

电子科技大学
计算机科学与工程学院

标准实验报告

(实验) 课程名称 软件开发综合实验

电子科技大学教务处制表

电子科技大学

实验报告

小组：XXX	组长（学号）、组员（学号）、组员（学号）
组长姓名：张镕麒	学号：2019081301026
一、实验室名称：基础实验大楼 A502	
二、实验项目名称：文件备份软件	
<p>三、实验目的：</p> <p>本课程的教学目标是为学生提供理论与实践相结合的基础平台，以计算机科学与技术专业必修课程《程序设计（C 与 C++）》、《数据结构与算法》和《软件工程》三门课程所涉及的主要知识为核心，基于具体的工程项目，加深学生对理论知识的理解，使学生能够解决计算机系统的复杂工程问题，达到选择或设计恰当的逻辑结构、存储结构及相应的算法的培养目标，培养学生在实际软件生产开发过程中对其所学知识的综合运用能力，提升学生的计算思维能力、算法设计与分析能力、程序设计与编程实现能力、团队协作能力、计算机系统的认知、分析、设计和运用能力，工程实践能力。</p>	
<p>四、实验内容：</p> <p>实验内容：</p> <ol style="list-style-type: none">设计并实现一款数据备份软件，以项目组形式推进，每组最多三人。基于软件工程方法学进行项目推进，经历从需求分析、系统设计、编码实现、软件测试的整个软件生命周期。实验最终成果包括一款基本可用的软件及其对应文档。软件应包括指明的完整功能，重点考察其正确性、易用性、健壮性。软件文档应包括：需求分析说明书、系统设计文档、软件测试报告，重点考察其规范性、一致性、可读性。采用现代化软件开发工具辅助项目开发，包括但不限于：项目管理工具，UML 建模工具，集成开发环境，版本控制工具，软件测试工具。 <p>该实验包括必选的基本要求和可任意选择的扩展要求：</p> <p>基本要求：</p>	

各小组“独立”实现一款数据备份软件（对应基础分总分 30 分）：

数据备份：将目录树中的文件保存到指定位置

数据还原：将目录树中的文件恢复到指定位置

扩展要求：

各项目组根据自身情况自行选择扩展要求（对应扩展分总分）。

文件类型支持（10 分）：支持特定文件系统的特殊文件（管道/软链接/硬链接等）

元数据支持（10 分）：支持特定文件系统的文件元数据（属主/时间/权限等）

自定义备份（10 分）：允许用户筛选需要备份的文件（路径/类型/名字/时间/定时）

压缩解压（10 分）：通过文件压缩节省备份文件的存储空间

打包解包（10 分）：将所有备份文件拼接为一个文件保存

加密备份（10 分）：由用户指定密码，将所有备份文件均加密保存

实时备份（10 分）：自动感知用户文件变化，进行自动备份

图形界面（10 分）：实现友好易用的 GUI 界面

网络备份（30 分）：将数据备份软件从单机模式扩展为网盘模式（10 分），还涉及到的功能包括：用户管理（5 分）、元数据管理（5 分）、传输加密（5 分）、增量备份（5 分）等。

其它功能：视功能难度讨论加分。

五、实验器材（设备、元器件）：

PC 多台，可连接互联网。

六、实验步骤及操作：

需求分析说明书（10 分）

1. 任务概述

1.1 引言

数据是企业重要的生产资料，关键数据的丢失可能会给个人或者企业致命一击。比如在 911 事件中，Bank NewYork 在数月后因数据的丢失被迫破产清盘。为什么后果如此严重？因为数据是计算机系统存在的原因和基础，数据往往是不可再生的。一旦发生数据丢失，企业就会陷入困境：客户资料、技术文件、财务账目等客户、交易、生产数据可能被破坏得面目全非。

1.2 综合描述

本软件是一款基于 Linux 系统的数据备份软件，可以对于给定的文件目录进行备份。在备份的同时也可支持多种用户自定义的行为，如文件夹打包、文件压缩、文件加密。适用于适用于对数据安全需求较高的人群。

1.2.1 产品的状况

本产品是文件备份系列软件的第一代产品。其编写语言主要为 C++，核心技术完全基于互联网开源技术。为确保本产品的正确运行，请确定运行环境一定为 Linux 系统。

涉及到的系统接口主要有 dirent.h，sys/stat.h，unistd.h 等文件操作相关 Linux 自带接口。

1.2.2 产品的功能

文件夹备份

文件夹打包成. tar 文件
. tar 文件解包为文件夹
. tar 文件压缩为. tar. huf 文件
对. tar. huf 文件进行加密
对. tar. huf 文件进行解密
. tar. huf 文件解压为. tar 文件
文件名筛选自定义备份
文件类型筛选自定义备份
文件路径筛选自定义备份
文件修改时间筛选自定义备份

1.2.3 用户类和特性

- 1. 对文件保存有安全需求的用户。该类用户需要对文件进行拷贝备份，以备当原文件受损时使用备份文件还原。
- 2. 在第 1 类用户的基础上对文件存储空间需求的用户。该类用户需要降低备份文件的存储空间，从而降低存储成本。
- 3. 在第 2 类用户的基础上对文件有安全需求的用户。该类用户需要对备份文件进行加密，以免无权限人员获取备份文件内容从而导致泄露。
- 4. 在第 1 类用户的基础上对文件有筛选需求的用户。该类用户只需要选择必要的文件进行备份，从而避免无意义的存储空间浪费。

1.3 运行环境

1.3.1 基本配置

文件备份软件系统所需的基本配置如下：

- (1) 硬件平台：无特殊要求
- (2) 操作系统：Linux

1.3.2 其他配置

- (1) 内存：4GB 以上
- (2) 硬盘：128MB 以上
- (3) 网络：无要求

2. 功能需求

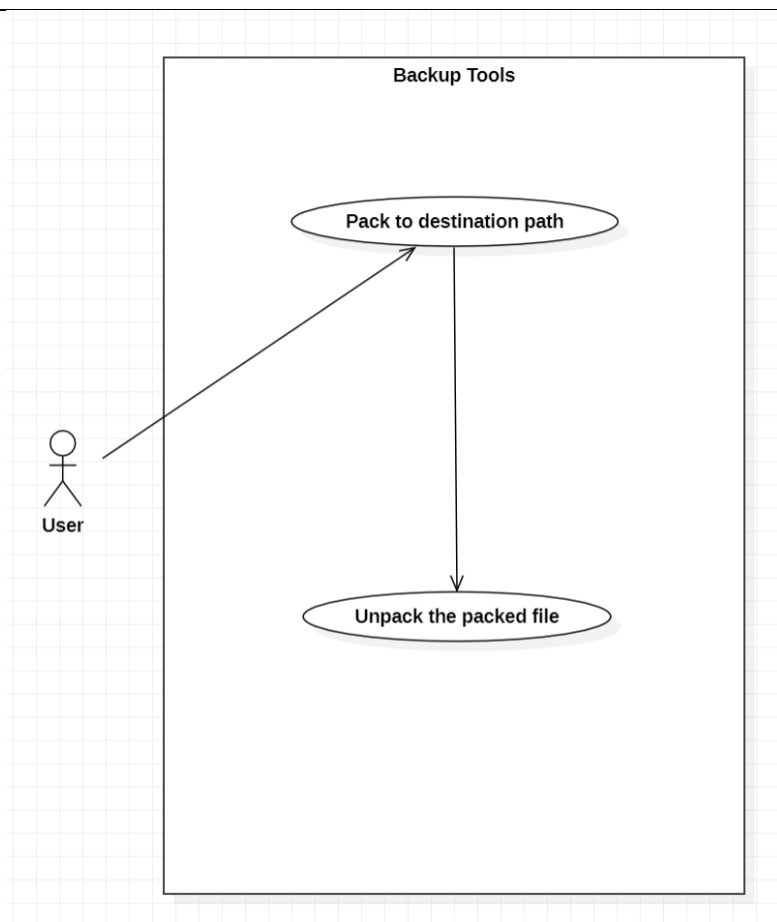
2.1 功能划分

文件夹（自定义）备份（拷贝）	文件压缩解压
文件打包解包	文件加密解密

2.2 系统用例

2.2.1 用例编号 1

2.2.1.1 用例图

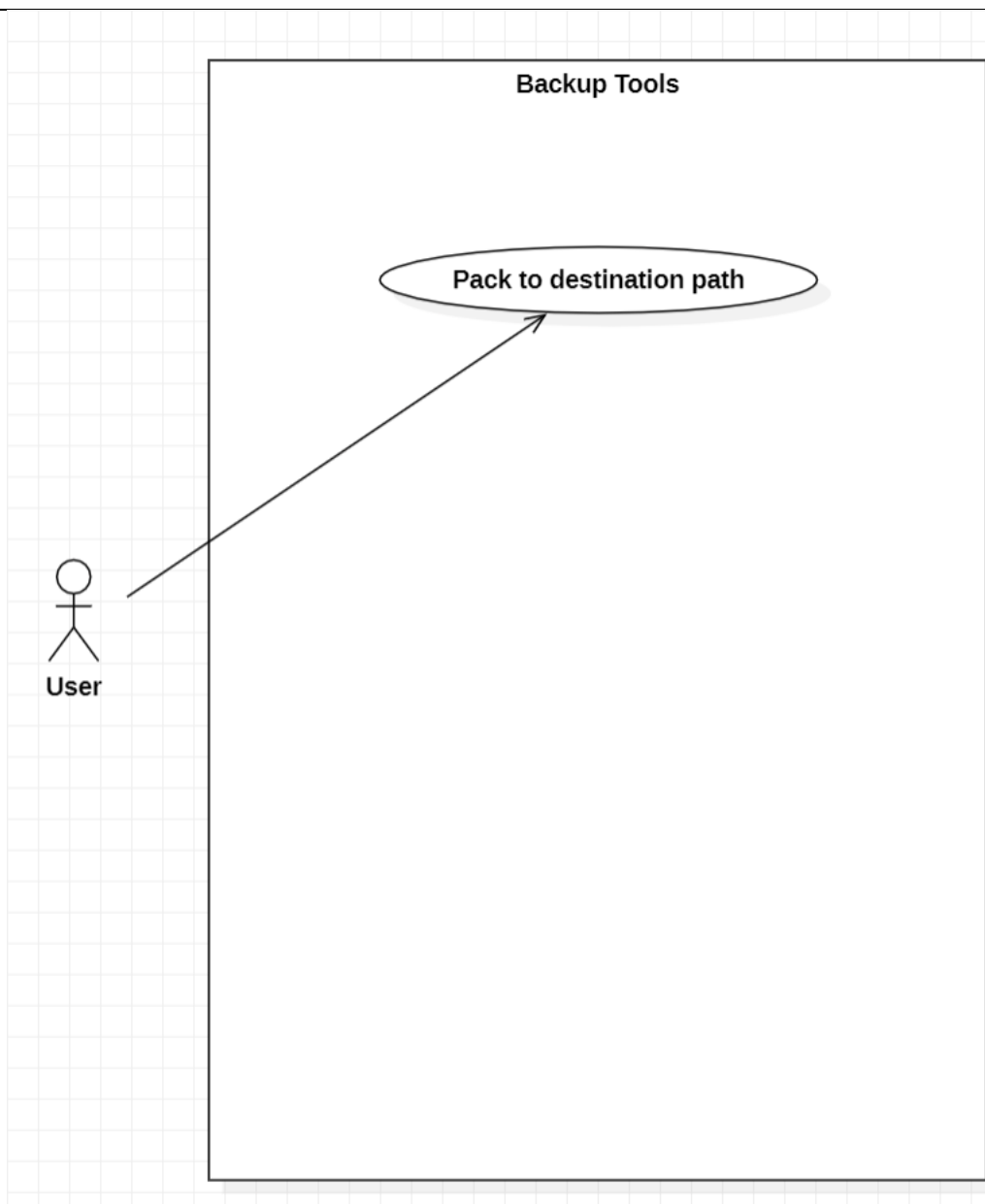


2.2.1.2 用例描述

用例标识	1	用例名称	Backup a folder to destination path(original)
创建人	张镕麒	创建日期	2022 年 11 月 6 日
语境目标	用户将文件夹备份到指定路径（基础模式）		
前置条件	原文件夹存在，目标文件夹不存在		
成功的结束状态	成功备份文件夹		
失败的结束状态	备份文件夹失败		
参与者	User		
基础用例	Pack to destination path, Unpack the packed file		
事件流	基本流程	1、用户输入原路径与目的路径文件夹	
		2、用户选择基础备份模式	
		3、成功备份文件夹	
	异常流程	4.1、未输入路径	
		4.2、目的路径已存在	
非功能需求	时间短、存储空间需求少		

2.2.2 用例编号 2

2.2.2.1 用例图



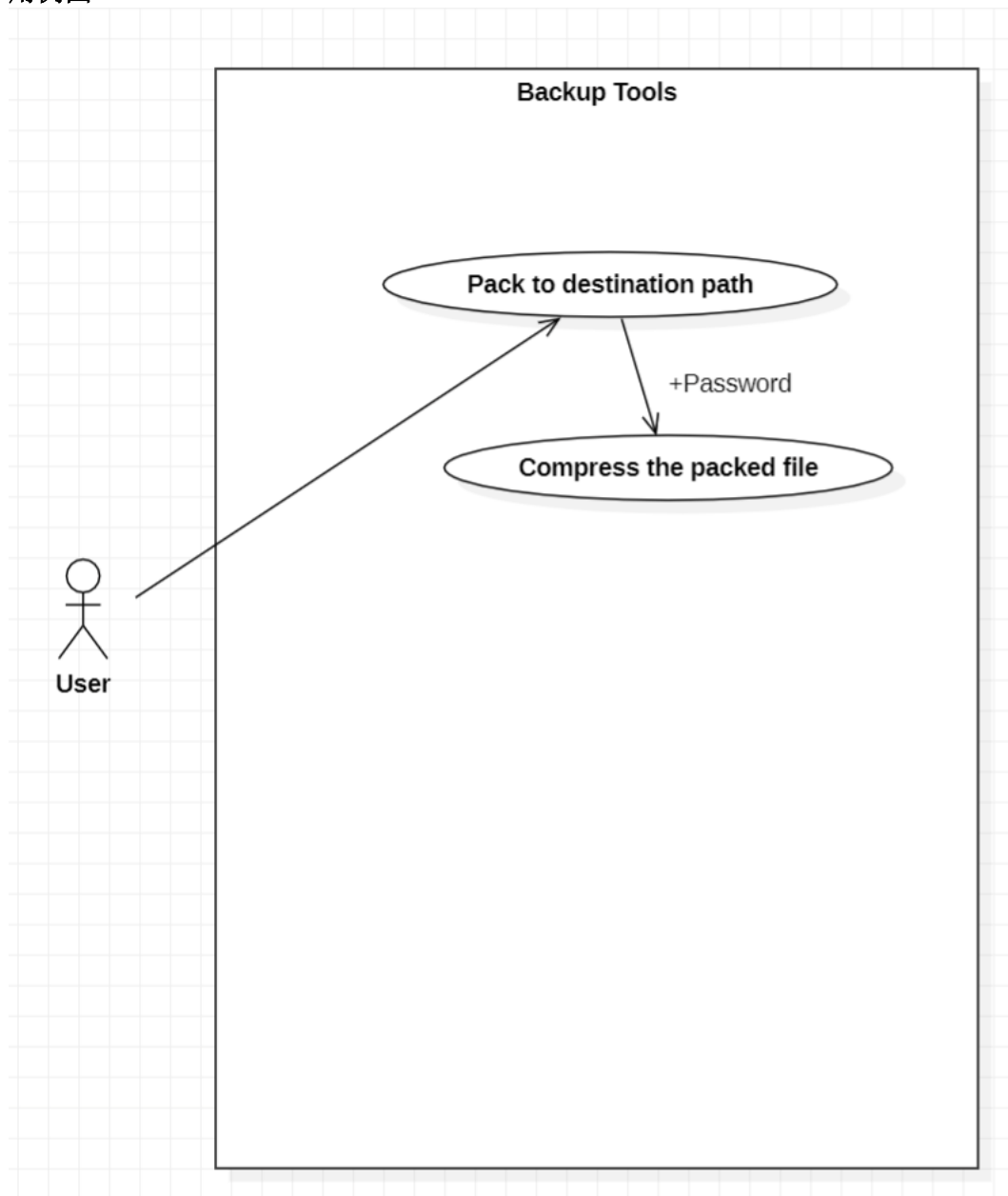
2.2.2.2 用例描述

用例标识	2	用例名称	Backup a folder to destination path(Pack)
创建人	张镕麒	创建日期	2022 年 11 月 6 日
语境目标	用户将文件夹备份到指定路径（打包模式）		
前置条件	原文件夹存在，目标文件夹不存在		
成功的结束状态	成功打包备份原文件夹		
失败的结束状态	打包备份失败		
参与者	User		
基础用例	Pack to destination path		
事件	基本流程	1、用户输入原路径与目的路径文件夹	
		2、用户选择打包备份模式	

流		3、成功打包备份文件夹
	异常流程	4.1、未输入路径
	非功能需求	安全性
	补充规格说明书	无

2.2.3 用例编号 3

2.2.3.1 用例图



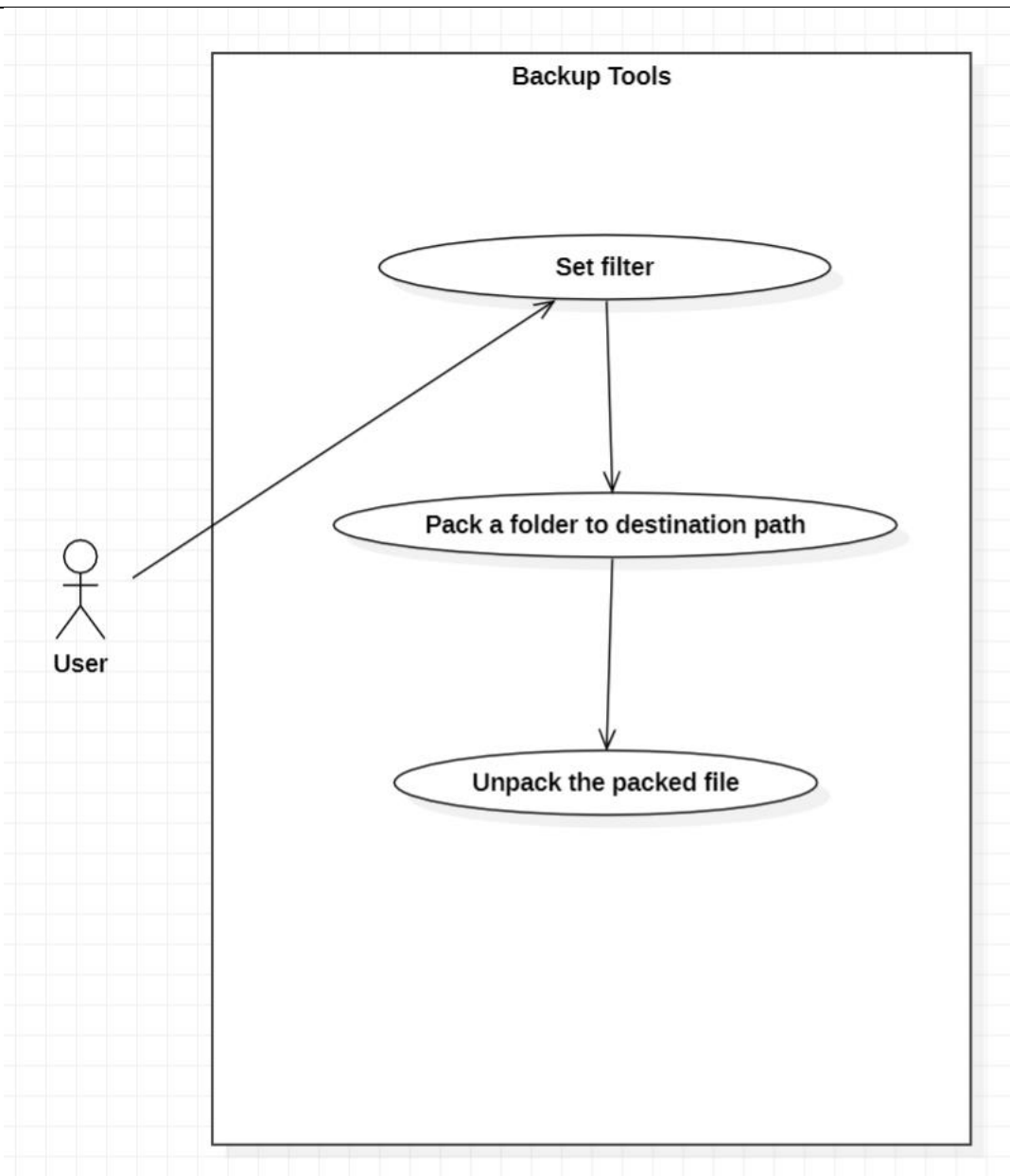
2.2.3.2 用例描述

用例标识	3	用例名称	Backup a folder to destination path(Compress)
创建人	张镕麒	创建日期	2022 年 11 月 6 日
语境目标	用户将文件夹备份到指定路径（压缩模式）		
前置条件	原文件夹存在，目标文件夹不存在		
成功的结束	成功压缩备份文件夹		

		状态	
		失败的结束状态	压缩备份文件夹失败
		参与者	User
		基础用例	Pack to destination path, Unpack the packed file, Compress the packed file, Decompress the compressed file
事件流	基本流程		1、用户输入原路径与目的路径文件夹
			2、用户选择打包备份模式
			3、用户输入压缩密码（若选择加密备份）
			4、成功压缩备份文件夹
	异常流程		5.1、未输入路径
			5.2、目的路径已存在
			5.3、未输入密码（若选择加密备份）
			5.4、输入密码过长（若选择加密备份）
	非功能需求		时间短、存储空间需求少、加密足够安全

2.2.4 用例编号 4

2.2.4.1 用例图



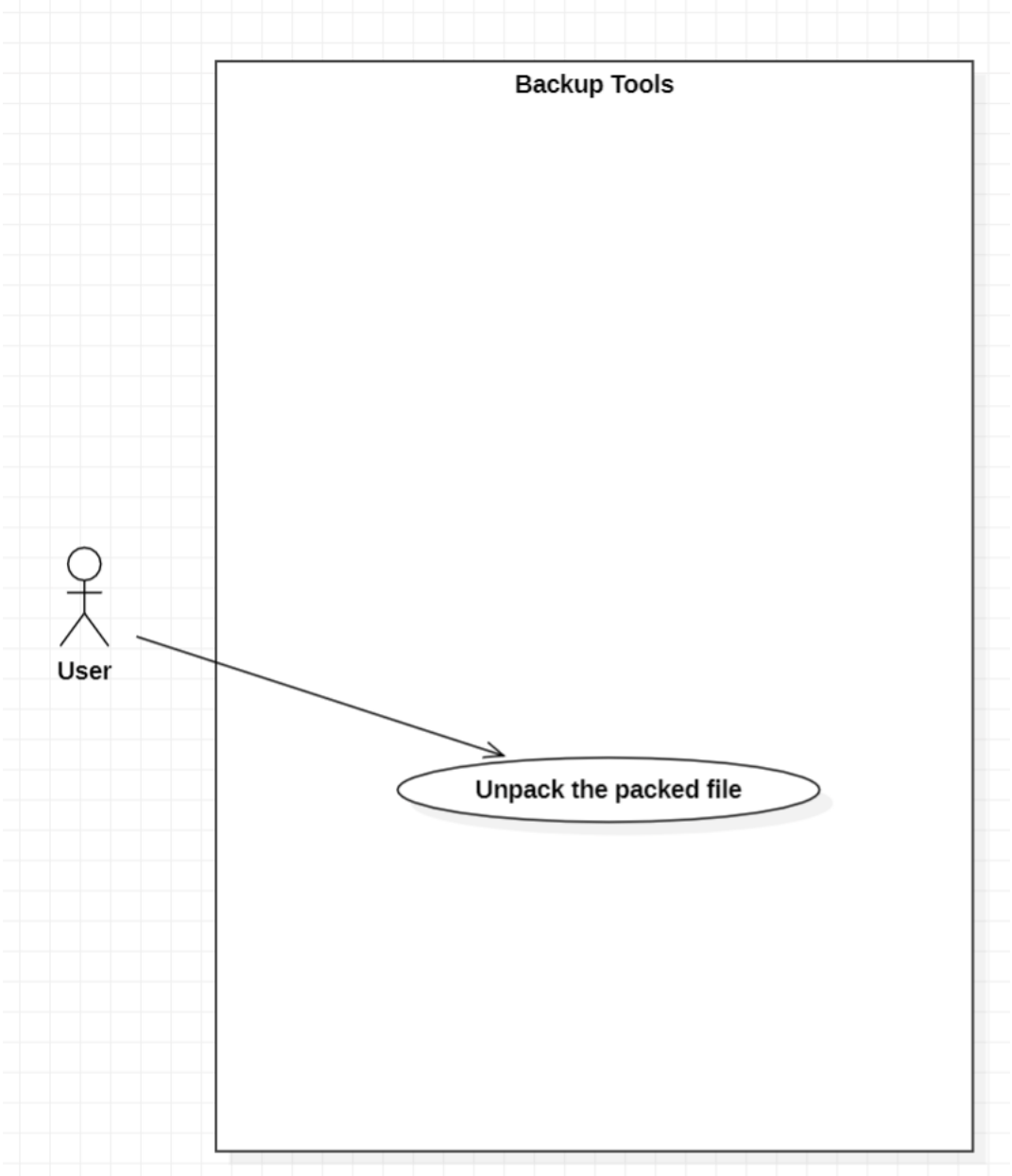
2.2.4.2 用例描述

用例标识	4	用例名称	Backup a folder to destination path(filter)
创建人	张镕麒	创建日期	2022 年 11 月 6 日
语境目标	用户将文件夹备份到指定路径（筛选模式）		
前置条件	原文件夹存在，目标文件夹不存在		
成功的结束状态	成功筛选备份文件夹		
失败的结束状态	筛选备份文件夹失败		
参与者	User		
基础用例	Set filter, Pack to destination path, Unpack the packed file		
事件	基本流程	1、用户输入原路径与目的路径文件夹	
		2、用户选择筛选备份模式	

流		3、用户输入筛选模式与筛选内容
		4、成功筛选备份文件夹
异常流程		5.1、未输入路径
		5.2、目的路径已存在
		5.3、未选择筛选模式
		5.4、未输入筛选内容
非功能需求		时间短、存储空间需求少

2.2.5 用例编号 5

2.2.5.1 用例图



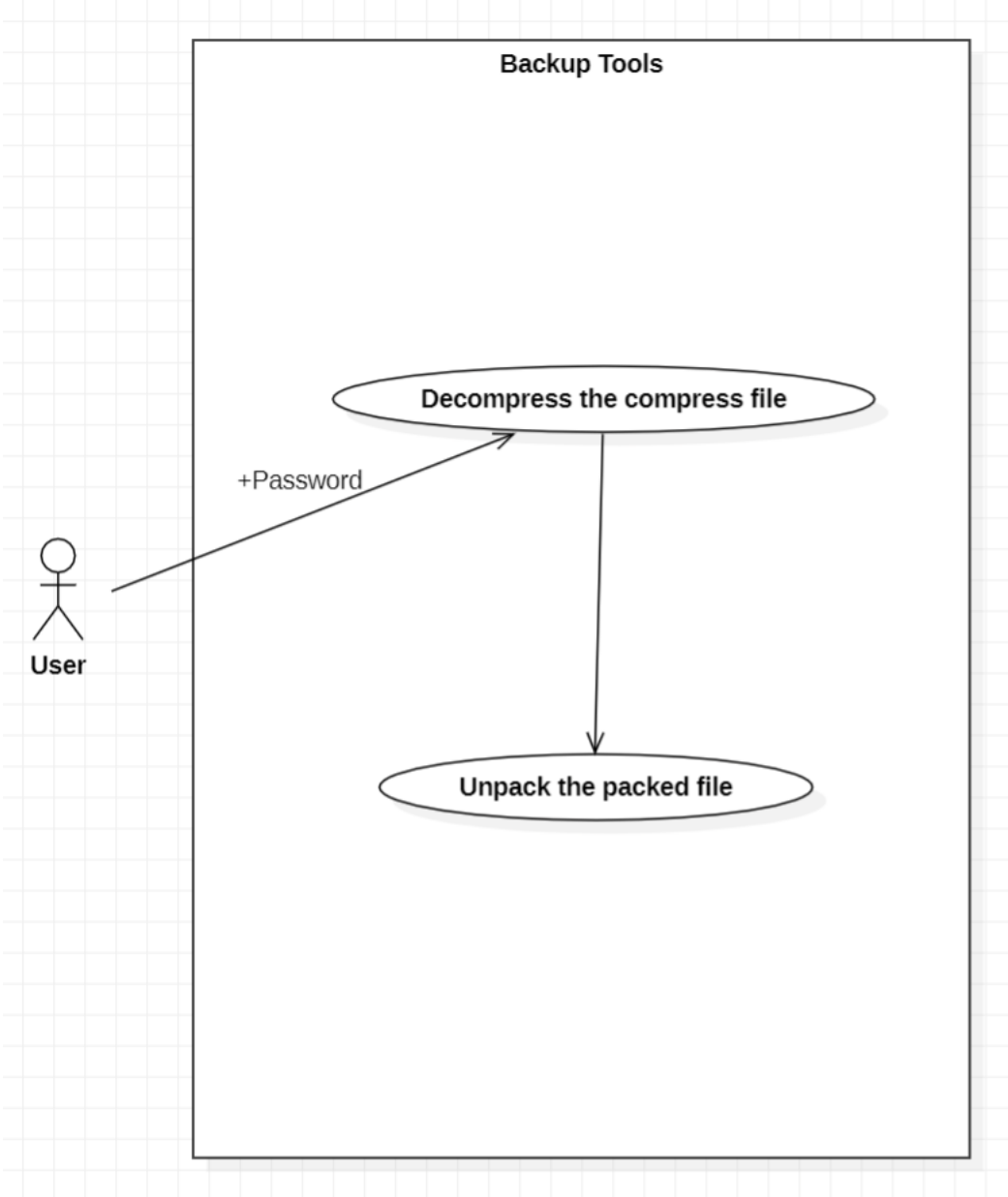
2.2.5.2 用例描述

用例标识	5	用例名称	Unpack the previously packed file
创建人	张镕麒	创建日期	2022 年 11 月 6 日
语境目标	用户解包用例 2 产生的打包文件		

前置条件		原文件存在
成功的结束状态		成功解包文件
失败的结束状态		解包文件失败
参与者		User
基础用例		Unpack the packed file
事件流	基本流程	1、用户输入原路径与目的路径文件夹
		2、用户选择解包模式
		3、成功解压文件
	异常流程	4.1、未输入路径
非功能需求		时间短、存储空间需求少

2.2.6 用例编号 6

2.2.6.1 用例图



2.2.6.2 用例描述

用例标识	6	用例名称	Decompress the previously compressed file
创建人	张镭麒	创建日期	2022 年 11 月 6 日
语境目标	用户解压用例 3 产生的压缩文件		
前置条件	原文件存在		
成功的结束状态	成功解压文件		
失败的结束状态	解压文件失败		
参与者	User		
基础用例	Decompress the compressed file, Unpack the packed file		
事件流	基本流程	1、用户输入原路径与目的路径文件夹	
		2、用户选择解压模式	
		3、用户输入密码（若适用）	
		4、成功筛选备份文件夹	
	异常流程	5.1、未输入路径	
		5.2、密码错误	
非功能需求	时间短、存储空间需求少		

3. 外部接口需求

3.1 用户界面

本产品的用户界面基于 C++Qt 编写，包含 Backup 页、Pack 页、Unpack 页、Compress 页、Decompress 页、Filter-Path、Filter-Path 页、Filter-Type 页、Filter-Time 页。下面将详细介绍。

3.1.1 用户界面 1：Backup 页

Backup Tools

Backup

Pack

Unpack

Compress

Decompress

Source

Select a folder

Destination

Select a folder

☒ Pack?

☒ Compress?

☒ Encrypt?

Password

(Max length = 32)

Filter

Reset

☐ Keep temporary files?

Run

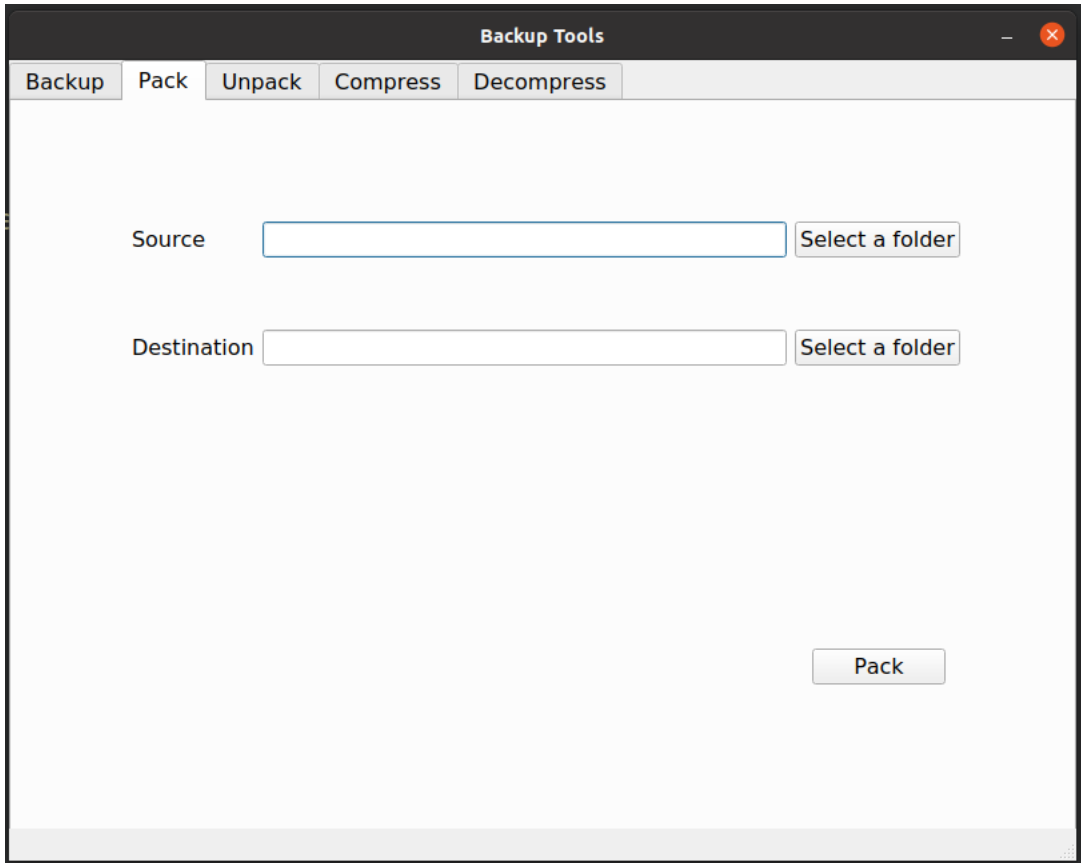
在 Backup 页面中，首先可以在 Source 行与 Destination 行输入文件路径，路径后的按钮将弹出文件路径选择子窗口。下面分别是 Pack、Compress、Encrypt 选项，当且仅当前一个选项被选择后，后一个选项才会显示。根据选项的选择情况分别对应普通备份模式、打包备份模式、压缩备份模式、加密压缩备份模式。当 Encrypt 选项被选中后，Password 行将会出现，用户必须在该行输入加密压缩密码。

当点击 Filter 按钮后，将弹出 Filter 窗口，当设置了合适的筛选器后将在该页显示筛选器的筛选模式，点击 Reset 按钮后将退出筛选模式。

当选择 Keep temporary files 选项后，备份过程将保留程序在备份过程产生的临时文件。

点击 Run 按钮后，程序将按照给定的设置运行。

3.1.2 用户界面 2: Pack 页



在 Pack 页面中，首先可以在 Source 行与 Destination 行输入文件路径，路径后的按钮将弹出文件路径选择子窗口。

点击 Pack 按钮后，程序将按照给定的设置运行。

3.1.3 用户界面 3: Unpack 页

Backup Tools

Backup

Pack

Unpack

Compress

Decompress

Source

Select a tar file

Destination

Select a folder

☐ Remove empty folders?

Unpack

在 Backup 页面中，首先可以在 Source 行与 Destination 行输入文件路径，路径后的按钮将弹出文件路径选择子窗口。

当选择 Remove empty folders 选项后，解包过程产生的空文件夹将被删除。

点击 Unpack 按钮后，程序将按照给定的设置运行。

3.1.4 用户界面 4: Compress 页

Backup Tools

Backup Pack Unpack Compress Decompress

Source

Destination

☒ Encrypt? (Max length=32)

☐ Keep temporary files?

在 Compress 页面中，首先可以在 Source 行与 Destination 行输入文件路径，路径后的按钮将弹出文件路径选择子窗口。当 Encrypt 选项被选中后，Password 行将会出现，用户必须在该行输入加密压缩密码。

当选择 Keep temporary files 选项后，备份过程将保留程序在备份过程产生的临时文件。

点击 Compress 按钮后，程序将按照给定的设置运行。

3.1.5 用户界面 5: Decompress 页

The image shows a software window titled "Backup Tools" with a dark header bar. Below the header is a tabbed interface with five tabs: "Backup", "Pack", "Unpack", "Compress", and "Decompress". The "Decompress" tab is currently selected. The main area of the window contains the following elements:

- Source:** A text input field followed by a button labeled "Select a huf file".
- Destination:** A text input field followed by a button labeled "Select a folder".
- Password(if applicable):** A text input field with a label "(Max length=32)" to its right.
- Unpack?:** A checked checkbox.
- Remove empty folders?:** An unchecked checkbox.
- Keep temporary files?:** An unchecked checkbox.
- Decompress:** A button located at the bottom right of the main area.

在 Decompress 页面中，首先可以在 Source 行与 Destination 行输入文件路径，路径后的按钮将弹出文件路径选择子窗口。

当选择 Unpack 选项后，将进入解压解包模式，Remove empty folders 选项将会出现。

当选择 Remove empty folders 选项后，解包过程产生的空文件夹将被删除。

当选择 Keep temporary files 选项后，备份过程将保留程序在备份过程产生的临时文件。

点击 Decompress 按钮后，程序将按照给定的设置运行。

3.1.6 用户界面 6: Filter-Path、Filter-Name、Filter-Type 页

The image shows a software window titled "Form" with a dark header bar. The main area has a light gray background and contains the following elements:

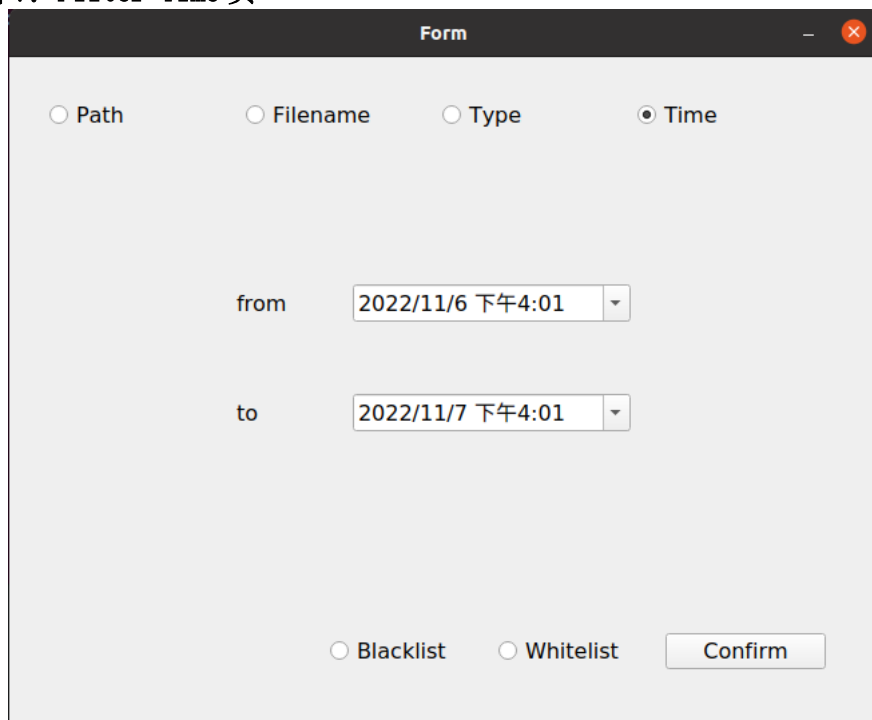
- Filter Type Selection:** Four radio buttons labeled "Path", "Filename", "Type", and "Time". The "Path" button is selected.
- Tips:** The text "Tips: One for each line" is displayed above a large, empty rectangular text input area.
- Filter List Selection:** Two radio buttons labeled "Blacklist" and "Whitelist" are located at the bottom left.
- Confirm:** A button labeled "Confirm" is located at the bottom right.

首先根据上方的选项确定页面，在 Filter-path、Filter-name、Filter-type 页面中，下方将会出现文本编辑框，在编辑框内用户需要按行输入筛选的内容。

在下方的 Blacklist 与 Whitelist 选项将会决定上方筛选内容是黑名单或白名单。

点击 Comfirm 按钮后，程序将按照给定的设置执行。

3.1.7 用户界面 7: Filter-Time 页



首先根据上方的选项确定页面，在 Filter-Time 页面中，下方将会出现日期选择构件，用户需要给定筛选的时间区段——起始时间与结束时间。

在下方的 Blacklist 与 Whitelist 选项将会决定上方筛选内容是黑名单或白名单。

点击 Comfirm 按钮后，程序将按照给定的设置执行。

3.2 软件接口

本软件使用的 Linux 提供的系统接口如下

- dirent.h: 访问、遍历文件夹
- sys/stat.h: 获得文件信息
- sys/types.h: 获得文件信息
- unistd.h: 获得文件信息
- pwd.h: 获得用户信息
- grp.h: 获得组信息
- utime.h: 更改文件修改时间
- time.h: 获得文件修改时间
- fcntl.h: 访问文件

本软件使用的 C++ 工具库如下:

- queue: 使用优先队列
- iostream: 使用输入输出流
- vector: 使用优先队列
- string: 使用字符串

本软件使用的第三方工具库如下:

- Qt 中的 QWidget: 调用子窗口
- Qt 中的 QVector: 用于窗口间传输信息
- Qt 中的 QMessageBox: 用于提示信息
- Qt 中的 QVariant: 用于窗口间传递结构化信息
- Qt 中的 QPushButton: 用于美化用户界面

- Qt 中的 QStringList: 用于获取文本信息
- Qt 中的 QDebug: 用于 Debug
- Qt 中的 QDate: 用于获得日期
- Qt 中的 QDateTime: 用于获得日期时间
- Qt 中的 QFileInfo: 用于文件名相关串操作
- Qt 中的 QString: 用于获取文本信息
- Qt 中的 QMainWindow: 调用父窗口
- Qt 中的 QFileDialog: 调用文件选择子窗口

4. 其它非功能性需求

4.1 性能需求

响应时间尽量低、临时文件占用存储空间尽量少。

4.2 安全性需求

无。

4.3 软件质量属性

易用性足够高，尽量提供人性化功能。
给出尽量全面的出错提示，方便用户找到出错问题。

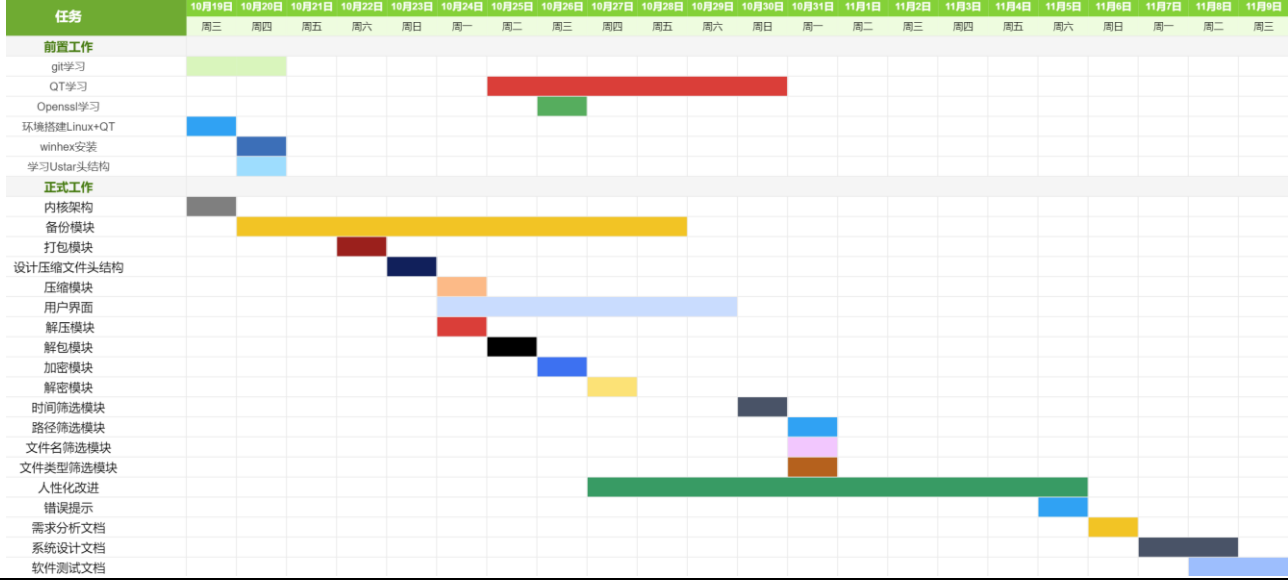
5. 项目规划

5.1 人员分工

项目由一位成员分工具体安排如下（例）。

角 色	主要职责	负责模块	人员	备注
项目经理 PM &程序员 DEV	<div><div></div> 项目全面负责</div> <div><div></div> 项目设计</div> <div><div></div> 主要框架/模块编写</div> <div><div></div> 项目进度控制</div> <div><div></div> 模块编写</div>	<div><div></div> 代码框架</div> <div><div></div> 代码汇总</div> <div><div></div> 打包\解包模块</div> <div><div></div> 压缩\解压模块</div> <div><div></div> 加密\解密模块</div> <div><div></div> 筛选模块</div> <div><div></div> 用户界面</div> <div><div></div> 其他人性化功能</div> <div><div></div> 测试与 Debug</div> <div><div></div> 文档编写</div>	张镕麒	

5.2 项目总体规划



系统设计文档（20 分）

1. 开发环境和工具

1.1 开发环境

- (1) 硬件平台：联想 Y9000P 2022 版笔记本电脑
- (2) 操作系统：Linux
- (3) IDE：Qt 5.11

1.2 依赖库

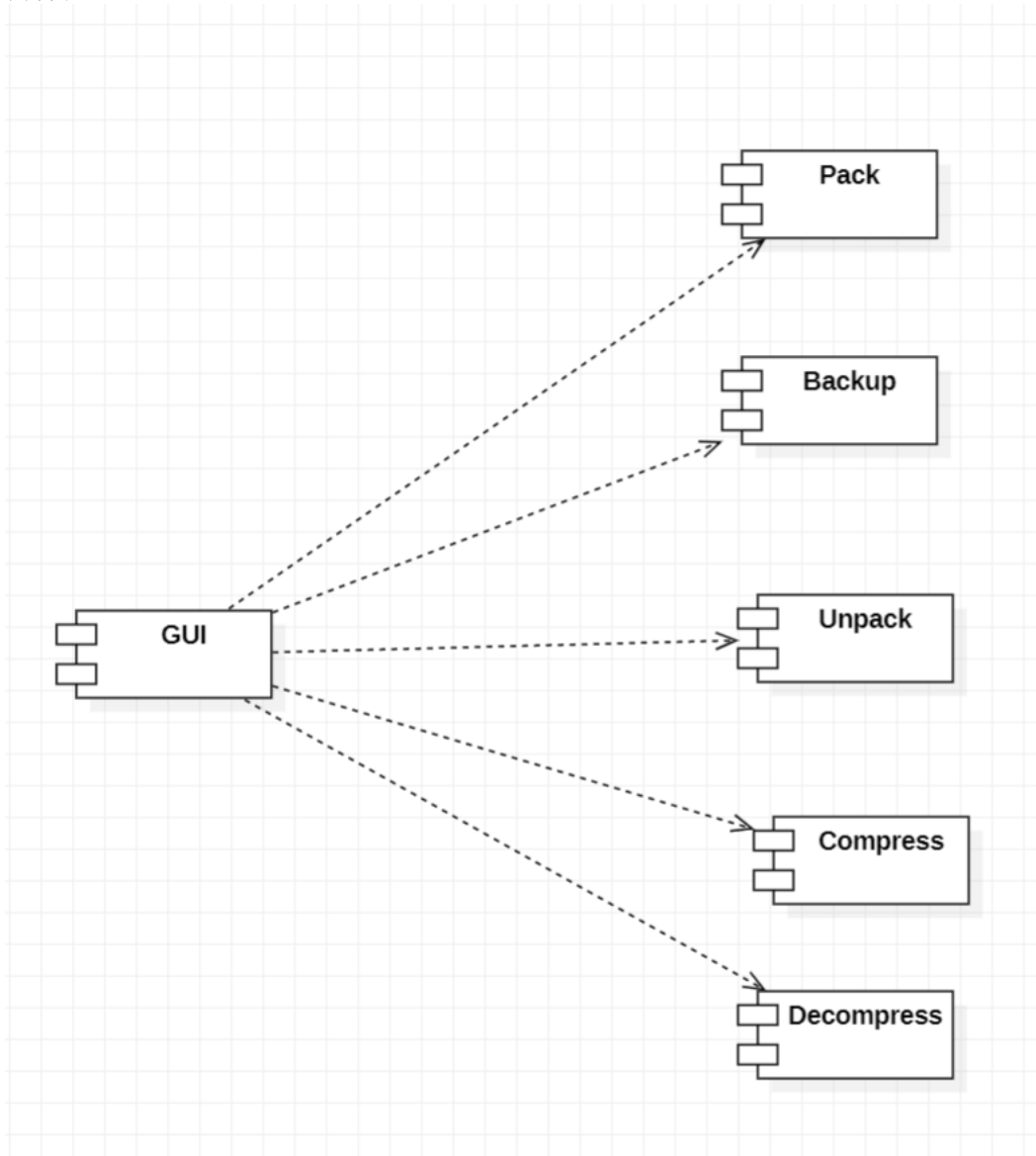
- (1) QT：版本 5.11，用于图形界面编程。
- (2) OpenSSL：用于文件加密。.

2. 总体设计

2.1 系统结构设计

2.1.1 顶层系统结构

2.1.1.1 构件图

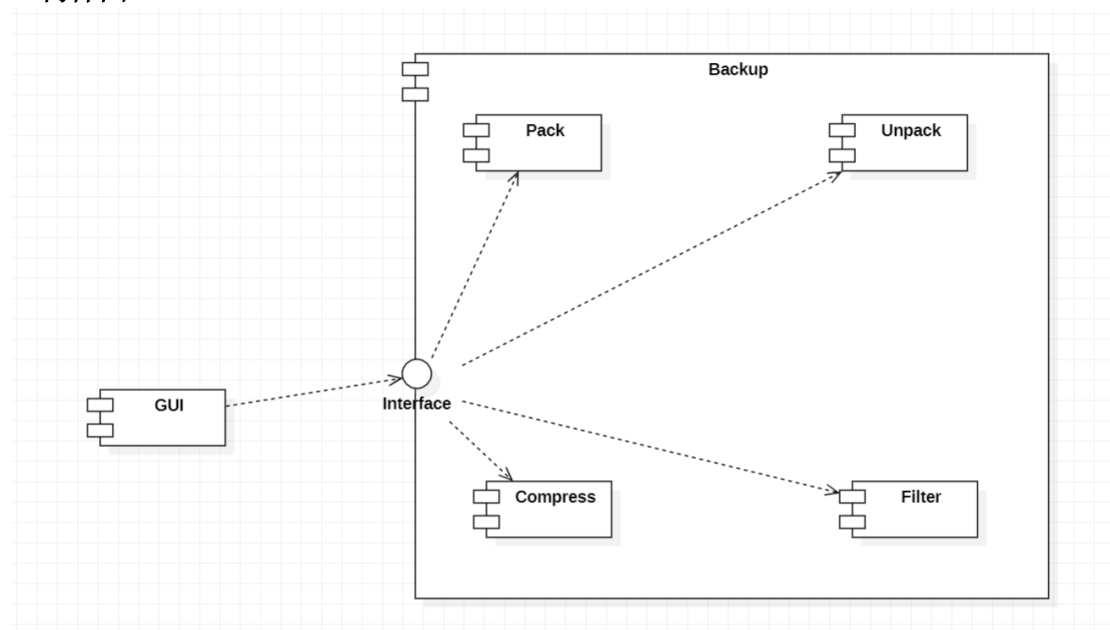


2.1.1.2 构件图描述

用户通过 GUI 访问到的功能有五种，分别为 Backup、Pack、Unpack、Compress、Decompress。

2.1.2 子系统一结构

2.1.2.1 构件图

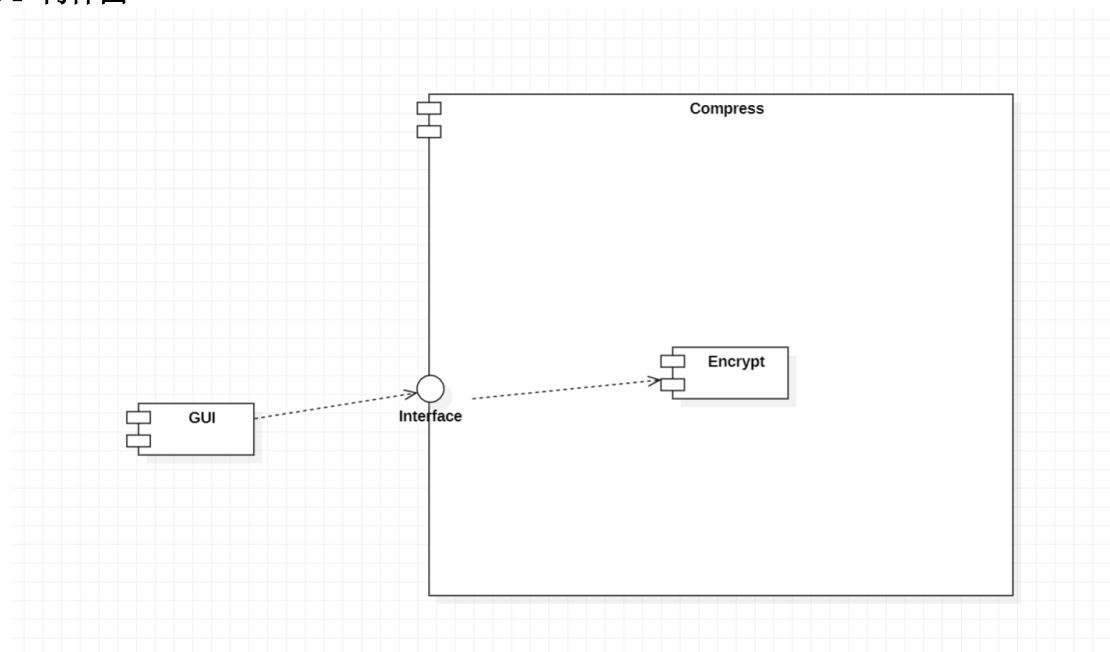


2.1.2.2 构件图描述

Backup 模块根据备份模式的不同，会分别调用子模块：Pack、Unpack、Compress、Filter。

2.1.3 子系统二结构

2.1.3.1 构件图

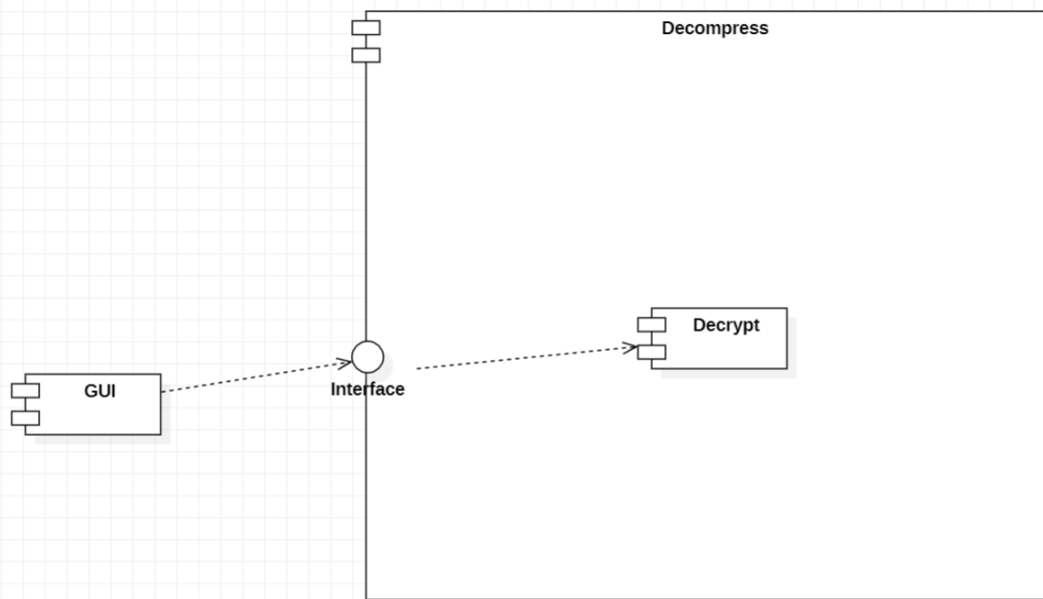


2.1.3.2 构件图描述

Compress 模块根据压缩模式的不同，会选择性地调用 Encrypt 模块。

2.1.4 子系统三结构

2.1.4.1 构件图



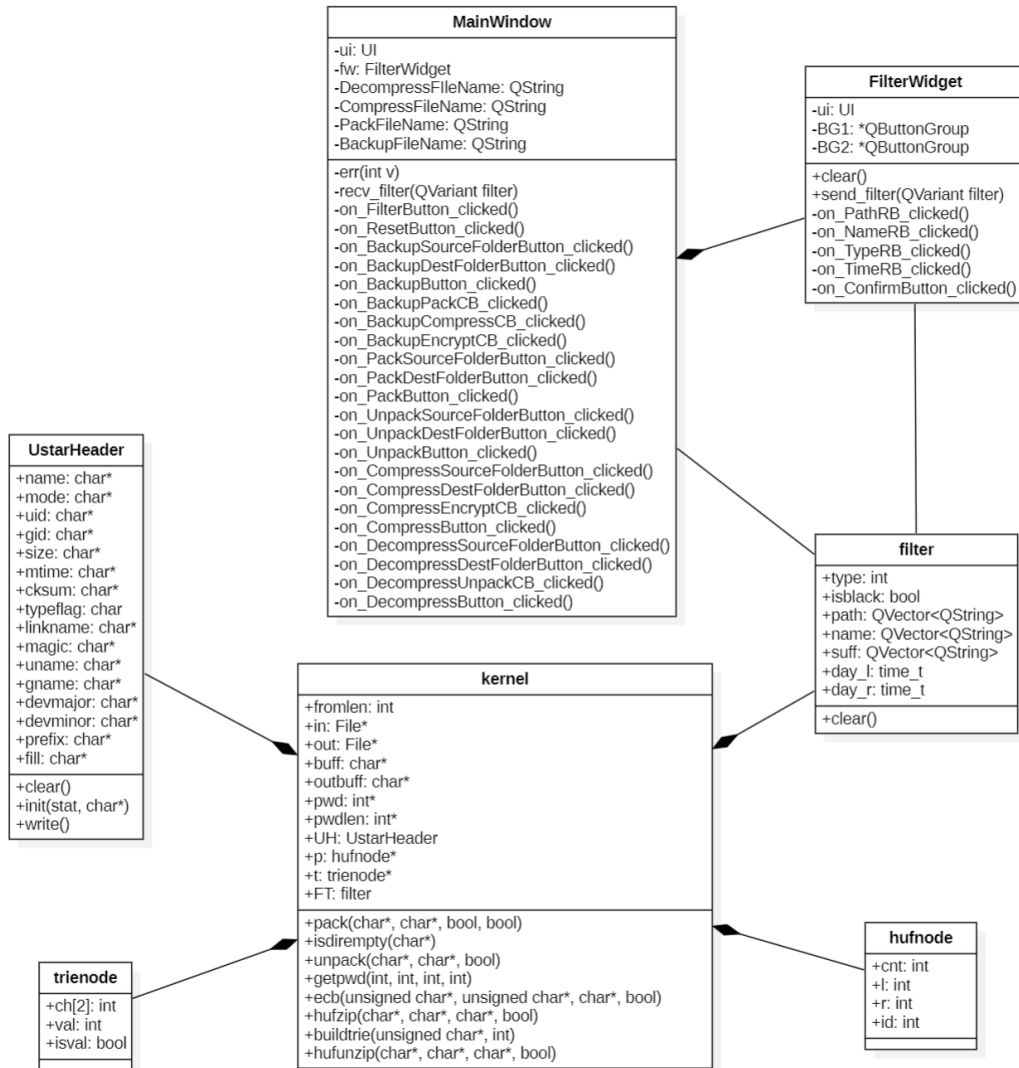
2.1.4.2 构件图描述

Decompress 模块根据解压模式的不同，会选择性地调用 Decrypt 模块。

3. 静态建模

3.1 系统对象模型

3.2 类（对象）描述



3.2.1 MainWindow 类

3.2.1.1 类描述

是图形界面中主窗口对应的类，可以对图形界面中的交互进行反馈，当程序启动时该类被构造，当程序结束时该类被解构。

3.2.1.2 属性描述

- ui: Qt 中的 ui 设计对象
- fw: FilterWidget 类，用于筛选
- DecompressFileName、CompressFileName、PackFileName、BackupFileName: 保存原路径中的文件名，用于自动生成目的路径文件夹

3.2.1.3 方法描述

- err(int v): 根据输入的错误类型进行报错
- recv_filter(QVariant filter): 获取从 FilterWidget 处传来的 filter 结构结构体
- on_FilterButton_clicked(): 显示 FilterWidget 窗口
- on_ResetButton_clicked(): 删除缓存中的筛选信息
- on_BackupSourceFolderButton_clicked(): 弹出文件路径选择窗口
- on_BackupDestFolderButton_clicked(): 弹出文件路径选择窗口
- on_BackupButton_clicked(): 分别根据不同的备份模式，执行不同的备份过程
- on_BackupPackCB_clicked(): 选中后进入打包备份模式，并显示压缩备份选项。否则隐藏压缩备份选项及后续选项。

- `on_BackupCompressCB_clicked()`: 选中后进入压缩备份模式，并显示加密备份选项。否则隐藏加密备份选项及后续选项。

- `on_BackupEncryptCB_clicked()`: 选中后进入加密备份模式，并显示密码输入框。否则隐藏密码输入框。

剩余方法为其余页面的类似处理过程，此处不再赘述。

3.2.2 FilterWidget 类

3.2.2.1 类描述

是图形界面中筛选窗口对应的类，可以对图形界面中的交互进行反馈，当程序启动时该类被构造，当程序结束时该类被解构。

3.2.2.2 属性描述

- `ui`: Qt 中的 ui 设计对象
- `BG1`: 筛选类型按钮组，包括路径、文件名、文件类型、修改时间
- `BG2`: 筛选类型按钮组，包括黑名单、白名单

3.2.2.3 方法描述

- `clear()`: 清空缓存中的 `filter` 结构体内容，调用 `filter` 类中的 `clear()` 函数。
- `send_filter(QVariant filter)`: 向 `MainWindow` 传送 `filter` 结构体数据
- `on_PathRB_clicked()`: 选中后进入文件路径筛选模式，显示文本框
- `on_NameRB_clicked()`: 选中后进入文件名筛选模式，显示文本框。
- `on_TypeRB_clicked()`: 选中后进入类型筛选模式，显示文本框。
- `on_TimeRB_clicked()`: 选中后进入修改时间筛选模式，显示时间选择控件。
- `on_ConfirmButton_clicked()`: 保存当前筛选信息，并调用 `send_filter()` 将筛选信息传向 `MainWindow`。

3.2.3 filter 类

3.2.3.1 类描述

存储筛选信息。当程序启动时该类被构造，当程序结束时该类被解构。

3.2.3.2 属性描述

- `type`: 表示筛选的类型，值分别为 0 到 4 表示不筛选、路径筛选、文件名筛选、文件类型筛选、修改时间筛选
- `isblack`: 表示筛选的类型，值分别为 0 和 1 表示黑名单模式、白名单模式。
- `path`: 存储用户输入的路径筛选信息。
- `name`: 存储用户输入的文件名筛选信息。
- `suff`: 存储用户输入的文件类型（后缀）筛选信息。
- `day_l`: 存储用户输入的修改时间最小值。
- `day_r`: 存储用户输入的修改时间最大值。

3.2.3.3 方法描述

- `clear()`: 清空全部筛选信息

3.2.4 UstarHeader 类

3.2.4.1 类描述

存储被打包的文件夹内容与属性。当程序启动时该类被构造，当程序结束时该类被解构。

3.2.4.2 属性描述

- `name`: 文件名或目录名
- `mode`: 文件的权限信息
- `uid`: 文件拥有用户的 id
- `gid`: 文件拥有用户所在组的 id
- `size`: 文件大小，以 Byte 为单位
- `mtime`: 文件修改时间，以时间戳方式显示
- `cksum`: 文件校验和，为文件头除校验和位置所有字节的和加上 256
- `typeflag`: 文件类型
- `linkname`: 文件的链接名（若适用）
- `magic`: Ustar 标志位

- **uname:** 文件拥有用户的名称
- **gname:** 文件拥有用户所在组的名称
- **devmajor:** 版本号（若适用）
- **devminor:** 版本号（若适用）
- **prefix:** 文件名补充位（若适用）
- **fill:** 补零位

3.2.4.3 方法描述

- **clear():** 清空所有参数信息
- **init(stat, char*):** 确定所有参数信息以便写入
- **write():** 向 out 文件指针写入该 Ustar 文件头。

3.2.5 hufnode 类

3.2.5.1 类描述

用于构建哈夫曼树，表示哈夫曼树的一个节点。默认根为 0 号结点。当程序启动时该类被构造，当程序结束时该类被解构。

3.2.5.2 属性描述

- **cnt:** 该节点对应的词频，合并后词频相加
- **l:** 该节点的左儿子 id
- **r:** 该节点的右儿子 id
- **id:** 该节点的 id

3.2.5.3 方法描述

无

3.2.6 trienode 类

3.2.6.1 类描述

用于构建字典树，表示字典树的一个节点。当程序启动时该类被构造，当程序结束时该类被解构。默认根为 0 号结点。

3.2.6.2 属性描述

- **ch[0]:** 该节点的左儿子 id
- **ch[1]:** 该节点的右儿子 id
- **val:** 该节点对应的解码内容。
- **isval:** 该节点是否是终止节点。

3.2.6.3 方法描述

无

3.2.7 kernel 类

3.2.7.1 类描述

含有所有内核函数的实现。当程序启动时该类被构造，当程序结束时该类被解构。

3.2.7.2 属性描述

- **fromlen:** 目的路径字符串 from 的长度，用于拼接以及削减字符串。
- **in:** 输入文件指针。
- **out:** 输出文件指针。
- **buff:** 输入文件内容缓存。
- **outbuff:** 输出文件内容缓存。（有必要与 buff 同时存在，在压缩时需要两个同时使用）
- **pwd:** 存储原码为下标时对应的密码。
- **pwdlen:** 存储原码为下标时对应的密码长度（以比特为单位）。
- **UH:** UstarHeader 文件头实例。
- **p:** 哈夫曼树结点类实例。
- **t:** 字典树结点类实例。
- **FT:** filter 类实例。

3.2.7.3 方法描述

- **pack(char*, char*, bool, bool):** 将给定文件夹进行打包的函数，实现过程需要递归搜索文件夹的内容，读取到的每一个文件需要判断是否符合筛选条件，黑名单与白名单的实现需要额外设置

状态位。最终生成.tar 文件。

● **isdirempty(char*)**: 对于给定文件夹，返回文件夹是否存在文件。特殊情况是多层嵌套的文件夹，当每层文件夹都不包含文件时也返回父文件夹为空。由于 Linux 删除文件夹需要保证文件夹为空，故函数执行时若某子文件夹为空，则会直接删除该文件夹。当且仅当解包时用户选择删除空文件夹时被调用

● **unpack(char*, char*, bool)**: 将给定打包文件进行解包的函数。首先按照 Ustar 格式读取前 512 字节，并检查 magic 位与校验和。验证通过后将文件内容写入到文件中，并且调用系统 API 修改元数据。若用户选择删除空文件则此时调用 isdirempty 函数进行空文件夹删除。最终生成文件夹。

● **getpwd(int, int, int, int)**: 获得原码与密码对应关系的函数。当 hufzip 函数执行后，哈夫曼树已经建好，此时遍历该树即可获得密码。实现时选择使用递归方式遍历哈夫曼树，密码根据向左儿子还是右儿子分别进行左移运算与左移或一运算。同时记录深度以备解码使用。

● **ecb(unsigned char*, unsigned char*, char*, bool)**: 加密函数，对于给定的文本段，通过调用 openssl 函数获得对应加密秘钥下生成的加密文本段。由于 ecb 是非对称加密，所以通过 bool 参数控制加密模式或解密模式。由于 openssl 只支持固定长度的秘钥，故对于用户输入的不定长秘钥，函数会自动补齐至最大值，若用户不选择加密压缩，事实上也会通过默认秘钥对文件进行加密，从而实现过程上的统一性。

● **hufzip(char*, char*, char*, bool)**: 对于给定的文件，将其加密压缩。首先初始化哈夫曼树的结点，然后统计原文件中的词频，每个单词（Byte，不超过 256 种）作为哈夫曼树的叶节点进行合并。由于合并会产生新节点，且每次将两个结点合并成一个新节点，故总结点数不超过 512 个。当哈夫曼树建好后执行 getpwd 函数获取密码表。通过下列格式写入到临时文件中。

Magic 位，char 型，默认为 huffman，占 8 字节，用于检测文件类型		
Size 位，int 型，占 4 字节，压缩后文件比特大小		Alpha 位，int 型，占 4 字节，密码表项数
明文，占 1 字节	密文比特长度，占 1 字节	密文，自动补齐至最短的整字节长度
明文，占 1 字节	密文比特长度，占 1 字节	密文，自动补齐至最短的整字节长度
明文，占 1 字节	密文比特长度，占 1 字节	密文，自动补齐至最短的整字节长度
.....		
压缩后且补全尾空余的文件内容		

临时文件生成后按照文本段的格式进行加密，首先需要将末尾的空余进行补齐，补齐内容为空余 Byte 个数。之后遍历每段，将该文本与秘钥传递至 ecb 函数中，获得秘文，写入到.tar.huf 文件中。

● **buildtrie(unsigned char*, int)**: 建立字典树，执行该函数将插入一个单词至字典书中。字典树的边为 01 比特，单词由 01 比特构成。将密码表中的密文插入到字典树中，并将明文存储到尾结点处。

● **hufunzip(char*, char*, char*, bool)**: 首先将文件通过 ecb 函数解密，之后按照上述压缩格式进行读取，对于密码表中的每一项，将其插入到字典树中。随后读取加密文件内容，在符合缓存区长度、加密文件长度、密码单项长度的前提下读取，并依次获取明文。全部解码完成后获得.tar 文件。

4. 动态建模

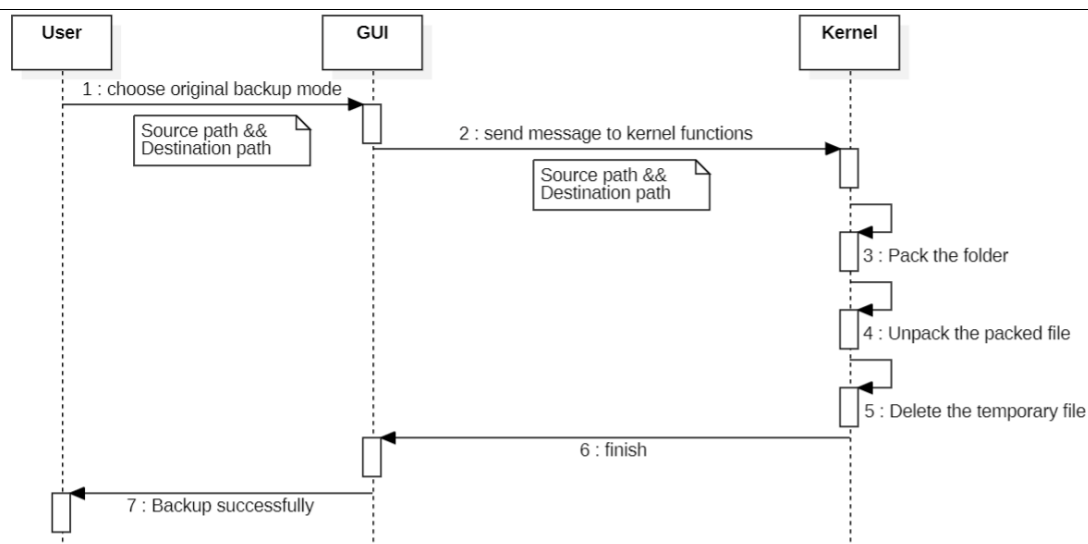
4.1 系统场景

4.1.1 场景：普通备份模式

4.1.1.1 场景描述

用户选择普通备份模式，此时程序输入为要备份的文件夹路径，输出为文件夹。

4.1.1.2 顺序图



4.1.1.3 流程说明

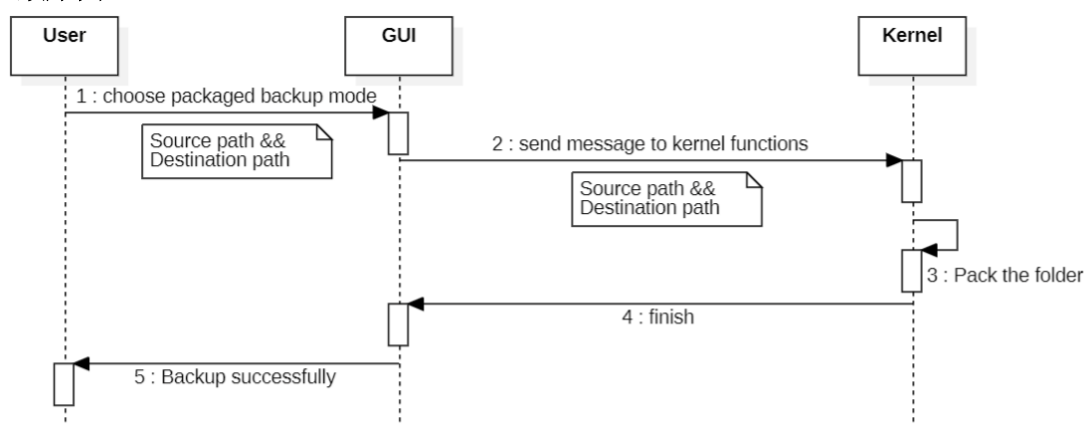
1. 用户选择普通备份模式，输入文件路径。
2. GUI 将信息传送给内核函数。
3. 内核执行打包程序至目的路径。
4. 内核将打包好的文件解包至目的路径。
5. 内核删除打包文件。
6. 内核函数执行结束返回。
7. 若执行出错，GUI 向用户提示错误信息。

4.1.2 场景：打包备份模式

4.1.2.1 场景描述

用户选择打包备份模式，此时程序输入为要备份的文件夹路径，输出为. tar 文件。

4.1.2.2 顺序图



4.1.3.3 流程说明

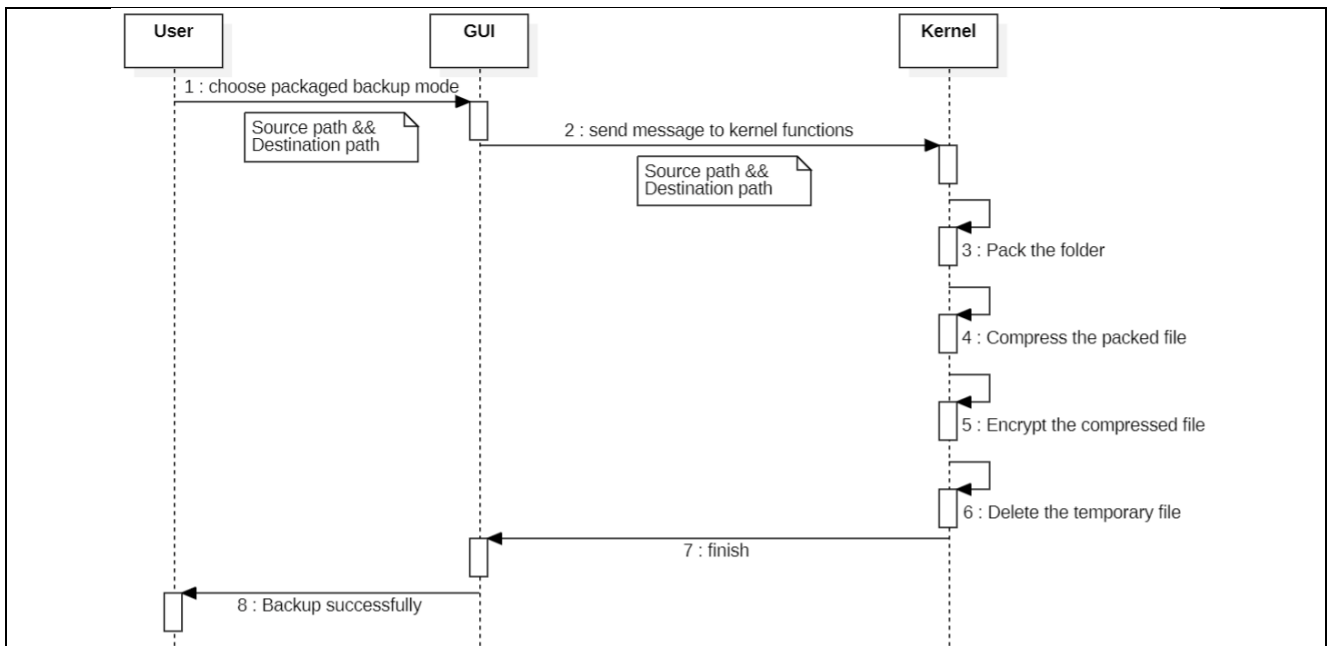
1. 用户选择打包备份模式，输入文件路径。
2. GUI 将信息传送给内核函数。
3. 内核执行打包程序至目的路径。
4. 内核函数执行结束返回。
5. 若执行出错，GUI 向用户提示错误信息。

4.1.4 场景：加密压缩备份模式与非加密压缩备份模式

4.1.4.1 场景描述

用户选择加密备份模式或非加密压缩备份模式，此时程序输入为要备份的文件夹路径，输出为. tar. huf 文件。

4.1.4.2 顺序图



4.1.4.3 流程说明

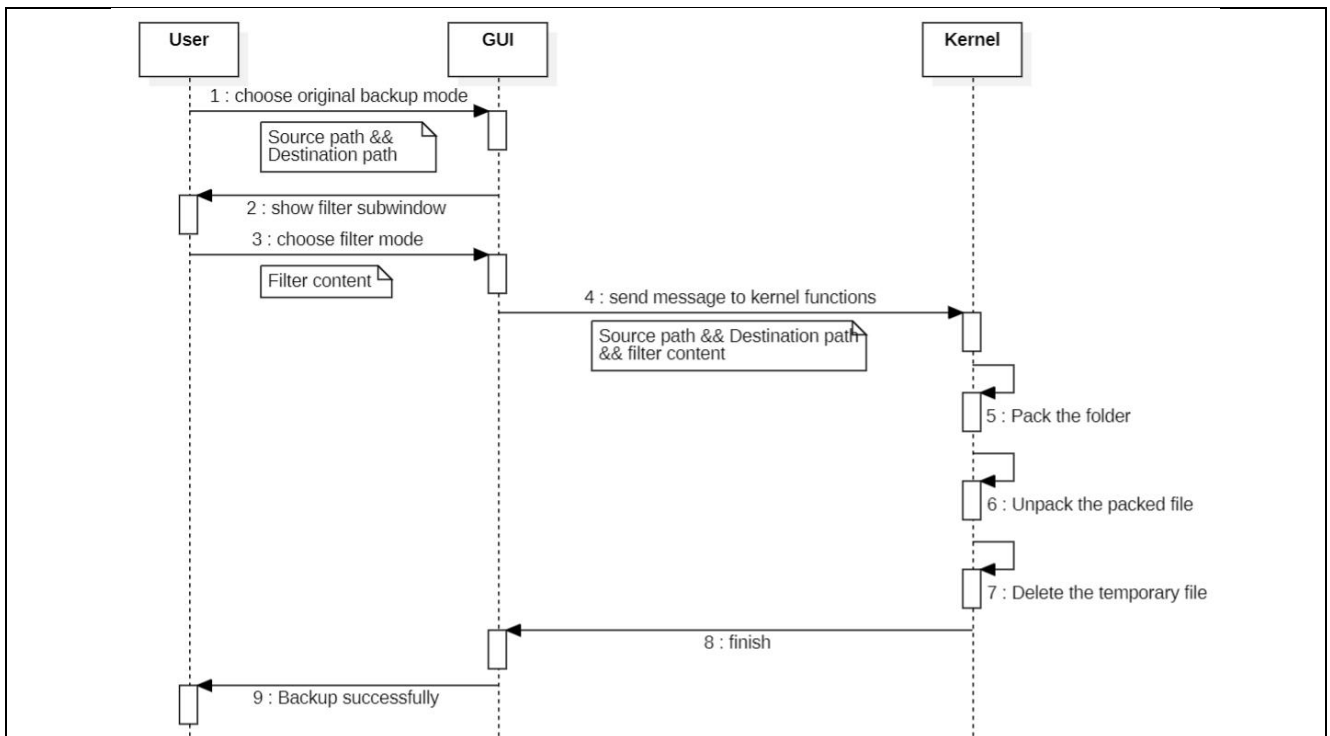
1. 用户选择打包备份模式，输入文件路径。
2. GUI 将信息传送给内核函数。
3. 内核执行打包程序至目的路径。
4. 内核执行压缩程序压缩打包好的文件。
5. 内核执行加密程序加密压缩好的文件。（由于程序实现上非加密模式也要使用默认密码加密，故都需要执行加密模块）
6. 内核执行删除程序删除临时文件。
7. 内核函数执行结束返回。
8. 若执行出错，GUI 向用户提示错误信息。

4.1.5 场景：筛选普通备份模式

4.1.5.1 场景描述

用户选择筛选普通模式，此时程序输入为要备份的文件夹路径，输出为文件夹。

4.1.5.2 顺序图



4.1.5.3 流程说明

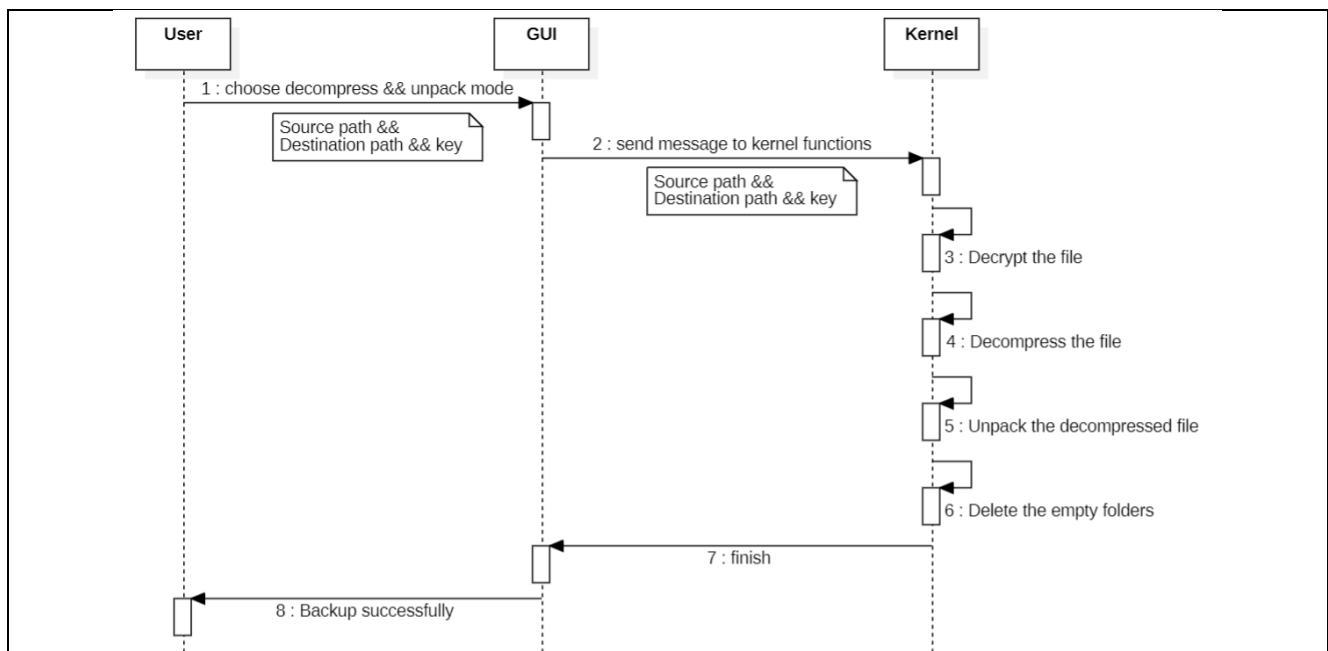
1. 用户选择筛选普通备份模式，输入文件路径。
2. GUI 弹出筛选子窗口。
3. 用户选择筛选模式以及确定筛选内容。
4. GUI 将信息传递给内核函数。
3. 内核执行打包程序至目的路径。
4. 内核将打包好的文件解包至目的路径。
5. 内核删除打包文件。
6. 内核函数执行结束返回。
7. 若执行出错，GUI 向用户提示错误信息。

4.1.6 场景：解压解包模式

4.1.6.1 场景描述

用户选择解压解包模式，此时程序输入为. tar.huf 文件，输出为文件夹。

4.1.6.2 顺序图



4.1.6.3 流程说明

1. 用户选择解压解包模式，输入文件路径与密钥。
2. GUI 将信息传送给内核函数。
3. 内核执行解密程序解密原文件。
4. 内核执行解压程序解压解密出来的文件
5. 内核执行解包程序解包解压出来的文件。
6. 内核执行删除程序删除空文件夹。
7. 内核函数执行结束返回。
8. 若执行出错，GUI 向用户提示错误信息。

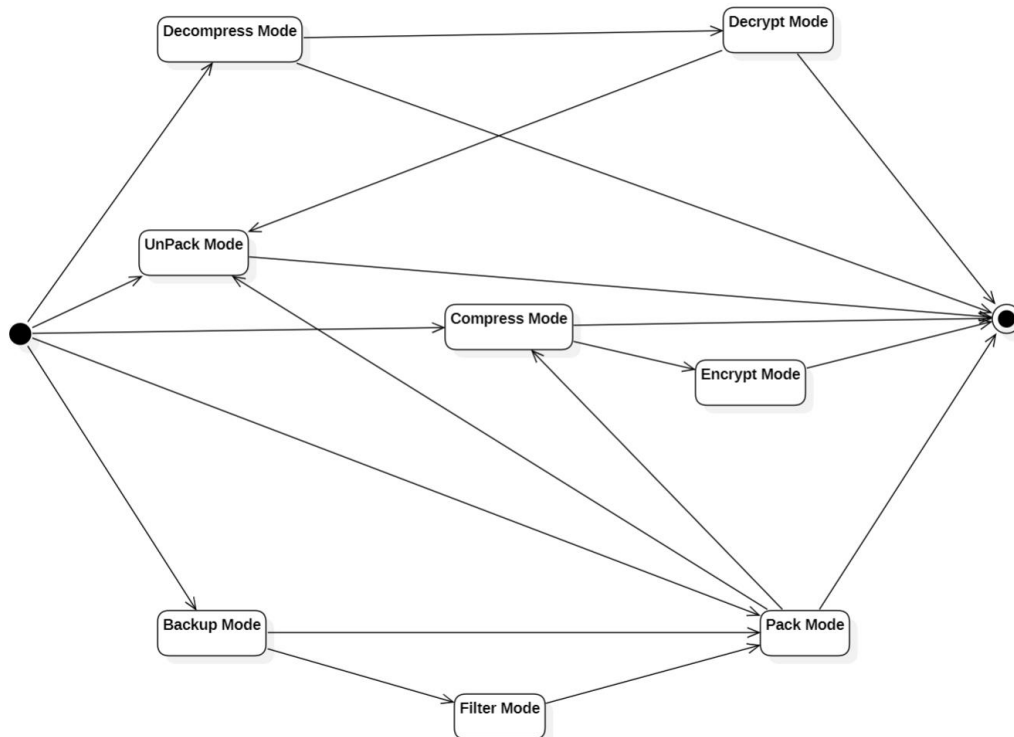
4.2 系统状态

4.2.1 系统状态切换

4.2.1.1 状态描述

下图为本程序十种基础状态切换图。

4.2.1.2 状态转换图



4.2.1.3 状态图说明

1. 起始状态，无前置状态，后置状态为备份、打包、解包、压缩、解压状态。
2. 备份状态，前置状态为起始状态，后置状态为过滤、打包状态。
3. 打包状态，前置状态为起始、备份、过滤状态，后置状态为压缩、解包（当且仅当普通备份模式）、终止状态。
4. 解包状态，前置状态为起始、解密、打包状态，后置状态为终止状态。
5. 压缩状态，前置状态为初始、打包状态，后置状态为加密、终止状态。
6. 解压状态，前置状态为初始状态，后置状态为解密、终止状态。
7. 加密状态，前置状态为压缩状态，后置状态为终止状态。
8. 解密状态，前置状态为解压状态，后置状态为解包、终止状态
9. 筛选状态，前置状态为备份状态，后置状态为打包状态。
10. 终止状态，前置状态为打包、加密、加密、压缩、解包、解压、解密状态，无后置状态。

软件测试报告（20 分）

1. 引言

为了尽可能的找出软件的不足，提高软件的质量，促进软件的成功验收，专门制定了本大纲。其主要目的在于为所要进行的测试工作制定各种必要的准则和规范，以及在有关方面协议的基础上对测试工作进行合理组织与管理。

2. 功能测试

2.1 打包模块

程序版本号：1.0	模块名：打包模块
测试用例编号：TestCase-Pack-01	用例级别：重要
用例名称：打包文件夹	测试时间：2022/11/10
预置条件：原目录存在。	

测试输入：原目录与目的目录	
操作步骤	1) 选择 Pack 页面。 2) 输入文件路径。 3) 执行程序。
预期结果：成功打包文件夹	
实际输出：成功打包文件夹	
测试人员：张镭麒	

2.2 解包模块

程序版本号：1.0	模块名：解包模块
测试用例编号：TestCase-Unpack-01	用例级别：重要
用例名称：解包. tar 文件	测试时间：2022/11/10
前置条件：.tar 文件存在，目的目录不存在。	
测试输入：.tar 文件路径与目的目录	
操作步骤	1) 选择 Unpack 页面。 2) 输入文件路径。 3) 执行程序
预期结果：成功解包. tar 文件	
实际输出：成功解包. tar 文件	
测试人员：张镭麒	

程序版本号：1.0	模块名：解包模块
测试用例编号：TestCase-Unpack-02	用例级别：普通
用例名称：删除空文件夹	测试时间：2022/11/10

预置条件： .tar 文件存在，目的目录不存在，.tar 文件存在空文件夹。	
测试输入： .tar 文件路径与目的目录	
操作步骤	1) 选择 Unpack 页面。 2) 输入文件路径。 3) 选择删除空文件夹选项。 4) 执行程序。
预期结果： 成功解包.tar 文件且删除了空文件夹。	
实际输出： 成功解包.tar 文件且删除了空文件夹。	
测试人员： 张镕麒	
2.3 压缩模块	
程序版本号： 1.0	模块名： 压缩模块
测试用例编号： TestCase-Compress-01	用例级别： 重要
用例名称： 压缩.tar 文件	测试时间： 2022/11/10
预置条件： .tar 文件存在。	
测试输入： .tar 文件路径与目的目录	
操作步骤	1) 选择 Compress 页面。 2) 输入文件路径。 3) 执行程序。
预期结果： 成功压缩.tar 文件。	
实际输出： 成功压缩.tar 文件。	
测试人员： 张镕麒	
程序版本号： 1.0	模块名： 压缩模块
测试用例编号： TestCase-Compress-02	用例级别： 普通

用例名称：删除临时文件		测试时间：2022/11/10
预置条件：.tar 文件存在。		
测试输入：.tar 文件路径与目的目录		
操作步骤	1) 选择 Compress 页面。 2) 输入文件路径。 3) 选择删除临时文件选项。 4) 执行程序。	
预期结果：成功压缩.tar 文件且删除了临时文件。		
实际输出：成功压缩.tar 文件且删除了临时文件。		
测试人员：张镕麒		

2.4 解压模块

程序版本号：1.0		模块名：解压模块
测试用例编号：TestCase-Decompress-01		用例级别：重要
用例名称：解压.tar.huf 文件		测试时间：2022/11/10
预置条件：.tar.huf 文件存在。		
测试输入：.tar.huf 文件路径与目的目录		
操作步骤	1) 选择 Decompress 页面。 2) 输入文件路径。 3) 执行程序	
预期结果：成功解压.tar.huf 文件		
实际输出：成功解压.tar.huf 文件		
测试人员：张镕麒		

程序版本号：1.0		模块名：解压模块
-----------	--	----------

测试用例编号：TestCase-Decompress-02		用例级别：普通
用例名称：解压解包.tar.huf 文件		测试时间：2022/11/10
预置条件：.tar.huf 文件存在，目的目录不存在。		
测试输入：.tar.huf 文件路径与目的目录		
操作步骤	1) 选择 Decompress 页面。 2) 输入文件路径。 3) 选择解包选项。 4) 执行程序。	
预期结果：成功解压解包.tar.huf 文件		
实际输出：成功解压解包.tar.huf 文件		
测试人员：张镕麒		

2.5 加密模块

测试用例如下表所示：

程序版本号：1.0		模块名：加密模块	
测试用例编号：TestCase-Encrypt-01		用例级别：重要	
用例名称：加密压缩.tar 文件		测试时间：2022/11/10	
预置条件：原.tar 文件存在。			
测试输入：原.tar 文件路径与目的目录			
操作步骤		1) 选择 Compress 页面。 2) 输入文件路径。 3) 选择 Encrypt 选项 4) 输入密码 5) 执行程序。 6) 通过 winhex 查看文件二进制内容。	
预期结果：成功加密压缩.tar 文件			
实际输出：成功加密压缩.tar 文件。			

测试人员：张镕麒

2.6 解密模块

程序版本号：1.0		模块名：解密模块
测试用例编号：TestCase-Decrypt-01		用例级别：重要
用例名称：解密解压. tar. huf 文件		测试时间：2022/11/10
预置条件：原. tar. huf 文件存在。		
测试输入：原. tar. huf 文件路径与目的目录		
操作步骤	1) 选择 Decrypt 页面。 2) 输入文件路径。 3) 输入正确的密码。 4) 执行程序。	
预期结果：成功解密解压. tar. huf 文件		
实际输出：成功解密解压. tar. huf 文件		
测试人员：张镕麒		

2.7 备份模块

程序版本号：1.0		模块名：备份模块
测试用例编号：TestCase-Backup-01		用例级别：重要
用例名称：普通备份文件夹		测试时间：2022/11/10
预置条件：原目录存在，目的目录不存在。		
测试输入：原目录与目的目录		
操作步骤	1) 选择 Backup 页面。 2) 输入文件路径。 3) 执行程序。	
预期结果：成功普通备份文件夹。		

实际输出： 成功普通备份文件夹。
测试人员： 张镕麒

程序版本号： 1.0	模块名： 备份模块
测试用例编号： TestCase-Backup-02	用例级别： 重要
用例名称： 打包备份文件夹	测试时间： 2022/11/10
预置条件： 原目录存在。	
测试输入： 原目录与目的目录	
操作步骤	1) 选择 Backup 页面。 2) 输入文件路径。 3) 选择 Pack 选项。 4) 执行程序。
预期结果： 成功打包备份文件夹。	
实际输出： 成功打包备份文件夹。	
测试人员： 张镕麒	

程序版本号： 1.0	模块名： 备份模块
测试用例编号： TestCase-Backup-03	用例级别： 重要
用例名称： 压缩备份文件夹	测试时间： 2022/11/10
预置条件： 原目录存在。	
测试输入： 原目录与目的目录	
操作步骤	1) 选择 Backup 页面。 2) 输入文件路径。 3) 选择 Pack 选项。 4) 选择 Compress 选项。 5) 执行程序。

预期结果： 成功压缩备份文件夹。
实际输出： 成功压缩备份文件夹。
测试人员： 张镕麒

程序版本号： 1.0	模块名： 备份模块
测试用例编号： TestCase-Backup-04	用例级别： 重要
用例名称： 加密备份文件夹	测试时间： 2022/11/10
预置条件： 原目录存在。	
测试输入： 原目录与目的目录	
操作步骤	1) 选择 Backup 页面。 2) 输入文件路径。 3) 选择 Pack 选项。 4) 选择 Compress 选项。 5) 选择 Encrypt 选项。 6) 输入密码。 7) 执行程序。
预期结果： 成功加密备份文件夹。	
实际输出： 成功加密备份文件夹。	
测试人员： 张镕麒	

程序版本号： 1.0	模块名： 备份模块
测试用例编号： TestCase-Backup-05	用例级别： 普通
用例名称： 筛选模块复用与重置	测试时间： 2022/11/10
预置条件： 原目录存在，目的目录不存在。	
测试输入： 原目录与目的目录	

操作步骤	1) 选择 Backup 页面。 2) 输入文件路径。 3) 设置筛选条件。 4) 执行程序。 5) 更改文件路径。 6) 执行程序。 7) 清空筛选条件。 8) 更改文件路径。 9) 执行程序。
预期结果： 第一次备份被筛选，第二次备份被筛选，第三次备份没有被筛选，筛选模块的复用与重置功能正常。	
实际输出： 第一次备份被筛选，第二次备份被筛选，第三次备份没有被筛选，筛选模块的复用与重置功能正常。	
测试人员： 张镕麒	

2.8 筛选模块

程序版本号：1.0		模块名：筛选模块
测试用例编号：TestCase-Filter-01		用例级别：重要
用例名称：筛选路径 & 黑名单		测试时间：2022/11/10
预置条件：原目录存在，目的目录不存在。		
测试输入：原目录与目的目录		
操作步骤	1) 选择 Backup 页面。 2) 输入文件路径。 3) 点击 Filter 按钮。 4) 选择 Path & Blacklist 选项。 5) 输入筛选内容。 6) 点击 confirm 按钮。 7) 执行程序。	
预期结果：路径筛选功能正常，黑名单功能正常。		
实际输出：路径筛选功能正常，黑名单功能正常。		
测试人员：张镕麒		

程序版本号：1.0		模块名：筛选模块	
测试用例编号：TestCase-Filter-02		用例级别：重要	
用例名称：筛选路径 & 白名单		测试时间：2022/11/10	
预置条件：原目录存在，目的目录不存在。			
测试输入：原目录与目的目录			
操作步骤		1) 选择 Backup 页面。 2) 输入文件路径。 3) 点击 Filter 按钮。 4) 选择 Path & Whitelist 选项。 5) 输入筛选内容。 6) 点击 confirm 按钮。 7) 执行程序。	
预期结果：路径筛选功能正常，白名单功能正常。			
实际输出：路径筛选功能正常，白名单功能正常。			
测试人员：张镕麒			

程序版本号：1.0		模块名：筛选模块	
测试用例编号：TestCase-Filter-03		用例级别：重要	
用例名称：筛选文件名 & 白名单		测试时间：2022/11/10	
预置条件：原目录存在，目的目录不存在。			
测试输入：原目录与目的目录			
操作步骤		1) 选择 Backup 页面。 2) 输入文件路径。 3) 点击 Filter 按钮。 4) 选择 Name & Whitelist 选项。 5) 输入筛选内容。 6) 点击 confirm 按钮。 7) 执行程序。	

预期结果： 文件名筛选功能正常，白名单功能正常。
实际输出： 文件名筛选功能正常，白名单功能正常。
测试人员： 张镕麒

程序版本号： 1.0	模块名： 筛选模块
测试用例编号： TestCase-Filter-04	用例级别： 重要
用例名称： 筛选路径 & 黑名单	测试时间： 2022/11/10
前置条件： 原目录存在，目的目录不存在。	
测试输入： 原目录与目的目录	
操作步骤	1) 选择 Backup 页面。 2) 输入文件路径。 3) 点击 Filter 按钮。 4) 选择 Type & Blacklist 选项。 5) 输入筛选内容。 6) 点击 confirm 按钮。 7) 执行程序。
预期结果： 文件类型筛选功能正常，黑名单功能正常。	
实际输出： 文件类型筛选功能正常，黑名单功能正常。	
测试人员： 张镕麒	

程序版本号： 1.0	模块名： 筛选模块
测试用例编号： TestCase-Filter-05	用例级别： 重要
用例名称： 筛选时间 & 白名单	测试时间： 2022/11/10
前置条件： 原目录存在，目的目录不存在。	
测试输入： 原目录与目的目录	

	操作步骤	1) 选择 Backup 页面。 2) 输入文件路径。 3) 点击 Filter 按钮。 4) 选择 Time & Whitelist 选项。 5) 输入筛选内容。 6) 点击 confirm 按钮。 7) 执行程序。	
	预期结果:	时间筛选功能正常, 白名单功能正常。	
	实际输出:	时间筛选功能正常, 白名单功能正常。	
	测试人员:	张镕麒	

3. 代码测试（可选）

仅对系统关键模块的源代码进行抽查，检查模块代码编写的规范性，批注的准确性，是否存在潜在性错误，以及代码的可维护性。包括：命名规范检查、注释检查、接口检查、数据类型检查、限制性检查。
推荐使用 lint 工具对代码进行全面静态分析，并给出代码检查结果。针对无法修改的告警，给出对应理由。

4. 性能测试（可选）

根据系统设计指标，或者对被测软件提出的性能指标，测试软件的运行性能，例如：传输连接最长时间、传输错误率、计算精度、记录精度、响应时限和恢复时限等。
测试系统的能力最高实际限度，即检查软件在一些超负荷情况下，功能实现的情况。例如：要求软件进行某一行行为的大量重复、输入大量的数据或大数值数据、对数据库进行大量复杂的查询等。

5. 健壮性测试（可选）

采用人工的干扰使应用软件、平台软件或者系统硬件出错，中断正常使用，检测系统的恢复能力，以及程序的内存、临界资源等在负载压力下的正确性。

6. 测试结果分析

对所测试的模块在功能、代码、性能、健壮性等方面进行的测试结果汇总。

测试模块	测试项目	测试结果	备注
打包模块	打包文件夹	测试通过	单元测试
解包模块	解包.tar 文件	测试通过	单元测试
	删除空文件夹	测试通过	单元测试
压缩模块	压缩.tar 文件	测试通过	单元测试
	删除临时文件	测试通过	单元测试
解压模块	解压.tar.huf 文件	测试通过	单元测试
	解压解包.tar.huf 文件	测试通过	集成测试

加密模块	加密压缩.tar 文件	测试通过	集成测试
解密模块	解密解压.tar.huf 文件	测试通过	集成测试
备份模块/	普通备份文件夹	测试通过	集成测试
	打包备份文件夹	测试通过	单元测试
	压缩备份文件夹	测试通过	集成测试
	加密备份文件夹	测试通过	集成测试
	筛选模块复用与重置	测试通过	集成测试
筛选模块	筛选路径 & 黑名单	测试通过	单元测试
	筛选路径 & 白名单	测试通过	单元测试
	筛选文件名 & 白名单	测试通过	单元测试
	筛选文件类型 & 黑名单	测试通过	单元测试
	筛选时间 & 白名单	测试通过	单元测试

七、实验结论:

包含的功能:

- 文件备份: 通过调用系统 API 遍历文件夹获取文件信息,
- 文件打包: 通过模仿 Ustar 头格式, 将文件的信息写入打包文件中。
- 文件解包: 通过模仿 Ustar 头格式, 利用打包文件的信息构造解包后的文件。
- 文件压缩: 使用哈夫曼编码, 统计文件词频, 使用哈夫曼树进行编码, 构造压缩文件头, 将加密后的文件内容与解码表写入压缩文件。
- 文件解压: 根据压缩文件获取解码表以及文件内容, 构造字典树, 然后将文件内容在字典树上匹配, 从而获得压缩前文件内容。
- 文件加密: 调用 openssl 库, 使用 AES_ecb_encrypt 函数对内容成段加密
- 文件解密: 调用 openssl 库, 使用 AES_ecb_encrypt 函数对内容成段解密
- 元数据支持: 通过 sys/stat.h 头文件调用 stat 函数获取文件信息并存储到打包文件中, 在解包时根据该信息, 通过 utime.h、grp.h、pwd.h 等头文件的函数修改新生成的文件的
- 自定义路径备份: 通过 Qt 提供的窗口间结构化信息传递的传递方式获取筛选内容, 在备份的遍历过程中检查当前状态是否满足筛选条件, 若不满足则返回, 否则将该文件加入打包文件中。
- 自定义类型备份: 通过 Qt 提供的窗口间结构化信息传递的传递方式获取筛选内容, 在备份的遍历过程中检查当前状态是否满足筛选条件, 若不满足则返回, 否则将该文件加入打包文件中。
- 自定义名字备份: 通过 Qt 提供的窗口间结构化信息传递的传递方式获取筛选内容, 在备份的遍历过程中检查当前状态是否满足筛选条件, 若不满足则返回, 否则将该文件加入打包文件中。
- 自定义日期备份: 通过 Qt 提供的窗口间结构化信息传递的传递方式获取筛选内容, 在备份的遍历过程中检查当前状态是否满足筛选条件, 若不满足则返回, 否则将该文件加入打包文件中。
- GUI 界面: 除满足基础的备份等要求外, 提供多种人性化功能:
 - 1、提供选择文件路径子窗口, 便于输入路径。
 - 2、选定备份模式后, 自动补充目的路径以及后缀。
 - 3、解压选择目的路径后自动补充文件名。
 - 4、满足单次备份与还原需求。即在 Backup 界面提供多种备份模式 (如打包后加密压缩), 在 Decompress 界面提供一次性解压解包模式。
 - 5、筛选条件可以在多次备份过程中自动保持不变。

- 6、筛选提供黑白名单功能，筛选更灵活。对于路径筛选而言，实现时需要注意其逻辑的非对称性。遍历时若符合黑名单要求，需要停止遍历，不符合黑名单要求，需要继续遍历。而遍历时若符合白名单要求，需要继续遍历，如果原文件夹不符合白名单要求，也需要继续遍历。（即父目录不处于白名单而父目录的子目录处于白名单中）
- 7、对应第六点白名单的情况，此时会产生空文件夹，程序提供了删除空文件夹的功能。
- 8、程序提供了保存临时文件的功能，以便用户使用程序出错时可以找到问题原因。
- 9、选择备份方式时，程序会自动显示可能的模式。即基础模式只显示打包选项，打包模式才显示压缩选项，压缩模式才显示加密选项，加密模式才显示密码输入框。
- 10、提供丰富的报错种类。

八、总结及心得体会：

效率上，从本课程分数要求上说，从编写代码到写文档的这一过程，基本本人每天的得分效率是 10 分每天，特例是学习阶段（累计约三天，没有得分），以及完成自定义备份时（当天完成 15 分内容）。考虑到分数最终减 120 分，一切顺利的情况下，基本上本课程要想拿到高分最少需要 20 天完成（单人），从大学三年所有实验课中考虑，这门课的工作量算是相当大的。

工作量大的原因主要有几方面。本实验要求的诸多功能需要自学，有的甚至需要花费大量的时间。不同学生的知识背景不同，有的人不会使用 Qt，有的人没使用过 Linux，有的人不会云存储。并且事实上由于缺少对应的先修课程，大部分同学对这些内容都是不了解的，导致了很多队伍即便凑够三人难以完成大部分额外要求，所以个人认为该课程作为必修课难度偏高。

除此之外，在本课程的自学过程中，经常需要翻阅英文文档也是一大难点，如调用 openssl 库函数、Ustar 头文件、以及其他 Linux 系统 API，均缺少足够丰富的中文资料。我相信这也是难倒不少同学的原因之一。

本次实验让我完整地体验到了软件开发的过程，在先前校外校暑期夏令营中本人也完整参与了一次软件开发项目，但是当时没有撰写文档，体会并不是十分深刻。此次试验也让我深入体会了 Debug 的过程，在此期间，经常是包括学习博客与 Debug 教程博客共二十余个页面长期停留在我的浏览器上，经常解决一个问题又引出来了许多新的问题，最终一个一个解决才能回到最初的位置，完成预想的功能，对我而言是一次在本科学习过程中难得的经历。

九、对本实验过程及方法、手段的改进建议：

个人认为文件备份并不是十分贴合于面向对象开发方法，因为在本实验中要求实现的功能基本上都是对文件的属性及内容的流水线化处理，不同的处理过程（压缩解压打包解包）差异很大，缺少经典的明显的类之间的继承。个人认为更贴合结构化开发方法，事实上本代码正是使用结构化开发实现的内核函数，使用面向对象开发方法实现的用户界面。

此外，对于第八点中提到的难点问题，希望课程组可以做出改善，如：提供相关详细资料，示例代码等。在做实验的过程中感觉实验的要求个人觉得有些不明确，存在模棱两可的理解，在深入理解的过程中也花费了很久的时间。

报告评分：

指导教师签字：