跨平台数据采集器开发方案

By Xian2207, 13689903575, wszhangxian@126.com

1 目标

（1）使用 C++ 开发数据采集器，用于采集以下数据：进程、服务列表、端口、注册表、带宽、网络连接、用户列表、系统日志、进程闯进、进程暂停、进程删除、进程挂起、文件新增、文件读取、文件修改、文件删除、防火墙告警信息、杀毒软件弹窗监控、用户登录、用户注销、新增用户、删除用户、资源使用率变化；

（2）预留数据上传、行为识别接口；

（3）支持跨平台方案，Windows 发行版（含补丁版本）不少于 100 个；Linux 发行版不少于 75 个。

2 周期

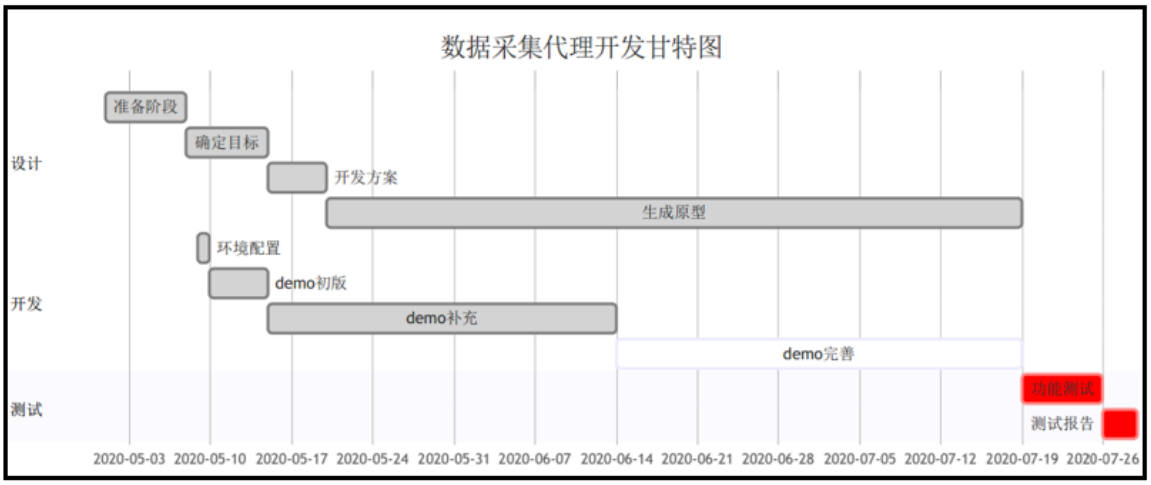

图 1-数据采集器开发甘特图

说明：

+ 灰色代表完成；
+ 白色框代表正在进行；
+ 红色框代表未完成。

表 1-数据采集器开发方案具体计划

序号	类型	任务	起始时间	截止时间	状态	备注
1	设计	准备阶段	501	508	完成	
		确定目标	509	515	完成	获取新数据内容
		开发方案	516	520	完成	cmake开发
		生成原型	521	720	完成	
2	开发	环境配置	509	509	完成	
		demo初版	509	515	完成	Win版
		demo补充	515	614	正在	Win版
		demo完善	615	720	正在	Win+Linx版
3	测试	功能测试	721	728	未完成	
		测试报告	728	731	未完成	

3 解决方案

3.1 微软跨平台方案

原理：利用 Visual Studio(VS)安装 Linux C++开发环境。分别创建 Windows 和 Linux 项目，写入代码。前者直接编译和生成 exe 应用，后者需 Linux 远程服务器或 VMware 或 Windows Subsystem Linux 运行代码，从而生成 binary 文件。

优点：VS 既可以开发 Windows 应用，又可以开发 Linux 应用。

缺点：如果没有远程 Linux Server/Desktop，用户须额外安装 Linux OS。VM 和 WSL 有开发缺陷，前者占物理机内存和磁盘空间；后者需特定机型，无界面，且需额外配置。

逻辑流程：

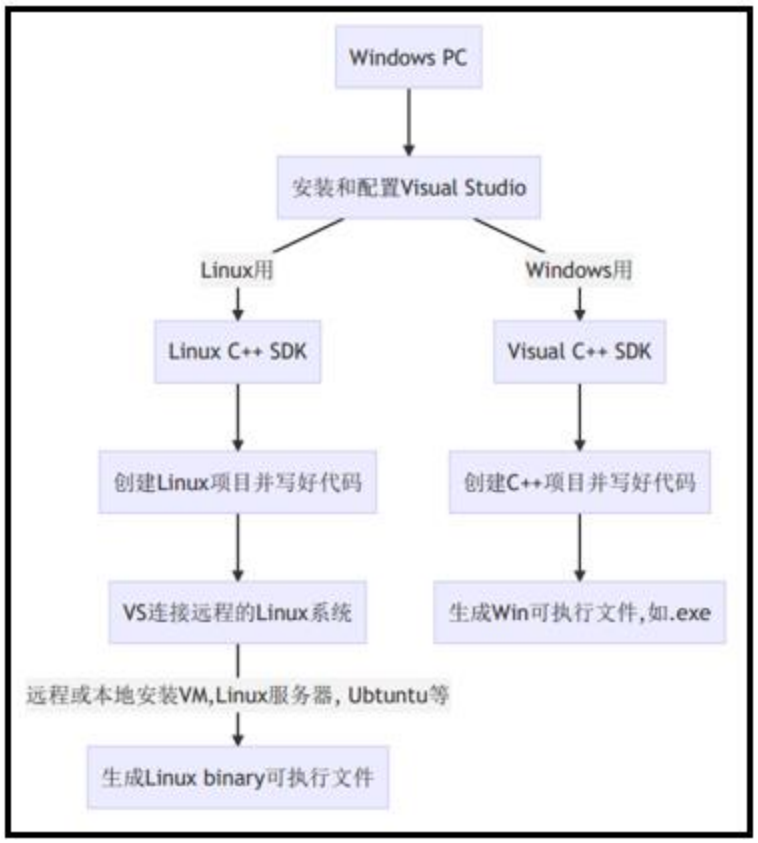

图 2-微软跨平台开发方案

3.2 双系统方案

原理：配置双系统，Windows 下开发 exe 应用，Linux 下开发 binary。

优点：系统分开，环境各自配置，互不影响，单独开发 Win/Linux 应用。

缺点：开发人员需配置双系统。

强制：不要在 Windows 里配置 MinGW 的 gcc/g++进而规避双系统，因为 MinGW 自带的 C/C++ 库有部分不同于纯 Linux OS 的 C/C++库。

逻辑流程：

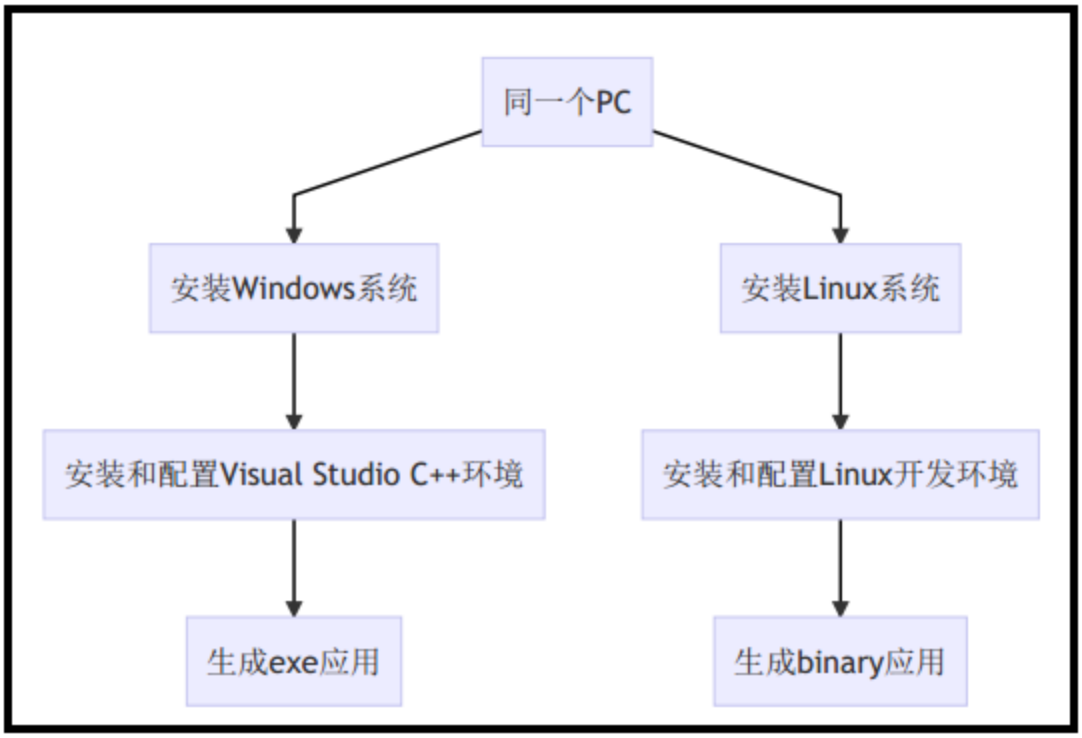

图 3-双系统开发方案

3.3 CMake 方案

原理：类似双系统开发方案，但推荐用 VS/VSCode/txt/vim 写好源码（例如头和源文件），编写 CMakeLists，最后用 CMake 根据目标平台（Windows/Linux 等）生成 exe。

优点：源码可跨平台编译出针对目标平台的应用程序。

缺点：需要掌握 CMake 语法。

逻辑流程：

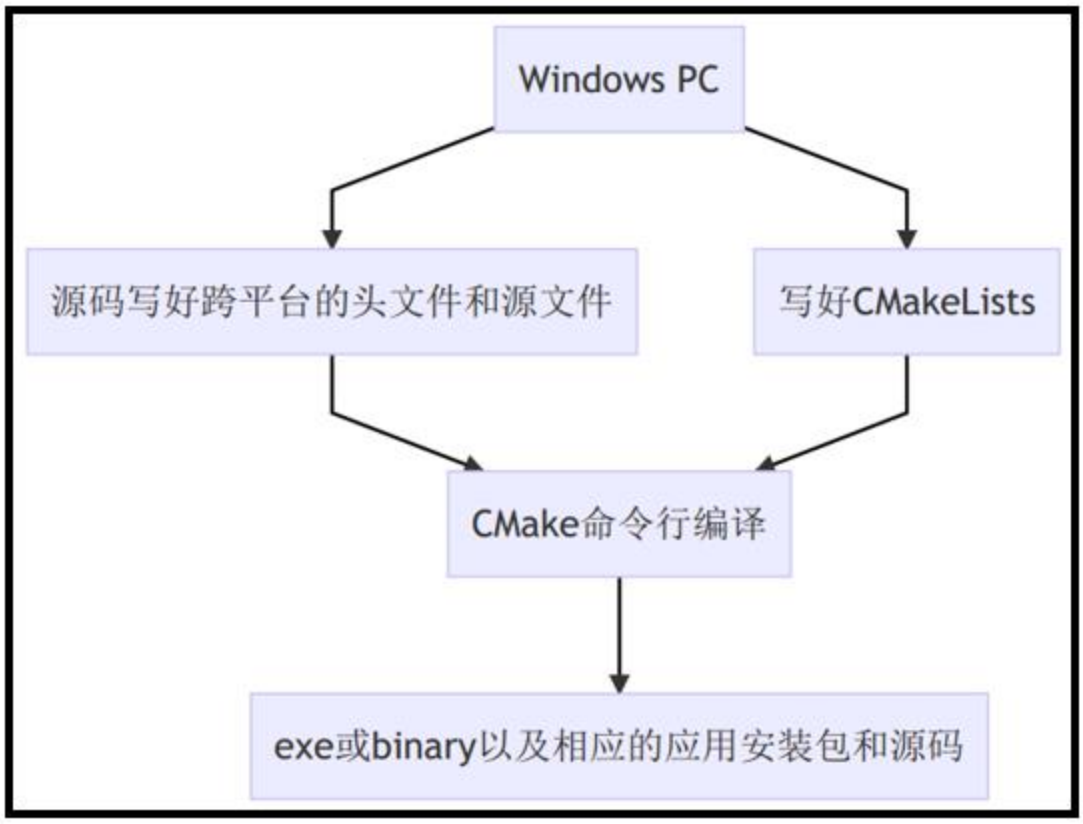

图 4-CMake 开发方案

3.4 脚本方案

原理：使用脚本语言 Python 或 Go 语言开发数据采集器。

优点：Go/Python 使用简单，开发便捷，支持各种平台。

缺点：需要安装对应的 SDK 或编译器运行环境。

逻辑流程：

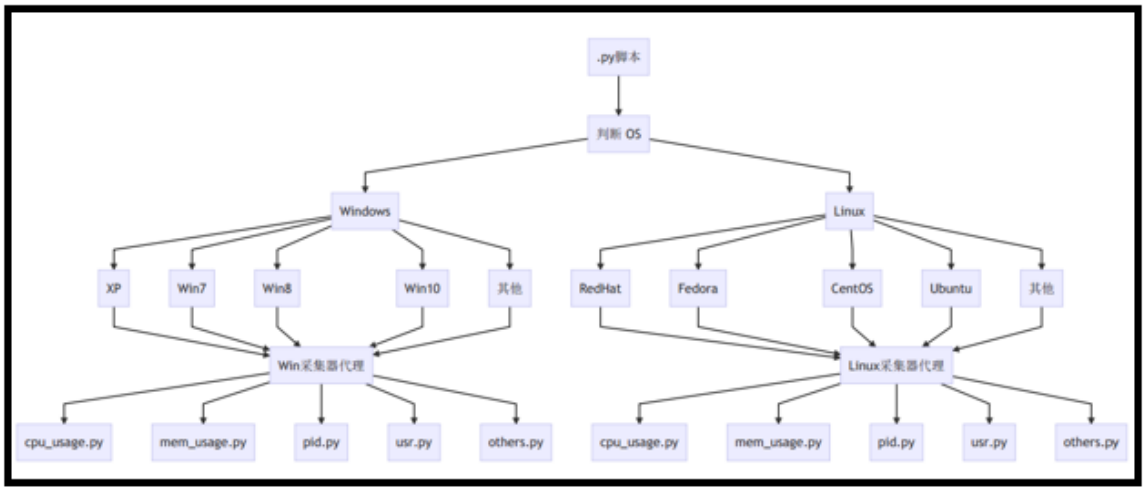

图 5-脚本开发方案

4 设计标准

4.1 CPU 使用率

CPU 使用率：它与 CPU 时钟精度有关，Windows 2000 professional 及以上版本，例如 Win7, Win8, Win2010 等，时间精度为 10-16ms。假设一个周期是 40ms，前 20ms 休眠让 CPU 处于虚度状态（即$Idle_{unused}$），后 20ms 执行某段程序（即CPU_{used}），则使用率的定义为

$$CPU_{usage} = \frac{CPU_{used}}{CPU_{used} + Idle_{unused}} \cdot 100\% = \frac{20}{20 + 20} \cdot 100\% = 50\%$$

4.2 内存使用率

内存使用率：使用的内存占总内存的比率，计算公式如下

$$Memory_{usage} = \frac{Memory_{used}}{Memory_{used} + Memory_{unused}} \cdot 100\%$$

4.3 其他数据

以磁盘容量为例，例如 C 盘，使用空间占磁盘总空间的比率，公式如下

$$Disk_{usage} = \frac{Disk_{used}}{Disk_{used} + Disk_{unused}} \cdot 100\%$$

磁盘容量分为三部分：

+ 总体容量：原始磁盘总大小，单位 byte；

+ 可用容量：磁盘剩余空间里的可用大小，单位 byte；

+ 剩余容量：磁盘剩余空间（含可用和坏道后不可用大小），单位 byte。

4.4 数据上传

由 CPU 使用率、内存使用率、磁盘使用率等组成的数据结构，以 TCP 连接方式由客户端发送给服务端。

5 代码结构

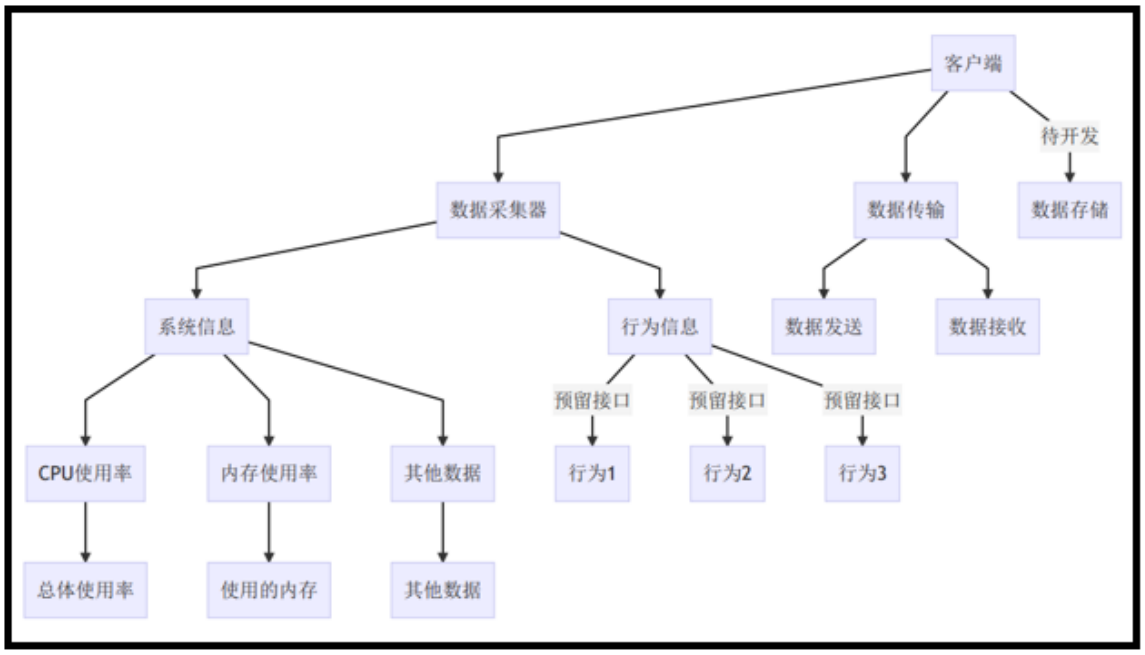

图 6-数据采集器代码结构

说明：

+ 并发模式：多线程根据任务数目和数据状态决定是否并发；

+ 数据存储：待开发。

6 跨平台部署

6.1 Windows

第一步：安装依赖

+ cmake-3.x.x-win64-x64.msi

+ Visual Studio (C++)

+ 源码

第二步：安装数据采集器

+ 方式 1：demo-1.0-Win64.exe（推荐）

+ 方式 2：经源码编译后安装

```
# Target Win64
mkdir build
cd build
cmake .. -G "Visual Studio XX 201X WinXX"
cmake --build . --config Release

```

```
# Target Win32
mkdir build
cd build
cmake ..
cmake --build . --config Release
```

第三步：卸载

+ 方式 1：安装包

双击 start > all program > demo > uninstall 进行 Windows 程序标准卸载。

+ 方式 2：源码安装

直接删除源码文件夹，例如 “demo”。

6.2 Linux

第一步：安装依赖

+ cmake

+ 源码：demo-1.0-source.zip

第二步：安装数据采集器

+ 方式 1：安装包（推荐）

```
sudo dpkg -i demo.1.0_amd64.deb
```

+ 方式 2：源码安装

```
# Target type depends on Linux compiler type
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/path/to/demo ..
make
make install

# Run
cd /home/path/demo
./demo
```

第三步：卸载

+ 方式 1：安装包

```
sudo dpkg -r <package_name>
# alternatively: sudo apt-get remove <package_name>
```

+ 方式 2：源码安装

直接删除源码文件夹，例如 “demo”。

7 参考资料

[1] https://baike.baidu.com/item/跨平台/8558902?fr=aladdin
[2] https://devblogs.microsoft.com/cppblog/using-visual-studio-for-cross-platform-c-development-targeting-windows-and-linux/
[3] https://blog.csdn.net/u011520181/article/details/81702460?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase
[4] https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-gettickcount
[5] https://en.wikipedia.org/wiki/Computer_memory
[6] https://www.techopedia.com/definition/11411/disk-usage-du