中的 documentation/no-windows 文档中找到更多的信息。

4.4.6. [registry] 段
有了以上信息，下面查看一下 wine.conf/~/.wine/config 中处理注册表的选项。

LoadGlobalRegistryFiles
控制是否尝试装载全局注册表，如果它存在的话。

LoadHomeRegistryFiles
控制是否尝试装载用户的注册表文件(在用户的主目录中的 .wine 子目录中)。

LoadWindowsRegistryFiles
控制 Wine 是否尝试从在现存的 MSWindows 安装中的真实的 Windows 注册表中装载注册信息。

WritetoHomeRegistryFiles
控制是否把注册信息写到用户的注册表文件中(目前，这是没有选择的，就是说如果你把它关闭了， Wine 根本就不能把注册表保存到磁盘上；你退出 Wine 之后，你的变动就消失了。)

UseNewFormat
这个选项被废弃了。Wine 现在总是使用新格式；前些时候去除了对旧格式的支持。

PeriodicSave
如果设置这个选项为一个非零的值,它指定你想以一个给定的时间间隔把注册表保存到磁盘上。如果你未设置它，则只在 wineserver 终止的时候把注册表保存到磁盘上。

SaveOnlyUpdatedKeys
控制是把整个注册表保存到用户的注册表文件中,还是只保存用户实际上变更了的子键。考虑到用户的注册表将屏弃任何全局注册表文件和 Windows 注册表文件,通常应该只保存用户修改了的子键；这种方式下,对全局或 Windows 注册表其余部分的变动仍可以影响这个用户。

编写：Petr Tomasek <tomasek@etf.cuni.cz>;Nov 14 1999

修改：Andreas Mohr <am ohr@codeweavers.com>;Jan 25 2000

(提取自 wine/documentation/cdrom-labels)

直到不久前，你只可能在 wine config 文件中通过手动设置来指定驱动器卷标和系列号。现在，wine 也可以直接从驱动器读取它们。对在 CD-ROM 上发布的许多 Win 9x 游戏和安装程序这是很有用的，它们检查卷标。

4.5.1. 支持什么?

文件系统 类型 注释
FAT 系统 硬盘、软盘 读取标签和系列号
ISO9660 光盘 只读取标签

4.5.2. 如何设置?
如果你在 ~/.wine/config 文件的[Drive X]段中指定了一个 Device= 行，则自动的读取标签和系列号。注意如果你这样设置它，则这个设备必须存在和可以访问。

如果你不这样做，则你应该在 ~./wine/config 中给出固定的 "Label"= 或 "Serial"= 条目，如果没有给出设备则 Wine 返回这些条目。如果它们不存在，则 Wine 将返回缺省值(标签 DriveX 和系列号 12345678)。

如果你给出一个 "Device"= 条目只是为了原始扇区访问,而不从这个设备读取卷信息(例如，你希望有一个固定的，预先配置的标签)，则你需要指定 "ReadVolInfo"= "0" 来告诉 Wine 跳过卷读取。

4.5.3. 例子
这里是光盘和软盘一个例子；从光盘和软盘二者的设备上读取标签；只从软盘上读取系列号:

```
[Drive A]
"Path"= "/mnt/floppy"
"Type"= "floppy"
"Device"= "/dev/fd0"
"Filesystem"= "msdos"

[Drive R]
"Path"= "/mnt/cdrom"
"Type"= "cdrom"
"Device"= "/dev/hda1"
"Filesystem"= "win95"
```

下面是屏弃 CD-ROM 标签的一个例子:

```
[Drive J]
"Path"= "/mnt/cdrom"
"Type"= "cdrom"
"Label"= "X234GCDSE"
;注意这里的这个设备不是真的需要有一个固定的标签
"Device"= "/dev/cdrom"
"Filesystem"= "msdos"
```

4.5.4. 要做/开放要点

只有光盘是 iso9660 和光盘标签驻留在第一轨道上时才可以读取它。

最好检查 FAT 超级块(现在只检查一个字节)。

支持标签/系列号写。

标签可以长于 11 个字符?(iso9660 有 32 个字符)。

读取 ext2 卷标如何?....

HonestQiao 回复于：2005-09-13 14:19:33

4.6. Dll 加载
编写：Ove Ken <ovek@winehq.com>;

(提取自 wine/documentation/dll-overrides)

wine.conf 指令(directive)[DllDefaults]和 [DllOverrides]是有些混乱的主题。这些指令的最终目的是足够清楚的，通过 - 给出一个选择，Wine 应该使用自己的内置 DLL，或者应该使用在现存的 Windows 安装中找到的 .DLL 文件? 本文档将解释这些特征是如何工作的。

4.6.1. DLL 类型

native
一个"固有的"DLL 是为真实的 MicrosoftWindows 写的一个 .DLL 文件。

builtin
一个"内置的"DLL 是一个 WineDLL。它们可以要么是 libwine.so 的一部分，要么是在新近的版本中，在 Wine 在需要的时候可以装载的一个特殊的 .so 文件。

elfdll
一个"elfdll"是有着一个特殊的 Windows 式样的文件结构的一个 Wine .so 文件，它尽可能的与 Windows 相接近，并且通过使用特殊的 ELF 装载器和连接器技巧，可以无缝的与"固有的"DLL 进行动态连接。Bertho Stultiens 正在做这项工作，但这个特征还没有合并到 Wine 中(因为政治上的原因和缺乏时间)，所以这个 DLL 类型此时在官方的 Wine 中不存在，"内置的"DLL 类型获取了 elfdll 的一些特征(比如动态装载)，所以"elfdll"的功能可能即将融入"内置"类型中。

so
一个固有的 Unix .so 文件，加上在装载库时生成的(?on the fly)调用惯例转换 thunk。(with calling convention conversion thunks generated on the fly as the library is loaded)对于"glide"这样的在 Windows 和 Unix 二者上使用相同的 API 的库，这是非常有用的。

译注：历史上，在实现 Algol60 传名调用(call by name)的时候，把实际参数做为一个无参数的子程序来对待，传统上把这个无参数的子程序叫做 thunk。

4.6.2. [DllDefaults]段

DefaultLoadOrder
如果正在处理的 DLL 未在 [DllOverrides]段中找到，这个选项指定 Wine 应当以什么次序查找可获得的 DLL 类型。

4.6.3. [DllPairs]段
有的时候，在缺省的配置文件中有一个叫 [DllPairs]的段，它已经被废弃了，因为组对信息已经被嵌入到 Wine 自身中了。(本段的目的只不过是如果用户尝试组对(pair codependent)不同类型的 16-bit/32-bit DLL 则发起警告。) 如果你的 wine.conf 或 ~/.wine/config 中仍然有它，你删除它是安全的。

4.6.4. [DllOverrides]段
本段指定如何处理特定的 DLL，特别是，如果你从一个真实的 Windows 配置中得到一些"固有的"DLL，那么就要在这里指定是否使用它们。因为内置的 DLL 仍不能与固有的 DLL 无缝的混合，特定的 DLL 依赖可能有问题，但在 Wine 中对许多流行的 DLL 配置存在着工作项目(workaround)。参见 WWN 的 [16]状态页来找出你要用的 DLL 是否在 Wine 中被实现了。

当然也可以通过显式的使用 Wine 的 --dll 命令行选项来屏弃这些设置(详情参见手册页)。下面是对选择你的最优的配置的提示(列出 16/32-bit DLL 对):

krnl386,kernel32
它们的固有版本永远不能工作，所以不用尝试了。保持为 builtin。

gdi,gdi32
图形设备接口。尝试运行固有的 GDI 没什么作用。保持为 builtin。

user,user32
窗口管理和标准控件。有时可能要使用 Win95 的 native 版本(如果依赖于它的所有其他 DLL，比如 comctl32 和 comdlg32，也运行 native 版本的话)。但是，在地址空间独立(Address Space Separation)之后就不可能了，所以保持为 builtin。

ntdll
NT 内核 API。尽管没有很好的编制文档，它的 native 版本是永远不能工作的。保留为 builtin。

w32skrnl
Win32s (在 Win3.x 中)。它的 native 版本可能是永远不能工作的。保留为 builtin。

wow32

NT 的 Win16 支持库。它的 native 版本可能是永远不能工作的。保留为 builtin。

system
Win16 内核材料。它的 native 版本是永远不能工作的。保留为 builtin。

display
显示器驱动程序。明确的保留为 builtin。

toolhelp
工具帮助器例程。这很少出问题。保留为 builtin。

ver,version
版本。很少有用和处理它(mess with)。

advapi32
注册表和安全特征。它的 native 版本是否工作是两可的。

commdlg,comdlg32
通用对话框，比如颜色选择器、字体对话框、打印对话框、打开/保存对话框，等等。尝试 native 版本是安全的。

commctrl,comctl32
通用控件。它们是工具条、状态条、列表控件、等。尝试 native 版本是安全的。

shell,shell32
Shell 接口(桌面、文件系统、等)。它是 Windows 中未编制文档最严重的部分之一，你可能走运的能使用它的 native 版本，如果你需要的话。

winsock,wsock32
Windows 套接口。它的 native 版本不能在 Wine 下工作，所以保留为 builtin。

icmp
给 wsock32 的 ICMP 例程。如同 wsock32，保留为 builtin。

mpr
由于 thunking 要点，它的 native 版本可能不工作。保留为 builtin。

lzexpand,lz32
Lempel-Ziv 压缩。 Wine 的 builtin 版本应当工作的很好。

winaspi,wnaspi32
高级 SCSI 外设接口。它的 native 版本可能不工作。保留为 builtin。

crtdll
C 运行时库。它的 native 版本很容易的比 Wine 的版本工作的好。

winspool.drv

打印机缓冲池。 你好象没有那么走运使用它的 native 版本。

ddraw
DirectDraw/Direct3D (直接绘制/直接三维)。因为 Wine 没有实现 DirectX HAL，现在它的 native 版本不能工作。

dinput
DirectInput(直接输入)。它的 native 版本是否工作是两可的。

dsound
DirectSound (直接声音)。可能运行它的 native 版本，但不要依仗它。

dplay/dplayx
DirectPlay (直接播放)。它的 native 版本应该工作的非常好，如果是完全的话。

mmsystem,winmm
多媒体系统。它的 native 版本好象不能工作。保留为 builtin。

msacm,msacm32
音频压缩管理器。如果你把 msacm.drv 设置为相同的，它的 builtin 版本工作的非常好。

msvideo,msvfw32
Windows 视频。 可以安全的(和推荐)尝试 native 版本。

mcicda.drv
CD 音频 MCI 驱动程序。

mciseq.drv
MIDISequencerMCI 驱动程序(MID 回放)。

mciwave.drv
Wave 音频 MCI 驱动程序(WAV 回放)。

mciavi.drv
AVIMCI 驱动程序(AVI 视频回放)。最好使用 native 版本。

mcianim.drv
Animation MCI 驱动程序。

msacm.drv
音频压缩管理器。设置为与 msacm32 相同。

midimap.drv
MIDIMapper。

wprocs
这是 Wine 用于 thunking 目的的一个伪装 DLL。它的 native 版本不存在。
[/color]

4.7. 键盘
编写：Ove Ken <ovek@winehq.com>;

(提取自 wine/documentation/keyboard)

现在 Wine 需要知道你的键盘布局(layout)。这个要求来自一些应用程序的需求，它们需要获得正确的键盘扫描码，原因是它们直接读取这些扫描码，而不是接受从 X 服务器返回的字符。这意味着 Wine 现在需要有一个从 X 键到这些程序所需要的扫描码的映射。

在启动的时候，Wine 尝试着识别活跃的 X 布局，方法是查看它是否匹配任何定义的表。如果是，所有的事情都正常。如果不是，你需要定义它。

要想定义它，打开文件 windows/x11drv/keyboard.c 并查看现存的表。复制它做为一个备份，特别是在你不是使用 CVS 的时候。

你实际上需要做的是找出每个键需要生成的那个扫描码。在 main_key_scan 表中查看，它看起来如下:

```
static const int main_key_scan[MAIN_LEN]=
{
/* 这是我的 (102-键) 键盘布局，如果不匹配你的键盘是很遗憾的 */
0x29,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B,0x0C,0
x0D,
0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,0x1B,
0x1E,0x1F,0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x2B,
0x2C,0x2D,0x2E,0x2F,0x30,0x31,0x32,0x33,0x34,0x35,
0x56 /* 第 102 个键(实际上在 l-shift 的右边)*/
};
static const char main_key_US_phantom [MAIN_LEN][4]=
{
"` ~","1 !","2 @ ","3 #","4 $","5 %","6 ^ ","7 & ","8 *","9 (","0 )","- _","= + ",
"qQ ","wW ","eE ","rR ","tT ","yY ","uU ","iI ","oO ","pP ","[ {","] }",
"aA ","sS ","dD ","fF ","gG ","hH ","jJ ","kK ","lL ","; :","' \"","\\ |",
"zZ ","xX ","cC ","vV ","bB ","nN ","m M ",", <",". > ;","/ ?",
"< > ;" /* 幽灵键 */
};
```

接着，把印在按键上的字符赋予每个扫描码。这为 US 101 键键盘(的次序)做的。它可以在 keyboard.c 的顶部找到。它还显示了如果没有第 102 键，你可以跳过它。

但是，对于多数国际化的 102 键键盘，我们使它易于你的使用。这些键盘的扫描码布局已经非常匹配在 main_key_scan 中的物理键盘布局了，所以你要做

的所有事情就是完成在你主键盘上生成字符的所有的键(除了空格键之外)，并把它们组织到一个适当的表中。只有第 102 个键是个例外，它通常在最后一行的第一个字符的左边(通常是 Z)，它必须放到在最后一行之后的单独一行中。

例如，我的挪威(Norwegian)键盘看起来如下

```
?!"#?% & / ()= ?` Back-
|1 2@ 3?4$ 5 6 7{8[9]0}+ \?space

Tab Q W E R T Y U IO P?^
■
Enter
Caps A S D F G H JK L??*
Lock'

Sh->;Z X C V B N M ;:_Shift
ift< ,.-

CtrlAltSpacebarAltGrCtrl
```

注意第 102 个键，它是 <>; 键，在 Z 的左边。在主字符右侧的那个字符是由 AltGr 键生成的字符。

定义这个键盘如下:

```
static const charm ain_key_NO [MAIN_LEN][4]=
{
"|?,"1!","2\"@ ","3#?,"4?","5%","6&","7/{","8([","9)]","0=}","+?","\\?,
"qQ ","wW ","eE","rR","tT","yY","uU ","iI","oO ","pP","连","╚~",
"aA","sS","dD","fF","gG","hH","jJ","kK","lL"," ","嫫","*",
"zZ","xX","cC","vV","bB","nN","m M ",",;",".:","-_",
"<>;"
};
```

除了 " 和 \ 需要用反斜杠引用起来，和第 102 个键在单独的一行之外，它是非常直接的。

你写完了一个这样的表之后，你需要把它添加到 main_key_tab[] 布局索引表中。这看起来如下:

```
static struct {
WORD lang,ansi_codepage,oem _codepage;
const char (*key)[MAIN_LEN][4];
}main_key_tab[]= {
...
```

```
...
{MAKELANGID(LANG_NORWEGIAN,SUBLANG_DEFAULT),1252,865,
&main_key_NO},
...
```

在你增加了这个表之后，重新编译 Wine 并测试它的工作。如果检测你的表失败，尝试运行

```
wine --debugmsg +key,+keyboard >;& key.log
```

并查看结果的 key.log 文件来找到关于你的布局的错误消息。

注意 LANG_* 和 SUBLANG_* 定义在 include/winnls.h 中，你可能需要用它找出给你的语言分配的编号，并在调试消息输出中找到它。这个编号是 (SUBLANG *0x400 + LANG)，所以，例如 LANG_NORWEGIAN (0x14) 和 SUBLANG_DEFAULT (0x1) 的组合将是 (十六进制的)14 + 1*400 = 414，因为我是挪威人，我将在调试消息输出中查找 0414 以便找出为什么不能检测到我的键盘。

一旦它工作了，请提交到 Wine 计划。如果你使用 CVS，你需要在你的主 Wine 目录中做

```
cvs -z3 diff -u windows/x11drv/keyboard.c >;layout.diff
```

，接着把 layout.diff 提交到 <wine-patches@winehq.com>；并加上关于它是什么的一个简要的说明。

如果你不使用 CVS，你需要做

```
diff -u the_backup_file_you_made windows/x11drv/keyboard.c >;
layout.diff
```

并按上面解说的那样提交。

如果你做的正确，它将被包括到下一次 Wine 发行中，而所有使用扫描码的有问题的应用程序(尤其是远程控制应用程序)和游戏将荣幸的使用你键盘布局，并且你将不再得到这些闹心的 fixme 消息了。

祝你好运。

4.8. 处理字体
4.8.1. 字体
编写：Alex Korobka <alex@aikea.msunysb.edu>；

(提取自 wine/documentation/fonts)

Note:Wine 包含了 fnt2bdf 实用工具。可以在 tools 目录中找到它。可以在 wine headquarters:http://www.winehq.com/tools.html 找到到本文档中提及的工具的链接。

4.8.1.1.如何转换 Windows 字体
如果你要访问一个 Windows 安装，你应该使用 fnt2bdf 实用工具(可在 tools 目录中找到)来把位图字体(VGASYS.FON、SSERIFE.FON、和 SERIFE.FON)转换成 X 窗口系统可以识别的格式。

用 fnt2bdf 提取位图字体。

使用 bdftopcf 把第一步生成的 .bdf 文件转换成 .pcf 文件。

把 .pcf 文件复制到字体服务器目录中，它的目录通常是 /usr/lib/X11/fonts/misc (你可能要有超级用户的特权)。如果你想建立一个新字体目录，则你必须把它添加到字体路径上。

为把字体复制到其中的那个目录运行 mkfontdir。如果你已经在 X 中了，你应该运行 xsetfp rehash 来使 X 服务器使用这个新字体。

编辑 ~/.wine/config 文件来删除给你刚安装的字体的别名。

WINE 运行可以不需要这些字体，但感观可能非常不同。还有，一些应用程序尝试装载它们定制的字体(on the fly)(WinWord 6.0)，并且因为 WINE 仍未实现这个，它转而输出象下面这样的一些东西；

STUB:AddFontResource(SOMEFILE.FON)

你也可以转换这个文件。注意这个 .FON 文件可能不持有任何位图字体，而 fnt2bdf 在这种情况下会失败。还要注意尽管上述消息不会消失，WINE 通过使用你从 SOMEFILE.FON 提取的字体绕过(work around)问题。fnt2bdf 将只适用于 Windows 3.1 字体。它不适用于 TrueType 字体。

怎样处理 TrueType 字体? 有很多商业字体工具可以把它们转换成 Type1 格式但结果的字体是远离主流的 (stellar)。使用它们的其他方式是使用有呈现

(render) TrueType 能力的字体服务器(Caldera 有一个，在 sunsite 和它的镜像的 Linux/X11/fonts 中有自由的 xfstt，如果你在 FreeBSD 上你可以使用 /usr/ports/x11-servers/Xfstt 中的 port。还有一个 xfsft，它使用 freetype 库，参见 documentation/ttfserver)。

但是，将来有可能通过 FreeType 呈现器支持固有 TrueType(hint,hint 

4.8.1.2. 如何向 ~/.wine/config 添加字体别名
许多 Windows 应用程序假定总是存在最初的 Windows 3.1 发布中包含的字体。缺省的 Wine 建立许多把它们映射成现存的 X 字体的别名:

Windows 字体 ...被映射成... X 字体
"MS Sans Serif" -> ;"-adobe-helvetica-"
"MS Serif" -> ;"-bitstream -charter-"
"Times New Roman" -> ;"-adobe-times-"
"Arial" -> ;"-adobe-helvetica-"

没有给 "System " 字体的别名。还有，对应用程序在运行时安装的字体不建立别名。建议的处理这个问题的方式是转换缺少的字体(参见前面)。如果证明这是不可能的，如在 TrueType 字体的情况下，你可以通过向 [fonts] 添加一个别名强迫字体映射器选择一个接近的相关 X 字体。确保 X 字体实际上存在(使用 xfontsel 工具)。

AliasN = [Windows 字体],[X 字体]<, 可选的 "屏蔽 X 字体" 标志>;

例子:

Alias0 = System ,--international-,subst
Alias1 = ...
...

注释:

在序列 {0,...,N} 中不能有间隙(gap)，否则不读在第一个间隙之后的所有别名。

通常字体映射器以下列方式把 X 字体名转换成 Windows 程序可见的字体名字:

X 字体 ...被展示为... 提取的名字
--international-... -> ;"International"
-adobe-helvetica-... -> ;"Helvetica"
-adobe-utopia-... -> ;"Utopia"
-misc-fixed-... -> ;"Fixed"
-... -> ;
-sony-fixed-... -> ;"Sony Fixed"
-... -> ;

注意因为 -misc-fixed- 和 -sony-fixed- 是不同的字体，Wine 修改第二个提取的名字来确保 Windows 程序可以区分它们，原因是在字体选择对话框中只出现提取的名字。

"屏蔽"别名替换最初的提取的名字，所以在这个例子的情况下我们将有下列映射:

X 字体 ...被映射成... 提取的名字
--international-... -> ;"System "

"非屏蔽"别名对用户是透明的，他们不替代提取的名字。

当对一个别名可获得一个固有 X 字体的时候，Wine 丢弃这个别名。

如果你不访问在第一段落中提及的 Windows 字体，你应该尝试用非屏蔽别名替换"System "字体。 xfontsel 应用程序可向你展示 X 可获得的字体。

Alias..= System ,...bold fontwithoutserifs

还有，一些 Windows 应用程序要求字体而不指定这个字体的字型名字。在多数 Windows 安装中字体表以 Arial 为开始，但是 X 字体表以在 fonts.dir 中的第一行的字体为开始。所以 WINE 使用下列条目来确定首先检查哪个字体。

例子:

Default= -adobe-times-

注释:

最好使可缩放的字体家族(包括粗体和斜体)成为缺省的选择，原因是映射器检查所有可获得的字体，直到完美的匹配了要求的高度和其他属性或者到达了 字体表的结束。典型的 X 安装在 ../fonts/Type1 和 ../fonts/Speedo 目录中有可缩放的字体。

4.8.1.3. 如何管理一个缓存的字体矩阵
WINE 在 ~/.wine/.cachedmetrics 文件中存储可获得的字体的详情。你可以把它复制到其他地方并向 ~/.wine/config 中的 [fonts]段添加这个条目:

FontMetrics = <file with metrics>;

如果 WINE 检测到在 X 字体配置中的变动,它将从头重建字体矩阵并用新信息重写 ~/.wine/.cachedmetrics 。这个过程要持续一会。

4.8.1.4. 太小或太大的字体
Windows 程序可以要求 WINE 呈现有用点指定的高度的一个字体。但是，点-到-象素的比率依赖于你的显示器的真实的物理大小 (15″、17″、等...)。X 尝试着提供一个估计的比率，而可能与你的实际大小有很到不同，你可以通过向 [fonts]段添加下列条目来选择这个比率:

Resolution = <integervalue>;

一般的，更高的数给你更大的字体。尝试实验 60 - 120 范围的值。96 是好的起点。

4.8.1.5. 启动时的 "FONT_Init: failed to load ..." 消息
最常见的情况是在你的字体目录之一当中有一个破碎的 fonts.dir 文件。你需要重新运行 mkfontdir 来重建这个文件。详情参见它的手册页。如果你因为不是 root 而不能在机器上运行 mkfontdir，使用 xset -fp xxx 来删除破碎的字体路径。

4.8.2. 设置一个 TrueType 字体服务器
编写：???

(提取自 wine/documentation/ttfserver)

依从下列指示来在你的系统上设置一个 TrueType 字体服务器。

获得 freetype-1.0.full.tar.gz

阅读文档，解包，配置和安装

测试库，比如 ftview 20 /dosc/win95/fonts/times

获得 xfsft-beta1e.linux-i586

安装它并在启动时开始它，比如在一个 rc 脚本中。参见 xfs 的手册页。

依从 <william c@ dai.ed.ac.uk>；给出的提示

我是从 http://www.dcs.ed.ac.uk/home/jec/progindex.html 得到 xfsft。我总是运行它。下面是我的 /usr/X11R6/lib/X11/fs/config:

```
clone-self = on
use-syslog = off
catalogue = /c/windows/fonts
error-file = /usr/X11R6/lib/X11/fs/fs-errors
default-point-size = 120
default-resolutions = 75,75,100,100
```

明显的，/c/windows/fonts 是我的 Win95 C：驱动器上 Windows 字体所在的地方；对于 Win31 是 /mnt/dosC/windows/system 。在 /c/windows/fonts/fonts.scale 中我有

```
14
arial.ttf
-monotype-arial-medium-r-normal--0-0-0-0-p-0-iso8859-1
arialbd.ttf -monotype-arial-bold-r-normal--0-0-0-0-p-0-iso8859-1
arialbi.ttf -monotype-arial-bold-o-normal--0-0-0-0-p-0-iso8859-1
ariali.ttf
-monotype-arial-medium-o-normal--0-0-0-0-p-0-iso8859-1
cour.ttf
-monotype-courier-medium-r-normal--0-0-0-0-p-0-iso8859-1
courbd.ttf
-monotype-courier-bold-r-normal--0-0-0-0-p-0-iso8859-1
courbi.ttf
-monotype-courier-bold-o-normal--0-0-0-0-p-0-iso8859-1
couri.ttf
-monotype-courier-medium-o-normal--0-0-0-0-p-0-iso8859-1
times.ttf
-monotype-times-medium-r-normal--0-0-0-0-p-0-iso8859-1
timesbd.ttf
-monotype-times-bold-r-normal--0-0-0-0-p-0-iso8859-1
timesbi.ttf
-monotype-times-bold-i-normal--0-0-0-0-p-0-iso8859-1
timesi.ttf
-monotype-times-medium-i-normal--0-0-0-0-p-0-iso8859-1
symbol.ttf
-monotype-symbol-medium-r-normal--0-0-0-0-p-0-microsoft-sy
```

m bol
w ingding.ttf
-m icrosoft-w ingdings-m edium -r-norm al--0-0-0-0-p-0-m icrosoft-s
ym bol

在 /c/w indow s/fonts/fonts.dir 中我有完全相同的字体。

在 /usr/X11R6/lib/X11/XF86Config 中我有

FontPath "tcp/localhost:7100"

在其他 FontPath 行的前面。就是它了！作为一个有趣的阶段性的意外收获(by-productofcourse)，所有指定 Arial 的 web 页在 Netscape 中以 Arial 出现 ...

关闭 X 并重新启动(并调试你在设置这些事时做出的错误)。

测试，比如 xlsfont | grep arial

希望这有所帮助...[/color]

4.9. 在 W ine 中打印
在 W ine 中如何打印文档...

4.9.1. 打印
编写：Huw D M Davies < h.davies1@physics.ox.ac.uk > ;

(提取自 w ine/docum entation/printing)

在 W ine 中打印可以通过两种方式。它们二者都在 alpha 阶段。

使用一个外部的 w indow s 3.1 打印机驱动程序。

使用内置的 W ine Postscript 驱动程序(+ ghostscript 为非 postscript 打印机生成输出)。

注意现在 W inPrinter(廉价，要求宿主计算机显式的控制打印头的哑巴打印机)不能与它们的 W indow s 打印机驱动程序一起工作。不清楚它们以后是否会。

4.9.1.1. 外部打印机驱动程序

现在只有 16 bit 驱动程序可以工作(注意这包括 win9x 驱动程序)。要使用它们，添加

printer=on

到 wine.conf(or ~/.wine/config)的 [wine]段。如果 CreateDC 的驱动程序参数是一个 16 bit 驱动程序，这个选项让它继续进行(proceed)。你可能还需要添加

"TTEnable"= "0""TTOnly"= "0"

到 ~/.wine/config 的 [TrueType]段。给驱动程序接口的这个代码在 graphics/win16drv 中。

4.9.1.2. 内置 Wine PostScript 驱动程序
通过把一个驱动程序内置到 Wine 中启用 PostScript 文件打印。参见下面的安装指导。给 PostScript 驱动程序的代码在 graphics/psdrv 中。

4.9.1.3. 缓冲池
缓冲池(Spooling)是非常原始的。wine.conf 的 [spooler]段把一个端口 (比如，LPT1:)映射到一个文件上或通过一个管道映射一个命令上。例如下面的一行

"LPT1:"= "foo.ps""LPT2:"= "|lpr"

把 LPT1:映射到文件 foo.ps 而把 LPT2:映射到 lpr 命令。如果一个作业被发送到一个未列出的端口，则建立以这个端口为名字的文件，比如为 LPT3:建立一个叫 LPT3:的文件。

4.9.2. Wine PostScript 驱动程序
编写：Huw D M Davies<h.davies1@physics.ox.ac.uk>;

(提取自 wine/documentation/psdriver)

当完成了这些的过程时候就允许 Wine 生成 PostScript 文件而不需要一个外部的打印机驱动程序。应该可以通过 ghostscript 过滤输出来打印到一个非 PostScript 打印机。

4.9.2.1. 介绍
驱动程序在被建造在 Wine 当中的时候表现的如同它就是一个叫做 wineps.drv 的 DRV 文件。尽管它模仿一个 16 bit 驱动程序，但如同 win9x 驱动程序那样，它可以与 16 和 32 bit 应用程序二者一起工作。

要安装它则添加

"Wine PostScript Driver"= "WINEPS,LPT1:"

到 win.ini 的 [devices] 段和

"Wine PostScript Driver"= "WINEPS,LPT1:,15,45"

到 [PrinterPorts] 段来设置它为缺省的打印机，还要添加

"device"= "Wine PostScript Driver,WINEPS,LPT1:"

到 ~/.wine/config 的 [windows] 段和 ??? [sic]

你还需要向注册表添加特定的条目。最简单的方式是定制 documentation /psdrv.reg 的内容(见后)并使用 Winelib 程序 programs/regapi/regapi。例如，如果你把 Wine 源代码树安装在 /usr/src/wine 中，你可以使用下列命令系列:

cp /usr/src/wine/documentation/psdrv.reg ~

vi ~/psdrv.reg

编辑 psdrv.reg 的复件来适合你的要求。作为一个最小化，你必须为每个打印机指定一个 PPD 文件。

regapisetValue < ~/psdrv.reg

你需要给你想使用的(type 1 PostScript)字体的 Adobe FontMetric (AFM) 文件。你可以从 ftp://ftp.adobe.com/pub/adobe/type/win/all/afmfiles 得到它们。目录 base17 或 base35 是开始的好地方。 注意它们只是字体矩阵而不是字体本身。现在这个驱动程序不下载额外的字体，所以你只能使用载这个打印机上存在的字体。(实际上，驱动程序可以使用在 PPD 文件中列出的任何字体，对于每个字体它都有一个 AFM 文件。如果你使用的字体未在你的打印机中或在 Ghostscript 中安装，你需要使用一些方式来把字体嵌入到打印作业中或把字体下载到打印机中。还要注意仍不能在它的 DSC 注释中正确的列出要求的字体，所以依赖于这些注释来下载正确的字体到打印机的一个打印管理器可能不能正确的工作。)

接着在你的 wine.conf (或 ~/.wine/config) 中建立 [afm dirs] 段并为每个包含你要使用的 AFM 文件的目录添加下列形式的一行:

"dir<n>;" = "/unix/path/name/"

你还需要给你的打印机的一个 PPD 文件。它描述这个打印机的特定特征。比如安装了那些字体，如何手动进纸(feed)等。Adobe 在它自己的 web 站点上有许多这种文件，看一下 ftp://ftp.adobe.com/pub/adobe/printerdrivers/win/all/。 参见上面的信息来配置驱动程序使用这个文件。

要启用彩色打印机你需要把在 PPD 中的 *ColorDevice 条目设置为 true，否则驱动程序将生成灰度的输出。

注意你不需要在 wine.conf 的 [wine] 段中设置 printer=on，它启用通过外部打印机驱动程序的打印，而不影响内置的 PostScript 驱动驱动程序。

如果你走运的话现在就可以从 Wine 生成 PS 文件了！

我测试它使用了 win3.1 notepad/write、Winword6 、Origin4.0，和 32 bit 应用程序如 win98 wordpad、Winword97、Powerpoint2000，有着一定程度的成功 - 你应当可以弄出点什么东西，它可能不在适当的位置。

4.9.2.2. 要做/缺陷

驱动程序读 PPD 文件，但忽略所有约束并不让你指定你是否有额外的东西比如信封进纸器(feeder)。 你将在打印设置对话框中发现输入箱(bin)比一般选择更大。我只真正测试了在 hp4m 6_v1.ppd 文件上的 ppd 分析。

没有 TrueType 下载。

StretchDIBits 使用 level2 PostScript。

高级设置对话框。

许多功能的部分实现。

ps.c 正在变得混乱(mess)。

Notepad 开始文本经常比边距(margin)设置要左许多。但是 win3.1 pscript.drv (在 wine 下)也是这样。

可能更多...

如果你想得到帮助请与我联系这样我们可以避免重复。

Huw D M Davies <h.davies1@physics.ox.ac.uk>;

第 5 章. 运行 Wine

目录

编写：John R. Sheets < jsheets@codeweavers.com >;

5.1. 如何运行 Wine

Wine 是一个非常复杂的软件，有多种方式调整如何运行它。除了非常少的例外，你可以通过配置文件激活与命令行参数相同的设置特征。本章中，我们将简要的讨论这些 参数，并把它们与相应的配置变量相匹配。你可以调用 wine --help 命令来得到所有 Wine 的命令行参数的一个列表:

```
用法:./wine [选项] 程序名字 [参数]

选项:
--debugmsg 名字 开启和或关闭调试消息
--desktop 几何 使用给定几何的一个桌面窗口
--display 名字 使用指定的显示器
--dll 名字 启用或停用内置的 DLL
--dosverx.xx 模仿的 DOS 版本(例如，6.22)
只在与 --winverwin31 一起时有效
--help,-h 显示这个帮助信息
--language xx 设置语言(Br,Ca,Cs,Cy,Da,De,En,Eo,Es,Fi,Fr,Ga,Gd,Gv,
Hr,Hu,It,Ja,Ko,Kw,Nl,No,Pl,Pt,Sk,Sv,Ru,Wa 之一)
--managed 允许窗口管理器来管理建立的窗口
--synchronous 开启同步显示模式
--version,-v 显示 Wine 版本
--winverxxxx 模仿的版本
(win95,nt40,win31,nt2k,win98,nt351,win30,win20)
```

你可以按需要指定任何参数。典型的，你想使你的配置文件成为一个合理的缺省设置；在这种情况下，你可以运行 wine 而不用显式的列出任何选项。在少见的情况下，你可能想屏弃命令行上的特定参数。

在选项之后，你应该加上你希望 wine 去执行的文件的名字。如果可执行文件在配置文件的 Path 参数中的目录里，你可以简单的给出可执行文件的名字。但是，如果文件不在 Path 中，则你必须给出到可执行文件的完整路径(用 Windows 格式，而不是 UNIX 格式!)。例如，给出下列的一个 Path:

```
[wine]
"Path"="c:\windows;c:\windows\system;e:\;e:\test;f:\"
```

要运行 c:\windows\system \foo.exe 你可以用:

```
$ wine foo.exe
```

但是，你必须使用下面的命令运行文件 c:\myapps\foo.exe:

```
$ wine c:\myapps\foo.exe
```

最后，如果你想向你的 windows 应用程序传递任何参数，你可以把它们列在尾部，在可执行文件名之后。这样，要运行虚构的 foo.exe Windows 应用程序并加上它的 /advanced 模式参数，在--managed 模式下调用 Wine，你的命令将如下:

```
$ wine --managed foo.exe /advanced
```

换句话说，影响 Wine 的选项应当在 Windows 程序名字之前，而影响 Windows 程序的选项在它的后面。[/color]

5.2. 命令行选项

5.2.1. --debugmsg [通道]

Wine 仍不完善，并且许多 Windows 应用程序仍然不能在 Wine 下运行而没有 bug(但它们中的许多程序在本地 Windows 下运行也不能没有 bug!)。为了易于人们找出(track down)导致每个 bug 的原因。Wine 提供了许多用于窃听的调试通道。

每个调试通道在活跃的时候，将触发把日记消息显示到你调用 wine 的控制台上。你可以把消息从它重定向到一个文件中并在你有空时检查它。但是要事先警告你!一些调试通道可以生成难以置信的大量日记消息。最多产的犯罪分子(offender)有 relay，它在每次调用一个 win32 函数的时候吐出(spits out)一个日记消息，win ，它跟踪 windows 消息传递，当然还有 all ，它是所有现存的调试单一通道的一个别名。对于一个复杂的应用程序，你的调试日志文件可能很容易的就达到 1MB 和更多。依赖于你运行程序多长时间，一个 relay 经常可以生成多于 10MB 日志消息。记录日志使 Wine 减慢许多。所以除非你真的想要日记文件，否则不要使用 --debugmsg。

在每个调试通道中，你可以进一步指定一个 message class，来过滤出不同严重程度的错误。四个消息类是:trace、fixme、warn、err。

要开启一个调试通道，使用形式 class+channel。要关闭它，使用

class-channel。要在同一个 --debugmsg 选项中列出多于一个通道，用逗号分隔它们。例如，要求在 heap 通道中的 warn 类消息，你可以项下面这样调用 wine:

```
$ wine --debugmsg warn+heap program_name
```

如果你去掉了消息类，wine 将显示这个通道的所有四类消息:

```
$ wine --debugmsg +heap program_name
```

如果你想查看除了 relay 通道的所有日志消息，你可以象下面这样去做:

```
$ wine --debugmsg +all,-relay program_name
```

下面是在 Wine 中所有调试通道和类的一个主列表。在以后的版本中可能增添(或减去)更多的通道。

```
allacceladvapianimateaspiatom avifile bitblt
bitmap caretcdrom class clipboard clipping combo comboex
comm commctrlcommdlg console crtdllcursor datetime dc
ddemldraw debug debugstr delayhlp dialog dinputdll
dosfs dosmem dplay driverdsound edit elfdllenhm etafile
event execfile fixup fontgdiglobalgraphics
header heap hook hotkey icmp icon imagehlp imagelist
imm int int10 int16 int17 int19 int21 int31
io ipaddress joystick key keyboard ldt listbox listview
localmcimcianim mciavim cicdam cim idim ciwavem di
menu message metafile midim mmaux mmio mmsys mmtime
module monthcalm prm sacm msg msvideo nativefont nonclient
ntdllodbc ole opengl pager palette pidlprint
process profile progress prop propsheetpsapipsdrv ras
rebar reg region relay resource richedit scrollsegment
seh selector sendmsg server setupapisetupx shellsnoop
sound static statusbar storage stress string syscolor system
tab tape tapitask text thread thunk timer
toolbar toolhelp tooltips trackbar treeview ttydrv tweak typelib
updown ver virtualvxd wave win win16drv win32
wing wininetwinsock winspoolwnetx11 x11drv
```

关于调试通道的详情，请查看 Wine 开发者指南。

5.2.2. --desktop [几何]
缺省的,wine 在你的正规桌面上运行应用程序。Wine 应用程序与本地 X11 应用程序混合在一起。窗口相互交叠,并且你可以在相互关系中调整它们的大小。通常,当你最小化 Wine 窗口的时候,它们缩小(collapse)成在你的桌面左下角的一个小图标,躲避你的其他非 Wine 窗口的行为。但是,如果你运行在 --managed 模式中,你的 Wine 应用程序将象其他程序那样最小化。

有时,你可能要把 Wine 窗口限制于你的桌面中小一些的一个区域中。这由 --desktop 选项来控制。 当你把这个选项传递给 wine 的时候,它将建立这么大的一个窗口并作为 Wine 桌面而不再借用正规的桌面空间。Wine 将接着把应用程序窗口放置到这个新桌面窗口中。如果你最小化这个应用程序,它将在它自己的桌面窗口的左下角图标化(iconize)。

--desktop 选项的几何信息使用标准的 X11 几何格式,例如,"640x480" 是 640 象素宽和 480 象素高的一个桌面窗口。你还可以在几何中指定桌面窗口的左上角的坐标,但你的窗口管理器可能选择屏弃你的要求。下列调用将在坐标(10,25)打开一个新的 640 x 480 桌面窗口:$ wine --desktop 640x480+10+25 foo.exe

更常见的是,你去掉起点坐标,而只使用高度和宽度:$ wine --desktop 640x480 foo.exe

5.2.3. --display
缺省的,wine 在 $DISPLAY 环境变量中的那个 X 显示器上显示它的窗口。通常,$DISPLAY 被设置为 :0,它把所有窗口发送到你的当前宿主主机上的主视频监视器上。

要包窗口发送到在同一个系统上的一个不同的监视器上,你需要把 :0 变更为一个不同的数,例如,设置为 :1 来发送到第二监视器上。你还可以指定其他系统。如果你登录到一个系统 alpha,但想让 wine 在网络上的一个其他系统上运行,比如 beta,你可以使 $DISPLAY 的值为 beta:0。

你还可以在你的 wine 命令行使用 --display 选项声明你的显示器的值。上面的例子的命令行如下:

$ wine --display="beta:0" foo.exe

5.2.4. --dll

5.2.5. --dosver

5.2.6. --help

5.2.7. --language

5.2.8. --managed

5.2.9. --synchronous

5.2.10. --version

5.2.11. --winver

第 6 章. 找出和报告缺陷

6.1. 如何报告一个缺陷

编写：Gerard Patel

(提取自 wine/documentation/bugreports)

有两种方式来报告一个缺陷。其一是使用一个简单的 perl 脚本，如果你不想花很长时间来生成报告则建议你使用这种方式。它被设计来供所有人使用，从新手到高级开发者。你也可以通过困难的方式制作一个缺陷报告 -- 高级开发者可能有此偏好。

6.1.1. 简单的方式

要使这种方法工作你的计算机上必须有 perl。要找出你是否有 perl，可运行 which perl。如果它返回象 /usr/bin/perl 这样的东西，则你有可运行的 Perl。否则跳到(skip on down to)"困难方式"。如果你不确信，继续进行好了。当你运行这个脚本时，如果你没有 perl，状况是会是显而易见的。

把目录改变为 <dirs to wine>/tools

键入 ./bug_report.pl 并给随着这个目录。

向 comp.emulators.ms-windows.wine newsgroup 发送一个消息并加上 "Nice Formatted Report" 附件。如果可能的话，上载完整的调试输出到一个 web/ftp 服务器并在你的消息中提供地址。

6.1.2. 困难的方式

一些简单的建议可以使你的缺陷报告更有用(这样更容易得到回答和修理):

尽可能多的传送信息。

这意味着我们更多的信息而不是一个简单的 "我运行 MSWord 的时候它崩溃了。你知道为什么吗?" 至少包括下列信息:

你使用的 Wine 版本(运行 wine -v)

你使用的操作系统,什么发布(如果是的话),和什么版本

编译器和版本(运行 gcc -v)

Windows 版本,如果你安装了的话

你正在尝试运行的程序,它的版本号,和从中获取这个程序的一个 URL(如果可获取的话)

你启动 wine 的命令行

你认为有关的或有帮助的任何其他信息,如在 X 问题的情况下 X 服务器版本,libc 版本等。

加上 --debugmsg +relay 选项重新运行程序(比如,wine --debugmsg +relay sol.exe)。

如果在运行你的程序时 Wine 崩溃了,这些信息对于对我们找出导致崩溃的原因很重要。这可能输出大量(好多 MB)信息,所以最好输出到一个文件中。在 Wine-dbg>; 提示符出现的时候,键入 quit。

你可能想要尝试 +relay,+snoop 而不是 +relay,但是请注意 +snoop 是非常不稳定的并且经常比一个简单的 +relay 更早崩溃!如果在这种情况下,则请只使用 +relay!! 在多数情况下加上 +snoop 时的缺陷报告是没用的!

要跟踪输出请使用下列命令:

所有 shell:

```
$ echo quit | wine -debugmsg +relay [other_options] program_name
> & filename.out;
$ tail -n 100 filename.out >; report_file
```

(这将把 wine 的调试信息只打印到文件接着自动退出。使用这个命令可能是个好主意，因为 wine 打印输出太多的调试信息，它们会溢出终端，吞噬 CPU 。)

tcsh 和其他 csh 式样的 shell:

```
$ wine -debugmsg +relay [other_options] program_name |& tee filename.out;
$ tail -100 filename.out >; report_file
```

bash 和其他 sh 式样的 shell:

```
$ wine -debugmsg +relay [other_options] program_name 2>&1 | tee filename.out;
$ tail -100 filename.out >; report_file
```

report_file 将包含最后一百行调试输出，包括寄存器复制和回溯(backtrace)，这是信息中最重要的部分。即使你不理解它们的意思，也请不要删除它们。

传送你的报告到新闻组 comp.emulators.ms-windows.wine

在你的帖子中，包括第 1 部分的所有信息，并插入在第二部分中的输出文件中的文本。如果你这样作了，你收到一些有帮助的响应的机会就会很大。

6.1.3. 问题和注释

如果读了本文档之后还有一些东西搞不明白，或者认为可以解释的更好，或者是应该包括的，请向 comp.emulators.ms-windows.wine 发帖子来让我们知道如何改进这个文档。