一、 项目名称

赛题编号： 59

赛题名称： python 编写更新提醒软件

二、 团队信息

（一） 团队名称

团队名称：Hollow Man

团队编号：1798858424

（二）团队成员信息

Hollow Man Hollow Man

本项目为 Hollow Man (@hollowman6) 个人项目，独立开发。

三、 项目详细方案

（一） 对题目的技术背景分析

背景：

Centos 提供了 DNF 自动更新和 Cockpit Web 控制台更新 2 中自动更新方式，都有局限性，此项目预期能够看到 openEuler 中软件包更新的提醒，但不强制或不自动全量更新，而是有使用者手动选择需要更新的软件包，并且预期能够进一步提供可选择更新列表进行一键式更新。

项目要求：

鉴于目前社区未提供更新提醒软件，这对于操作系统而言是一款易用性问题，同时在桌面系统中该问题更为突出，因而需要自研一款 openeuler 特有的更新提醒软件。

产出标准：

1，基于 openEuler 社区版本，功能实现满足轻量化要求，不能对系统造成性能负载，通过 python/shell 开发。

2， 具备定时轮询功能，并且通过配置实现轮询时间，更新提示内容，显示风格，日志路径，更新提示用户范围，源地址配置等设置。

3，提供一键式全量升级，部分升级等命令接口，升级完成后提示用户重启系统。

4，兼容 yum/dnf 升级机制。

5，满足 x86_64、aarch64 等多架构支持

6，支持在线云场景部署及离线场景部署

可行性分析：

1，项目将通过 python3 脚本，在安装有 python3 解释器的系统中直接运行。

2，软件可以通过 python3 中的 os 库调用系统的 yum/dnf 升级命令（或者相关 API），并且通过分析执行的输出结果，来获取可以更新的软件列表，提示用户选择一键式全量升级或者部分升级等相应操作，随后再次调用 yum/dnf 命令进行升级，升级完成后提示用户重启系统，因而兼容 yum/dnf 升级机制，同时支持在线云场景部署及离线场景部署。

3，软件可以分别编写 CLI 界面和 UI 界面。CLI 界面为直接输出可以更新的软件包数量，提示用户进行手动执行 yum/dnf 命令更新。

4，UI 界面预期使用 PyQT5 制作（因为 PyQT5 制作的图形界面更美观，当然考察到 PyQT5 占用内存过大并且需要额外安装 python 库，用 Python3 自带的 Tkinter 库也在考虑范围中。）

5，UI 界面可以配置实现轮询时间，更新提示内容，显示风格，日志路径，更新提示用户范围，源地址配置，在更新时是否自动检查可用的磁盘空间，移除不需要的依赖，以及官方软件仓库 yum/dnf 源地址配置等设置。用户还可以在首选项中配置是否在更新执行完毕之后自动清除缓存，优先包管理器配置等设置。软件会将这些配置保存在或更新在 config.json 文件中，每次软件运行时会自动读取其中的内容，用户也可以手动修改其中的配置，重启软件后生效。

6，定时轮询功能可通过 shell 脚本来实现，对于在软件中通过配置的管理员指定的用户，配置 crontab 通过 shell 脚本来每隔用户指定的时间启动 python3 脚本获取是否有存在最新软件。同时，在 /etc/rc.d/rc.local 中添加启动项，shell 脚本判断当前用户在软件中通过配置的管理员指定的用户范围中时则执行 python3 脚本获取是否有存在最新软件。

7，因为 python 解释器在 x86_64、aarch64 等多架构下都可运行，因而无需担心此问题 。

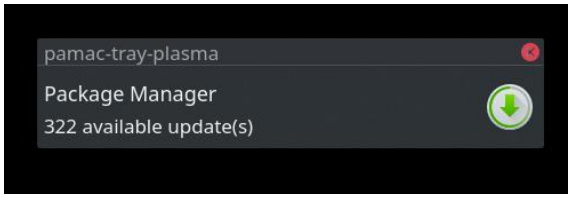
（二）对实现题目的方案设计

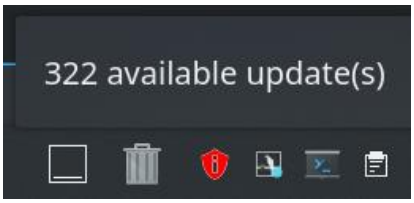
1、架构和流程

仿照 Arch Linux 下 KDE Plasma 桌面环境 pacman 包管理器软件更新机制，根据上述技术背景分析，可以进一步细化我们的设计方案。

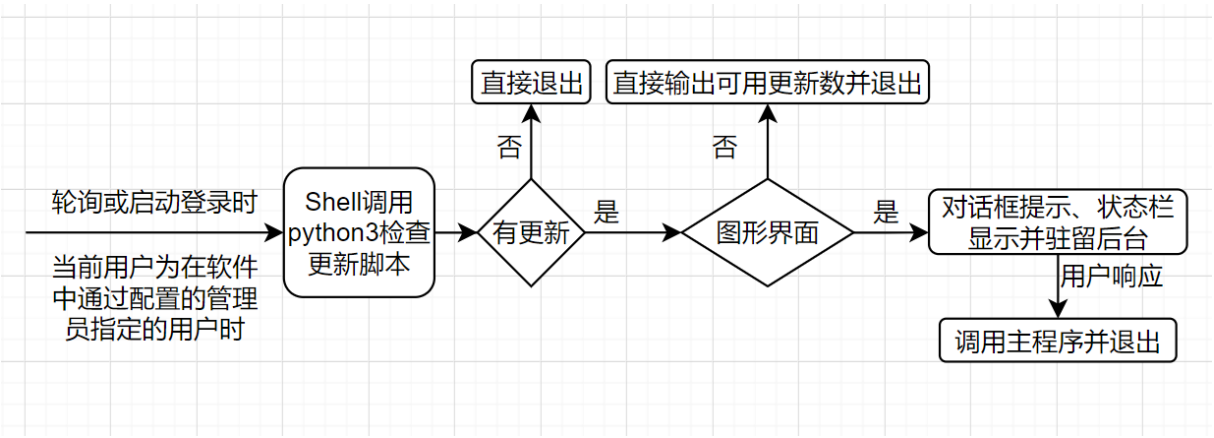
软件将分为两个部分。

第一个部分为轻量级的 Python3 软件更新检查脚本，用于轮询时和登录启动时，当判断为在软件中通过配置的管理员指定的用户时，通过 shell 调用执行，其功能仅为执行检查更新功能，如果有更新并且在桌面环境下，则通过 Python3 自带的轻量级 Tkinter 库弹出对话框，显示可用的更新数量，并且在系统状态栏处显示“可用更新”图标并驻留后台，方便用户随时打开进行进一步操作，如果无更新则自动退出。当用户点击对话框或者点击状态栏图标时，则调用打开主程序，此检查更新脚本退出；如果有更新并且在非桌面环境下，则直接输出显示可用更新数量并退出。





流程图：



第二个部分为主程序部分，此部分计划用 PyQT5 或 Tkinter 库编写，当打开后程序将显示可供更新的软件列表，如果没有可更新的软件程序将显示“目前所有软件都为最新状态”。默认软件列表为全选中状态，用户可以选择忽略指定的更新或者忽略全部更新。

被忽略的更新版本号及名称都会通过 python 执行 sqlite3 记录进入数据库，在下一个版本未出现之前将不再提示此更新。程序每次检查更新之后都会读取此忽略数据库来删减从 yum/dnf 获取的可更新软件列表。如果检查到数据库中某一项出现了更新的版本，数据库中的这一项记录将会被删除，同时再次提

醒用户是否更新，这样做维护了数据库的最小化。

下面这张图是 Arch Linux 下 KDE Plasma 桌面环境 pacman 包管理器软件更新界面：

软件列表中将显示可更新软件包名称，可更新到的版本号，当前版本号，以及来源（yum/dnf，同一软件包优先 dnf，可配置优先源）。用户可以通过名称，软件来源，大小，日期等对软件列表进行选择，右侧复选栏可以选中，来选择是否升级该软件包。**同时，除此以外，软件还会提供更新列表搜索功能，方便用户直接选中。**底部将显示总下载量大小，用户点击应用就可以开始进行进一步操作。

当用户点击应用后，软件就会自动生成事务摘要供用户确认。

下面这张图是 Arch Linux 下 KDE Plasma 桌面环境 pacman 包管理器软件更新事务概要确认界面：

用户确认完毕后即开始软件包的下载。软件界面下方将显示当前软件的操作以及总进度条和预计剩余时间，用户可以在下载过程中随时点击取消，软件将停止下载。当下载完毕进行安装时取消按钮将变为灰色，同时阻止用户进行关机等操作，防止产生软件包依赖不完整问题。升级完毕后提示用户重启系统。

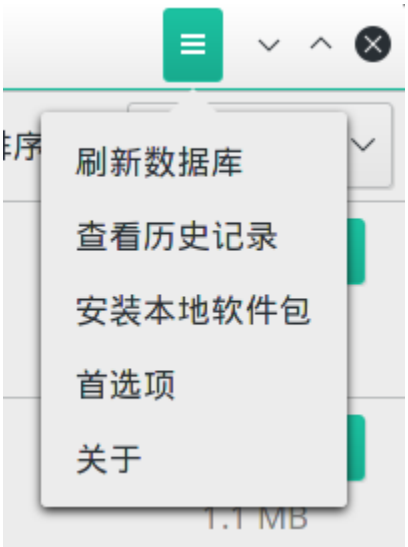
下面这张图是 Arch Linux 下 KDE Plasma 桌面环境 pacman 包管理器软件更新界面：

用户还可以点击相关按钮查看实时进展详情（日志）。

下面这张图是 Arch Linux 下 KDE Plasma 桌面环境 pacman 包管理器软件更新实时进展详情界面：

菜单栏可以提供刷新数据库，查看更新日志等功能。

下图是 Arch Linux 下 KDE Plasma 桌面环境 pacman 包管理器软件菜单栏界面：

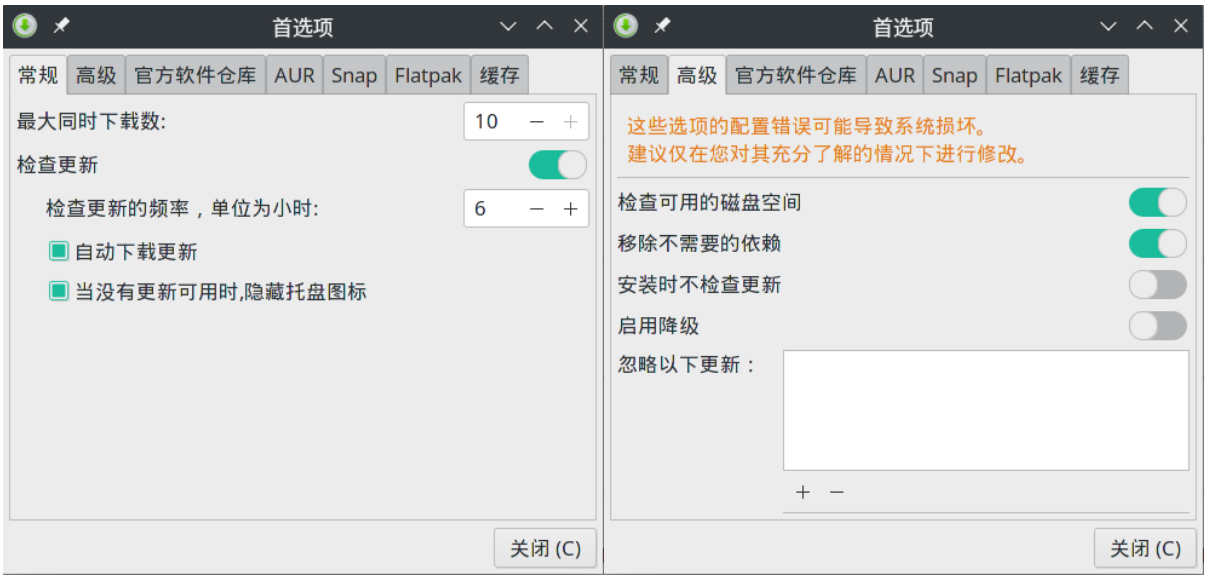


下图是 Arch Linux 下 KDE Plasma 桌面环境 pacman 包管理器软件查看安装历史记录界面：

日志将会将 yum/dnf 命令执行过程中的输出加上时间戳之后进行记录，用户可以搜索这一日志获得单条记录。

在首选项界面中，用户可以配置是否启用自动检查更新，以及自动检查更新的频率，并且还可以配置更新提示内容显示风格，日志路径，更新提示用户范围，在更新时是否自动检查可用的磁盘空间，移除不需要的依赖，以及官方软件仓库 yum/dnf 源地址配置，优先包管理器配置等设置。用户还可以在首选项中配置是否在更新执行完毕之后自动清除缓存。软件还会在界面中提供清除缓存按钮。软件会将这些配置保存在或更新在 config.json 文件中，每次软件运行时会自动读取其中的内容。

下四张图是 Arch Linux 下 KDE Plasma 桌面环境 pacman 包管理器软件首选项配置界面：



2、工作进度计划

2020 年 12 月-2021 年 1 月中旬：

因为这个时候我需要忙于大学期末考试，因而没有太多时间做任务，可以在这个期间学习 yum/dnf 更新机制，同时补充项目所需知识，进一步完善项目规划。

2021 年 1 月中旬-1 月底：

编写轻量级的 Python3 软件更新检查脚本。

2021 年 2 月：

编写主程序部分。

2021 年 3 月：

测试程序，提交作品至仓库，准备决赛答辩。

为可能的项目延期预留的时间。

3、实现难点分析

项目主要难点为 yum/dnf 和 UI 之间的信息交换传递。预期可以通过 API 解决。如果没有相关 API，则可以通过正则表达式分析 yum/dnf 的输出信息，经过处理和用户选择后通过 python 系统调用执行。

（三）对技术实现提供验证方案

将会在 x86_64、aarch64 等多架构下进行测试，以及在华为云 ECS 中运行 OpenEuler 系统来实现在线云场景部署测试，同时在自己的电脑上运行 OpenEuler 系统来测试离线场景部署。测试主要内容为对软件具有的功能能否分别在这些平台上正常运行进行测试，如果不能则进行修改完善。