Process Monitor Documentation

System Monitoring Application for Windows

Generated on August 7, 2025

Version 1.0

1 Overview

Process Monitor is a Windows-based system monitoring application designed to track and display real-time CPU and memory usage of running processes. Built using the Windows API and C++, it provides a graphical interface to view process details, historical usage data, and system-wide resource consumption. The application also supports configurable alerts for high CPU or memory usage and saves historical data to a file.

2 Features

- **Real-Time Process Monitoring**: Displays a list of running processes with their PID, CPU usage (%), and memory usage (MB).
- **Historical Data Tracking**: Maintains a 60-second history of CPU and memory usage for each process and displays average usage.
- **System-Wide Metrics**: Shows total CPU and memory usage across all processes.
- **Configurable Alerts**: Allows users to set a CPU usage threshold (default: 80%) for alerts. Memory alerts are triggered when total memory usage exceeds 80% of system memory.

- **Data Persistence**: Saves process history to a file ($process_history.txt) for later analysis. **Responsive UI** : *Adjusts layout dynamically when the window is resized.*

3 System Requirements

- **Operating System**: Windows (tested on Windows 10/11)
- **Libraries**: Requires user32.lib, comctl32.lib, and psapi.lib (linked via #pragma comment)
- **Compiler**: C++ compiler compatible with Windows API (e.g., MSVC)
- **Dependencies**: Common Controls library (initialized via InitCommonControlsEx)

4 Installation

1. **Clone or Download the Source Code**: Obtain the source code (e.g., ProcessMonitor.cpp).
2. **Compile the Code**:
   - Use a C++ compiler like MSVC in a Windows environment.
   - Ensure the required libraries (user32.lib, comctl32.lib, psapi.lib) are available.
   - Build the project to generate the executable.
3. **Run the Executable**: Execute the compiled .exe file to launch the application.

5 Usage

1. **Launch the Application**:
   - Run the executable. A window titled "Process Monitor" will appear.
2. **View Process Information**:
   - The top list view displays current processes with columns for Process Name, PID, CPU Usage (%), and Memory Usage (MB).
   - The bottom list view shows average CPU and memory usage over the last 60 seconds for each process.
3. **Refresh Data**:

- Click the "Refresh" button to update the process list and save historical data to process$_history.txt$.

4. **Set CPU Alert Threshold**:
   - Enter a value in the "CPU Alert Threshold (%)" text box (default: 80.0).
   - If a process's CPU usage or total memory usage exceeds the threshold, an alert dialog will appear.
5. **Monitor System Usage**:
   - Total CPU and memory usage are displayed at the bottom of the window.
6. **Review Historical Data**:

- Open process$_history.txtintheapplicationdirectorytoviewsavedprocessdata, includingtimestampsandusa$

6 File Output

- **File**: process$_history.txt$**Content** : $Containstimestampedentrieswithtotal CPU/memoryusageandper-processCPU/memoryhistory.$

- **Location**: Saved in the same directory as the executable.
- **Format**:

```
Timestamp: [Date and Time]
Total CPU Usage: [Percentage]%
Total Memory Usage: [MB] MB
Process: [Name] (PID: [PID])
CPU History: [Values]
Memory History (MB): [Values]
------------------------
```

7 Notes

- The application updates process data only when the "Refresh" button is clicked to avoid excessive CPU usage.
- Memory alerts are based on 80% of total system memory, calculated at startup.
- The UI is resizable, and controls adjust automatically to fit the window size.

8 Troubleshooting

- **No Processes Displayed**: Ensure the application has sufficient permissions to access process information.
- **Alert Dialogs Not Appearing**: Verify that the CPU threshold is set to a reasonable value and that processes are exceeding it.

- **File Not Saving**: Check write permissions in the application directory for process$_h$*istory.txt*.