TẬP ĐOÀN CÔNG NGHIỆP - VIỄN THÔNG QUÂN ĐỘI
Công ty an ninh mạng Viettel
*****

DEV CORE

CÔNG CỤ THEO DÕI TÀI NGUYÊN TIẾN TRÌNH

Người thực hiện: Nguyễn Trọng Hiếu

Vị trí: Sinh viên thực tập

Mã nhân viên: sv_hieunt

Người hướng dẫn: Nguyễn Ngọc Anh

Phòng: Sản phẩm chống tấn công có chủ đích

Hà Nội, 09/2023

Tóm tắt

Từ khóa: lập trình hệ thống, Linux, Windows, tiến trình, giám sát.

MỤC LỤC

1 Giới thiệu **1**
1.1 Đặt vấn đề 1

2 Cơ sở lí thuyết **3**
2.1 Tệp tin (File) 3
2.1.1 Đọc/ghi vào cuối/xóa file 3
2.2 Logs 3
2.2.1 Cấu trúc thư mục logs 3
2.2.2 Design Pattern cho Logs 4
2.3 Tệp tin cấu hình 4
2.3.1 JSON 4
2.3.2 Registry trên Windows 6
2.3.3 Tệp tin cấu hình (.conf) trên Linux 6
2.4 Daemon 7
2.4.1 Định nghĩa 7
2.4.2 Services 7
2.4.3 Cách gán chương trình để khởi động cùng hệ thống bằng C++ . . 7
2.5 Process 8
2.5.1 Hiển thị thông tin process bằng C++ 8
2.5.2 Giao tiếp liên tiến trình (Interprocess Communication) 10

3 Thiết kế **11**
3.1 Sơ đồ Use Case (Use Case Diagram) 11
3.2 Sơ đồ UML 11

4 Triển khai **12**
4.1 Các vấn đề chung 12
4.1.1 ANSI hay UNICODE 12
4.1.2 Giao tiếp liên tiến trình 12
4.1.3 Lệnh giao tiếp giữa CTA và CTB 12
4.2 CTA 13
4.2.1 CTA nhận cấu hình 13
4.2.2 Hoạt động nhận/gửi logs 13
4.3 CTB 14
4.3.1 Kiểm tra file cấu hình đầu vào 14
4.3.2 Nhận logs từ CTA 14
4.4 Các lỗi gặp phải trong quá trình code 14
4.4.1 Các thư viện include sai thứ tự 14
4.4.2 Không sử dụng std::shared_ptr<>() cho các class cần chia sẻ . . . 14

4.4.3 Các vấn đề với Windows Named pipes 14

5 Công cụ và thư viện **16**

6 Kết luận **17**
6.1 Hạn chế 17
6.2 Cải tiến trong tương lai 17
6.3 Kết luận 17

References **18**

1. Giới thiệu

1.1 Đặt vấn đề

Viết công cụ theo dõi tài nguyên chiếm dụng (cpu, ram, disk, network) của process bất kỳ, và ghi ra log sự kiện process chiếm dụng tài nguyên vượt ngưỡng.

Yêu cầu chức năng:

- Công cụ gồm 1 chương trình service/daemon tự động chạy khi máy tính khởi động (CTA), và 1 chương trình quyền thường chạy thủ công (CTB).
- CTA thực hiện theo dõi tài nguyên chiếm dụng của các process theo cấu hình được chỉ định.
- CTA nhận cấu hình từ CTB, cấu hình có dạng json chứa danh sách process và thông số cần theo dõi: [process name, cpu (%), memory (MB), disk (MB/s), network (KB/s)]. Ví dụ:

```
[
  {
    "process": "chrome.exe",
    "cpu": 10,
    "memory": 200,
    "disk": 1,
    "network": 500
  },
  {
    "process": "devenv.exe",
    "cpu": 50,
    "memory": 100,
    "disk": 10,
    "network": 100
  }
]
```

- CTA lưu lại cấu hình ở dạng registry (trên Windows), hoặc file cấu hình (.conf) (trên Linux). Khi CTB không gửi cấu hình thì CTA sử dụng cấu hình cũ.

- Khi có process vượt ngưỡng thông số được cấu hình, CTA gửi thông tin cho CTB, CTB ghi ra file log mỗi dòng là một sự kiện, gồm các thông tin:
  - Date time
  - Process ID
  - Process name
  - Type (cpu/memory/disk/network)
  - Value
- Trong trường hợp có sự kiện vượt ngưỡng nhưng CTB không chạy, CTA lưu lại sự kiện trên hàng đợi, đến khi CTB chạy sẽ gửi toàn bộ log cho CTB.

Yêu cầu phi chức năng:

- Công cụ có 2 phiên bản, 1 phiên bản chạy trên Windows, 1 phiên bản chạy trên Linux.
- Sử dụng OS native API.
- Trong quá trình hoạt động, công cụ không chiếm quá 5% CPU và 100 MB memory (càng thấp càng tốt).
- Code không có các lỗi lập trình an toàn.
- Sử dụng ngôn ngữ C++, cấu trúc theo hướng đối tượng.
- Code clean, comment đầy đủ.

2. Cơ sở lí thuyết

2.1 Tệp tin (File)

Trong lập trình hệ thống máy tính, thuật ngữ "tệp tin" được sử dụng để chỉ một đơn vị lưu trữ dữ liệu trên hệ thống tệp tin. Một tệp tin (file) là một tập hợp các dữ liệu có cấu trúc hoặc không cấu trúc được tổ chức và lưu trữ trên hệ thống máy tính.

Mỗi tệp tin có một tên duy nhất và thường được đặt trong một thư mục cụ thể. Tệp tin có thể chứa các loại dữ liệu khác nhau như văn bản, mã nguồn, hình ảnh, âm thanh, video, và nhiều loại dữ liệu khác.

Trong lập trình, tệp tin thường được sử dụng để đọc và ghi dữ liệu. Các ngôn ngữ lập trình cung cấp các thư viện và chức năng để làm việc với tệp tin, cho phép đọc và ghi dữ liệu, di chuyển trong tệp tin, xóa và đổi tên tệp tin, và thực hiện các thao tác khác liên quan đến quản lý tệp tin trên hệ thống máy tính.

2.1.1 Đọc/ghi vào cuối/xóa file

1. Windows

   - **Đọc và ghi:** MSDN - Opening a File for Reading or Writing.
   - **Ghi vào cuối file:** Stack Overflow - How to append data on a file in win32
   - **Xóa file:** MSDN - DeleteFile function (winbase.h) và Delete Files with C++ on windows. Cần lưu ý rằng tất cả các handle trỏ đến file phải đóng thì mới có thể xóa file (MSDN - Closing and Deleting Files).

2. Linux

2.2 Logs

Trong lập trình và quản trị hệ thống máy tính, logs là những tập tin ghi lại các thông tin về hoạt động của hệ thống hoặc ứng dụng.

2.2.1 Cấu trúc thư mục logs

- File log được lưu tại cùng địa chỉ với file thực thi (cả CTA và CTB). Stack Overflow - How do I get the directory that a program is running from?. Lưu ý rằng CTA và CTB cần nằm tại thư mục khác nhau để tránh xung đột.
- Các tệp tin trong thư mục được đặt tên có ý nghĩa logs_yy_mm_dd.log trong đó yy, mm, dd lần lượt là là năm, tháng, ngày phát hiện ra sự cố.

- Trong bài tập này, tệp tin log sẽ có nhiều dòng, mỗi dòng là một sự kiện ngăn cách nhau bởi dấu phẩy, gồm các thông tin sau:
  - Thời gian phát hiện sự cố (năm, tháng, ngày, giờ, phút, giây)
  - Process ID
  - Process name
  - Type (cpu/memory/disk/network)
  - Value

2.2.2 Design Pattern cho Logs

Có thể tham khảo: C++ Logger Design

2.3 Tệp tin cấu hình

Tập tin cấu hình là tập tin dùng để lưu trữ các thông số cấu hình, thiết lập ban đầu của phần mềm, hệ điều hành hoặc các dịch vụ.

Một số đặc điểm chính của tập tin cấu hình:

- Chứa các giá trị mặc định hoặc thiết lập ban đầu của phần mềm hoặc dịch vụ.
- Có thể được sửa đổi để thay đổi cách thức hoạt động của phần mềm.
- Có dạng văn bản chứa cặp khóa-giá trị cấu hình cách nhau bởi dấu phẩy, dấu nháy hoặc dấu ngắt hàng.
- Tên file thường mang ý nghĩa liên quan đến phần mềm như apache.conf, mysql.conf.
- Đặt tại thư mục cấu hình chung hoặc thư mục của từng phần mềm.

Trong bài tập này, chúng ta sẽ sử dụng tập tin cấu hình định dạng JSON khi CTA nhận cấu hình từ CTB, lưu cấu hình dưới dạng Registry trên Windows hoặc dưới dạng tập tin cấu hình (.conf) trên Linux.

2.3.1 JSON

1. Khái niệm

   JSON (JavaScript Object Notation) là định dạng dữ liệu phổ biến dùng để truyền thông tin giữa máy tính và các phần mềm ứng dụng khác.

   Đây là một định dạng dữ liệu dạng văn bản dễ đọc và viết bởi người và máy. Nó dễ hiểu hơn so với XML.

   Một số điểm chính của JSON:

- Là dữ liệu có cấu trúc dạng đối tượng, gồm key-value pairs.
- Sử dụng cú pháp giống như kiểu dữ liệu trong JavaScript.
- Có thể convert dễ dàng sang các ngôn ngữ lập trình khác như Python, Java, C#, PHP,...
- Được sử dụng rộng rãi trong Web server-client communication, REST APIs.
- Nhiều phần mềm mã nguồn mở dùng JSON để truyền định dạng config.
- Có nhiều thư viện hỗ trợ parse, tạo JSON dễ dàng.

2. JSON trên Windows C++

   Yêu cầu:

   - Windows SDK.
   - C++ 17 trở lên.
   - Windows 11 Version 22H2 - June 2023 (theo như comment (nội dung: "Windows 11 Version 22H2 - June 2023 Samples Update") trong Github của tài liệu tham khảo Github - MSDN JSON Samples).

   Tài liệu tham khảo xem tại MSDN - Windows.Data.Json Namespace và Github - MSDN JSON Samples (chọn cppwinrt để xem code C++).

   Ngoài ra có thể xem thêm Stack Overflow - Parse JSON with windows.data.json.h in C.

   Code mẫu:

```
#include <winrt/windows.data.json.h>

#define json_char TCHAR

using namespace winrt;
using namespace winrt::Windows::Data::Json;


int main() {

    auto obj = JsonObject(
        JsonObject::Parse(
            LR"({"main":"Hello World"})"
            )
        );
```

```
    return 0;
}
```

3. JSON trên Linux C++

   Có thể tham khảo thư viện C++ sau đây để thao tác với tập tin JSON: nlohmann/json

2.3.2 Registry trên Windows

Registry trong hệ điều hành Windows là một cơ sở dữ liệu trung tâm chứa thông tin về cấu hình, cài đặt và các thiết lập cho hệ thống và các ứng dụng. Nó là một phần quan trọng của cơ chế quản lý cấu hình của Windows.

Registry lưu trữ thông tin dưới dạng "cặp khóa-giá trị" (key-value pairs), với các khóa (keys) là các thư mục và các giá trị (values) là các thông số liên quan. Các thông tin trong Registry có thể bao gồm cấu hình hệ thống, cấu hình phần cứng, cài đặt ứng dụng, các thông tin đăng nhập, và nhiều thông tin quan trọng khác.

Registry có cấu trúc phân cấp, tương tự như cấu trúc thư mục trên hệ thống tệp tin. Các khóa có thể chứa các khóa con và giá trị được gán cho các khóa cụ thể. Các ứng dụng và hệ thống Windows có thể truy cập và sửa đổi Registry để lưu trữ và truy xuất thông tin cấu hình.

Quản lý Registry là một phần quan trọng của việc quản lý và sửa đổi cấu hình hệ thống trong Windows. Người dùng thông thường không nên thay đổi Registry thủ công (bằng tay) mà nên sử dụng các công cụ và giao diện người dùng cung cấp bởi Windows để truy cập và chỉnh sửa.

2.3.3 Tệp tin cấu hình (.conf) trên Linux

Tập tin cấu hình Linux, thường được gọi là "Linux config file," là một tệp tin chứa các cài đặt và cấu hình cho các thành phần và ứng dụng trong hệ thống Linux. Các tệp tin cấu hình này có thể điều chỉnh cách hoạt động của hệ điều hành, các dịch vụ mạng, ứng dụng và nhiều thành phần khác.

Các tệp tin cấu hình Linux thường được viết bằng văn bản thuần túy và thường được lưu trữ trong thư mục **/etc** hoặc các thư mục con của nó. Mỗi thành phần hoặc ứng dụng trong hệ thống có thể có một hoặc nhiều tệp tin cấu hình riêng biệt.

Các tệp tin cấu hình Linux thường được chỉnh sửa bằng một trình chỉnh sửa văn bản như vi hoặc nano. Việc chỉnh sửa các tệp tin cấu hình này đòi hỏi quyền root hoặc quyền quản trị hệ thống để đảm bảo tính toàn vẹn và bảo mật của hệ thống.

Tài liệu tham khảo:

- Configuration Files in Linux | Baeldung on Linux

2.4 Daemon

2.4.1 Định nghĩa

Daemon là một tiến trình chạy trên nền (background process) để thực hiện các tác vụ phục vụ cho hệ thống hoặc cho các ứng dụng khác.

Một số đặc điểm chính của daemon:

- Chạy tuần tự từ lúc khởi động hệ điều hành đến khi tắt máy.
- Không được tương tác trực tiếp với người dùng.
- Thường đảm nhiệm các tác vụ phục vụ quan trọng như vào/ra, quản lý tập tin, dịch vụ mạng, cơ sở dữ liệu,...
- Có thể được cài đặt bởi hệ điều hành hoặc các ứng dụng.
- Chạy không ngừng một cách tự động và cung cấp dịch vụ theo yêu cầu.

2.4.2 Services

1. Windows Service
   - Cách tạo một Windows Service đơn giản: Simple Windows Service in C++.
   - Cách quản lí một Windows Service: sách Windows System Programming (4th Edition), trang 472.
2. Linux Service
   - Cách cài đặt một Service: systemd/Services

2.4.3 Cách gán chương trình để khởi động cùng hệ thống bằng C++

1. Windows:

   Ta sẽ sử dụng Registry để đăng kí chương trình chạy khi khởi động:

   - Sử dụng *RegCreateKeyEx()* để tạo khóa tại **HKEY_CURRENT_USER**, thư mục **Software\\Microsoft\\Windows\\CurrentVersion\\Run**.
   - Đăng kí giá trị mới (tên chương trình) cho khóa bằng *RegSetValueEx()*.
   - Đóng khóa bằng *RegCloseKey()*.

   Tham khảo code chi tiết tại Stack Overflow - Add Application to Startup (Registry).

2. Linux

   Thêm chương trình vào thư mục **$HOME/.config/autostart**.

   Tham khảo tại Stack Overflow - C ++ add applications to Startup on linux programmaticaly from inside program code

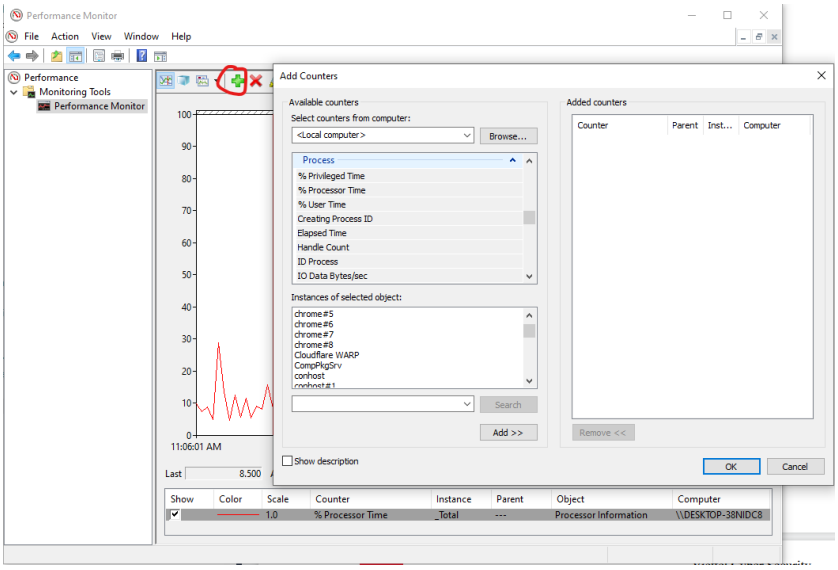
2.5 Process

2.5.1 Hiển thị thông tin process bằng C++

1. Windows

   (a) CPU (%) và Memory chiếm dụng (MB)

   - Sử dụng Windows PDH Process V2 để lấy thông tin về CPU và memory chiếm dụng.
   - Code mẫu sử dụng Windows PDH: MSDN - PDH Examples và Stack Overflow - Counting number of threads on windows server; counter path (cần thêm PDH_FMT_NOCAP100 ở tham số **dwFormat** cho hàm *PdhGetFormattedCounterValue*).
   - Thông tin về Counter Name: MSDN - Retrieving Counter Names and Help Text
   - Cách xem những query khả dụng: Tìm kiếm "Performance Monitor", sau khi mở, chọn "Performance Monitor", sau đó chọn "Add" (Ctrl + N), kéo xuống "Process" ta được ví dụ như ảnh sau:



Hình 1: Performance Monitor trên Windows

Ở ảnh này ta thấy, nếu muốn hiển thị PID của conhost, ta dùng query "\Process(conhost)\IDProcess", nếu muốn hiển thị số handle của Chrome thứ 8, ta dùng "\Process(chrome#8)\Handle Count". Tương tự với Process V2, ta chỉ thay số thứ tự bằng PID là được, ví dụ chrome#3976.

(b) Tốc độ sử dụng tài nguyên đĩa (disk) và mạng (network)

- MSDN - Configuring and Starting the NT Kernel Logger Session: Tạo một provider tên là "NT Kernel Logger" để cung cấp các bản ghi sự kiện. Lưu ý cần **#define INITGUID** ở đầu file code để bật các System GUID.
- MSDN - NT Kernel Logger Constants: Các GUID của NT Kernel Logger. Ở đây ta chú ý đến các GUID của **DiskIO**, **Thread**, **TCPIP**, **UDPIP**.
- MSDN - EVENT_TRACE_PROPERTIES structure (evntrace.h): Cấu trúc quan trọng nhất của provider. Các tham số quan trọng:
  - *Wnode.Flags* = **WNODE_FLAG_TRACED_GUID**
  - *Wnode.Guid* = **SystemTraceControlGuid**
  - *EnableFlags*:
    - **EVENT_TRACE_FLAG_NO_SYSCONFIG**
    - **EVENT_TRACE_FLAG_DISK_IO**
    - **EVENT_TRACE_FLAG_NETWORK_TCPIP**
    - **EVENT_TRACE_FLAG_THREAD**
- MSDN - Retrieving Event Data Using MOF: Code mẫu về xử lí dữ liệu thu thập được từ provider. Vì WMI sắp xếp các tên thuộc tính của các class không theo thứ tự nên trong đây đã có code mẫu xử lí về việc lấy thông tin từng phần tử trong class (tên, vị trí, kích thước phần tử).

2. Linux

(a) CPU (%) và Memory chiếm dụng (MB) Sử dụng trường các trường thông tin trong **/proc/<PID>/status** và **/proc/<PID>/stat**. Tham khảo tại:

- The /proc Filesystem — The Linux Kernel documentation, bảng 1.2.
- Stack Overflow - How to calculate the CPU usage of a process by PID in Linux from C?.

Hoặc sử dụng thư viện **taskstats.h**, các tài liệu:

- The struct taskstats — The Linux Kernel documentation.
- Code mẫu tại Google Source Android - Taskstats example.

Về quản lí bộ nhớ trong Linux, xem Stack Overflow - What is RSS and VSZ in Linux memory management. Thứ ta sẽ cần hiển thị ra cho memory chiếm dụng của process là RSS.

(b) Tốc độ sử dụng tài nguyên đĩa (disk)

Sử dụng trường các trường thông tin trong **/proc/<PID>/io**:

- The /proc Filesystem — The Linux Kernel documentation
- Stack Overflow - Calculating Disk Read/Write in Linux with C++

2.5.2 Giao tiếp liên tiến trình (Interprocess Communication)

1. Windows

   Sử dụng **Named Pipes** để giao tiếp giữa CTA (process daemon tự động chạy khi máy tính khởi động) và CTB (process chạy thủ công).

   Tài liệu và code tham khảo cho Windows Interprocess Communications:

   - MSDN - Pipes (Interprocess Communications).
   - Chương 11 - Interprocess Communication (trang 397), sách Windows System Programming (4th Edition).

2. Linux

   Tương tự Windows, ta sẽ sử dụng **Named Pipes** để giao tiếp giữa CTA và CTB.

   Tài liệu và code tham khảo cho Linux Interprocess Communications:

   - OpenSource - A guide to inter-process communication in Linux (.pdf)
   - OpenSource - Inter-process communication in Linux: Using pipes and message queues

3. Thiết kế

3.1 Sơ đồ Use Case (Use Case Diagram)

3.2 Sơ đồ UML

4. Triển khai

4.1 Các vấn đề chung

4.1.1 ANSI hay UNICODE

Các hàm API của WIN32 sử dụng UNICODE (2-byte). Tuy nhiên, trên thực tế khi làm việc với cả Tiếng Việt có dấu, chỉ cần ANSI (1-byte) là đủ.

Vì vậy ta cần thống nhất việc đọc (file config) và ghi file (logs) sẽ đều phải thực hiện trên bộ kí tự ANSI, còn việc xử lí các dữ liệu đầu vào thế nào thì sẽ tùy hệ điều hành mà code thực thi, cụ thể hơn là biến đổi dữ liệu đầu vào thành **UNICODE cho hàm API Windows** và **ANSI cho Linux**.

4.1.2 Giao tiếp liên tiến trình

Dựa vào yêu cầu của bài, thực tế chúng ta chỉ cần một server được tạo trên CTA và một client trên CTB. Vì vậy có thể tham khảo code mẫu cho:

1. Windows
    - **Server**: MSDN - Multithreaded Pipe Server. Lưu ý tham số **nMaxInstances** trong *CreateNamedPipe()* sẽ có giá trị là **1**.
    - **Client**: MSDN - Named Pipe Client.
2. Linux
    - OpenSource - Inter-process communication in Linux: Using pipes and message queues (Ví dụ số 2, 3).

4.1.3 Lệnh giao tiếp giữa CTA và CTB

Trong quá trình giao tiếp, cần phải sử dụng các lệnh sau để bên còn lại biết được mình cần phải làm gì với dữ liệu được gửi tới:

Tên lệnh	Mô tả
CTB_NOTI_CONFIG	CTB báo với CTA là có config mới.
CTA_SEND_LOGS	CTA tiến hành gửi logs cho CTB.

4.2 CTA

4.2.1 CTA nhận cấu hình

1. Nhận cấu hình

   (a) Nhận cấu hình mới từ CTB

   (b) Sử dụng lại cấu hình cũ

      i. Windows (Registry)
      Đọc file cấu hình từ **Computer/HKEY_LOCAL_MACHINE/SOFTWARE/CtaProcessMonitoring/ProcsesConf**

      ii. Linux (Configuration file)
      Đọc file cấu hình từ **/home/<username>/.CtaProcessMonitoring/ProcessConf**

2. Lưu lại cấu hình

   (a) Windows (Registry)
   Cấu hình sẽ được lưu vào **Computer/HKEY_LOCAL_MACHINE/SOFTWARE/CtaProcessMonitoring/ProcsesConf** trong Registry. Khởi tạo thư mục nếu chưa tồn tại.

   (b) Linux (Configuration file)
   File cấu hình sẽ được lưu vào **/home/<username>/.<app_name>/ProcessConf**. Xem Configuration Files in Linux | Baeldung on Linux, phần 3.1 - Traditional Configs để biết thêm chi tiết.

   Cần lưu ý rằng, trước khi lưu lại cấu hình mới, ta cần xóa sạch cấu hình cũ cũng như tạo mới các thư mục, tệp tin cho cấu hình mới nếu cần thiết.

4.2.2 Hoạt động nhận/gửi logs

1. Lưu lại sự kiện trên hàng đợi nếu CTB chưa hoạt động

   - Hàng đợi: Dữ liệu được lưu vào hàng đợi trong bộ nhớ của Service.

2. Gửi toàn bộ logs cho CTB

   - Khi CTB hoạt động, CTA sẽ gửi toàn bộ log cho CTB và giải phóng bộ nhớ dữ liệu.

4.3 CTB

4.3.1 Kiểm tra file cấu hình đầu vào

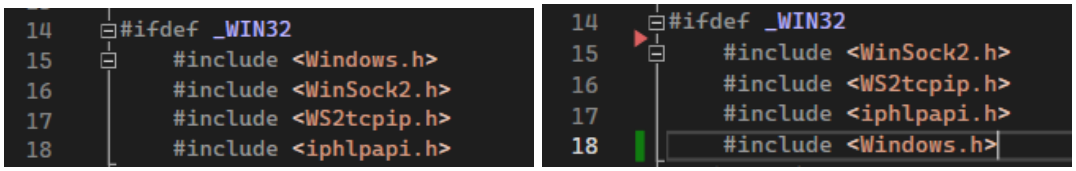
4.3.2 Nhận logs từ CTA

CTB nhận dữ liệu thông qua Pipeline và đẩy dữ liệu vào file log của mình.

4.4 Các lỗi gặp phải trong quá trình code

4.4.1 Các thư viện include sai thứ tự

Một lỗi tương tự giống như Stack Overflow - Can't include Winsock2.h in MSVC 2010.



Hình 2: Lỗi include thư viện (trái) và cách khắc phục (phải)

4.4.2 Không sử dụng std::shared_ptr<>() cho các class cần chia sẻ

Vấn đề xảy ra khi ta muốn chia sẻ class nhưng lại không thể dùng các toán tử '&' thông thường như khi dùng C. *std::shared_ptr<>()* sinh ra để khắc phục việc đó.

Tham khảo:

- Quora - How do you include a class object inside another class (C++, class, compiler errors, populate, development)?
- cppdeveloper - [Smart Pointers] std::shared_ptr trong C++

4.4.3 Các vấn đề với Windows Named pipes

1. Quyền truy cập giữa CTA và CTB.
   - Khi CTA là một Services (chạy với quyền Admin), nếu chỉ khởi tạo named pipes với thông số mặc định thì CTB (chạy với quyền User) sẽ không thể truy cập vào được.
   - Giải pháp là sử dụng hàm *SetSecurityInfo* (SetSecurityInfo function (aclapi.h)) với các tham số:
     - *ObjectType* = **SE_KERNEL_OBJECT** (pipe is kernel object).

  - *SecurityInfo* = **DACL_SECURITY_INFORMATION**
  - *pDacl* = **NULL**
- Theo SetSecurityInfo function (aclapi.h) thì "If the value of the *SecurityInfo* parameter includes the **DACL_SECURITY_INFORMATION** flag and the value of *pDacl* is set to **NULL**, full access to the object is granted to everyone."
- Ngoài ra, để có quyền sửa quyền truy cập của pipe, ta cần thêm giá trị **WRITE_DAC** vào tham số *dwOpenMode* của CreateNamedPipeA function (winbase.h) (theo như yêu cầu của **DACL_SECURITY_INFORMATION** trong SECURITY_INFORMATION).

2. Đọc ghi đồng bộ/bất đồng bộ trong Named Pipes.

   - Trong quá trình code, một lỗi xảy ra khi cứ sau một lúc thì cả server lẫn client đều rơi vào trạng thái WAIT rất lâu. Lí do là trong BLOCKING mode, mặc định pipe sẽ đọc rồi ghi lần lượt (xảy ra sâu trong hệ thống nên dù user có tạo 2 thread riêng biệt thì vẫn bị ảnh hưởng). Với việc đọc ghi lần lượt này, nếu thread đọc của server thực hiện việc đọc mà bên client không gửi gì nữa thì thread ghi của server cũng sẽ rơi vào trạng thái BLOCKING dẫn đến việc đọc ghi của server bị treo hoàn toàn.
   - Giải pháp: thêm giá trị **FILE_FLAG_OVERLAPPED** vào tham số *dwOpenMode* của CreateNamedPipeA function (winbase.h).
   - Giải pháp có được khi ta đọc mô tả của giá trị **FILE_FLAG_OVERLAPPED** trong tham số *dwOpenMode*: "This mode enables the thread that started the operation to perform other operations while the time-consuming operation executes in the background. For example, in overlapped mode, a thread can handle simultaneous input and output (I/O) operations on multiple instances of a pipe or **perform simultaneous (cùng lúc) read and write operations on the same pipe handle**. If overlapped mode is not enabled, functions performing read, write, and connect operations on the pipe handle do not return until the operation is finished. "

5. Công cụ và thư viện

6. Kết luận

6.1 Hạn chế

Chương trình vẫn còn nhiều hạn chế do kiến thức của thực tập sinh (người viết) còn hạn chế. Các hạn chế của chương trình có thể kể đến như:

- Với một cái tên process, mới chỉ giám sát được duy nhất 1 PID trong khi có thể có nhiều process có tên giống nhau.
- Chưa thể thống kê được tài nguyên mạng sử dụng của linux.

6.2 Cải tiến trong tương lai

Với những hạn chế trên của chương trình, người viết xin đề xuất các cách cải thiện sau cho từng vấn đề:

- Sử dụng vector để chứa PID của các process có cùng tên, cải tiến các hàm tìm kiếm, quản lí, thống kê.
- Học thêm về mạng máy tính.

6.3 Kết luận

Tham chiếu