Wine 中文用户指南

前言

在 easywine 论坛上已经找到新版的翻译文档，但是很多地方不怎么满意，我这几天正好需要这方面的知识，就对着现在的《wine user guide》翻译和参考 easywine 论坛上那篇已经翻译好的文章修改了一下，自己花了一点时间，站在前人的肩上完成的，拿出来，希望对英文不好的朋友使用 wine 时有帮助。（注：我最近需要用 wine 调试一些程序，自己英文不好，英文看着吃力，翻译成中文再看，为了以后的效率，翻译成中文的看着顺点儿）

里面有几个地方我也没有翻译好，不熟，例如 wineconsole 那里。要是发现错误的地方，请指正。

本作品依据 WineHQ 的 《wineusr-guide》（html 档）译出。所依据的该文档的版本为：
2008 年 3 月 24 日 的版本
本作品翻译完成于 2008 年 4 月 1 日

目录

生词本

表的列表

第一章 介绍

内容标签：
1.1 概览/关于
1.2 什么是 wine？
1.3 wine 的版本
1.4 wine 以外的选择

1.1 概览/关于

1.1.1 本文档的目的和目标读者

本文档，名为《Wine 用户指南》，既是一个简单的安装指南，又是一个扩展性的参考指南。本文适合新的 Wine 用户，也适合有经验的 Wine 用户。它以在文档中提供所有配置特性和支持域来提供完全的一步一步的安装和配置指导，同时也提供起重要作用的扩展性的参考资料。

1.1.2 更多问题和评论

如果在检视这份指南，FAQ，和其他相关文档后仍然有一些东西您不能指出，您可以给我们发送邮件。《邮件列表》章节包含许多邮件列表和IRC频道，它们都是不错的地方，您可以在那里获取帮助和提出建议。如果您有自己独到的见解，并确信您能更好地表述，您可以在Wine的文档本身添加一个 《bug 报告》或者《发布一个补丁》文件。

1.1.3 内容概览/采取的步骤

为了能够使用使用 Wine ,您必须首先做一个安装的工作。这个指南将帮助您把您的系统从空的 ,没有 Wine 的状态变为一个值得炫耀的新鲜的，最新 Wine 安装。
第一步，获取Wine ,举例说明多种方法来把获取的Wine的文件装到您的计算机上。
第二步，配置Wine ,展示如何依您的个人需求来配置一个Wine的安装。
最后一步，运行Wine ,包含特殊的步骤，可使一特定应用程序在Wine下更好运行，
并提供有用的连接，以便您可获取更多的帮助。

1.1.4 快速开始

安装和运行 wine 的大略过程如下:

- 从 获取wine 获取一个当前你的linux发行版的wine也可以看参考 Wine 下载页. 新用户或部分用户也可以下载使用rpm的发行版。
- 配置wine使用winecfge 命令，对大多数用户来说默认的配置已经是可以使用的了。
- 测试 wine 是否安装好 运行 wine 的 Windows 3.1 版本程序可以使用这样的命令： *wine winefile*
- 运行wine使用这样的命令： *wine filespec/appname.exe*

你在运行命令之前首先应该安装好软件，例如： *wine /media/cdrom/setup.exe* 或者同样的其他软件.

1.2 什么是 wine？

1.2.1. Windows 和 Linux

不同的程序是为不同的操作系统设计的，大多数情况下不能在其他系统上运行。例如，Windows 程序，不能在 Linux 上运行，因为它们包含有 Linux 系统不能理解的指令，除非它们被转换到 Windows 环境下。同理，Linux 程序，也不能在 Windows 操作系统下运行，因为 Windows 不能解释其所有的指令。

这个情况呈现出一个根本性的问题，在那些想要同时想要在 Windows 和 Linux 下运行软件的人面前。通常的解决方法是在同一台计算机上同时安装这两套系统，这被称为“双引导”。当需要 Windows 程序时，使用者重启并进入 Windows 来执行之。当需要 Linux 程序时，使用者重启并进入 Linux。这个方法呈现出巨大的困难：不仅是使用者必须经受经常重启的折磨，而且两个平台的程序不能同时运行。在系统上安装 Windows 也增加了额外的负担：该软件昂贵，需要独立的磁盘分区，并不能阅读多数的文件系统格式，使得在各个系统间共享数据变得困难。

1.2.2 什么是 wine，它如何能帮助我？

Wine 使得在任何“类 Unix”操作系统(特别是在 Linux)上运行 Windows 程序成为可能。在其核心，Wine 是一个 Windows 应用程序接口(API)库，作为一个 Windows 程序和 Linux 之间的桥梁。想象 Wine 是一个兼容层，当 Windows 程序尝试执行一个通常情况下 Linux 不能理解的功能(函数)，Wine 将把该程序的指令翻译成 Linux 能够理解的指令。例如，一个程序要求系统创建一个 Windows 的按钮或文本编辑文件，Wine 将其转换为以使用标准 X11 协议的视窗管理器的命令形式的 Linux 的等价物。

如果您有权获取 Windows 程序的源代码，Wine 也能用来重新编译您的程序为 Linux 能更容易理解的格式。Wine 仍然需要用来启动重新编译后的程序，但是在 Linux 里本地地编译 Windows 程序有许多优点。更多信息尽在《Winelib 使用者指南》。

1.2.3 Wine 的特性

贯穿其开发过程，Wine 已经在它的支持的特性和能运行的程序方面不断成长。下面是一个不完整的这些

特性的列表：
•支持运行 Win32(Win 95/98, NT/200/XP), Win16(Win 3.1)和 DOS 程序。
•选择性地使用外部的第三方 DLL 文件(比如包含于 Windows 中的那些)。
•基于 X11 图形显示，允许远程显示到任何 X 终端，甚至文本终端。
•整合于桌面或可混合视窗。
•游戏的 DirectX 支持。
•能很好支持各种声卡驱动，包括 OSS 和 ALSA。
•支持选择输入设备。
•打印：PostScript 接口驱动(psdrv)到标准 Unix PostScript 打印服务。
•调制解调器，串设备支持。
•Winsock TCP/IP 网络支持。
•扫描仪，CD 刻录机，及其他设备的 ASPI 接口(SCSI)支持。
•先进的 unicode 和外语支持。
•特性完全的 Wine 调试器(Wine debugger)和可配置的追踪记录信息来使问题捕捉更容易。

1.3 wine 的版本

1.3.1 从 WineHQ 来的 Wine

Wine是一个开源项目，有许多不同版本的Wine，因此，您可以从中选择。标准的Wine版本来自于断断续续的发布(大概一月两次)，可以从互联网上下载预先打包好的二进制形式和现成的用来编译的源代码形式。作为一个选择，通过使用在Git服务器上的最近的可用的源代码，您可以安装Wine的开发版。参见下一章，《获取Wine》，您可以了解更多细节。

1.3.2 Wine 的其他版本

有许多程序是以这样或那样的方式衍生于标准 Wine 代码库。它们中的一些是来自于积极捐助 Wine 项目的公司的产品。
这些产品尝试显得突出或区别于标准版本的 Wine，并提供更好的兼容性，更容易的配置，以及商业化的支持。如果您想要这类东西，考虑购买这类产品也是个好主意。

表 1-1. 各种不同的 Wine 产品

产品	描述	发布形式
CodeWeavers CrossOver Office	CrossOver Office 允许您在 Linux 中安装您喜爱的 Windows 应用程序，而不需要一个微软操作系统许可证。CrossOver 包含一个容易使用的，单击使用的界面，它使得安装 Windows 应用程序简单而快捷。	商业版；提供 30 天不限制任何功能的演示版。
CodeWeavers CrossOver Office Server Edition	CrossOver Office 服务器版允许您运行您喜爱的 Windows 办公应用程序在一个 Linux 下的发布的瘦客户端环境，而每个客户机不需要微软操作系统许可证。CrossOver Office 服务器版允许您满足字面上地数百并发的用户的需求，全部来自一个单独的服务器。	

1.4 wine 以外的选择

除了使用 wine 以外还有许多种办法来运行软件。如果你正在考虑使用 wine 来运行应用程序，当你遇到困难，你要考虑一下这些办法的可行性。

1.4.1 Native 应用程序

许多 Windows 应用程序，特别是更经常使用的，比如媒体播放器，即时通讯，以及文件共享程序都有非常好的的开源代替品。此外，相当大量的 Windows 程序已经被直接地移植到了 Linux ,完全地排除了对 Wine (或 Windows)的需要。
（译者注：这里原标题是 Native Applications,Native 这个单词可以译成原来的，原生的，但是在 wine 的使用中，我个人感觉使用 Native 这个单词表示比较好，就像后面的 builtin 程序，本文中也不将这个词翻译成中文，而依然使用 builtin 来表示。）

1.4.2 另一个操作系统

可能最显而易见的方法来使得一个 Windows 应用程序运行是简单地在 Windows 上运行它。然而，安全性，许可证费用，向后兼容性，以及机器效能问题将使这一方法变得困难，所以 Wine 是如此有用。

另外一个选择是使用 **ReactOS** ,它是一个完全开源的 Windows 之外的选择。ReactOS 很大程度上与 Wine 项目共享代码。不是在 Linux 的顶端运行 Windows 应用程序，而是在 ReactOS 内核的顶端运行之。ReactOS 也提供与 Windows 驱动程序文件的兼容性，允许使用硬件而无需起作用的 Linux 驱动程序。

1.4.3 虚拟机

您可以不在您的机器上全新地安装操作系统，而可以在软件层面运行一个虚拟机，并在其上安装一个不同的操作系统。这样，您可以运行一个 Linux 系统并同时运行 Windows 与您的应用程序一起在一个虚拟机里同时地在同一硬件上。虚拟机不仅允许您在同一硬件上安装并运行不同版本的 Windows ，而且允许您运行其他操作系统，包括 ReactOS。
有许多不同的虚拟机，而且其中一些也能在不同平台上模拟 x86 硬件。开源的 Bochs 和 QEMU 能虚拟地运行 Windows 和 ReactOS 。其他的，商业虚拟机有 VMware 和 微软的 VirtualPC 。
然而，使用虚拟机有巨大的缺点。不像 Wine,这些程序是模拟器，所以不可避免的会有实实在在的速度下降。更为重要的是，在虚拟机里运行应用程序使得应用程序不能被完全地整合于当前环境之内。例如，您不能让 Windows 系统托盘图标或应用程序快捷方式出现在您的 Linux 桌面的系统托盘图标或应用程序快捷方式的旁边。因为 Windows 应用程序必须完全地驻留于虚拟机之内。

第二章 获取 wine

内容标签：
2.1 wine 安装方法
2.2 从包安装 wine
2.3 从源码安装 wine

2.1 wine 安装方法

一旦您已经决定 Wine 正合适您的需求，下一步是决定您想怎样安装之。有 3 种方法安装来自 WineHQ 的 Wine ，每一种都有其优点和缺点。

2.1.1 用软件包包安装

目前最为容易的安装 Wine 的方法是使用预先打包好的 Wine 的版本。这个包包含可以运行的二进制文件，它们特别地为您的发行版而编译。通常它们的功能性和完整性已被打包者测试过。
包是推荐的安装 Wine 的方法。我们轻松的获取到wine在wine下载页。这些总是最近可用的包。由于受欢迎，在其他地方在官方发行版仓库都能发现 Wine 包。然而它们可能有时是过时的，取决于您的发行版。包也易于升级，有些发行版可以无缝地升级 Wine ，只需几次点击。从源代码包创建您自己的可安装的二进制包也是可能的，虽然它已经超出了本指南的范围。

2.1.2 从一个源归档安装

有时 Wine 包并不完全符合您的需求。可能它们在您的架构或发行版上不可用，或者您需要使用您自己的编译器优化选项来创建 Wine ,并关闭一些设置。或者可能您需要在编译之前修改源代码特定的部分。

作为一个开源项目，您可以对 Wine 源代码自由地做上述之事，它将随每个 Wine 版本发布。这个安装方法可以通过下载一个 Wine 源归档并从命令行编译来完成。如果您觉得做这些事很舒服并有特殊的需求，这个选择正合适您。

获取 Wine源归档是件容易的事。每个版本，我们都以 tar.gz 格式弄一个源代码包放在wine下载页。从源代码编译安装 Wine 比使用包安装要难一些，但是我们将深入讲解之并试图帮助您完成这个方法。

2.1.3 从 Git 树安装

如果您希望试验 Wine开发的最新成果 ,或者您自己想帮助开发 Wine,您可以从我们的 Git 服务器下载最近的源代码。关于从 Wine Git仓库 下载的操作说明书可以在http://www.winehq.org/site/git获得。

请注意，通常的关于使用开发版本的警告对其有效。Git服务器 上的源代码很大程度上是未经测试的，甚至不能正确地编译。但是，它是测试下一版 Wine 能如何工作的最佳途径，并且如果您要修改源代码，最好是获取最近的拷贝。Git仓库 对应用程序所有者也有用，他们可以测试应用程序是否仍然可以在下一个Wine发布下正确工作，或者近期的补丁是否使情况有所改善。如果你对帮助我们使得一个应用程序在 Wine 下工作感兴趣，请参阅 《how to》文档。

2.2 从一个包安装 wine

2.2.1 安装一个全新的包

在一个全新的系统上安装一个包是非常直截了当的。简单地下载并安装包使用任何您的发行版提供的工具。通常在安装之前不需要明确地移除旧版本，因为现代的 Linux 发行版应该能自动地升级并替换之。但是，如果你从源代码安装了 Wine，在安装一个 Wine 包之前，您应该移除它。参阅《卸载从源代码安装的 Wine》获取正确的指导。

2.2.2 不同的发行版

Wine 在巨大数量的不同的 Linux 发行版上工作，也能在其他类 Unix 系统，比如 Solaris 和 FreeBSD，每一种都有其特定的安装和管理包的方法。但是，幸运地，相同的基本思想对所有它们的都起作用，而安装 Wine 应该不比安装任何其他软件更难，不管您用的是什么发行版。卸载 Wine 包也很简单，并且在现代的 Linux 发布里通常是通过与安装包相同的易用界面来完成。

我们将不包括在各种系统打包和包管理的安装或卸载 Wine 包的方法的特点。但是，最新近的安装注记为特定发行版能够在 WineHQ 网页的 HowTo找到。如果您需要更多的关于指出如何简单地安装一个 Wine 包的帮助，我们建议您查阅您的发行版的文档，支持论坛，或 IRC 频道。

2.3 从源代码安装 wine

在你从源码安装 wine 以前，请先卸载你机器上安装的任何 wine 的安装包。安装 wine 需要使用命令行终端和 wine 的完整的源代码。当你下载源码从 Git 服务器或者从一个档归文档（即下载好的压缩包）中把源码解压出来，在命令行下按照下面的向导操作。

2.3.1 获取创建依赖关系

在 Wine 运行时，使用许多开源的库。虽然 Wine 并不严格地依赖于这些库而且能够在没有它们中的大多数时编译，但是在编译时拥有这些库，Wine 的功能性将得到提升。在过去，许多用户问题是由于人们在从源创建 Wine 时，没有必要的开发库所致；由于这个以及其他的原因，我们高度推荐通过二进制包安装，或者通过创建能够自动满足其依赖关系的源代码包安装。

如果您希望手动安装创建依赖关系，有许多种方法可以检视您是否缺少一些有用的开发库。最为直接的方法是在您编译 Wine 之前，查看 configure 程序的输出，以确定任何重要的东西是否缺少。如果是那样，简单地安装缺少的东西，然后在编译之前重新运行 configure。您也可以检查 configure 修改了的文件，(include/config.h.in)并检视是否有文件 configure 试图寻找但没有找到的开发库。

2.3.2 编译 wine

一旦您已经安装了您需要的创建依赖关系，您已经做好了编译包的准备。在终端窗口，在进入到 Wine 源代码树后，运行下面的命令：

```
$ ./configure
$ make depend
$ make
# make install（或者$sudo make install）
```

末尾的命令要求 root 权限。尽管您决不应该以 root 运行 Wine ,但将需要以这样的方式安装 Wine。

2.3.3 卸载从源代码安装的 Wine

要卸载从源代码安装的 Wine, 您需要再一次在终端里进入到您用来安装 Wine 的同一个目录。然后运行下面的命令：

```
# make uninstall
```

这个命令将需要 root 权限，并且应该从您系统上移除所有的 Wine 二进制文件。但是，它将不移除您的 Wine 配置 以及位于您用户的家目录 （主目录）里的应用程序，所以您可以自由地安装另一版本的 Wine 或者手动地删除该配置。

第三章 配置 wine

内容标签：
3.1 使用 winecfg
3.2 使用注册表和注册表编辑器
3.3 配置其他东西

最最通常的配置变更可以通过使用 winecfg 工具来达成。我们将经历一个简单的，一步一步的介绍，这个介绍是对 Winecfg 的介绍。并概要的给出可用的设置选项。在下一节我们将看到您可以通过使用 regedit 做出更高级的变更，也提供一个完全的参考给所有的 Wine 配置设定。最后，一些您可能想要配置的东西超出了 Winecfg 和 regedit 的范围，我们将看看它们。

3.1 使用 winecfg

在过去，Wine 使用一个特殊的配置文件，它能在 ~/.wine/config 中找到。如果您仍然使用一个提及这个文件的 Wine 的版本 (即 2005 年六月以前的版本)。您应该在您要做任何其他事情之前进行升级。现在所有的设定都直接地存储于注册表中并且在 Wine 启动时被 Wine 存取。

Winecfg 应该已经与 Wine 程序的其他部分被一齐安装到了您的计算机上。如果您不能指出如何启动它，尝试运行命令：
$ /usr/local/bin/winecfg

或者可能仅仅是：$ winecfg

当该程序启动了，您将注意到在窗口的上方有如下的一排标签：

•Applications (应用程序)

•Libraries (库)

•Graphics (图形)

•Desktop Intergration (外观)

•Drivers (驱动器)

•Audio (声音/音频)

•About (关于)

在 Applications 和 Libraries 标签里更改设定将最能对运行一个程序产生影响。其他的设定偏重于使

Wine 本身以您希望的方式运转。

注意：*Applications (应用程序), Libraries (库),* 和 *Graphics (图形)* 标签是联系在一起的！ 如果您在 Applications (应用程序) 标签里选择了 Default Settings(默认设定)，在 Libraries (库), 和 Graphics (图形)标签里的所有更改将被变更为对所有的应用程序生效。如果您已经在 Applications (应用程序)标签里配置了一个特定的应用程序并且选择了它，那么在 Libraries (库), 和 Graphics (图形)里所做出的更改将仅仅影响该应用程序。这样允许定制设定给特定的应用程序。

3.1.1 应用程序设定

Wine 有能力模仿不同版本的 Windows 的运转。一般情况下，最大的区别是 Wine 以 Win9x 版本 或 NT 版本 方式运转。一些应用程序要求特定的运转模式以运行，而更改此设置可能使一个运行有错误的应用程序工作。最近，Wine 的默认 Windows 版本变成了 Windows 2000。人所共知，如果您选择 Windows 98 ，许多应用程序将工作得好些。
在该标签内您将注意到有一个 *Default Settings （默认设定）*入口。如果您选择之，您将看见给所有应用程序的当前默认 Windows 版本。一个问题多多的应用程序最好在默认设定之外单独地配置。要这么做：
1 点击 *Add application （添加应用程序）*按钮。
2 浏览直到您定位了该 .exe 文件。
3 当它被添加后您可以选择特定的 Windows 版本，Wine 将为该程序模拟之。

3.1.2 库设定

同样，一些应用程序需要特定的库 以便运行。Wine 重新制造了 Windows 系统库--即所谓的 Native DLL 和完全地定制版本设计来以完全相同的方式作用而不需要来自微软的许可证。Wine 的 built-in 版本 有许多广为人知的不足，但是许多情况下其功能性是足够的。仅使用 built-in DLL 确保您的系统是 非微软的。但是，Wine 有能力载入 native Windows DLL。

3.1.2.1 DLL Overrides

（译者注：为了简单，我们这里只要记住 built-in 就是属于 wine 的 dll，native 就是 windows 系统带过来的 dll，这样简单的理解认为就够了）

不是总能使用 built-in DLL 运行一个应用程序的。有时 native DLL 简单地工作得更好。在你已定位了一个在一个 Windows 系统上的 native DLL 后，您将需要把它放在一个合适的地方以便 Wine 找到它，而后配置使它被使用。一般说来您需要把它放在一个您已经配置为 c:\windows\system32 的目录 （获取更多信息在 驱动 节）。有四个 DLL 您应该决不应该尝试使用 native 的版本：kernel32.dll, gdi32.dll, user32.dll, 和 ntdll.dll。这些库要求低级的 Windows 内核存取而其不存在于 Wine 中。请记住。

当你已经拷贝了 DLL 后要告诉 Wine 尝试使用它。您可以配置 Wine 来在 native DLL 和 built-in DLL 之间选择在两个不同的级别。如果您有 *Default Settings（默认设定）*选择在 *Applications（应用程序）*标签，您所做的改变将影响所有的应用程序。或者，您可以通过在 Applications （应用程序）标签里改写全局设定在“每一个应用程序” 级别。

要添加并 override FOO.DLL，键入“FOO”在标有 *New override for library (新的库的 override)* 标签的盒子里：并点击 *Add (添加)* 按钮。要改变 DLL 如何运转，选择它在 Existing overrides（存在的 overrides）: 盒子和选择 Edit（编辑）。您也可以选择 native only（仅本地），builtin only（仅内建），或全部关闭。

3.1.2.2 关于系统 DLL 的助记

Wine 团队已经决定要创建假的 DLL 文件来欺骗许多程序，这些程序要检查文件是否存在以决定某功能是否可用（比如 Wincosk 和 TCP/IP 网络）。如果这对你来说是一个问题，您可以创建空文件于配置的 c:\windows\system32 目录来使得程序 认为 DLL 在那里，而 Wine 的 内建 DLL 将被载入当程序真的需要它时。（很不幸，tools/wininstall 自己并不创建这些空文件。）

应用程序有时也从物理文件检查版本资源（例如，确定 DirectX 的版本）。空文件在这种情况下不起作用，有必要安装完全版本的资源的文件。这个问题正在解决中。与此同时，您仍然需要抓一些真的 DLL 文件来让这些应用程序检验不出来。

当然有 Wine 当前没有很好（或根本没有）实现的 DLL。如果您没有真的 Windows 可以从中拷贝必要的 DLL，您总是可以从一个 Windows DLL 归档站点得到一些，这些站点可以从 Internet 搜索引擎搜索得到。请确保遵守任何您抓到的 DLL 的许可证；有些是可再分发的，而有些不是。

3.1.2.3 缺失的 DLL

万一 Wine 抱怨缺失一个 DLL，您应该检查是否该文件为一个 公共可用 DLL 或一个属于您的程序的 客户 DLL（通过在 Internet 上搜索其名字）。在您已定位该 DLL 后，您需要确保 Wine 能够使用之。DLL 通常在下述位置被载入，并依从下述顺序：

1、程序被启动的目录。
2、当前目录。
3、Windows system 目录。
4、Windows 目录。 （即 Windows 根目录）
5、PATH 变量目录。

简单地说：要不把要求的 DLL 放到您的程序目录（可能显得丑陋）,要不就放到 Windows 的系统目录。另外，如果可能，您可能不应该使用基于 NT 的 native DLL，因为 Wine 对 NT API 的支持比它对 Win9x API 支持要弱（可能导致比没有配置 Windows 的更差的与 NT DLL 的兼容性）。

3.1.3 图形设定

有基本地五个不同的图形设定您可以配置。对大多数人来说默认值就很好了。

首先一些设定主要影响游戏，并有些明显。您能够阻止鼠标从一个 DirectX 程序（例如，一个游戏）的窗口中离开，其默认值是那个 box 被勾选。有许多的原因使得您可能想那样做，至少包括：如果把鼠标配

置为局限在一个更小的区域，玩游戏会更加容易；另一个打开此选项的另一个原因，是为了更精确的控制鼠标。Wine 偏移鼠标的位置，来模拟 Windows 鼠标的运作方式 。相似地，"desktop double buffering"（桌面双倍缓冲，指的是"渲染的时候并不直接写前台，而是首先在后台一个缓冲中渲染，然后交换缓冲，可以获得比较稳定的动画图像的写屏技术"）它允许更平滑的屏幕刷新，游戏可以从中获益，默认值是打开之。作为交换，内存使用量将增加。

您可能会发现 *Emulate a virtual desktop（模拟一个虚拟桌面）* 非常有用。在这个情况下，所有程序将在一个独立的窗口中运行。您可能会发现这个非常有用，特别是在测试有错误的游戏（可能不成功地）改变屏幕分辨率时。限制他们在一个窗口中能允许对他们的更多控制以减少实际可能的费用。您可能想要尝试的分辨率大小是 640x480 (默认值) 或 800x600。

最后，您可以配置一些 Direct3D 设定。在大多部分这些设定是自动探测的，但是您可以强制它们以一个特定的方式运行。有些游戏探测潜在的系统来检视它是否支持特殊的某些特性。通过关闭这些 Wine 将不会报告能力用一个某一方式呈报游戏。这将导致游戏运行得更快以图形的质量的费用或者图形或游戏根本不能运行。

3.1.4 驱动设定

Windows 要求一个相当严格的启动器配置，Wine 模拟之。大多数人熟悉该标准的符号："A:" 驱动器代表软盘，"C:" 驱动器代表主系统盘，等等。Wine 使用相同的概念并将这些驱动器映射到当前的本地文件系统上。

Wine 的驱动器配置相对地简单。在 Winecfg 的 *Drives* 标签您将看见用来添加和移除可用的驱动器的按钮。一个新的 entry 将会被制作并且一个默认的驱动器 mapping 将会出现。您可以在 *Path:* 框里改变这些驱动器指向哪里。如果您不确定具体的路径，您可以选择"Browse"（浏览）并寻找之。移除一个驱动器就简单得只需要选择之并点击"Remove"（移除）。

Winecfg 能自动地探测您系统上可用的驱动器。推荐您在尝试手动配置驱动器前使用之。简单地点击"*Autodetect"（自动探测）* 按钮来使 Wine 寻找您系统上的驱动器。

也许您有兴趣在 Winecfg 之外配置您的驱动器设定。幸运地，这种情况也很简单。所有的驱动器设定居于一个特殊的目录~/.wine/dosdevices。每个"驱动器"简单地是一个到其真实所在的连接。Wine 自动地建立 2 个驱动其在您首次运行它的时候：

```
$ ls -la ~/.wine/dosdevices/
lrwxrwxrwx   1 wineuser wineuser     10 Jul 23 15:12 c: -> ../drive_c
lrwxrwxrwx   1 wineuser wineuser      1 Jul 23 15:12 z: -> /
```

要添加其他驱动器，例如您的 CD-ROM，创建如下连接即可： $ ln -s /mnt/cdrom ~/.wine/dosdevices/d: 请注意给连接使用的 DOS-风格 命名习惯——其格式是一个字母后接一个冒号，诸如 "a:"。所以如果您连接您的 C: 驱动器指向 ~/.wine/drive_c，您可以认为说 c:\windows\system32 即是说 ~/.wine/drive_c/windows/system32 的意思。

3.1.5 音频设定

Wine 可以使用几种不同的音频子系统工作，您可以在“Audio”（音频/声音）标签下选择他们。“Autodetect”（自动探测）按钮能为你把它全部配置出来，或者您可以手动地选择一个驱动。老一点的 Linux 发行版使用 2.4 内核或更早的版本典型地使用“OSS”（开放声音系统）。新一点的 2.6 内核已经转换到了“ALSA”（Advanced Linux Sound Architecture，先进 Linux 声音架构）。“aRts”（the Analog Realtime Synthesizer，模拟[类比]实时合成器）无效了因为最近的维护不足，如果您正在使用 GNOME 您或许也可以使用“EsounD”（The Enlightened Sound Daemon，Enlightened 桌面管理器的声音守护进程）。OSS 和 ALSA 声音驱动得到了最充分的测试，所以如果可能的话，推荐您坚持使用它们。如果您需要使用“Jack”或“NAS”您可能已经知道为什么了。

DirectSound 设定主要被游戏使用。您可以选择什么级别的硬件加速您想要，但是对大多数人来说“Full”（完全）比较好。

3.1.6 集成桌面

Wine 能够载入 Windows 的主题，如果您有可用的主题。当然，这肯定不是使用 Wine 或者应用程序所必须的，它确实能允许您定制一个程序的外观和感观。Wine 支持新一点的 MSStyles 型主题。不像老一点的 Windows Plus!型主题，uxthem 引擎支持特殊格式的 .msstyles 文件，这种 .msstyles 文件能改变所有的 Windows 控件 的主题。这多少类似于现代 Linux 桌面版已支持多年的 主题 的性质。

如果您喜欢试验之：

1、下载一个 Windows XP 主题。请确保它含有一个 .msstyles 文件。

2、创建一系列新目录于您的 假 Windows 驱动器：

```
$ mkdir -p ~/.wine/drive_c/windows/Resources/themes/主题名字
```

3、移动（剪切）.msstyles 到那个 主题名字 目录。

4、使用 Winecfg 的 *Desktop Intergration（集成桌面）* 标签来选择新主题。

3.2 使用注册表和注册表编辑器

您在 Winecfg 里做的所有设定变更，除了驱动器设定，基本上都存储在注册表里。在 Windows 里，这是一个应用程序和操作系统配置的中央存储仓库。同样地，Wine 实现一个注册表，而一些在 Winecfg 里找不到的设定能在注册表里变更。（实际上有更多的机会您需要深入注册表来变更一个应用程序设定而非在 Wine 本身。）

现在，Wine 本身使用注册表来存储设定的事实已引起争议。有的人争辩道这样它就太像 Windows 了。相反地，有很多事物需要考虑。首先，不可能避免实现一个注册表，很简单，因为应用程序期望能存储其设定于那里。以便 Wine 存储和读取在一个单独配置文件里的设定将会要求一套独立的代码来基本上做 Win32 API 所做的，而这些 Wine 已经实现。最后，不像 Windows, Wine 的注册表是用纯文本写成的，并且能使用您喜爱的文本编辑器进行修改。即使大多数神志情形的系统管理员（以及 Wine 开发者）疯狂地咒骂 Windows 注册表畸形态的特性，Wine 也有必要设法以某种方法支持之。

3.2.1 注册表结构

OK...就这么滴吧，let's 深入注册表一点，来看看它是如何展开的。Windows 的注册表是一个精心制作的树形结构，连大多数 Windows 程序员也不完全明白它是怎样展开的，它有不同的 " hives"（麻疹，可能是贬义词骂人用）和大量的连接在其中；要完全覆盖它的内容超出了本文档的范畴。但是您现在就可能需要了解如下的注册表键：

HKEY_LOCAL_MACHINE
这个基本的根键（在 Win9x 里它被存储于隐藏文件 system.dat 里）包含所有与当前 Windows 安装有关系的东西。它通常被简写为 HKLM 。

HKEY_USERS
这个基本的根键（在 Win9x 里它被存储于隐藏文件 user.dat 里）包含该安装的每一个用户的配置信息。

HKEY_CLASSES_ROOT
它是一个到 HKEY_LOCAL_MACHINE\Software\Classes 的连接。它包含一些数据，它们描述一些事物，诸如文件关联，OLE 文档处理器，以及 COM 类。

HKEY_CURRENT_USER
它是一个到 HKEY_USERS\your_username 的连接。例如，您的个人配置。

3.2.2 注册表文件

现在，您可能想知道的是注册表如何翻译成 Wine 的结构。在上面描述的注册表布局事实上存在于三个不同的文件，这些文件存在于每个用户的 ~/.wine 目录中。

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 （例如，默认用户的设定）

这些文件在您首次使用 Wine 时被 **wineprefixcreate** 自动地创建。一系列的全局设定被存储于 c:\windows\inf\wine.inf 并且被 rundll32.exe 程序处理。当您首次运行 Wine 时 wine.inf 被处理来组装初始的注册表。要获取更详细的信息，您可以查看 wineprefixcreate 脚本来检视这一切是如何完成的。在升级 Wine 后，**wineprefixcreate** 也可以用来升级默认的注册表键。

正如我们所提到的，您可以编辑那些 .reg 文件，使用任意您想要的文本编辑器。请确保您这么做的时候 Wine 不在运行，否则所有的变更将丢失。

3.2.3 使用注册表编辑器

要读取并变更注册表，一个简单一点的方法是使用工具——regedit。类似于它所代替的 Windows 程序，regedit 用来提供一个系统级别的注册表视图，包含所有的键。简单地运行 *regedit* 它就应该弹出来。您将会立即注意到在文本文件中显示得像密码般的键被组织得井井有条，层次分明。

进入到注册表中，点击左边的键名，并深入下一层。要删除一个键，点击之并从 Edit（编辑）目录选择“Delete”（删除）。要添加一个键或值，定位您想要放它的地方并从 Edit（编辑）目录选择选择“New”（新建）。同样地，您修改一个存在的键或值，在右手边的窗格里高亮之，并从 Edit（编辑）目录选择选择“Modify”（修改）。另一个达到同样效果的方法是右键单击键或值。

可能对 Wine 使用者来说，比较感兴趣的是存储于 HKEY_CURRENT_USER\Software\Wine 里的设定。大多数您在 Winecfg 里变更的设定存储于此目录的此区域。

3.2.4 系统管理窍门

使用上述文件结构，一个系统管理员就有可能配置系统以便一个系统级别的 Wine 安装（以及应用程序）能被所有用户共享，并仍然让所有用户拥有他们自己的私有（个人化）配置。一个管理员能够在已经安装了 Wine 和任何他希望用户能使用的 Windows 应用程序软件后，拷贝由此得到的 system.reg 并改写全局注册表文件（我们假设它会居于 /usr/local/etc 这里），使用如下命令行：

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

并且可能甚至符号连接这些回管理员的帐户，使得以后更容易在系统级别安装应用程序：

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

您也可以尝试过对 user.reg 做相同的事，但是那个文件包含用户特定的设定。每一个用户应该拥有他们自己的那个文件的版本并拥有修改它（用户自己的版本）的权限。

您将想注意驱动器 mappings。如果您正在共享 system.reg 文件您将想确保每一个用户的注册表设定与 ~/.wine/dosdevices 中的启动器 mappings 相兼容。概括来说，您保持您的驱动器 mappings 与 **wineprefixcreate** 提供的默认配置越接近，这越容易管理。您可能或不能共享一些或全部的真的“C:”驱动器上您原先已经安装了的应用程序。一些应用程序要求可以写特殊的设定到驱动器，特别是那些为 Windows 95/98/ME 设计的应用程序。

注意 *tools/wineinstall* 脚本过去做一些这种事如果您以 root 用户身份从源码安装了 Wine ，但是现在它不再做这些。

最后请注意：当心您对管理员帐户所做的一切 —— 如果您拷贝或连接管理员的注册表到全局注册表，任何用户可能有能力读取管理员的首选项，这对存储于这里的敏感信息可能不好（密码，个人信息，等等）。

只使用管理员帐户来安装软件，而不是用来做日常工作；请使用普通用户帐户来做日常工作。

3.2.5 注册表键的完全列表

您将在developer's wiki找到一份最新的有用的注册表键和值的列表。

3.3 其他要配置的东西

这一节的意思是想提及其他的您可以配置的设定。它也当做一个技巧和窍门的集合。

3.3.1 串口和并口

串口和并口配置非常类似于驱动器配置 ——使用设备的名字简单地在 ~/.wine/dosdevices 创建一个符号连接。Windows 串口按照这么一个习惯命名：一个词“com”后面接一个数字。比如 com1,com2, 等等。类似地，并口使用“lpt”后接数字。比如 lpt1。您应该直接地连接它们到相应的 Unix 设备，诸如 /dev/ttyS0 和 /dev/lp0 。要配置一个串口和一个并口，运行下列命令：

```
ln -s /dev/ttyS0 com1
ln -s /dev/lp0 lpt1
```

3.3.2 网络共享

Windows 共享能被映射到 unc/ 目录里以便任何尝试存取 \\myserver\some\file 的东西将在 ~/.wine/dosdevices/unc/myserver/some/file/ 里查找。例如，如果您使用 Samba 来挂载 \\myserver\some 于（到）/mnt/smb/myserver/some，那么您可以做：

```
ln -s /mnt/smb/myserver/some unc/myserver/some
```

来使之在 Wine 中可用（别忘啦如果它不存在，请先创建该 unc 目录!）

3.3.3 字体

字体配置，曾经是个令人厌恶的问题，现在简单得多啦。如果您有一系列的 TrueType 字体在 Windows 里，您可以简单地拷贝 .ttf 文件们到 ~/.wine/dosdevices/c:\windows\fonts 中。
（译者注：c:\windows\fonts 指 Wine 的 虚拟 Windows 主盘 里的目录）

3.3.4 打印机

Wine 能直接地与 CUPS 打印系统 （CUPS, Common UNIX Printing System 通用 Unix 打印系统）互动，来查找您系统上可用的打印机。配置 Wine 的打印机就简单到确保您的 CUPS 配置工作。不过当它打印文档的时候 Wine 仍然需要 lpr 命令（源自 CUPS）。

如果你使用 CUPS,可以使用老的 BSD-Printing 系统（BSD 打印系统）:

- 全部的打印机已经被 Wine 安装在了/etc.printcap 文件中。
- Wine 需要一个 PPD0File 文件给每个打印机(wine 已经含有 generic.ppd)。
- 打印一个文档的时候调用 lpr 命令。

3.3.5 扫描仪

在 Windows 里，扫描仪使用 TWAIN API 来存取当前的硬件。Wine 的 built-in 的 TWAIN DLL 简单地将那些请求发送给 Linux 的 SANE 库。所以，要在 Wine 下利用您的扫描仪您首先需要确保您能够使用 SANE 存取之。在此之后您需要确保您有 xscanimage 可用。当前 xscanimage 是和“sane-frontends（sane 前端）包”一起发行的，但它可能不能安装到您的发行版上。现在已知扫描仪存取有一些问题存在。如果您发现它能够工作，请考虑升级本指南的这一节，并提供详细的关于在 Wine 下使用 SANE 的细节。（译者提示：您是怎么用的？在什么发行版下？安装了哪些包？怎么配置的？扫描仪型号？等等）

3.3.6. ODBC 数据库

在 Wine 里的 ODBC 数据库系统，类似于打印系统，被设计钩对 Unix 系统于一高水平。不是去确保所有的 Windows 代码在 Wine 下工作，而是使用一个合适的 Unix ODBC 供应商，比如 UnixODBC。因此，如果您配置 Wine 去使用 built-in odbc32.dll ，此 Wine DLL 将接入到您的 Unix ODBC 包，并让其做该工作。但是 ,如果您配置 Wine 去使用 native odbc32.dll ，它将尝试使用 native ODBC32 等驱动程序。

3.3.6.1 配置 Unix 上的 ODBC 数据库

要在 Wine 下使用 Unix ODBC 数据库的第一步当然就是要使 Unix ODBC 系统自己能工作起来啦。这个方法包括下载源代码或 RPM 包等。有多种 Unix ODBC 系统可用 ;笔者曾经使用过的一种叫 Unix ODBC (使用 IBM DB2 驱动). 也有 ODBC-ODBC 桥，它可以用来存取一个微软 Access 数据库。一般地，这类系统将包含一个工具，例如 isql，它将允许您从命令行存取数据以便您能检查该系统是否工作。

下一步就是要把 Unix ODBC 库和 built-in odbc32 DLL 关联起来。built-in odbc32 DLL 检查环境变量 LIB_ODBC_DRIVER_MANAGER 来寻找 ODBC 库 的名字。例如在笔者的 .bashrc 文件里有如下的行：

```
export LIB_ODBC_DRIVER_MANAGER=/usr/lib/libodbc.so.1.0.0
```

如果该环境变量没有设置，那么它寻找一个库名叫“libodbc.so”，这样您可以添加一个符号连接到您系统上与之等价的库文件。例如，以 root 身份您可以运行如下命令：

```
# ln -s libodbc.so.1.0.0 /usr/lib/libodbc.so
# /sbin/ldconfig
```

配置这个的最后一步是确保 Wine 被设定为运行 内建的 odbc32.dll 版本。通过修改 DLL 配置 可以达此

目的。这个 built-in DLL 仅仅作为调用的代码和 Unix ODBC 库之间的残余部分。

如果您遇见了任何问题 ,您可以在运行 Wine 之前 ,使用 WINEDEBUG=+odbc32 命令 ,就可以跟踪(调试) 发生了什么。一句警告。有些程序实际上有点骗人并绕过 ODBC 库 , 例如水晶报表引擎会去注册表检查 DSN。关于这个的修补方法被提供于 Unix ODBC 的主页那里有一节是关于在 Wine 下使用 Unix ODBC 的。

3.3.6.2 使用 Windows 上的 ODBC 数据库

Native 的 ODBC 驱动有报告表示可以为多种数据库 (包括 MSSQL 和 Oracle) 工作。事实上 , 有些如 MSSQL 只能在 linux 上通过一个 Winelib 应用程序被存取。不是仅仅拷贝 DLL 文件 ,多数 ODBC 驱动 要求一个基于 Windows 的安装向导被运行来配置诸如注册表键之类的东西。

为了安装 MSSQL 支持 , 您将首先需要到 microsoft.com 下载并运行 mdac_typ.exe 安装向导。为了配置您的 ODBC 连接您必须接下来在 Wine 下运行 CLICONFG.EXE 和 ODBCAD32.EXE。您可以在 windows\system32 目录里找到他们 , 在运行 mdac_typ 后。比较这些程序的输出和在 Windows 机器上有何不同。有些东西 , 比如协议 , 可能会缺失 —— 因为他们需要和操作系统一齐安装。如果是这样 , 您可能可以从一个 存在的 Windows 系统安装拷贝缺失的功能和任何需要的注册表键值。一个本地 native 的 Windows 安装配置在 Wine 上应该与在 Windows 本地以同样的方式工作。

已经在 Wine 下成功地测试的类型 :

数据库类型	可用性
MS SQL	100%

请报告任何其他能用的数据库到wine-devel邮件列表.

第四章 运行 Wine

内容标签：
4.1 基础使用：应用程序和控制面板
4.2 如何运行 wine
4.3 类似 Explorer 的图形化环境变量
4.4 wine 的命令行选项
4.5 环境变量
4.6 wineserver 命令行选项
4.7 设定 windows、Dos 环境变量
4.8 文本程序（CUI：控制用户接口）

本章将描述运行 Wine 的所有方面，例如，基本的 Wine 启用，各种 Wine 支持程序的命令行参数，等等。

4.1 基础使用：应用程序和控制面板

假定您正在使用假的 Windows 安装，您使用与您在 Windows 里安装应用程序相同的方法在 Wine 中：通过运行安装向导。您可以只接受默认安装路径，大多数安装向导会默认安装到 “C:\Program Files”，这很好。如果应用程序安装向导要求创建快捷方式图标，您可能会发现 Wine 在您的桌面和您的应用程序目录里创建图标。如果这事发生了，您可以通过点击那里启动应用程序。

标准的卸载东西的方法是对这种应用程序提供一个 unistaller，通常注册到“添加/删除应用程序”控制面板。要使用 Wine 上的类似程序，在终端里运行 uninstaller 程序（它位于在 Wine 源[代码]目录里的 programs/uninstaller/ 目录）：

```
$ uninstaller
```

有些程序安装关联的控制面板，典型的这类例子是 IE 浏览器和 QuickTime 播放器，您可以在终端中使用下列命令 Wine 来使用控制面板：

```
$ wine control
```

它将打开一个窗口，窗口中有已安装的控制面板，就像在 Windows 里一样。

如果应用程序不安装任何目录或桌面项目，您将需要从命令行运用该应用程序。请记住您把它安装在哪里了，然后使用类似下面的命令：

```
$ wine "c:\program files\appname\appname.exe"
```

将可能工作。目录并不区分大小写，但请记住一定带上双引号（半角英文）。有些程序并不总是使用显而易

见的名字来命名其目录和 EXE 文件，故您也许需要进入程序文件目录去看看哪里放了些什么。

4.2 如何来运行 Wine

您可以简单地调用 wine 命令行来获取简单的帮助信息：

```
Wine 20040405
Usage: wine PROGRAM [ARGUMENTS...]    运行特定的程序
       wine --help                    显示本帮助并退出
       wine --version                 输出版本信息并退出
```

第一个参数应该是您想要 Wine 执行的文件的名字。如果可执行文件在 Path 环境变量里，您可以直接出可执行文件名。但是，如果可执行文件不在 Path 里，您必须给出可执行文件的全路径(以 Windows 格式，而非 UNIX 格式！)。例如，给一个下列的 Path:

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

您可以使用如下命令来运行文件 c:\windows\system\foo.exe ：

```
$ wine foo.exe
```

但是，您必须用下面的命令来运行 c:\myapps\foo.exe：

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

（注意反斜线逃逸符 “\”！）

要获取运行文本模式(CUI,文本界面)可执行文件的详细信息，请参阅下面相关章节。

4.3 类似 Explorer 的图形化 Wine 环境

如果您喜欢使用一个图形化的界面来管理您的文件，您可能想考虑使 Winefile 。这个 Winelib 应用程序是来自 Wine 的并且能从其他 Wine 程序找到。它是来检视您驱动器配置和定位文件的有用方法。另外您能够直接从 Winefile 执行程序。请注意，很多功能没有被实现。

4.4 wine 命令行选项

4.4.1 --help

--help
显示简单的命令行帮助页。

4.4.2 --version

显示 Wine 版本。对验证您的安装有用。

4.5 环境变量

4.5.1. WINEDEBUG=[channels]

Wine 并不是完美的，并且许多 Windows 程序还是不能在没有 bugs 的情况下运行于 wine 下（但那么，很多程序也不能完美运行在 Native Windows 下(译者注 :个人感觉这里说的应该是 Native Windows DLL 下完美运行)）。为了使人比较容易追踪找到在每个 bug 后面的成因，wine 提供你一些 *debug channels*（找错方法）能让你轻易的找到错误。

当你激活每个排错的方法，将会引起登录信息被显示到你调用 **wine** 的控制台。在那里，你能在你的空闲时后重新传入给一个文件的讯息而且调查它。但是你首先要注意，一些侦错选项能产生难以置信那么多数量的记录信息。（打开这些选项）你能看见很丰富的（wine）的犯罪记录：

relay
每当 win32 的函数被调用它就会弹出一个日志（调用）信息

win
追踪 windows 信息传递，当然

all
这是个别名对于每个存在的调试选项。例如对于一个复杂的应用程序 ,你的侦错记录能容易地大于 1MB 或者更大。打开 relay 选项跟踪时，常产生记录信息超过 10 MB 大,关键是你运行了程序多久。 (你可以查看 RelayExclude 注册表值，修改 relay 的跟踪报告。).日志记录会相对降低一点点 wine 的速度，因此除非你真的做需要日志文件，否则不应使用 WINEDEBUG。

在每个侦错频道里面 ,你能更进一步指定一个 信息类别(message class)，过滤出不同的程度的错误。 四个信息类别是:
trace, （追踪）
fixme, （修复）

warn, （警告）
err（错误）

为了要打开一个侦错频道，使用格式
类别+频道

为了把它关掉,使用
类别-频道

为了要列出相同的多于一个的
WINEDEBUG
选项，用逗点隔开他们。 例如，需要警告类别的信息,使用
heap
调试选项。你可以像这样调用 **wine**:
$ WINEDEBUG=warn+heap wine program_name

如果你停止所要的信息类别（像下面一样的调用），**wine** 将会显示来四个类别的信息:
$ WINEDEBUG=heap wine program_name

如果你想看到除 relay 选项以外的所有记录信息，你可以像这样做:
$ WINEDEBUG=+all,-relay wine program_name

这里是一张 wine debug 选项清单和类别。很多选项将会增加（或减少）在以后的版本中。

表 4-1. Debug Channels(调试选项)

accel	adpcm	advapi	animate	aspi
atom	avicap	avifile	bidi	bitblt
bitmap	cabinet	capi	caret	cdrom
cfgmgr32	class	clipboard	clipping	combo
comboex	comm	commctrl	commdlg	computername
console	crtdll	crypt	curses	cursor
d3d	d3d_shader	d3d_surface	datetime	dc
ddeml	ddraw	ddraw_fps	ddraw_geom	ddraw_tex
debugstr	devenum	dialog	dinput	dll
dma	dmband	dmcompos	dmfile	dmfiledat
dmime	dmloader	dmscript	dmstyle	dmsynth
dmusic	dosfs	dosmem	dplay	dplayx
dpnhpast	driver	dsound	dsound3d	edit
enhmetafile	environ	event	eventlog	exec
file	fixup	font	fps	g711
gdi	global	glu	graphics	header
heap	hook	hotkey	icmp	icon
imagehlp	imagelist	imm	int	int21

int31	io	ipaddress	iphlpapi	jack
joystick	key	keyboard	listbox	listview
loaddll	local	mapi	mci	mcianim
mciavi	mcicda	mcimidi	mciwave	mdi
menu	menubuilder	message	metafile	midi
mmaux	mmio	mmsys	mmtime	module
monthcal	mpeg3	mpr	msacm	msdmo
msg	mshtml	msi	msimg32	msisys
msrle32	msvcrt	msvideo	mswsock	nativefont
netapi32	netbios	nls	nonclient	ntdll
odbc	ole	oledlg	olerelay	opengl
pager	palette	pidl	powermgnt	print
process	profile	progress	propsheet	psapi
psdrv	qcap	quartz	ras	rebar
reg	region	relay	resource	richedit
rundll32	sblaster	scroll	seh	selector
server	setupapi	shdocvw	shell	shlctrl
snmpapi	snoop	sound	static	statusbar
storage	stress	string	syscolor	system
tab	tape	tapi	task	text
thread	thunk	tid	timer	toolbar
toolhelp	tooltips	trackbar	treeview	ttydrv
twain	typelib	uninstaller	updown	urlmon
uxtheme	ver	virtual	vxd	wave
wc_font	win	win32	wineboot	winecfg
wineconsole	wine_d3d	winevdm	wing	winhelp
wininet	winmm	winsock	winspool	wintab
wintab32	wnet	x11drv	x11settings	xdnd
xrandr	xrender	xvidmode		

想了解关于侦错频道的较多的细节，点击进入 Wine开发者向导。

4.5.2 WINEDLLOVERRIDES=[DLL Overrides]

（译者注：为了简单，我们这里重申一遍：），built-in 就是属于 wine 的 dll，native 就是 windows 系统带过来的 dll，这样简单的理解认为就够了）
不是总能使用 Built-in DLL 运行应用程序。有时候 native DLL 运行得更好。虽然这些 DLL overrides 可以通过使用 winecfg 来设置，但是您可能希望使用 WINEDLLOVERRIDES 环境变量来设置使用他们。

例如 ,如果您想要 Wine 来使用本地(原生)ole32.dll, oleaut32.dll 和 rpcrt4 您可以像这么运行 **Wine**：
$ WINEDLLOVERRIDES="ole32,oleaut32,rpcrt4=n" wine program_name

要获取更多关于 DLL overrides 的信息，请参阅本指南的 《DLL overrides》（3.1.2.1）》章节。

4.5.3 开放源码（OSS）音频驱动设置

如果你正在使用 OSS 音频驱动而且有多个音频设备，（例如： /dev/dsp*,/dev/mixer*）你可以指定下面一种你喜欢的环境变量：

- AUDIODEV=[audio device]
- MIXERDEV=[mixer device]
- MIDIDEV=[MIDI device]

例如：
$ AUDIODEV=/dev/dsp4 MIXERDEV=/dev/mixer1 MIDIDEV=/dev/midi3 wine program_name

4.6 wineserver 命令行选项

Wineserver 通常自动地被 Wine 启动，不论第一个 wine 进程是什么时候启动的。但是，wineserver 有一些有用的命令行选项，如果您手动启动 wineserver，您就可以添加它们。例如，通过一个用户登录脚本或其他的什么。

4.6.1 –d<n>

为终端里启动的 wineserver 的 debug 输出设置 debug 级别<n>。换句话说：任何高于 0 的值都将启动 wineserver 特殊 debugging（调试）输出。

4.6.2 -h

显示 wineserver 命令行选项帮助信息。

4.6.3 -k[n]

杀死当前的 wineserver ，使用任意的信号 n。

4.6.4. -p[n]

这个参数使 wineserver 保持您想要的时间，n 即为该时间，单位为“秒”。它可以防止 wineserver 马上关闭。
通常，wineserver 会在最后一个使用 wineserver 的 wine 进程终止后立即退出。但是，由于 wineserver 在启动时载入了许多东西（诸如整个 Windows 注册表数据），其启动可能是相当慢的，所以

有时候通过使之保持从而保持其在所有 Wine 会话结束后持续存在。

4.6.5 -w

此参数使一个新启动的 wineserver 等待，直到当前活动的 wineserver 实例终止。

4.7 设置 Windows/Dos 环境变量

您的程序可能要求正确地设置一些环境变量以便其成功地运行。在这种情况下，您需要在 Linux shell 中设置这些环境变量，因为 Wine 将把全部的 shell 环境变量设定传递给 Windows 环境变量空间。例如，在 bash shell 中（其他 shell 可能有不同的语法 ！）：

```
export MYENVIRONMENTVAR=myenvironmentvarsetting
```

这将确保一旦您使用 Wine 启动您的程序，该 Windows 程序能存取 MYENVIRONMENTVAR 环境变量。如果您想要 MYENVIRONMENTVAR 设置能保持，那么您可以把该设定放入 /etc/profile 文件中，或者也可以放在 ~/.bashrc 文件里，如果您使用 bash 的话。

请注意，但是该规则也有例外：如果您想要更改 PATH, SYSTEM 或 TEMP 变量，那么您肯定不能那么做，因为那将改变 Unix 环境设定。所以，您应该把他们设置在注册表中。要设置他们，您需要启动 wine regedit 然后进入

HKEY_CURRENT_USER/Environment

键。现在您可以创建或修改您需要的变量的值

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

此设置指出了 windows system 文件们在何处。Windows system 目录应该居于被用来作为 Windows 设定的目录之下。这样，如果使用 /usr/local/wine_c_windows 作为 Windows 目录的话，那么 system 目录应该是 /usr/local/wine_c/windows/system 。此设置务必不要在末尾上写上斜线，并且请确保您有写入权限。

"Temp" = "c:\\temp"

这应该是您想存放您的临时文件的目录， /usr/local/wine_c/temp 在我们前面的例子里。再次提醒，不要有末尾的斜线，要有写入权限！！

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

类似于 Unix boxes 上的 PATH 设定。当 Wine 是运行像 wine sol.exe ，如果 sol.exe 存在于 Path 设定里指定的目录中，wine 将执行之（当然啦，如果 sol.exe 存在于当前目录中，wine 将执行当前目录中的那一个 sol.exe ）。请确保此变量始终有您的 windows 目录 和 system 目录 （比如，在此例子中的设置，它必须有 "c:\\windows;c:\\windows\\system" ）

4.8 文本型程序（CUI: 控制台用户界面）

文本模式程序是只使用文本作为输出的程序（惊奇啊！）。用 Windows 术语来说，它们被称为 CUI（Console User Interface，控制台用户界面）可执行文件，与 GUI（Graphical User Interface，图形用户界面）相对。Win32 API 提供了一套完整的 API 来处理这种情况，涵盖了从基本的特征（如文本打印）到高级的功能性（如全屏幕编辑，色彩支持，光标支持，鼠标支持等），中间的特性如：行编辑，或 原始的/加工过的 输入流 支持。

鉴于上述的宽广范围，以及当前在 Un*x 世界里的应用，Wine 提供三种不同的方法来运行控制台程序（亦称 CUI 可执行文件）：

•bare streams

•wineconsole with user backend

•wineconsole with curses backend

这里的名称有点灰色而略显费解。“bare streams” 意味著不提供 Wine 的额外支持，这些支持用来影射 Unix 控制台存取。另外两种方法要求使用特定的 Wine 程序 (wineconsole)，它提供扩展的环境。下面的表格描述了使用那三种方法您可以（和不可以）做什么

表 4-2. 三种 console 的基本区别

功能	Bare streams	Wineconsole & user backend	Wineconsole & curses backend
如何运行(假定可执行文件为 foo.exe)	$ wine foo.exe	$ wineconsole -- --backend=user foo.exe	$ wineconsole foo.exe 你也可以像选项一样使用： --backend=curses
对行 oriented CUI 应用程序的良好支持 (它们一行接一行地打印信息)	是	是	是
对全屏幕 CUI 应用程序的良好支持(包括但不限于色彩支持, 鼠标支持...)	否	是	是
甚至能在 X11 未运行时运行	是	否	是
实现	映射标准的 Windows 流到标准的 Unix 流 (stdin/stdout/stderr)	Wineconsole 将创建一个新的窗口 (这样，要求 USER32 DLL 可用) 在那里所有信息将被显示	Wineconsole 将使用已存在的 unix 控制台 (从程序被运行的那个) 在 (n)curses 库的支持（帮助）下，控制所有控制台表面来与用户交互

功能	Bare streams	Wineconsole & user backend	Wineconsole & curses backend
已知的限制			如果 2 个（或多个）Windows 控制台被用于相同的 Un*x 终端，将产生奇怪的表现和行为

4.8.1 CUI 可执行文件配置

当使用 wineconsole 时，有若干配置选项可用。Wine (Windows 也)以每个应用程序为基础，存储若干配置选项在注册表中。例如，这使得用户可以为一个给定的应用程序定义他想要给它的默认屏幕缓冲大小。

从今以后，只有 USER backend 允许您编辑那些选项（我们不推荐手动编辑注册表内容）。当您右键在控制台中单击时，将弹出一个菜单，您可以从下述选项中选择：

•Default（默认）: 这将编辑所有尚未被单独配置的应用程序共享的设定。所以，当首次运行一个应用程序时（在您的机器上，在您的帐号下）Wineconsole 将把从默认设定继承而来的设定作为应用程序的设定。以后，该应用程序将拥有它自己的设定，您可以按照您的意愿来修改之。

Properties（值）: 这将编辑应用程序的设定。当您已经完成了编辑，将提示您是否想：

1.Keep these modified settings only for this session (next time you run the application, you will not see the modification you've just made).

仅为此会话保持这些修改后的设定（下一次您运行该应用程序时，修改将消失）

2.Use the settings for this session and save them as well, so that next you run your application, you'll use these new settings again.

为此会话使用这些设定，并存储之，以便您下次运行您的应用程序，您将再一次使用这些新的设定。下面的列表列出了您可以配置的项目，以及其意义：

表 4-3. Wineconsole 配置选项

配置选项	意义
光标大小	定义光标的大小。可用的三个选项为：small (33% 字符高), medium (66%) 和 large (100%)
弹出菜弹	早些时候就说过，当用右键点击窗口，wineconsole 就会弹出。然而，这将是个问题，你在 wineconsole 中的应用程序希望点击右键的事件能够传给它，在你按下 control 或 shift 键时，同时点击右键打开弹出菜单（用 shift+其他 或者 control +其他键组合有另外的功能）。例如：当你没有按住 shift 键点击右键点击窗口后 ticking shift 会发送“事件”（event）给应用程序，你按住 shift 键点击右键时，窗口打开。
快速编辑	这个勾选框允许您决定左键单击事件是否将被解释为事件并发送给当前的应用程序(不勾选)或作为屏幕中被选中的矩形块以待之后拷贝到剪贴斑上 (勾选).
历史记录	这允许您设置控制台能回溯多少条命令。您也可以在连续键入若干条相同的命令时是否要存储它们全部的命令 (不勾选) 或最后一条 (勾选).
外观设定	外观设定（police）值选项卡允许您选择默认控制台的字体（ 字体文件, 大小, 背景和前景

配置选项	意义
	色).
屏幕缓冲 & 窗口大小	您所看见的控制台是由不同的两部分组成。一方面屏幕缓冲包含了所有的您的应用程序打印在屏幕上的信息，另一方面窗口显示屏幕缓冲的给定部分。请注意，窗口总是比屏幕缓冲小或者与之相等。较小的窗口大小将放上滚动条以便您能看见全部的屏幕缓冲的内容。
退出时关闭	如果它被勾选, 那么 wineconsole 将在应用程序终结时退出。反之, 它将保持打开直到用户手动地关闭之: 这将使您能在程序终结后看见它最后输出的信息。
编辑模式	当用户键入命令,他/她可以在许多不同编辑模式中选择: • Emacs: 可用 emacs 下的键设定. 例如, Ctrl-A 将使光标跑到编辑行的行首. 参阅您的 emacs 手册来获取命令的详细信息. • Win32: 标准的 Windows 控制台键设定 (主要使用箭头[方向]键).

第五章 问题捕捉报告 bug

内容标签：
5.1 如果有些程序仍不能运行怎么办？
5.2 如何报告一个 bug

5.1 如果有些程序仍然不能运行怎么办

有些时候您已经尝试了所有的方法，您甚至费了九牛二虎之力吃尽了苦头绞尽了脑汁，做了这令人恶心的事情，但仍然无助于使一些他妈的该死的程序在一些版本的 Wine 上工作。别泄气，我们在这里要帮助您……（换句话说，您想付出多少银子？）

5.1.1 验证您的 wine 配置

使用 $ wine --version 并观察其输出，确保您正在运行一个新近版本的 Wine。启动 winecfg 并检查各个设定以确保设定看上去比较正常。检查 ~/.wine/dosdevices 确保您的 c: 指向了您认为它应该指向的地方。

5.1.2 使用不同的 Windows 版本设定

在许多情况下使用不同的 Windows 版本设定 是有所助益的。

5.1.3 使用不同的启动路径

这个方法有时能有助益。请 wine prg.exe 和 wine x:\\full\\path\\to\\prg.exe 都试一试。
（译者注：请替换为您自己机器上的正确路径）

5.1.4 乱动 DLL 配置

在运行时使用 WINEDEBUG=+loaddll 来配置要载入哪些 DLL 以及载入 native 的还是 built-in 版本。然后请确保您有正确的 native 的 DLL 文件在您配置的 C:\windows\system 目录并在命令行或配置文件乱动 DLL 载入顺序设定。

5.1.5 检查您的系统环境！

仅仅一个想法 :会不会是您的 Wine 创建/执行环境出错了？请确保 Wine 所依赖的包没有任何问题(gcc, glibc，X 库，OpenGL(!)，……）例如当使用“错误的”头文件给“正确的”库时，有些人会遇到奇怪的不能找到东西的失败！!!(这导致数日的 debug ，不顾一切地尝试找出为什么那个低级的功能[函数]会以

那种超乎想象的方式失败……ARGH!）

5.1.6 使用不同的 GUI(窗口管理器)模式

通过配置文件来命令 Wine 使用 桌面模式 ，托管模式 或朴素的或者说丑陋的 “普通”模式 三者其一。也将可能使问题有所改善。

5.1.7 检查您的应用程序！

可能您的应用程序在使用某种防拷贝措施？许多防拷贝措施当前并不能在 Wine 上工作。不过，有些可能在将来可以工作。(CD-ROM 层并不真的是特性完全的。)

到 游戏拷贝世界 并尝试给您的游戏找一个象样的破解来破除该丑陋的防拷贝措施。不过,我还是希望您确实有一份该程序的合法拷贝…… :-)

5.1.8 检查您的 Wine 环境！

是否使用 Windows 分区运行会有戏剧性的影响。 配置 Wine 去做与过去所做相反的事。另外，安装 DCOM98 或 DCOM95 。这将非常有益。

5.1.9 重新配置 wine

有时候 wine 安装进程发生了改变，新版本的 Wine 依赖于这些改变。尤其是如果您的配置是很就以前创建的。重命名您原有的 ~/.wine 目录作为备份。使用适合您的 Wine 发行版的配置进程来创建新的配置。使用原有的 ~/.wine 目录里的信息作为参考。如果是源代码的 Wine 发行版，您可以您想用来配置 Wine 的用户身份来运行 tools/wineinstall 脚本。这是一个很安全的操作。以后，您可以删除新创建的 ~/.wine 目录，并把旧的那个重命名回去。(译者注：意味著您觉得新的配置不如旧配置好，想改回去。)

5.1.10 查看更多信息

很有可能有其他人已经尝试了做您要做的事，您会发现下列资源很有用：

•搜索 WineHQ的应用程序数据库 来查找与该程序有关的窍门、技巧。如果您的该程序的特定版本未被列于其中，您可以找一个不同版本的信息来帮助您解决问题。

•Frank's Corner 包含一个应用程序列表以及详细的安装、配置它们的指导。更多的帮助可以从其用户论坛得到。

•Google 其用处大小取决于您如何使用之。您可能会发现搜寻 Google组群 是很有用的。特别是 comp.emulators.ms-windows.wine组群。

•Freenode.net有一个关于 Wine 的 IRC 频道。您可以在这里聊关于 Wine 的内容 ,使用任何的 IRC 客户端，比如Xchat,(译者注：推荐使用mozilla或者firefox的那个插件，支持多国语文[一个对话框只限一种文字]，稳定性也不错)。您需要如下设定方可登录：server = irc.freenode.net，port = 6667，channel = #winehq

•如果您的程序需要 Visual Basic 运行时间环境，您可以从 此微软页面 下载之。

•如果您知道您缺失某个 DLL ，例如 mfc42，您可能会在 www.dll-files.com找到它。

•Wine的 邮件列表 也可能有所助益。特别是 wine-users 列表。Wine-devel 列表可能也是合适的。视乎您所遇到的问题的类型。如果您要张贴问题到 wine-devel 您应该准备好做一些有助于诊断该问题的工作。参阅下一节来找出如何debug，找出您的问题之源。

•如果所有其他的努力都失败了，也许您希望调查商业化的 wine 版本来检视您的应用程序是否被其支持。

5.1.11 Debug 之！

下一步所要做的就是找到您的问题之源。可能的问题范围广泛到从只是配置的问题到 Wine 中完全没有实现的功能的问题。下一小节将描述如何 file 一个 bug 报告以及怎么开始 debug 一个崩溃。要获取更多关于使用 Wine 的 debug 设施的信息。请参阅《 Wine 开发者指南》。

5.2 如何报告一个 bug

请报告所有的 bug 以及相关的信息到 Wine Bugzilla。请搜索 Bugzilla 数据库来确定您的问题是否已经被报告过。如果该 bug 已被报告过，请添加任何与已有的 bug报告 相关的信息。

5.2.1 所有 bug 报告

下面是一些关于如何使您的 bug 报告更有价值的建议（这样您的问题更有可能被回复并被修复）：
1、尽可能多地张贴相关信息
　这以为著我们需要比“MicroSoft Office word 在任何我运行它的时候崩溃。您知道为什么吗？” 更多的信息：
•您在使用 Wine 的哪一个版本（运行 wine --version）

•您所使用的操作系统的名称，什么发行版（如果是发行版的话），以及什么版本。（例如，Linux Red Hat 7.2）。

•编译器及其版本（Linux 下请运行 gcc -v）。如果您并未编译 Wine，那么请告诉我们您使用的 Wine 的包的名称以及您从何处获得之。

•Windows 版本，如果您通过 Wine 使用之。如果您没有使用 Windows，也请您告诉我们。

•您尝试运行的程序名，以及版本号，以及一个可以获取该程序的 URL（如果有的话）。

•您用来启动 Wine 的确切命令行。（例如，wine "C:\Program Files\Test\program.exe"）

•能够重现该 bug 所需的确切操作步骤。

•任何其他您认为可能相关或有用的信息，比如 X 服务器版本如果有 X 问题，libc 版本，等等。

2、再次运行该程序，并使用 WINEDEBUG 环境变量 WINEDEBUG=+relay 选项。（例如，WINEDEBUG=+relay wine sol.exe）

这将输出一些可能对 debug 该程序有用的额外信息到控制台。它也会减慢程序执行速度。有些情况下，在使用了 +relay 后，bug 消失了。如果这种情况发生了，请告诉我们。

5.2.2 崩溃

如果在运行您的程序时 wine 崩溃了，拥有这一信息对我们来说相当重要；有了这些信息，我们才有机会指出是什么导致了崩溃。当然啦，这将会输出相当多（若干 MB）的信息。所以最佳的方法是将其输出到一个文件。当 Wine-dbg> 提示符出现时，键入 quit。

您可能想试试用 +relay,+snoop 来代替 +relay ，但是请注意，+snoop 非常不稳定而经常比只有 +relay 更早崩溃！ 如果是这样，请只使用 +relay !! 一个使用 +snoop 代码里的一个崩溃的 bug 报告在大多数情况下是没用的！您也可以启用其他参数，取决于您所研究的程序的性质和类型。参阅 wine man page 获取全部参数的列表。

要获取回溯跟踪输出，请使用下列方法之一。

5.2.2.1 简单的方法

1、此方法可使得一个完全是新手的人也能在一个崩溃事件中提交一份有关的回溯跟踪日志。

为了让这种方法工作，您的计算机上 必须 有 Perl。要证实您是否有 perl，请运行 **which perl**。如果返回如 /usr/bin/perl 的信息，说明您已经装了 perl。否则，请跳到下一小节“困难的方法”。如果你不确定，就继续，当你试着运行 perl 脚本，它将明显的表现出来（译者注：表现出来你是否安装 perl）。

2、切换目录到 /tools

3、键入 ./bug_report.pl ，然后请按其说明操作。

4、张贴该 bug 到 Wine Bugzilla。在张贴一个 bug 报告前，请搜索 Bugzilla数据库 来检查您的问题是否已被找到。包括您自己的关于问题的详细描述以及相关信息。粘贴“很好格式化了的报告”到提交的 bug。不要剪切、复制 bug 描述里的报告 —— 它相当的大。请保存完整的 debug 输出，以防万一啥

时候 Wine 开发者需要。

5.2.2.2 复杂的方法

似乎只有最后的 100 行左右的回溯跟踪对于找出程序在什么地方崩溃是必要的。为了获得那最后 100 行，我们需要干如下的事：

1、重定向所有 WINEDEBUG 输出到一个文件。
2、使用 tail 命令把最后 100 行分离到另一个文件。

这可以使用下列方法之一来完成
所有 shells：

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

（这将打印 wine 的 debug 信息只到文件然后自己退出，使用此命令是个不错的主意，因为 wine 打印如此之多的 debug 信息以至于如洪水泛滥般淹没了终端，如猛兽般吃掉 CPU 时间片。）

tch 和其他类 csh shells：

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

bash 和其他类 sh shells：

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

report_file 现在将包含最后一百行的 debug 输出，包括注册表备份和回溯追踪，这些是最为重要的信息。请不要删除这一部分，即使您不知道它是什么意思。

张贴 bug 到 Wine Bugzilla。您需要粘贴第(2)部分输出的 report_file 文件，以及相关的信息来创建 bug 报告。请不要剪切、复制 bug 描述里的报告 —— 它相当的大，并将造成大量的 bug 报告。如果您这样做，您收到某种有用的回复的机率将会非常小。

请搜索 Bugzilla 数据库 来检查是否您的问题已经被报告过。如果已经被报告过，请粘贴输出的 report_file 文件到原来的 bug 报告并添加任何其他相关的信息。

生词本

Binary 可执行文件/二进制文件
一个处于机器可执行状态的文件，编译的格式：hex 数据（与源代码相对）。

Distribution 发行版

发行版通常是指一些“发行商”(独立软件商)发行操作系统 CD 光盘的方式(通常在 Linux 的 context 被提及)。Linux 环境能以许多不同的配置发布，例如：发行版可以被创建为适合游戏，适合科学计算应用程序，适合作服务器操作系统，或适合做桌面系统，等等。

DLL 动态链接库

DLL(Dynamic)是能被程序动态地载入并执行的文件。基本上它是程序的一个外部代码仓库。因为通常若干不同的程序重用相同的 DLL ，而不是在它们自身的文件中置入那些代码。这极大地降低了要求的存储空间。DLL 的等价物/术语/代名词是“库”。

Editor 编辑器

编辑器通常是用来创建或修改文本文件的程序。有许多图形模式和文本模式下的编辑器在 Linux 下可用。例如，图形化的编辑器有：nedit, gedit, kedit, xemacs, gxedit.

例如，文本模式的编辑器有：joe, ae, emacs, vim, vi.在一个 *终端* 中，简单地这么启动它们：

$ editorname

Filename

Environment variable 环境变量

环境变量是用于 shell 中的文本定义，用来存储重要的系统设定。在一个 bash shell 中(它是 Linux 中最为常用的 shell)您可以执行如下命令来检视所有的环境变量：

set

若您想改变一个环境变量，您可以运行：

export *MYVARIABLE=mycontent*

要删除一个环境变量，请使用：

unset *MYVARIABLE*

Git

Git 是一个快速，直接的代码管理器。最初的编写者用于大型仓库，例如 linux 内核源码。参考 Git 章节在 Wine 开发者向导里有更多的使用信息。

Package 包

一个包是一个 distribution specific format(发行版特定格式)的压缩文件。它包含了您想安装的一个特定程序的文件。包通常通过 dpkg 或 rpm 包管理器安装。

root 根，根用户，超级用户，系统管理员

root 是系统管理员的帐户名。为了以 root 用户身分运行程序，只需要打开一个 终端 窗口，然后运行：

$ su -

它将提示您输入您系统上的 root 用户的密码，在正确输入密码后，您将能进行要求特殊的 root 权限的系统管理任务(工作)，root 帐户提示符为：

#

相对地，'$' 是普通用户帐户的提示符。

Shell

shell 是使得拥护能与系统交互(互动)的工具。通常 shell 是基于文本并源于命令行的。例如 ,流行的 shell

包括：bash，tcsh 和 ksh。Wine 假定您使用 bash 来完成 Wine 的安装任务，因为这是 Linux 上最为流行的 shell。shell 通常运行于一个 终端 窗口。

Source code 源代码
源代码是一个程序在编译之前该程序所包含的代码。例如，它是程序的原始创建指令，它告诉编译器当程序被编译成 二进制 后会以何种样子呈现。

Terminal 终端
一个终端窗口通常是一个图形化窗口，它用来执行一种 shell。如果 Wine 要您打开一个终端，那么您需要点击您桌面上的某个图标，该图标为一个大黑窗口（或者，其他情况下，一个显示著象征性的[缩微的]shell）。Wine 假定您在终端窗口使用 bash shell。所以，如果您的终端窗口恰好使用著其他的 shell 程序，您只需要在终端窗口中键入：
bash

表的列表

表 1-1. 各种不同的 Wine 产品

产品	描述	发布形式
CodeWeavers CrossOver Office	CrossOver Office 允许您在 Linux 中安装您喜爱的 Windows 应用程序，而不需要一个微软操作系统许可证。CrossOver 包含一个容易使用的，单击使用的界面，它使得安装 Windows 应用程序简单而快捷。	商业版；提供 30 天不限制任何功能的演示版。
CodeWeavers CrossOver Office Server Edition	CrossOver Office 服务器版允许您运行您喜爱的 Windows 办公应用程序在一个 Linux 下的发布的瘦客户端环境，而每个客户机不需要微软操作系统许可证。CrossOver Office 服务器版允许您满足字面上地数百并发的用户的需求，全部来自一个单独的服务器。	

表 4-1. Debug Channels(调试选项)

accel	adpcm	advapi	animate	aspi
atom	avicap	avifile	bidi	bitblt
bitmap	cabinet	capi	caret	cdrom
cfgmgr32	class	clipboard	clipping	combo
comboex	comm	commctrl	commdlg	computername
console	crtdll	crypt	curses	cursor
d3d	d3d_shader	d3d_surface	datetime	dc
ddeml	ddraw	ddraw_fps	ddraw_geom	ddraw_tex

debugstr	devenum	dialog	dinput	dll
dma	dmband	dmcompos	dmfile	dmfiledat
dmime	dmloader	dmscript	dmstyle	dmsynth
dmusic	dosfs	dosmem	dplay	dplayx
dpnhpast	driver	dsound	dsound3d	edit
enhmetafile	environ	event	eventlog	exec
file	fixup	font	fps	g711
gdi	global	glu	graphics	header
heap	hook	hotkey	icmp	icon
imagehlp	imagelist	imm	int	int21
int31	io	ipaddress	iphlpapi	jack
joystick	key	keyboard	listbox	listview
loaddll	local	mapi	mci	mcianim
mciavi	mcicda	mcimidi	mciwave	mdi
menu	menubuilder	message	metafile	midi
mmaux	mmio	mmsys	mmtime	module
monthcal	mpeg3	mpr	msacm	msdmo
msg	mshtml	msi	msimg32	msisys
msrle32	msvcrt	msvideo	mswsock	nativefont
netapi32	netbios	nls	nonclient	ntdll
odbc	ole	oledlg	olerelay	opengl
pager	palette	pidl	powermgnt	print
process	profile	progress	propsheet	psapi
psdrv	qcap	quartz	ras	rebar
reg	region	relay	resource	richedit
rundll32	sblaster	scroll	seh	selector
server	setupapi	shdocvw	shell	shlctrl
snmpapi	snoop	sound	static	statusbar
storage	stress	string	syscolor	system
tab	tape	tapi	task	text
thread	thunk	tid	timer	toolbar
toolhelp	tooltips	trackbar	treeview	ttydrv
twain	typelib	uninstaller	updown	urlmon
uxtheme	ver	virtual	vxd	wave
wc_font	win	win32	wineboot	winecfg
wineconsole	wine_d3d	winevdm	wing	winhelp
wininet	winmm	winsock	winspool	wintab
wintab32	wnet	x11drv	x11settings	xdnd
xrandr	xrender	xvidmode		

表 4-2. 三种 console 的基本区别

功能	Bare streams	Wineconsole & user backend	Wineconsole & curses backend
如何运行(假定可执行文件为 foo.exe)	$ wine foo.exe	$ wineconsole -- --backend=user foo.exe	$ wineconsole foo.exe 你也可以像选项一样使用： --backend=curses
对行 oriented CUI 应用程序的良好支持 (它们一行接一行地打印信息)	是	是	是
对全屏幕 CUI 应用程序的良好支持(包括但不限于色彩支持, 鼠标支持...)	否	是	是
甚至能在 X11 未运行时运行	是	否	是
实现	映射标准的 Windows 流到标准的 Unix 流 (stdin/stdout/stderr)	Wineconsole 将创建一个新的窗口 (这样，要求 USER32 DLL 可用) 在那里所有信息将被显示	Wineconsole 将使用已存在的 unix 控制台 (从程序被运行的那个) 在 (n)curses 库的支持（帮助）下，控制所有控制台表面来与用户交互
已知的限制			如果 2 个（或多个）Windows 控制台被用于相同的 Un*x 终端，将产生奇怪的表现和行为

表 4-3. Wineconsole 配置选项

配置选项	意义
光标大小	定义光标的大小。可用的三个选项为：small (33% 字符高), medium (66%) 和 large (100%)
弹出菜单	早些时候就说过，当用右键点击窗口，wineconsole 就会弹出。然而，这将是个问题，你在 wineconsole 中的应用程序希望点击右键的事件能够传给它，在你按下 control 或 shift 键时，同时点击右键打开弹出菜单(用 shift+其他 或者 control +其他键组合有另外的功能)。例如：当你没有按住 shift 键点击右键点击窗口后 ticking shift 会发送“事件”（event）给应用程序，你按住 shift 键点击右键时，窗口打开。
快速编辑	这个勾选框允许您决定左键单击事件是否将被解释为事件并发送给当前的应用程序(不勾选)或作为屏幕中被选中的矩形块以待之后拷贝到剪贴斑上 (勾选).

配置选项	意义
历史记录	这允许您设置控制台能回溯多少条命令。您也可以在连续键入若干条相同的命令时是否要存储它们全部的命令 (不勾选) 或最后一条 (勾选).
外观设定	外观设定（police）值选项卡允许您选择默认控制台的字体（ 字体文件, 大小, 背景和前景色).
屏幕缓冲 & 窗口大小	您所看见的控制台是由不同的两部分组成。一方面屏幕缓冲包含了所有的您的应用程序打印在屏幕上的信息，另一方面窗口显示屏幕缓冲的给定部分。请注意，窗口总是比屏幕缓冲小或者与之相等。较小的窗口大小将放上滚动条以便您能看见全部的屏幕缓冲的内容。
退出时关闭	如果它被勾选, 那么 wineconsole 将在应用程序终结时退出。反之, 它将保持打开直到用户手动地关闭之: 这将使您能在程序终结后看见它最后输出的信息。
编辑模式	当用户键入命令,他/她可以在许多不同编辑模式中选择: • Emacs: 可用 emacs 下的键设定. 例如, Ctrl-A 将使光标跑到编辑行的行首. 参阅您的 emacs 手册来获取命令的详细信息. • Win32: 标准的 Windows 控制台键设定（ 主要使用箭头[方向]键).