项目申请书

项目名称：OpenFDE 的任务管理器

姓名：陈寒江

联系邮箱：17830831606@163.com

导师：蕉下客

2025 年 5 月 22 日

1 项目简介

1.1 开发背景

目前，OpenFDE 缺乏一款能够同时管控 Linux 和 Android 程序的任务管理器。当某些应用出现内存泄漏或用户同时开启多个大型应用时，可能会导致系统卡顿。在这种情况下，用户往往难以直观地排查问题原因，从而影响使用体验。

为了解决这一问题，计划开发一款专为 Android PC 化设计的任务管理器，用于监控并管理设备上正在运行的进程和应用程序。在为用户提供实时系统资源使用情况的基础上，进一步优化 Android 在 PC 端的使用体验。

1.2 产出要求

- 提供 Android 和 Linux 程序详细进程信息，包括 CPU、内存、磁盘和网络使用情况。
- 支持结束进程、启动进程。
- 提供服务管理功能，允许用户启动、停止系统服务。
- 图形化界面，易于操作并提供详细的帮助文档和提示，方便新手用户。

2 方案分析

阅读官方文档可知，OpenFDE 桌面环境采用了基于 Waydroid 的 Android App 容器化方案。该方案利用了容器技术以及 Linux 的命名空间功能（如进程、网络等），实现了 Android 环境与宿主机之间的资源隔离。这意味着，在 Waydroid 中运行的 Android Service 会以独立的进程形式存在，并且通过 Xserver 实现与 Linux 桌面的融合。后者运行在一个独立的 Android Service 容器中，使得 Linux 窗口程序（包括 GTK、QT 窗口程序）均可通过 X11 协议与其进行交互。

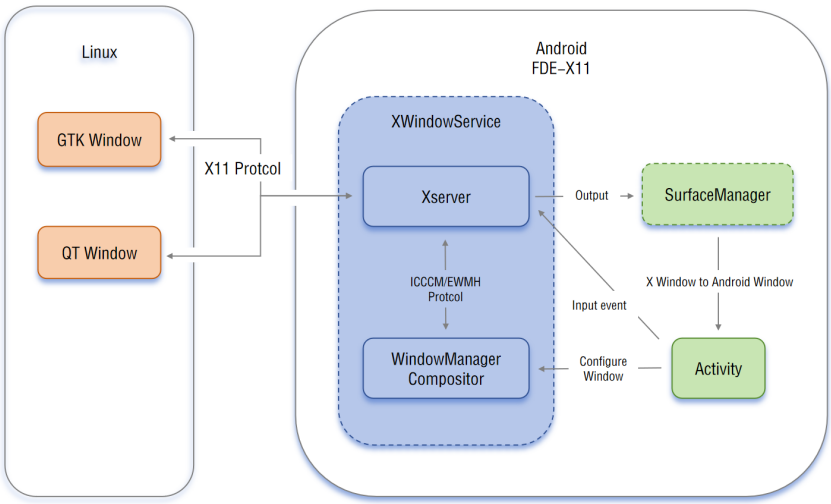

图 1: OpenFDE 桌面环境融合方案架构图

OpenFDE 的这一特点使得 Android Service 和普通的 Linux 进程在宿主机上能够共享同一套 Linux 任务管理器。

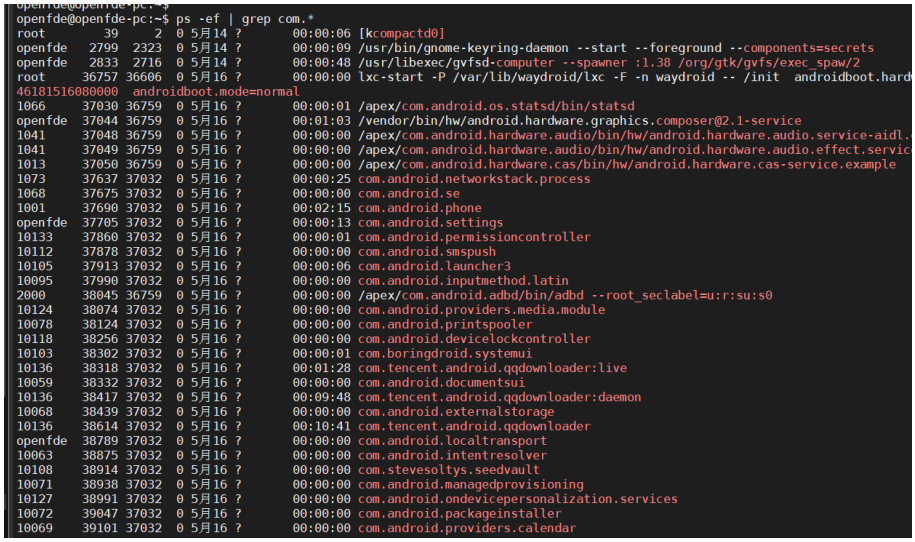

图 2: OpenFDE 下 PS 命令的运行结果

因此，本项目的关键在于直接从 Linux 任务管理器的设计思路出发，结合 OpenFDE 桌面环境中 Android App 容器化方案的特点，采用 C/S 架构设计 Linux 端任务管理器服务端及 PC 化 Android 端任务管理器客户端。

具体而言，任务管理器的服务端收集并处理进程的资源使用信息，包括但不限于 CPU 占用率、内存消耗、网络流量等指标。同时，服务端还需提供一套完整的调用接口，允许客户端访问这些数据，以确保跨平台通信的安全性和效率。经查阅资料，开源项目 `bashtop` 中的 `bashtop.psutil.py` 脚本可利用 `psutil` 得到所有进程信息，这完全满足 Linux 守护进程服务端的特点，可再次基础上扩展出调用接口。

对于客户端部分，则需要充分利用 Android UI 的响应式特点设计界面，使用户能够直观地查看和调度操作系统中所有类型的进程。

C/S 架构的一大难点在于选取通信方式，考虑到 OpenFDE 的基础 Waydroid 中的 Android 环境与宿主机之间通过 Linux 命名空间实现资源隔离，包括进程、网络等，宿主机原生程序与之进行 IPC 受到了容器的隔离，进一步导致访问受限，因此欲图通过 gbinder 实现进程间通信，必须确保宿主机上的程序能够正确解析 Android 服务使用的 AIDL 接口，而这一工作量非常大。

因此，根据异构环境通信的特点，另外选取了 gRPC 协议作为通信协议，这一选择相比 gbinder 方案具有独特优势，gRPC 利用了 HTTP/2 协议的多路复用、流式传输等特性，通信效率高，适合高性能场景，避免了本地 IPC 场景下突破进程隔离的较大工作。

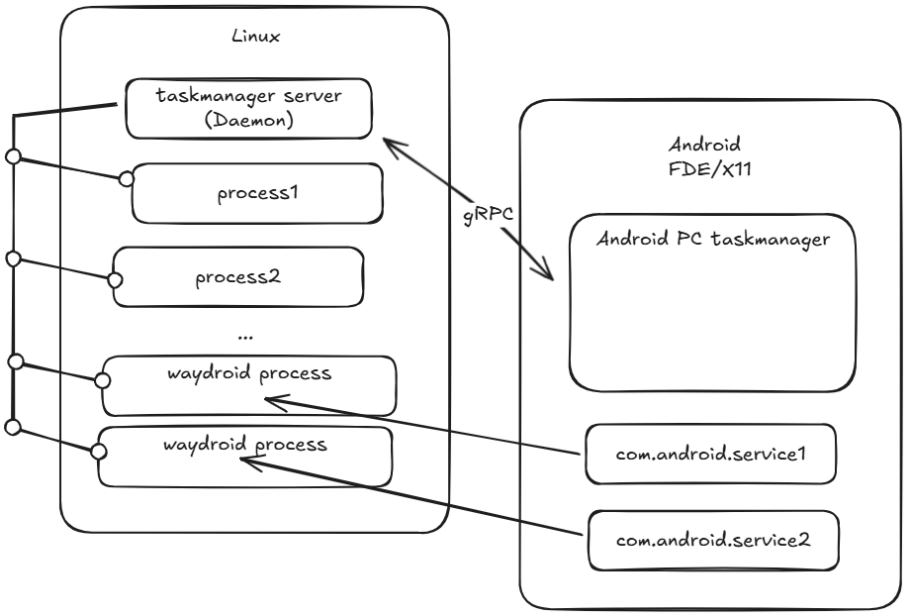

图 3: 任务管理器框架图

3 技术栈实践

开发工作从 Linux 端入手，服务端守护进程的实现参照 bashtop、btop、htop 等开源的流行交互式进程查看器的实现编写工具函数，将其注入到采用观察者模型为基础的服务端代码中当中，编程语言方面采用 Python 的超集 Cython 利用 `xmlrpc.server` 构建了一个轻量级的 RPC 服务器并实现发布-订阅模型，编程语言选择一方面是为了能在提高程序性能的同时，更好地借鉴 `bashtop.psutil.py` 的实现，另一方面为耦合 btop、htop 等进程查看器的已有 C/C++ 接口提供了方便之处，再者也能减少 Python 程序在后台保持长期运行时可能发生的内存泄露风险。

针对 Android PC 端，使用 Android Studio 创建一个 Jetpack Compose 工程，并特别选择了适用于大屏设备的应用选项，以支持 OpenFDE 桌面环境调整 App 窗口大小的需求。之所以采用这个流行的组合式声明 UI 框架，是因为它不仅完美契合了笔者丰富的前端开发经验，也能让开发者站在 Google 的肩膀上，享受近年来他们针对响应式 UI 适配工作的红利，在功能设计层面，可模仿成熟的开源竞品 Scene (`https://github.com/helloklf/vtools`)。通过引入 `grpc-android-kotlin` 依赖项，实现了与服务端的高效通信，基于 gRPC 的订阅机制能对远程服务接口进行优雅且高效的调用。

4 项目实施规划

项目将于 2025 年 7 月 1 日开始，至 9 月 30 日结束，共设计为三个月。实施规划如下：

- 环境搭建与前期准备 (7.1 - 7.7)
  - 使用 qemu-system-aarch64 虚拟 OpenFDE 主机环境，搭建 Android Studio 开发环境，adb 桥接两者的调试工作。
  - 配置 aarch64 Linux C++ 交叉编译工具链，配置 Python 开发工具链。
- Linux 端服务端原型开发与实现 (7.8 - 7.21)
  - 使用 Jetpack Compose 对任务管理器界面进行快速原型设计，留出订阅模型的接口，通

过 gRPC 编写 proto 标定数据传输规范，并尝试首次打通服务端与客户端的连接。

– 参照开源交互式进程查看器，使用 Cython 调用根据需求修改的 C/C++ 接口，参照 proto 重写发布模型代码。

- Android PC 端客户端功能开发与完善 (7.22 - 8.10)

  – Android PC 端联调通信接口层与任务管理器界面交互，完成进程信息显示的基本功能原型。

  – 添加任务管理器的进阶功能，实现仪表盘、CPU、网络等信息的界面原型。

- 联调测试与性能优化 (8.11 - 8.25)

  – 使用 Viztracer、cProfile、Valgrind 等工具，小步优化服务端内存占用，确保服务端守护进程在长期运行中不会发现内存泄漏问题。

  – 测试各种分辨率与窗口比例下的鲁棒性，完善对应的 UI 响应式优化工作。

- 文档整理与成果展示 (8.26 - 9.10)

  – 对源码的全部接口进行规范统一

  – 为源码添加配套注释和文档

  – 进行开发效果的初步展示

- 总结评估与项目收尾 (9.11 - 9.30)

  – 整理所有源代码、说明文档、展示应用

  – 完成项目总结报告

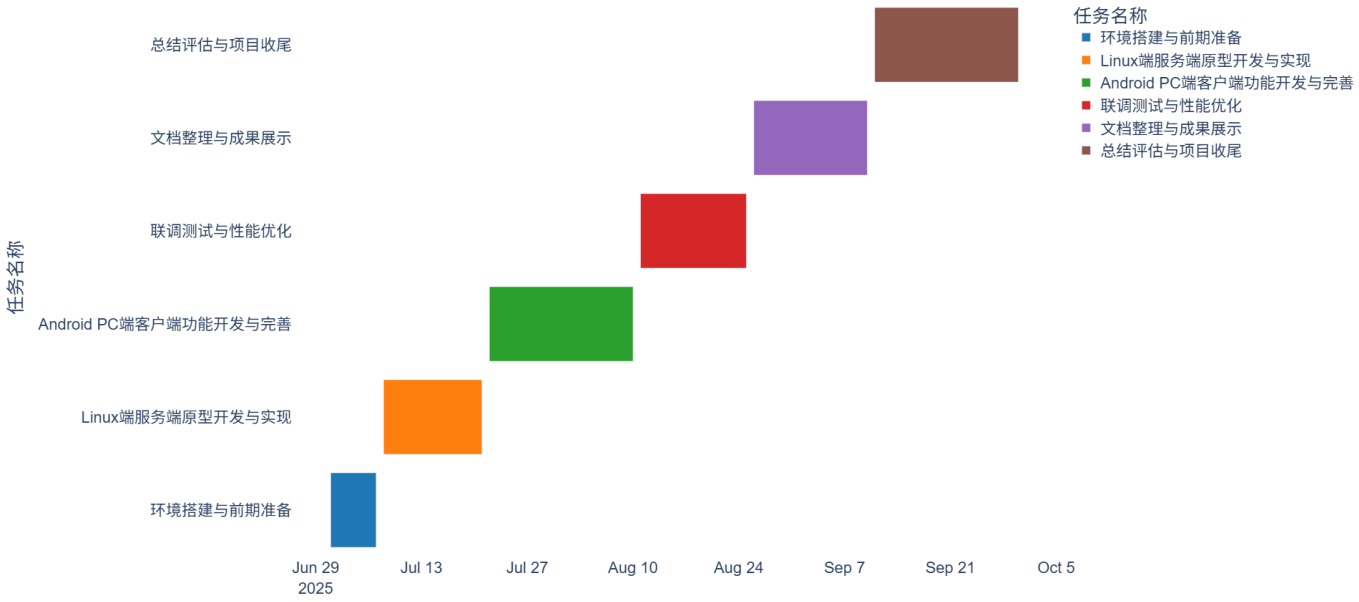

图 4: 任务管理器项目甘特图