

Forum:[Hack Forums](#)

Thread: [\[C++\] Ways to get list of processes running](#)

ByteDoktor - February 26, 2025, 12:24 pm

Now there are many ways to find the list of active processes on a system in C++, i will only cover a few of them, so if you're aware of any others that i have missed, please don't hesitate to reply with them.

First Method (Basic) (Windows Only)

```
Code #include <cstdlib>
#include <iostream>
using namespace std;

int main()
{
    // Print the objective of program
    cout << "Running tasklist command to retrieve "
         "list of running processes:"
         << endl;

    // Execute the tasklist command and display the output
    int returnValue = system("tasklist");

    if (returnValue != 0) {
        cerr << "Error executing tasklist command" << endl;
        return 1;
    }
    return 0;
}
```

Second Method (EnumProcess & OpenProcess)

The second is to simply use EnumProcesses functions that returns the PIDs and OpenProcess function.

```
Code #include <iostream>
#include "Windows.h"
#include "psapi.h"

BOOL PrintProcesses() {
    DWORD adwProcesses[1024 * 2], dwReturnLen1 = NULL, dwReturnLen2 = NULL,
        dwNmbrOfPids = NULL;
    HANDLE hProcess = NULL;
    HMODULE hModule = NULL;
    WCHAR szProc[MAX_PATH];

    if (!EnumProcesses(adwProcesses, sizeof(adwProcesses), &dwReturnLen1)) {
```

```

    printf("[!] EnumProcesses Failed With Error : %d \n", GetLastError());
    return FALSE;
}

dwNmbrOfPids = dwReturnLen1 / sizeof(DWORD);
printf("[i] Number Of Processes Detected : %d \n", dwNmbrOfPids);
for (int i = 0; i < dwNmbrOfPids; i++) {

    if (adwProcesses[i] != NULL) {

        if ((hProcess =
            OpenProcess(PROCESS_QUERY_INFORMATION | PROCESS_VM_READ,
                FALSE, adwProcesses[i])) != NULL) {

            if (!EnumProcessModules(hProcess, &hModule,
                sizeof(HMODULE),
                &dwReturnLen2)) {
                printf("[!] EnumProcessModules Failed[At Pid : % d] With Error: % d \n ",
adwProcesses[i], GetLastError());
            }
            else {
                if (!GetModuleBaseName(
                    hProcess, hModule,
                    szProc,
                    sizeof(szProc) /
                    sizeof(WCHAR))) {
                    printf("[!] GetModuleBaseName Failed[At Pid : % d] With Error: % d \n
",adwProcesses[i], GetLastError());
                }
                else {
                    wprintf(L "[%0.3d] Process \"%s\" - Of Pid : %d \n", i, szProc, adwProcesses[i]);
                }
            }

            CloseHandle(hProcess);
        }
    }
}

}

int main()
{
    PrintProcesses();
}

```

Source:

<https://learn.microsoft.com/en-us/windows/win32/api/psapi/nf-psapi-enumprocesses>

Method 3 (CreateToolhelp32Snapshot)

Using the CreateToolhelp32Snapshot method for process enumeration, a snapshot is taken, and a string comparison is used to check if a process name matches the target. However, this approach has a limitation: when multiple instances of the same process run at different privilege levels, the string comparison cannot distinguish between them.

Source:

<https://learn.microsoft.com/en-us/windows/win32/api/tlhelp32/nf-tlhelp32-createtoolhelp32snapshot>

```
Code #include "Windows.h"
#include <tlhelp32.h>
#include <ostream>
#include <iostream>
```

```
void PrintProcessList() {
    HANDLE hProcessSnap;
    PROCESSENTRY32 pe32;

    hProcessSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    if (hProcessSnap == INVALID_HANDLE_VALUE) {
        return;
    }

    pe32.dwSize = sizeof(PROCESSENTRY32);

    if (Process32First(hProcessSnap, &pe32)) {
        do {
            std::wcout << L"Process Name: " << pe32.szExeFile << std::endl;
        } while (Process32Next(hProcessSnap, &pe32));
    }

    CloseHandle(hProcessSnap);
}

int main() {
    PrintProcessList();
}
```

Thanks for reading my thread.