分 类 号：TP393.092　　单位代码：10183
研究生学号：200653H119　　密　　级：公 开

吉林大学

硕士学位论文

基于 Python+Tkinter 的 Linux GUI 辅助管理工具的设计与实现

Design and Implementation of GUI Aided Management tool in Linux Based on Python+Tkinter

作者姓名：李　悦
专　　业：软件工程
研究方向：网络应用技术
指导教师：徐高潮　教授
培养单位：计算机科学与技术学院

2009 年 11 月

基于 Python+Tkinter 的 Linux GUI 辅助管理工具的设计与实现

Design and Implementation of GUI Aided Management Tool in Linux Based on Python+Tkinter

作者姓名：李悦

专业名称：软件工程

指导教师：徐高潮

学位类别：软件工程硕士

答辩日期：2009 年 11 月 28 日

吉林大学博士(或硕士)学位论文原创性声明

本人郑重声明：所呈交学位论文，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：09 年 12 月 1 日

内容提要

当前，Linux 技术得到飞速发展，Linux 操作系统已经在各行各业得到广泛应用。特别是在个人桌面系统开始迅速普及。作为个人桌面系统，由于用户计算机水平有限，传统 Linux 繁琐的命令行操作成为普及 linux 的最大障碍。因此，如何将 Linux Shell 下的大量应用软件快速构建配置一个方便易用的图形化接口，从而有效提高操作的易用性是亟待解决的问题。

基于以上需求，本文基于 Python 语言，结合 Tkinter 和 Pwm 模块，为 Linux Shell 应用软件提供一个方便简洁的 GUI 图形化用户操作界面。同时，通过几个具有代表性的 Linux 交互命令程序的图形化系统辅助程序设计实例，详细阐述了 Python+Tkinter 实现 Linux 图形化辅助管理工具的细节和技术难点。

实验表明，本文实现的辅助管理系统实例运行稳定，操作简单，验证了本文提出的图形化实现方法的可行性和合理性。同时，对今后 linux 图形化辅助管理软件的设计与实现工作具有一定的参考价值和借鉴意义。

目录

第 1 章 前　言

1.1 Linux 系统的背景简介

随着 Linux 技术的不断提高，其强大的功能已为人们深刻认识，它已进入社会的各个领域并发挥着越来越重要的作用。Linux 操作系统是一套免费使用和自由传播的类 Unix 操作系统，最初主要用于基于 Intel x86 系列 CPU 的计算机上，随着 Linux 系统的普及，其他体系的计算机以及个人数字终端也在越来越多的采用 Linux 操作系统。Linux 操作系统是由通过互联网连接的成千上万的程序员参与设计和实现的。其目的是建立不受任何商品化软件版权制约的、广泛应用的 Unix 兼容产品。

Linux 操作系统的创立者是 Linus Torvalds，最初的设计思想是设计一个代替 Minix（一个操作系统教学演示程序）的操作系统。希望该操作系统可用于 386、486 或奔腾处理器的个人计算机上，并且具有 Unix 操作系统的全部功能。Linux 操作系统具有卓越的高效性和灵活性，能够在 PC 计算机上实现全部的 Unix 特性，具有真正的多任务、多用户的处理能力。 Linux 遵循 GNU 组织的公共许可证协议 GPL，是一套符合 POSIX 标准的操作系统。Linux 操作系统不仅包括完整的操作系统内核，而且还包括文本编辑器（例如 vi，emacs）、高级语言编译程序（例如 gcc）等多种应用软件。同时，它还包括带有多个窗口管理器的 X-Windows 图形用户操作界面，类似 Windows 系统，可以使用窗口、图标和菜单等多种图形控件对系统进行操作。

Linux 操作系统的全称是 GNU/Linux，它是由 GNU 组织的开源软件和 Linux 内核两个部分共同组成的一个操作系统，虽然这个系统诞生于 1992 年，比 Windows 操作系统要晚，但是与 Windows 相比它有很多独到的优势。

开放性：Linux 操作系统遵循 POSIX 开放系统互连（OSI）国际标准。

多用户：Linux 操作系统可以同时为不同用户使用，每个用户对自己的资源（例如：文件、目录、外设设备）具有特定的权限，彼此相对独立互不影响。

多任务：Linux 操作系统同时执行多个任务程序，而且各个程序的运行相对

独立。

友好的人机交互接口：Linux 操作系统向用户提供了两种接口形式：用户命令行 SHELL 操作方式和图形化 X-windows 操作方式。用户可以使用图形化界面方便快捷地操作系统，同时，系统管理人员能够使用 SHELL 命令行方式灵活高效地管理系统。

强大的网络功能：Linux 系统继承了 Unix 系统在网络方面的优势，完善的通过内核级实现的网络服务功能是 Linux 一大特点。

可靠的安全机制：Linux 系统采取了很多安全机制，包括对读、写、执行的控制，审计跟踪程序的执行，以及对授权的监控等，为网络多用户环境中的用户提供了必要的安全保障。

良好的可移植性：Linux 系统是一套可移植的操作系统，从最初诞生的 x86 平台到今天各种各样的数字终端、单片机、监控系统以及复杂的分布式网络计算环境和网格计算都可以看见 Linux 系统的身影。

1.2 开发图形化 Linux 辅助管理系统的必要性

作为 Linux 应用的一部分，使用图形化接口 GUI 对 Linux 系统进行操作和管理是十分必要的。在 Linux 系统中，管理工具通常是采用控制台 (SHELL 命令行) 的方式来进行操作的。复杂晦涩的 SHELL 命令和参数对初学者和一般用户来说是非常困难的，也对 Linux 系统的普及尤其是作为桌面操作系统照成巨大障碍。如何给传统的命令行操作方式的系统管理命令提供一套图形化的便于操作和管理的 GUI 机制是目前 Linux 系统普及中亟待解决的问题，也是推广普及 Linux 必须解决的重要课题。

本文针对以上问题，提出了采用 Python 脚本语言和图形化 Tkinter 模块建立快速图形化操作界面的方法，并通过具体实例验证了该方法的可行性和稳定性，为 Linux 系统的推广普及做了有益的探索。

1.3 Linux 辅助管理系统的设计目标分析

Linux 辅助管理系统的设计目标如下：

（1）接口的设计：对项目的所有功能实现 GUI 用户交互接口，这个功能用 Python 的 Tk 模块实现。

（2）解压缩程序的 GUI 模块：实现在 Linux 下程序的压缩和解压缩操作，用户通过与程序交互的图形化操作实现传统的命令行功能。

（3）定时自动关机程序的 GUI 模块：实现输入时间，自动关机的功能。

1.4 Linux 辅助管理系统的系统需求

本辅助管理系统是在 Linux 系统下对其 Shell 命令和 GUI 程序设计的一种扩展，该辅助管理系统是在 Slack Linux 系统下以 Python 语言编写的，接口部分基于 Python 的 Tkinter 模块，功能实现部分主要依靠调用 Linux 下的 Shell 命令实现，使用者仅仅需要简单的图形化操作即能知道该软件的使用方法。

1.5 Linux 辅助管理系统的功能需求分析

该辅助管理系统为了方便用户使用 Linux 系统，实现了一些简单的功能，用户可以通过其用户管理程序的模块，查看，删除，更改和创建用户和组，方便地实现对自己用户的管理和操作。另外一方面用户可以在该系统下实现对 Linux 下几乎所有压缩包的解压操作，其中软件通过对本机上所有压缩包档的搜索，列出所有的压缩文件，用户只需点击鼠标即可实现解压。最后，用户可以设定自己希望关机的时间，防止出现停电等意外情况的发生。

第 2 章 相关技术简介

2.1 Python 简介

2.1.1 什么是 Python

Python 是一种解释性的，面向对象的高层语言以及强大的网络服务器端脚本语言。与其他脚本语言一样，Python 代码类似于伪代码。Python 的语法规则和精巧设计使其便于多位程序员组成的开发团队阅读。该语言的语法并不丰富，但是很简洁。

Python 的创始人为 Guido van Rossum。1989 年 Guido 决心开发一个新的脚本解释程序，作为 ABC 语言的一种继承。

ABC 语言是 Guido 参加设计的一种教学语言。该语言非常优美和强大，是专门为非专业程序员设计的。但是 ABC 语言因为非开放性的限制使其没有得到广泛应用。Python 弥补了 ABC 语言的缺陷，与其它的语言如 C、C++和 Java 结合非常好[1-7]。同时，Python 语言也借鉴了 Modula-3 语言。多种语言特点的借鉴和完善使 Python 语言形成了自己的优势和特点。

2.1.2 Python 的特点

Python 是一种脚本语言，它的语法表达优美易读，具有很多优秀的脚本语言的特点：解释语言，面向对象，内建的高级数据结构，支持模块和包，支持多种平台，可扩展性。而且它还支持交互式方式运行，图形方式运行。它的语法有很多与众不同的特性：

1．多种运行方式：

Python 可以以命令行方式运行，同时，也可以交互式方式运行。图形集成环境是开发工作变得很简单高效。

2. 面向对象：

Python 是一个真正的面向对象语言。它甚至支持异常的处理。这种机制极大地减少了错误检查的工作量，加速了代码开发的速度。

3．模块和包：

Python 提供了各种各样的模块和包，丰富的模块库为程序设计者提供了快速开发方式。同时，与 JPython 共用的 Java 的类库也提高了 Python 的扩展性。

4．语言扩展：

C、C++或 Java 语言都可以为 Python 编写新的模块。同时，也可以与 Python 直接编译，或者采用动态库装入方式实现。

Python 语言的块语句的表示不是 C 语言常用的{}对，或其它符号对，而是采用缩近表示法。采用这种方式具有以下优势：首先，使用缩近表示法减少了视觉混乱，提高了代码的可阅读性；其次，它减少了程序员的自由度，更有利于统一风格，为代码维护减小开销 [1-7]。

2.1.3 Python 的主要技术特征

以上介绍了 Python 的基本特点，下面介绍 Python 语言的主要技术特征：

自动内存管理：Python 对象一旦变得无法利用就会被集中起来，Python 履行把其标志成“无用信息”的职责。

异常处理：异常处理支持在无需向代码中添加许多错误检查语句的情况下捕获错误。这个功能让 Python 程序从来不会崩溃，使得其总返回一个 traceback 消息。

丰富的核心库：目前已经开发完成许多扩展模块并已成为标准 Python 工具库的构成部分，程序员可以在任何 Python 应用程序中使用该工具库。

Web 脚本支持和数据处理：Python 使编写顺利运行于几个环境之中的 CGI 程序成为可能。通过利用 Python 的内置类和常规表达式方法，分析 XML、HTML、SHML 以及其他各类文本。

内置元素：Python 提供一个由有用内置元素组成的大型列表，这些内置元素带有正确处理它们所需要的许多专用操作。

开发流：尽管 Python 不需要任何编译或连结过程，但它支持字节编译。编译代码保存在名为字节代码的中间语言中，该中间语言可由带有 Python 虚拟机的任何系统加以访问。该特征提供类似于 Java 所提供的那类可移植性。应用程序无需编译就可以在几种不同的系统中使用。

可嵌入与可扩展：Python 可以嵌入到采用许多程序设计和脚本语言编写的应用程序之中。

物件分布：Python 可用于实现需要与其他应用程序中的对象进行对话的历程。

数据库：Python 具有与所有主要商用数据库的接口，提供有几个用于处理展开文件数据库的工具程序，并实现可以把整个对象保存到文件的持久性对象系统。但其中最棒的数据库特征是 Python 定义了一个标准数据库 API，正是它的存在使得应用程序容易移植到不同的数据库中。

GUI 应用程序：由于目前已为 Python 开发了许多图形用户界面绑定，因此可以创建实现 GUI 的应用程序，所创建的应用程序可以移植到许多系统调用、库、以及诸如 Windows MFC 之类的窗口系统中。

内省（introspection）：可以使用 Python 开发程序，以助于创建其他使用 Python 的程序。最重要的范例有调试器和配置文件程序。

第三方集成：也正是 Python 以上的特点，更确切地说是优点促使我们使用这种语言编写 Linux 辅助管理系统[1-7]。

2.1.4 Python 与 Tkinter

广义上来说，使用 Python 的 GUI 项目还有其他选项可供选择。但是，目前的 Python 已经选择支持 Tkinter 作为它的正式 GUI 实现。

Tkinter 是一个 Tk GUI API 的标准面向对象接口，最初由 Steen Lumholdt 编写，当时 Steen Lumholdt 在忙于利用 Python 改进 GUI。Tkinter 是一个成熟的跨平台接口，它为 GUI 应用程序提供小型的配件集。但是，这并不意味着要固定于这个集合。Tkinter 是可扩展的，即意味着可以使用第三方的配件程序包。Widget 是用户接口元素，例如 radio button（单选框）和 list box（列表框）等。

Tkinter 工具箱是一个强大的 GUI 框架，允许 Python 程序运行于 Windows、Unix 和 Macintosh 平台上。Tkinter 和其他工具箱之间的主要不同在于可移植性方面。几乎所有的工具箱都适用于一些特定系统，例如 KDE 绑定 Linux、Mac 工具箱绑定 Mac 都是支队一个特定平台提供支持的 GUI 实现，然而，Tkinter 却允许读者编写可在许多平台运行而不用任何改动的代码。

Tkinter 证明应用程序编程接口设计可与应用程序的事务历程分开创建。选择 Tkinter 作为自己的 GUI 环境时，主要必须考虑在何处防止正确的配件。完成可视化设计后，只需要把配件操作绑定到需要调用的具体函数上即可，至此图形接口准备停当。Tkinter 还允许快速地处理按钮和窗口 ，并定义其属性。设计和创建自己的接口后，可以改变应用程序的函数而不对 GUI 代码进行任何改变。

2.2 Tkinter 和 Pmw 控件简介

2.2.1 Tcl/Tk 概述

Tk 是一个流行的并经过认证的工具箱，由 John Ousterhout 开发，可处理 Windows、GUI 时间和用户交互作用。该工具箱作为 Tcl 的扩展加以提供。这就是为什么部分 Tkinter 是 Tcl 借口的原因所在。如果没有这些例程，管理 GUI 环境可能需要许多行代码的应用程序。

该工具箱最初是由 Ousterhout 在 Berkeleyde 加州大学开发作为 Tcl 的补充。在转到 Sun Microsystems 之后，Ousterhout 成立了一个名叫 Scriptics 的公司（现在叫 Ajuba），该公司致力于 Tk 和 Tcl 开发项目。

目前许多语言都是用 Tk，其中包括 Scheme、Perl 和 Python。Tkinter 是 Tk GUI 工具箱的 Python 接口。Tcl 是幕后语言，Tkinter 使用它和 Tk 工具箱通信。Tcl 和 Tk 都是开放源产品，目前正在进行开发的产品是 Scriptics 的工程师和 Tcl 用户社团的其他用户共同合作开发的一部分。Scriptics 负责源代码的并行版本系统（CVS，Concurrent Versioins System）库，并欢迎其他任何人提交源代码改动和小补丁。

2.2.2 Tkinter 的安装

下面介绍一下 Tkinter 的安装，对于 Linux 环境来说，Python 一般是在安装系统的时候直接装好的，操作系统安装的是 Python 2.4，但是 Tcl 和 Tk 则需要自己进行安装，下面给出了安装过程：

下载所需的包 Tcl/Tk8.4.1，下载文件 tcl8.4.13-src.tar.gz 和 tk8.4.1-src.tar.gz（官方网站 Scriptics 现在正在改版，下不到最新发布的 Tcl/Tk）

打开终端进入下载文件夹中（cd 命令），用解压命令 gunzip tcl8.4.1.tar.gz 和 gunzip tk.8.4.1.tar.gz 进行解压。

在终端下 cd unix 进入 unix 文件夹。

键入./configure --prefix=/usr --enable-shared 命令，以根据系统情况自动生成编译时所需的 Makefile 檔。

键入 make（回车），然后 make install 来安装 tcl 模块。

Tk 模块安装过程同上。

2.2.3 Pmw 控件组简介

在本项目中大量的使用了 Pmw 控件，因此需要来介绍一下 Pmw 控件：

Python 控件-----Pmw 是一个合成控件，以 Tkinter 为基类是完全在 Python 内写的。它们可以很方便地添加功能性应用，而不必写一堆代码。特别是，组合框和内部确认计划的输入字段放在一起是一个很有用的控件。

Pmw 的安装过程是很简单的，如下：

首先从 http://sourceforge.net/projects/pmw 上下载 Pmw.1.2.tar.gz 压缩包，进入到下载文件夹利用 gunzip 命令进行解压，把解压好的 Pmw 包复制到 Python2.4 →lib 文件夹中即可，这时在程序中 import Pmw 即可以调用 Pmw 的控件了。

2.3 Linux Shell 命令介绍

2.3.1 Shell 简介

Shell 是用户和 Linux 操作系统之间的接口。Linux 中有多种 shell，当前，默认 Shell 是 Bash Shell。Linux 系统的 Shell 作为操作系统的外壳，为用户与操作系统内核交互操作提供桥梁。

Shell 是一个命令语言解释器，具有一定数量的内建 shell 命令集。用户在提示符下输入的命令由 shell 先解释，然后交给 Linux 核心处理。

Shell 既可以交互方式提供处理，同时，也可以作为一种独立的程序设计语言来使用。shell 程序设计语言支持绝大多数高级语言中的常见方式，如函数、变量、数组和程序控制结构[8-12]。高效简洁的 shell 脚本程序为 Linux 系统管理员提

供了有力的帮助。

2.3.2 Shell 的种类

Linux 环境提供了多种 shell 为使用者提供方便。最常用的是 Bourne shell（sh）、C shell（csh）和 Korn shell（ksh）。

Bourne shell 是 UNIX 最初使用的 shell 种类。Linux 操作系统缺省的 shell 是 Bourne Again shell，它是 Bourne shell 的扩展，简称 Bash。Bash 在 Bourne shell 的基础上增加了很多特性，例如命令补全、命令编辑和命令历史表等功能，它还借鉴了很多 C shell 和 Korn shell 的优点，具有灵活强大的程序设计界面，同时又有很友好的用户接口。

C shell 是一种比 Bourne shell 更适合程序设计者使用的 shell，其语法类似 C 语言。Linux 为喜欢使用 C shell 的人提供了 Tcsh。Tcsh 是 C shell 的扩展，提供了包括命令行编辑、单词补全、拼写校正、历史命令替换、作业控制以及类似 C 语言的语法，它兼容 Bash shell，同时，又提供比 Bash shell 更多的提示符参数。

Korn shell 吸取了 C shell 和 Bourne shell 的优点，同时，完全兼容 Bourne shell。Linux 系统提供了 pdksh（ksh 的扩展），支持任务控制，可以在命令行上挂起、后台执行、唤醒或终止程序。

同时，Linux 操作系统还包括了一些流行的 shell 种类，例如 ash、zsh 等。每种 shell 都有一定的优势和用途，使用者可以根据自己的需要灵活选择。某些 shell 具有专利，选择的时候需要注意版权保护。用户在登录到 Linux 时由/etc/passwd 档来决定要使用哪个 shell[13-15]。

第 3 章 开发环境及工具介绍

3.1 运行环境

为了保证辅助管理系统运行的效率和可靠性，本系统开发环境配置如下：

（1）软件环境：

操作系统： Slax Linux

运行环境：Python 2.4

Shell 种类：Bash Shell

（2）硬件环境：

CPU：酷睿 E6200；

内存：2G；

Linux ext3 分区：10G；

Linux swap 交换分区：1G。

3.2 开发环境及工具

3.2.1 Vi 编辑器介绍

Vi 是 Visual interface 的简称，它是 Linux 环境中最常用的文字编辑器。Vi 与 Emacs 在 Unix 和 Linux 系统的编辑器中占有绝对的主导地位。Vi 提供了常用的文本编辑功能，例如输出、删除、查找、替换、块操作等，同时，使用者也可以根据自己的需要对其进行定制。定制功能是 Vi 的一大特色之处。

Vi 具有区别于传统编辑软件的操作模式，例如没有菜单，只有命令，且命令繁多。Vi 有 3 种基本工作模式：命令行模式、文本编辑输入模式和末行控制模式。

进行 Linux 下的程序设计开发工作，首先必须选择一种文本编辑器。本文选择 Vi 的增强版本 Vim 作为代码编辑器。

3.2.2 Vi+Python

使用 Vi 编辑器实现文本的编辑，并用调试或者编译工具对其实现编译并使其程序运行，这是所有的 Linux 程序员都了解的事情，这里亦是如此，由于 Python 语言是解释型语言，因此不同于 C 语言程序设计，对于其程序来说，只需要进行调试，而不需要进行编译，例如：将 Vi 编辑器编写好的代码程序保存到文件 abc 中，终端下用命令 python abc 进行执行即可实现对程序的调试和运行。

第 4 章 Linux 辅助管理系统分析及设计

4.1 系统分析

整个系统用例图如 4-1 所示：

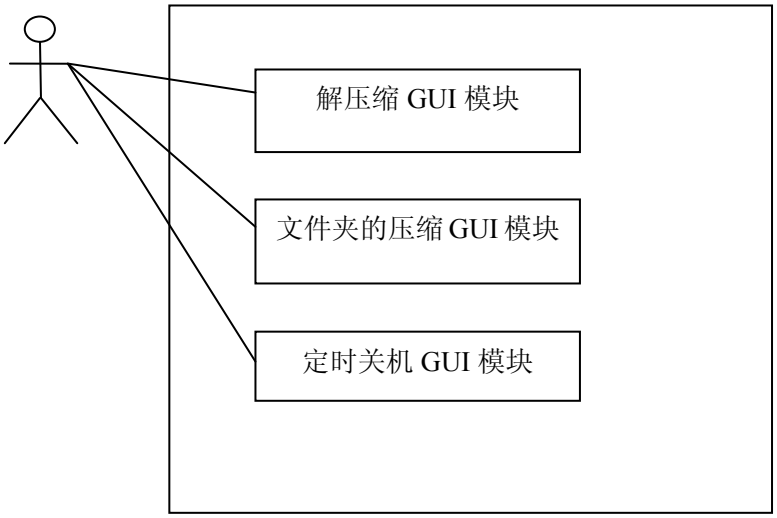

图 4-1 系统用例图

解压缩模块支持压缩文件的解压及打包文件的解包工作，流程图如图 4-2：

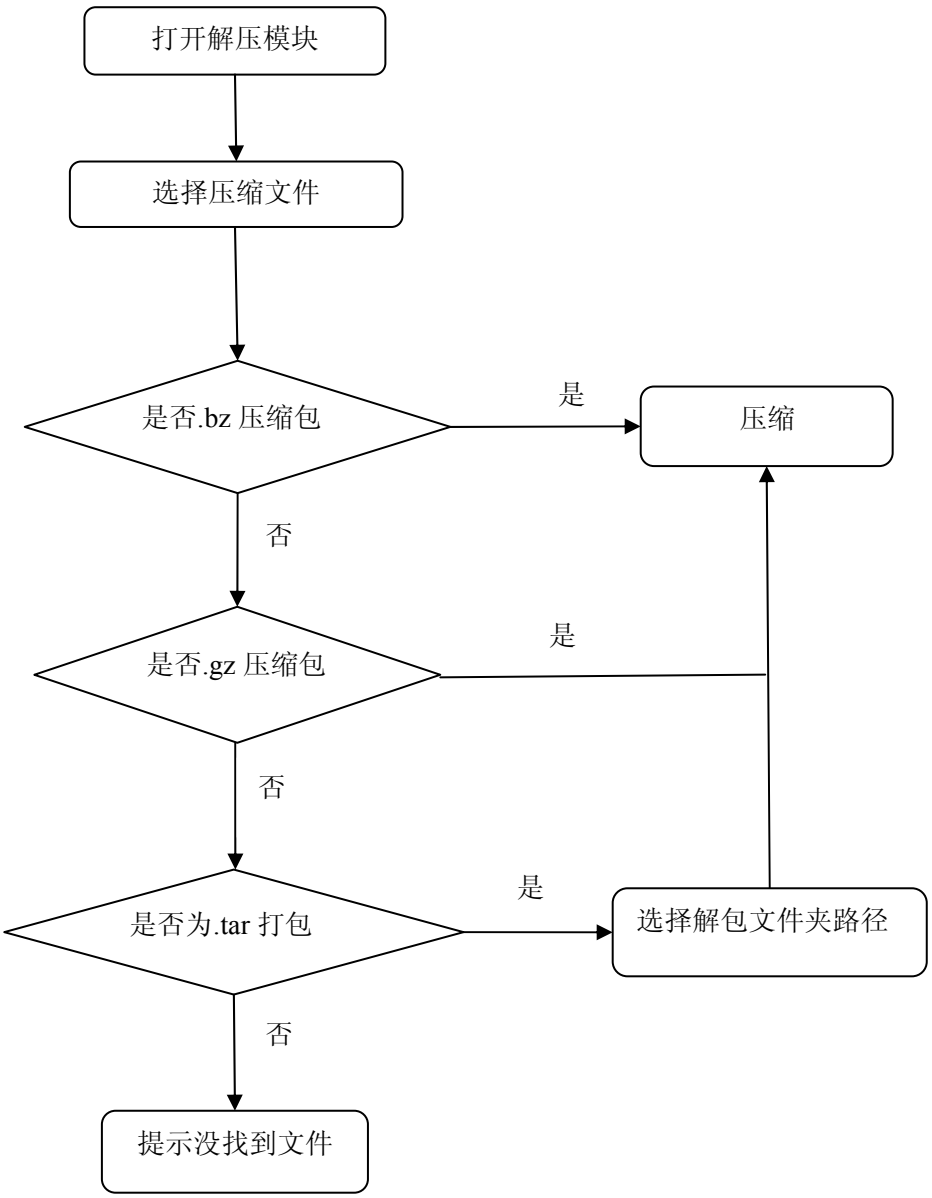

图 4-2 解压模块流程图

压缩模块支持文件夹的打包和压缩，流程图如图 4-3:

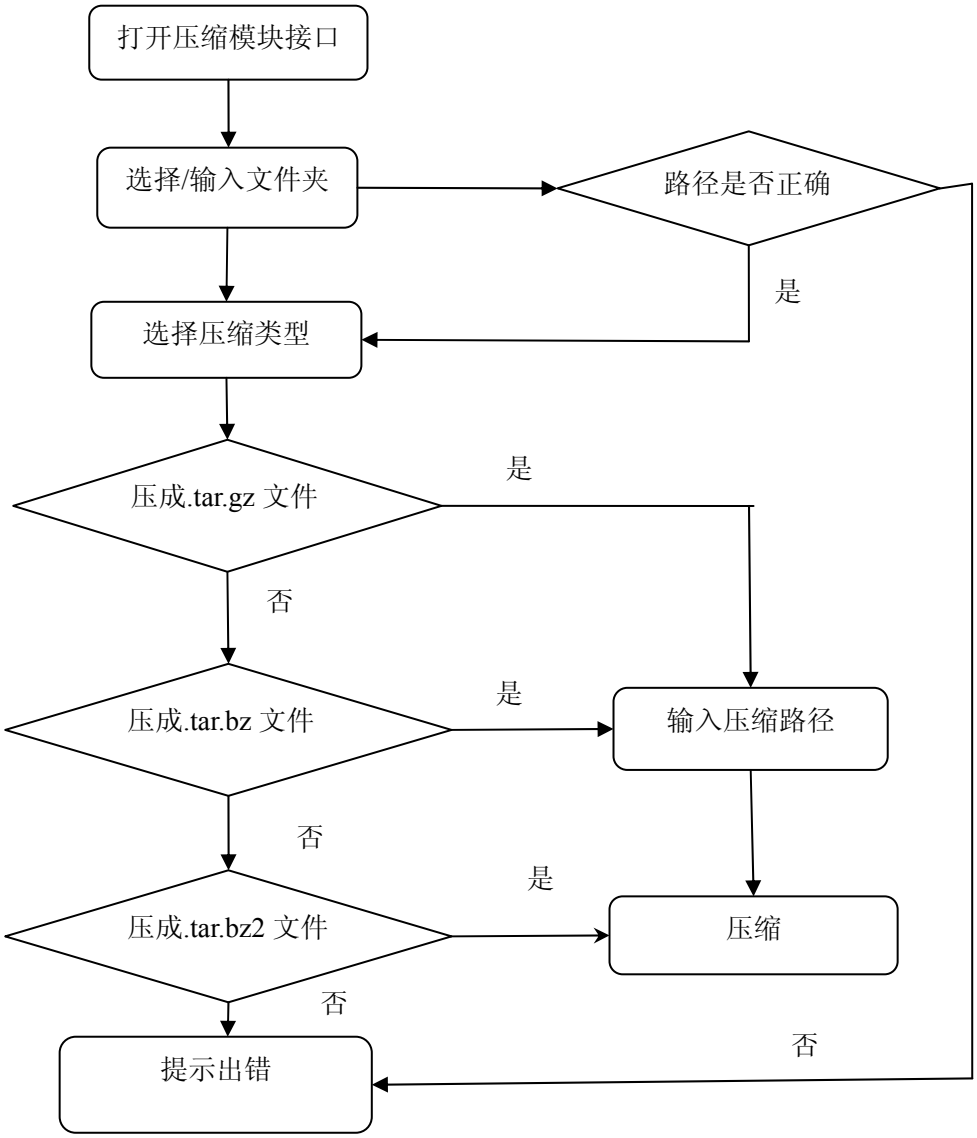

图 4-3 压缩模块流程图

定时关机模块流程图如图 4-4：

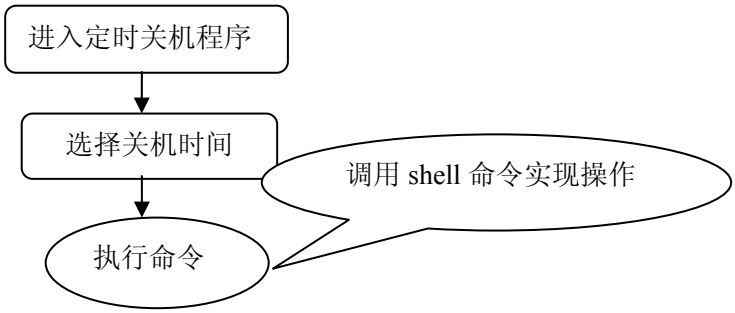

图 4-4 定时关机程序流程图

4.2 系统需要的 shell 命令说明

本系统的需要用到的 shell 命令如下[13-15]：

1．解压缩命令：

(1) .tar 包

解包： tar -xvf File_Name.tar

打包：tar -cvf File_Name.tar Dir_Name

（注：tar 是打包，不是压缩！）

---

(2) .gz 包

解压 1：gunzip File_Name.gz

解压 2：gzip -d File_Name.gz

压缩：gzip FileName

.tar.gz 和 .tgz

解压：tar zxvf File_Name.tar.gz

压缩：tar zcvf File_Name.tar.gz Dir_Name

---

(3) .bz2 包

解压 1：bzip2 -d File_Name.bz2

解压 2：bunzip2 File_Name.bz2

压缩： bzip2 -z File_Name

.tar.bz2

解压：tar jxvf File_Name.tar.bz2

压缩：tar jcvf File_Name.tar.bz2 Dir_Name

---

(4) .bz 包

解压 1：bzip2 -d File_Name.bz

解压 2：bunzip2 File_Name.bz

压缩：未知

.tar.bz

解压：tar jxvf File_Name.tar.bz

压缩：未知

---

(5) .Z 包

解压：uncompress File_Name.Z

压缩：compress File_Name

.tar.Z

解压：tar Zxvf File_Name.tar.Z

压缩：tar Zcvf File_Name.tar.Z Dir_Name

---

(6) .zip 包

解压：unzip File_Name.zip

压缩：zip File_Name.zip Dir_Name

.rar

解压：rar a File_Name.rar

压缩：rar e File_Name.rar

本项目中，选择实现 Linux 常用的“.tar”，“.gz”，“.bz”等几种类型来进行解压/压缩，或者打包的操作。

2．关机 shell 命令：

利用 Shutdown 命令实现自动关机，如下：

shutdown [-t 秒数] [-rkhncfF] 时间 [警告讯息]

-t 秒数 : 设定在切换至不同的 runlevel 之前, 警告和删除二讯号之间的延迟时间(秒).

-k : 仅送出警告消息正文, 但不是真的要 shutdown.

-r : shutdown 之后重新启动.

-h : shutdown 之后关机.

-n : 不经过 init , 由 shutdown 指令本身来做关机动作.(不建议用)

-f : 重新启动时, 跳过 fsck 指令, 不检查文件系统.

-F : 重新启动时, 强迫做 fsck 检查.

-c : 将已经正在 shutdown 的动作取消.

在本项目中，使用的是 shutdown －h <time>选项。

第 5 章 系统的实现

5.1 界面的代码实现

5.1.1 基础接口的实现

Linux 辅助管理的基础接口包括三个方面：

(1)解压/压缩模块的 Button 的实现

(2)定时关机的 Button 实现

(3)帮助 Button 的实现

在系统中，初始接口通过 App 类进行处理，解压/压缩和定时关机的 Button 分别实现了调用解压/压缩模块接口的功能，说明 Button 使用了 Tk 的控件组 Pmw 中的 About 控件，基于接口主类的 master 参数生成了一个说明接口，在其中包含了包括 Copyright，Tel，Email 等一系列信息。

基础接口如图 5-1 所示：

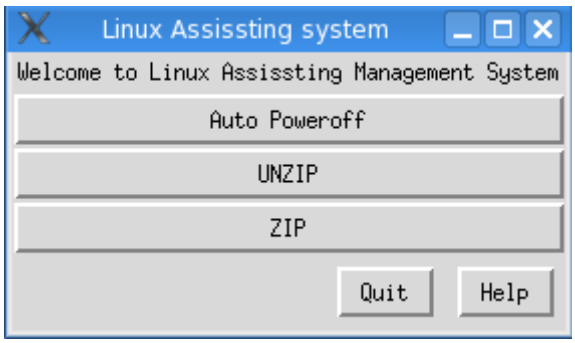

图 5-1 基础接口

代码如下：

```
class App:
----------------------省略初始化及接口制作部分--------------------
#create the help window
        Pmw.aboutversion('1.0')
      Pmw.aboutcopyright('Copyright Jason 2007\nAll rights reserved')
            Pmw.aboutcontact('For more information about this application
```

```
contact:\n'+
  'Tel:13630596776\nemail:zhangjunyao5518@hotmail.com')
self.about=Pmw.AboutDialog(master,applicationname='About this software')
        self.about.withdraw()
            #弹出输入接口的制作
def auto(self):
        wdw = Toplevel()
        windows.append(wdw)
        self.time = Pmw.TimeCounter(wdw,labelpos=W,label_text='Please enter the
time (HH:MM:SS)',min='00:00:00',max='23:59:59')
        self.time.pack(padx=20,pady=6,side=TOP)
           #关闭窗口
def closeWindow(win):
       win.destroy()
       if win in windows: windows.remove(win)
       if not windows: root.quit()
       windows.append(wdw)
           #显示说明窗口
def    help(self):
     self.about.show()
```

代码说明：首先由于 Python 语言是缩进式的语言，因此并没有出现像 C 或者 C++一样的大括号一样的格式，另外，在本段代码中，两次利用了 Pmw 的控件，分别是说明控件 Pmw.About 和 Pmw.TimeCounter 控件，前者实现说明功能，后者生成输入时间的窗口如图 5-2，在后面的代码中进行时间处理。

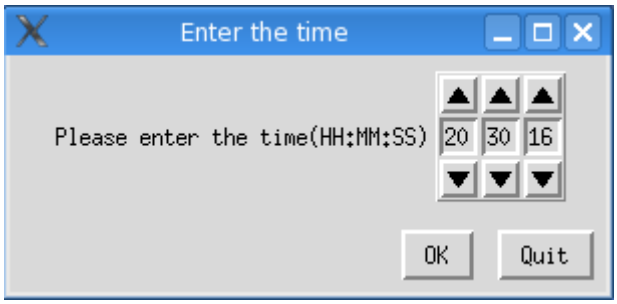

图 5-2 输入时间接口

5.1.2 解压/压缩接口的实现

解压/压缩的接口主要基于控件 Dialog，通过它进行接口生成和操作，其中还有一些细节问题和解面调用的生成过程，这里需要进行一些说明：

在基础界面上摁下 UNZIP 按钮，即可以得到解压缩的界面，要求输入一个文件的路径名称，如图 5-3 所示：

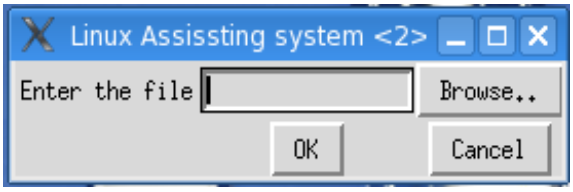

图 5-3 解压缩输入界面

在这个界面基础上，可以输入文件路径，或者摁下“Browse..”，即弹出文件选择的界面，这个界面的生成是调用了 Python 提供的 Dialog 库中的一个 tkFileDialog,代码如下：

```
import tkFileDialog
        filename = tkFileDialog.askopenfilename(title='open')
```

在上段代码中，askopenfilename 方法返回一个值，即打开的文件的名称，其默认路径是当前打开文件的路径(当然也可以自己设置默认路径)，界面效果如图 5-4 所示：

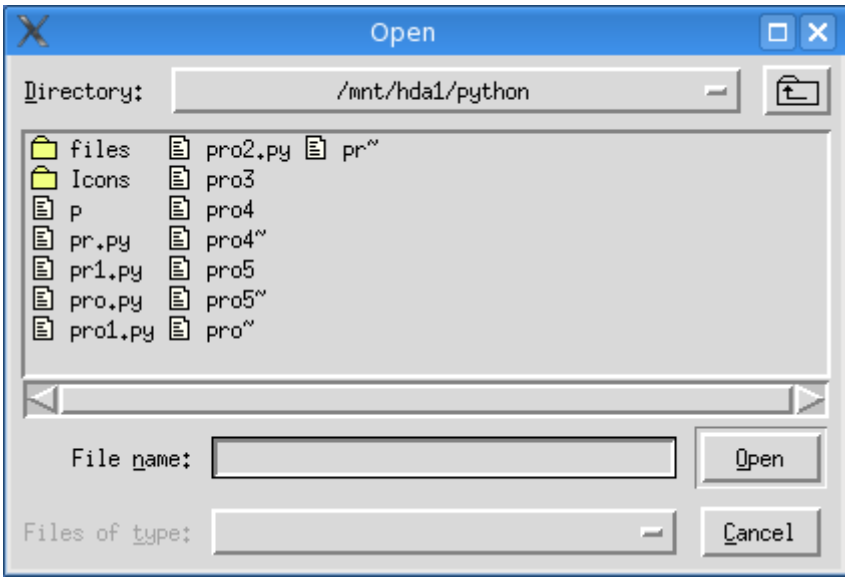

图 5-4 选择文件解面

ProgressBar 类：这个类基于 Tk 控件 Canvas（画布）生成进度条，用于解压缩过程中使处理事件的时候能更逼真，更生动些。

类 ProgressBar 的代码如下：

```
class ProgressBar:
  def __init__(self, master=None, orientation="horizontal",
                       min=0, max=100, width=100, height=18,
                   doLabel=1, appearance="sunken",
                   fillColor="blue", background="gray",
                       labelColor="yellow", labelFont="Verdana",
                       labelText="", labelFormat="%d%%",
                       value=50, bd=2):
----------------------------------------------------(省去初始化部分)
          self.canvas=Canvas(self.frame, height=height, width=width,bd=0,
                       highlightthickness=0, background=background)
      self.scale=self.canvas.create_rectangle(0, 0, width, height,
                   fill=fillColor)
              self.label=self.canvas.create_text(self.canvas.winfo_reqwidth() / 2,height / 2,
text=labelText, anchor="c", fill=labelColor,
  font=self.labelFont)
  self.update()
  self.canvas.pack(side='top', fill='x', expand='no')
```

```
def    updateProgress(self, newValue, newMax=None):
    if newMax:
        self.max = newMax
        self.value = newValue
        self.update()

 def    update(self):
    # Trim the values to be between min and max
    value=self.value
    if value > self.max:
     value = self.max
    if value < self.min:
        value = self.min
  # Adjust the rectangle
    if    self.orientation == "horizontal":
        self.canvas.coords(self.scale, 0, 0,
        float(value) / self.max * self.width, self.height)
     else:
        self.canvas.coords(self.scale, 0,
        self.height - (float(value) / self.max*self.he
                                self.width, self.height)
     # Now update the colors
     self.canvas.itemconfig(self.scale, fill=self.fillColor)
     self.canvas.itemconfig(self.label, fill=self.labelColor)
     # And update the label
     if self.doLabel:
        if value:
            if value >= 0:
                pvalue = int((float(value) / float(self.max)) * 100.0)
            else:
                value = 0
                self.canvas.itemconfig(self.label, text=self.labelFormat % value)
```

```
        else:
            self.canvas.itemconfig(self.label, text='')
    else:
        self.canvas.itemconfig(self.label, text=self.labelFormat % self.labelText)
        self.canvas.update_idletasks()
```

这段代码中，首先通过 Canvas 控件创建了一个框架，然后在其中建立矩形图形，再对所创建的图形进行颜色填充，最后通过 update 函数在发生事件的时候对其进行处理，即从最小值走到最大值。

利用这个类制作一个解压和压缩时的进度条，其界面如图 5-5 所示

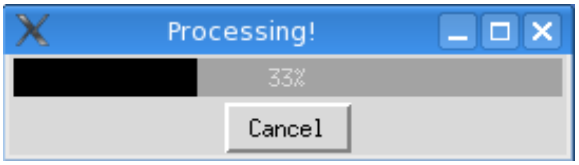

图 5-5 进度条界面

压缩界面的制作与解压界面大致相同，但有些细节上的差别，首先，摁下基础界面上的 ZIP 按钮，即弹出窗口要求输入文件夹名称，及选择要求压缩成的文件类型。如.gz，.zip 等。

代码如下：

```
def zip(self):
    wdw2=Toplevel()
    windows.append(wdw2)
    Label(wdw2,text='Enter the file').grid(row=0,column=0)
    entry=Pmw.EntryField(wdw2,labelpos=W,label_text='Select the
directory',validate=None)
    entry.grid(row=0,column=0)
    Button(wdw2,text='Browse..',command=self.com).grid(row=0,column=1)
    butto1=Button(wdw2,text='OK',command=self.extract)
    butto2=Button(wdw2,text='Cancel',command=sys.exit)
    butto1.grid(row=2,column=0)
    butto2.grid(row=2,column=1)
    combobox=Pmw.ComboBox(wdw2,label_text='Select the compress
```

```
type',labelpos=W,
        listbox_width=24,dropdown=1,scrolledlist_items=typ)
    combobox.grid(row=1)
    combobox.selectitem(typ[0])
```

在这段代码中，首先用 TopLevel 生成了一个窗口，然后在其中添加所需要的控件，引用了两个 Pmw 控件，一个是 Pmw.EntryField，另一个是 Pmw.ComboBox，分别用来添加输入框和组合框的。如图 5-6 所示：

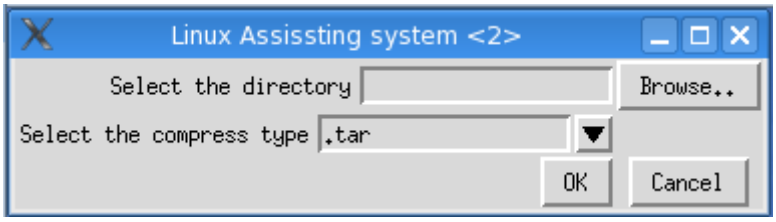

图 5-6 压缩界面

在压缩界面中，点击 Browse 按钮，进入选择文件夹界面，代码如下：

```
def com(self):
    co=tkFileDialog.askdirectory(title='Select the Directory')
```

这里调用了 tkFileDialog.askdirectory 组件，其返回值是文件夹的名字。如图 5-7：

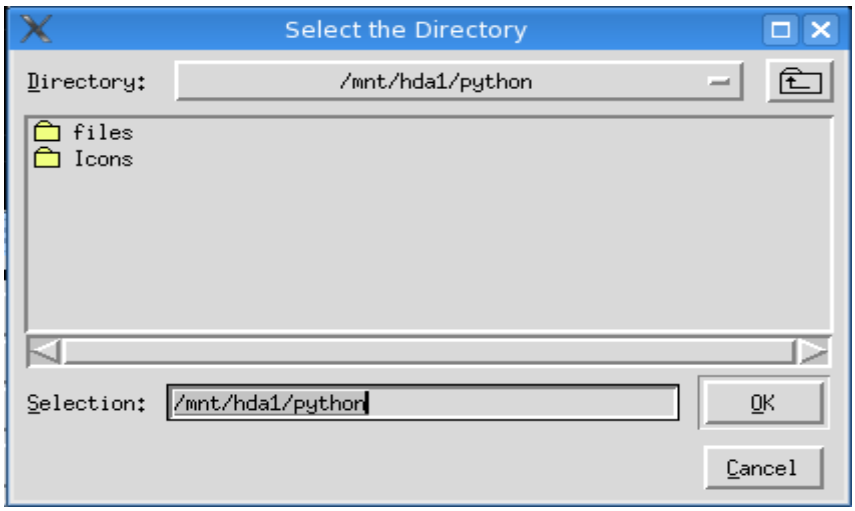

图 5-7 选择文件夹界面

5.2 事件响应及命令调用部分说明

时间响应部分大部分是包含在上面所说的类中，如 Button 中的一个属性 command，在基础接口的 quit(Button)中，command 调用了一个 Python 的函数库 os 中的函数 exit(0)，这样使得程序能安全退出。

在这里首先说明一下在 Python 程序中调用 Shell 命令的方法：

需要在程序的开始“import os”，这条语句使得程序调用 os 库。

在程序中需要调用 shell 命令的地方，os.system(‘shell commands’)，这里是调用了 os 库中的 system 函数，实现运行 shell 命令的功能[16-32]。

5.2.1 定时关机模块的事件处理

在定时关机模块中，有一个参数传递过程，即从接口上得到的输入的时间，在按 OK 按钮以后需要将参数传递给事件响应系统，然后调用 os.system 函数，调用 shell 命令“shutdown –h ‘time’”使得功能得以实现。

具体代码如下：

```
def TimeEventt(self):
    t=’shutdown –h %s &’ % self.time.getstring()
    os.popen(‘t’)
```

这里将 shutdown 等字符串赋给变量 t，然后使用 os.system 调用 t 的命令即可，这里选择了在后台执行这个命令(&)，以防止前台界面出现卡死的情况。

5.2.2 解压模块的事件处理

解压模块中大致分成两个部分，一个是压缩文件解压的部分，另外一个是.tar 文件的解包部分，这里根据不一样的文件类型选择调用不同的 shell 命令，并弹出不同的窗口，详细 shell 命令参见第四章第二节。

具体实现代码实现如下：

```
----------(省略非核心部分)----------
def unzip(self):
entryget= entry.get()
    filepath,filename = os.path.split(entryget) #将 filename 设置成全局类型的，以方
```

便后面调用

```
def extract(self, name):
    import re
t = 'tar xvf %s ' % name
    z = 'unzip %s' % name
    g = 'gunzip %s' % name
    b = 'bunzip %s' % name
    if re.match('.tar$',name):
p1=os.popen(t)
p1.close()
    if re,match('.zip$',name):
 p2=os.popen3(z)
 p2.close()
    if re.match('.gz$',name):
 p3=os.popen3(g)
 p3.close()
    if re.match('.bz2$',name):
p4=os.popen3(b)
p4.close()
```

在这段代码中，先将命令赋给变量,这里引用了正则表达式的 match 方法来判断 name 中是否有文件类型.tar, .gz 等等，并按照相应类型执行不同解压命令[16-32]。

5.2.3 压缩模块的事件处理

压缩模块将输入区输入的档或者文件夹名称传给事件响应系统，另外用户需要选上所想得到的压缩包的类型(利用 RadioButton 控件)，与解压缩相同，其调用不同的 shell 命令来进行压缩。

具体代码如下：

```
  def Compresstar(self, name, path):
      t1 = 'tar zcvf %s %s ' % name path
      p1=popen(t1)
      p1.close()
```

-------后面不同类型定义不同的函数，但是格式是相同的-----

这段代码的基本思路为在选择界面中选择好文件夹，在 ComboBox 中选择想要压缩的文件类型，点击 ok 键，弹出想要将压缩包放在的文件夹，即进行压缩工作，这里的 path 为输入路径的 asksavefile()返回的值[16-32]。

第 6 章 实例测试

本章用实际的用例来测试辅助管理系统。由于本论文篇幅的限制，选用 Linux 的 root 用户登录，以便实现一些特殊权限的更能，展示测试过程。

6.1 定时关机测试

运行项目，打开基础接口，点击 Auto Poweroff 按钮，得到结果如图 6-1 所示：

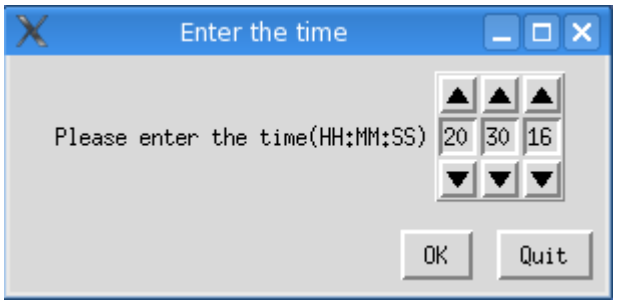

图 6-1 定时关机测试

当前时间为 20：30：16，在代码里添加一条语句，在后台 print 出关机的时间，然后输入一个时间，如：20：29：04 得到如图 6-2，然后在 20：31：00 自动关闭计算机。

图 6-2 定时关机测试 2

6.2 解压模块测试

在本组测试中，在路径/mnt/hda1/下载了一个.zip 的文件 pylongopgui.zip，如图 6-3 所示：

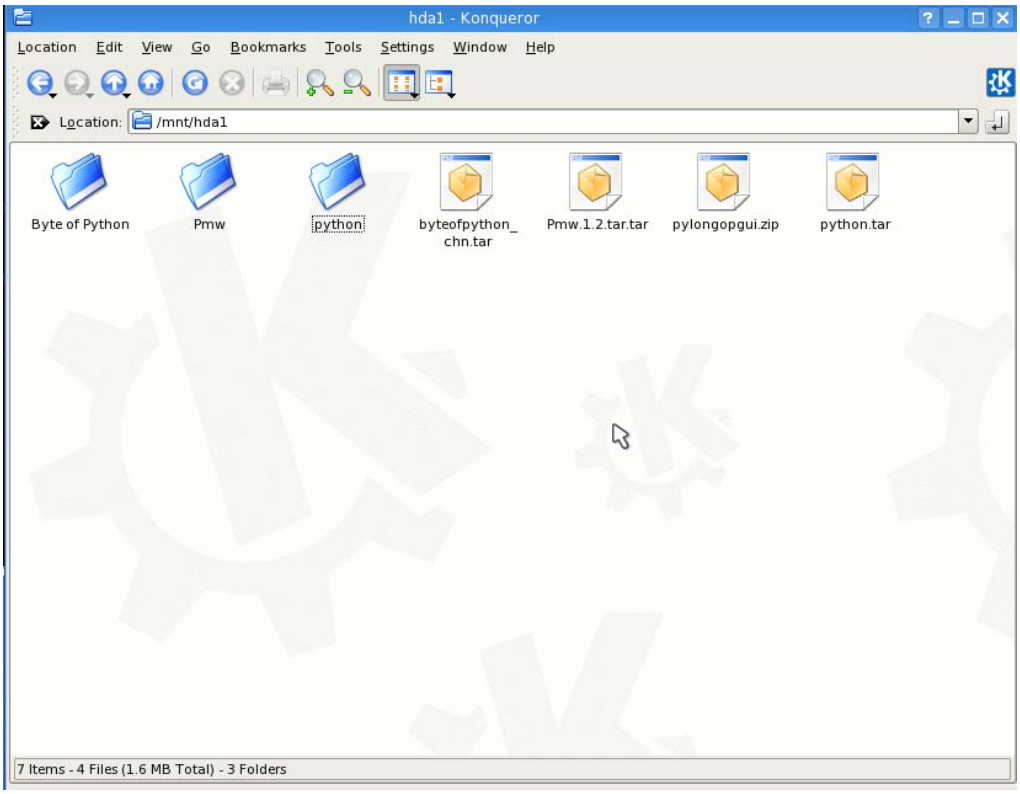

图 6-3 文件夹下的 ZIP 文件

在基础接口中，点击“ZIP”按钮，即可进入解压界面，选中一个文件，输入路径，如图 6-3，按下 OK 按钮，得到解压后的文档。若没有输入路径或者路径错误，将会得到对话框，提示错误如图 6-4：

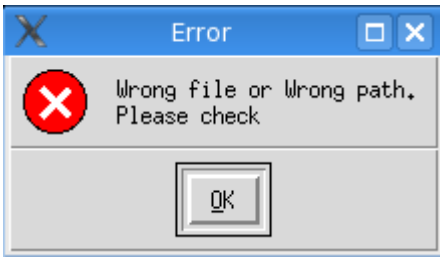

图 6-4 路径错误

输入路径正确，开始进行解压，调用进度条，解压完成以后得到文件夹在同样的路径下，如图 6-5 所示：

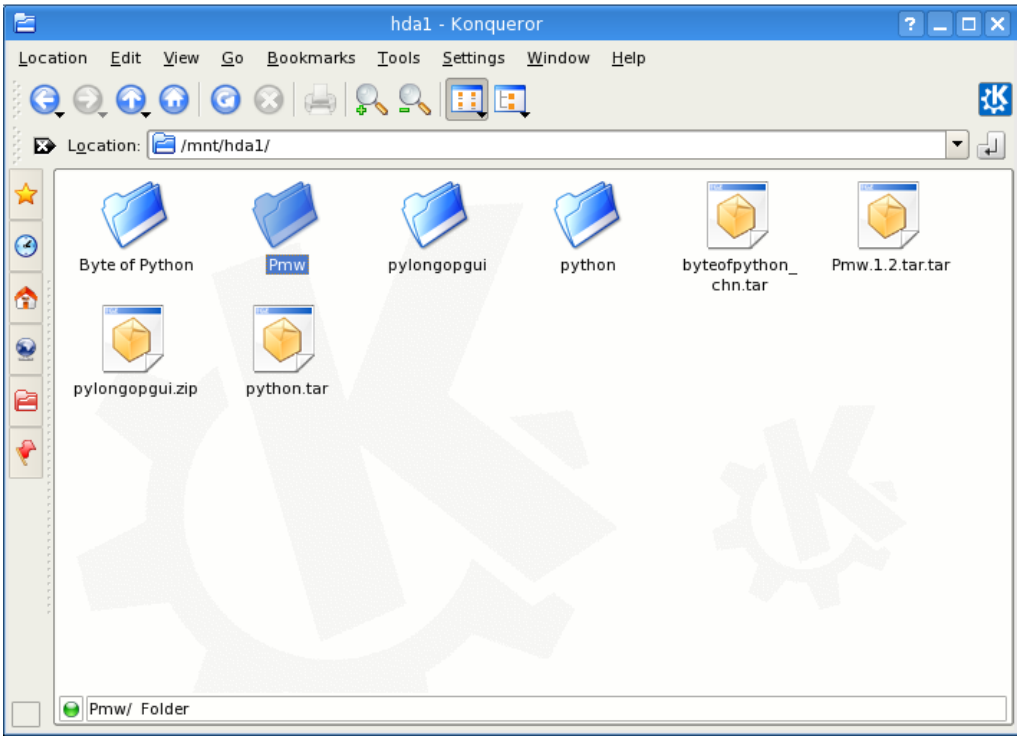

图 6-5 解压完毕后

6.3 压缩模块的测试

目录为/mnt/hda1/python 的文件夹需要压缩，如图 6-6 所示：

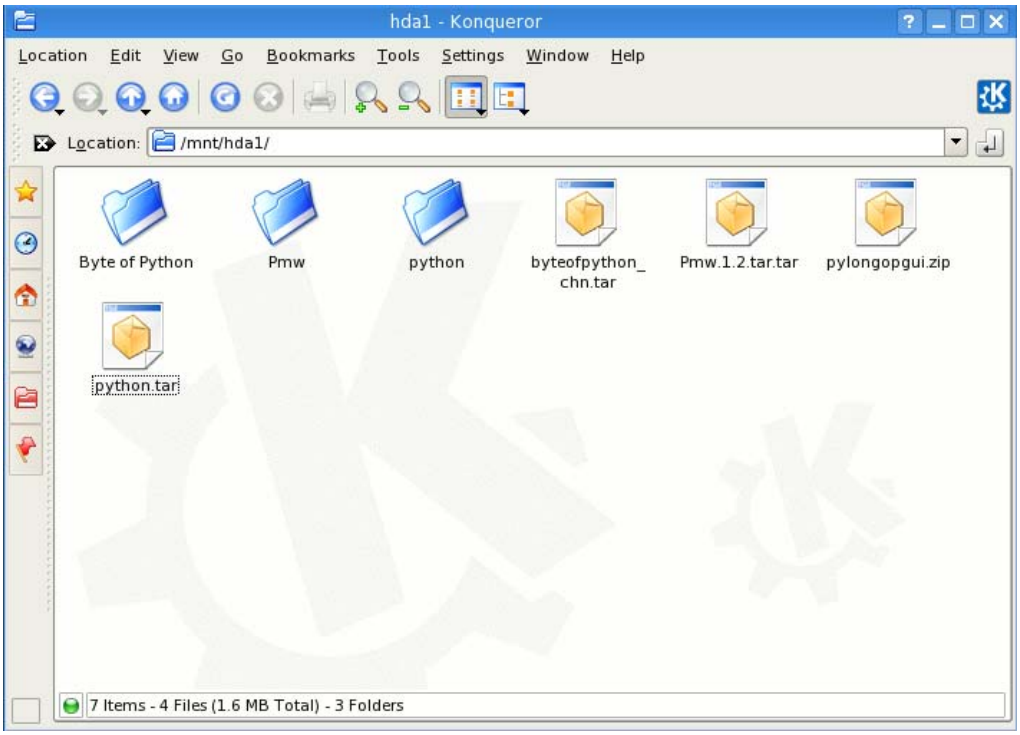

图 6-6 压缩前目录情况

进入基础界面后点击 ZIP 键，进入到选择文件界面选择文件夹如图 6-5：

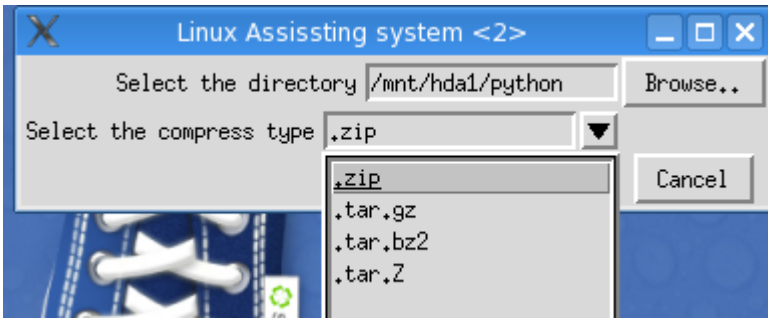

图 6-7 Zip 选择界面

选择以后，开始进行压缩，完成后生成 python.tar.gz 文件的文件夹如图 6-8 所示：

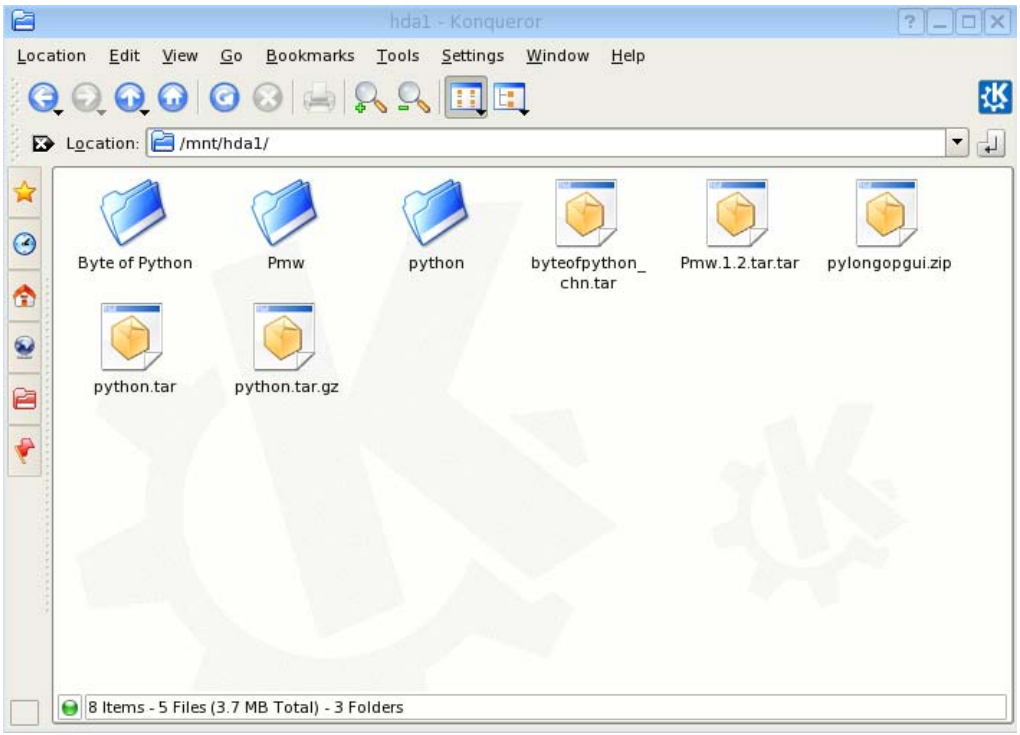

图 6-8 压缩后的文件夹

至此，程序测试完毕。

第 7 章 总结与展望

7.1 总结

随着 Linux 技术的日益发展，Linux 操作系统得到了普遍应用。特别是在个人桌面系统开始迅速普及。在个人桌面系统的普及中，由于用户计算机操作水平有限，传统 Linux 系统繁琐的命令行交互操作方式成为用户应用 Linux 系统的最大障碍，一个关键的问题是解决用户的易用性。GUI 图形化界面的接口为用户提供了方便快捷的操作。如何将 Linux Shell 下的大量应用软件配置一个快速方便的图形化接口，从而有效提高操作的易用性，更好地辅助系统管理是亟待解决的问题。

针对以上问题，本文基于当前广泛使用的 Python 语言，结合 Tkinter 和 Pwm 模块扩展，为 Linux Shell 应用软件提供一个方便简洁的图形化接口。本文对相关技术进行了详细介绍，对可行的 GUI 图形化方式进行了讨论，并指出 Python+Tkinter 方式快速构建图形化用户界面的可行性和优越性。同时，通过一个具体的压缩与解压命令以及定时自动关机 shell 指令的图形化系统辅助程序设计实例，详细阐述了 Python+Tkinter 实现 Linux 图形化辅助管理工具的细节和技术难点。最终完成的辅助管理系统具有良好的易用性和稳定性，达到了简化用户操作，提高 Linux 系统易用性的目的。

7.2 展望

本文实现的 Linux 图形化辅助管理系统只实现了最具代表性的几个常用 shell 交互命令的图形化用户交互界面应用，还有更多的复杂的命令行程序需要完成相关操作。进一步的工作主要包括以下几点：

1. 继续研究和分析 Python 及其 Tkinter 的实现机制和设计细节，优化和完善已有图形化方法；
2. 设计和完成具有更普遍应用价值和更方便快捷的通用图形化控件，为快速批量完成传统 shell 命令行程序的图形化整合工作提供可能；

3. 引入人机工程学、视觉艺术设计等交叉学科的知识，进一步在操作的人性化和视觉色彩、窗口布局等多方面进行优化提高。

参考文献

[1] Andre Lessa 著，张晓晖，张晓昕，王艳斌等译，深入学习：Python 程序开发 = Python developer's handbook[M]，北京：电子工业出版社，2001 年

[2] Lutz，Mark，Learning Python（第二版）[M]，南京：东南大学出版社，2005 年

[3] 肖建等编著，Python 程序设计基础[M]，北京：清华大学出版社，2003 年

[4] Martin C. Brown 著，康博译，Python 技术参考大全[M]，北京：清华大学出版，2002 年

[5] Grayson,John E.著，陈文志等译，Python 与 Tkinter 程序设计[M]，北京：国防工业出版社，2002 年

[6] Martin C. Brown 著，邱仲潘等译，XML 与 Perl、Python 和 PHP 程序设计指南[M]，北京：电子工业出版社，2002 年

[7] Dave Brueck，Stephen Tanner 著，陈河南，王晓娟等译，Python 2.1 宝典[M]，北京：电子工业出版社，2002 年

[8] Keogh,J.著，王崧等译，轻松学用 Linux 程序设计[M]，北京：电子工业出版社，2001 年

[9] K.Wall，M. Watson，M. Whitis 著,王勇等译，GNU/Linux 程序设计指南[M]，北京：清华大学出版社，2000 年

[10] David A.Rusling 等著，朱珂，涂二等译,Linux 程序设计白皮书[M]，北京：机械工业出版社，2000 年

[11] Goerzen,J.著,魏永明等译，Linux 程序设计宝典[M]，北京：电子工业出版社，2000 年

[12]杨树青，王欢著,Linux 环境下 C 编程指南[M]，北京：清华大学出版社，2007 年

[13] David Tansley 著，徐焱，张春萌等译，LINUX 与 UNIX Shell 程序设计指南[M]，北京：机械工业出版社，2000 年

[14] Bruce Molay 著，杨宗源，黄海涛译，Unix/Linux 程序设计实践教程[M]，北京：清华大学出版社，2004 年

[15] Petersen,R 著，梁普选等译，Linux 程序设计命令详解(第二版)[M]，北京：

电子工业出版社，2001 年

[16] 方敏，张彤，网络应用程序设计[M]，西安：西安电子科技大学出版社，2005 年

[17] 胡亮，李强，康健，Linux 试验教程[M]，长春：吉林大学出版社，2003 年

[18] 宋国伟，GTK+2.0 编程范例[M]，北京：清华大学出版社，2002 年

[19] Rikke D Giles, Graphical Interface Development with Glade2[M]，Addison Wesley，2003 年

[20] Stevens,UNIX Network Programming (Third Edition) [M]，Addison Wesley，2000 年

[21] George Lebl, Application Programming Using the Gnome Libraries[M]，Addison Wesley，1999 年

[22] Tony Gale, Ian Main, GTK+2.0 Tutorial[M]，2002 年

[23] 任泰，TCP/IP 协议与网络编程[M]，西安：西安电子科技大学出版社，2004 年

[24] Sarwar, AI-Saqabi 著，英宇，姚锋译，LINUX & UNIX 程序开发基础教程[M]，北京：清华大学出版社，2004 年

[25]唐靖飚，Unix 平台下 C 语言高级编程指南[M]，北京：电子工业出版社，2002 年

[26]Redhat Linux, http://www.redhat.com[EB/OL]

[27]Debian GNU/Linux, http://www.debian.org[EB/OL]

[28]Linux X-Windows System,http://www.experts-exchange.com/Operating_Systems/X_Windows/[EB/OL]

[29]GNU,GCC, the GNU Compiler Collection, http://gcc.gnu.org/[EB/OL]

[30]W.Richard Stevens, Bill Fenner, Andrew M.Rudoff, UNIX Network Programming, Volume 1: The Sockets Networking API, Third Edition[M], Addison-Wesley,2004 年

[31]Andrew S.Tanenbau，计算机网络（第 4 版）（英文影印版）[M]，北京：清华大学出版社，2005 年

[32]W.Richard Stevens，Stephen A.Rago,Advanced Programming in the UNIX Environment（Second Edition）[M]，北京：人民邮电出版社，2006 年

致 谢

从接受课题到现在完成毕业设计论文，衷心的感谢我的指导老师。他给予了精心的指导和热情的帮助，尤其在课题设计的前期准备阶段和后期调试阶段，导师提出许多宝贵的设计意见，使得我得以顺利的完成毕业设计开发工作。在和老师的学习中，老师渊博的知识、敏锐的思路和实事求是的工作作风给我留下了深刻的印象，这将使得我终身受益，谨此向老师表示衷心的感谢和崇高的敬意。

感谢在百忙之中为本文审稿的各位老师，谢谢！

摘 要

基于 Python+Tkinter 的 Linux GUI 辅助管理工具的设计与实现

随着 Linux 技术的日益发展，其强大的功能已为人们深刻认识，它已进入社会的各个领域并发挥着越来越重要的作用，特别是在个人桌面系统开始迅速普及。在个人桌面系统的普及中，一个关键的问题是 Linux 中应用程序的易用性。GUI 图形化界面的接口为用户提供了方便快捷的操作。如何将 linux Shell 下的大量应用软件配置一个快速方便的图形化接口，从而有效提高操作的易用性，更好地辅助系统管理是亟待解决的问题。

针对以上问题，本文基于当前广泛使用的 Python 语言，结合 Tkinter 和 Pwm 模块扩展，为 Linux Shell 应用软件提供一个方便简洁的图形化接口。同时，通过一个具体的压缩和解压命令的图形化系统辅助程序设计实例，详细阐述了用 Python+Tkinter 实现 Linux 图形化辅助管理工具的细节和技术难点。

在具体实现过程中，重点针对 Python 语言的特点和对 Tkinter 以及 Pwm 模块实现的具体细节进行了详细分析，在此基础上，结合 Linux 系统处理 shell 命令的具体实现机制，设计并实现了基于 python+pwm 模块的 tar、compress、uncompress 以及 shutdown 等系统命令的图形化用户交互界面。同时，考虑用户的操作习惯和对 Windows 系统的熟悉程度，在窗体风格、控件布局以及鼠标和键盘响应方式等多方面进行优化，以求操作的简单方便。

通过运行和调试，本文实现的 Linux GUI 辅助管理实例运行稳定，操作简单，验证了本文提出的图形化实现方法的可行性和合理性。同时，对今后更多 Linux Shell 下的图形化辅助管理软件的设计与实现工作具有重要参考价值和借鉴意义。

本文主要内容的组织如下：

第 1 章是前言，介绍了 Linux 操作系统的背景，并详细说明了开发 Linux 辅助管理系统的原因，Linux 辅助管理系统的设计目标、系统需求和功能需求等。

第 2 章介绍了开发本系统的相关技术和知识，主要包括：Python，Tkinter 控件，Pmw 控件和 Linux shell 命令。具体来说，本系统是在 Linux 系统下对其 Shell 命令和 GUI 设计程序设计的一种扩展。本系统在 Slack Linux 系统下以 Python 语言编写，接口部分主要基于 Python 的 Tkinter 模块，功能实现部分主要依靠调用 Linux 下的 Shell 命令来实现。

第 3 章介绍了用于开发本系统的开发环境和工具，主要是 Vi(Visual interface)编辑器+Python。使用 Vi 编辑器实现 Python 代码的编辑，然后调用 Python 解释工具对 Python 代码解释执行，亦可调试。

第 4 章介绍了本系统的系统分析和设计，详细说明了解压/压缩模块和定时关机模块的流程，同时对本系统用到的 Shell 命令进行了说明。

第 5 章介绍了本系统的详细设计和实现，主要包括界面代码的实现和事件响应及命令调用部分说明，并给出了相关核心代码。前者主要包括基础接口和解压/压缩接口的实现，后者主要包括定时关机模块的事件处理，定时关机模块的事件处理和压缩模块的事件处理。

第 6 章介绍了本系统运行和测试情况，主要包括定时关机测试、解压模块测试和压缩模块测试。通过测试，本文实现的辅助管理系统运行稳定，操作简单，验证了本文提出的图形化实现方法的可行性和合理性。

第 7 章对本文的研究作了一个总结，并对未来的工作提出了展望。

关键字:

Python，Tkinter，图形化辅助工具

Abstract

Design and Implementation of GUI Aided Management Tool in Linux Based on Python+Tkinter

With the development of the Linux OS technology，it has gradually become more and more important in nearly every aspects in people's lives. And it seems that Linux gets more favorite by enterprises during to its stability and other advantages.

Generally speaking, the GUI(Graphical User Interface) is essential for every OS, including the Linux OS. For Linux, the GUI design is a promising aspect, for it makes more convenient for the studying of the new learners. And GUI provides users with convenient and fast operation. How to configure a GUI for the software in Linux which help users operate the software more effectively is significative.

To solve the above problem, this paper uses the Python language, Tkinter and Pwm modules to develop a simple and convenient GUI to the Linux Shell application. And it uses a specific compression/decompression GUI aided system as a sample to describe the details and technical difficulties of developing Linux aided management tools with Python and Tkinter.

According to the experiment results, The GUI Aided Management developed in the paper runs stability, and requires basic principles of GUI programming. It proves that the GUI realization method proposed in the paper is feasible and rationale. And it is valuable for the later design and implementation of GUI aided management softwares in Linux later.

The mainly structure of this paper is as follows:

Chapter 1 mainly introduces the background of Linux OS, and describes the reason of developing the GUI aided management system in Linux, the design goals of the system, the system requirement of the system and the function requirement of the

system.

Chapter 2 mainly introduces the technology and knowledge used in developing the system, which includes the Python language, Tkinter module, Pmw module and Linux shell command. In details, the system is an extension of Shell commands and GUI programming in Linux. The system is developed in Python language in Slack Linux OS. The implement of the interface part is based on Tkinter module, and the implement of function part is based on Linux shell command.

Chapter 3 mainly introduces the develop environment and tools used: Vi(Visual interface) editor + Python.

Chapter 4 mainly introduces the system analysis and design of the system, and describes in details the flow chart of compression/decompression module and the timing shutdown module, and explains the Linux shell command used in the system.

Chapter 5 mainly introduces the detailed design and implement of the system. And it mainly describes the implement of interface, event handler and command invoking description. The former includes the implement of basic interface and the implement of compression/decompression interface. The latter includes the event handler of the compression module, the decompression module and the timing shutdown module.

Chapter 6 mainly describes the running and testing condition of the system. According to the experiment results, the system proposed in this paper, which runs stably and requires basic principles of GUI programming, is valuable for later studies.

Chapter 7 summarizes the study of the paper.

Keywords:

Python, Tkinter, GUI Aided Management Tool