<https://blog.csdn.net/yhd619/article/details/37988935>

 最近在学习用C++获取WINDOWS和LINUX  下的系统信息，对其大概方法做了一些总结，希望对新人有一些帮助，话不多说，先来代码大笑大笑。。。。。。。。。。。。。。。。。。。

      首先，我针对两个平台定义了一个接口类，即纯虚函数类，其windows 和 linux 的实现类可以实现该类，接口代码如下：

**[cpp]** [view plain](https://blog.csdn.net/yhd619/article/details/37988935) [copy](https://blog.csdn.net/yhd619/article/details/37988935)

1. #pragma once
2. #pragma comment(lib,"Kernel32.lib")
3. #pragma comment(lib,"Psapi.lib")
4. #include <Windows.h>
5. #include <TlHelp32.h>
6. #include<direct.h>
7. #include<winternl.h>
8. #include <Psapi.h>
9. #include <QMap>
10. //获取系统信息的接口
11. **class** ISysInfo
12. {
13. **public**:
14. **virtual** ~ISysInfo(){}
15. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
16. /\*函数说明：获取系统CPU当前使用率
17. /\*函数参数：nCpuRate->CPU使用率
18. /\*返回结果：返回结果状态
19. /\*作者时间：Bevis 2014-07-19
20. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
21. **virtual** **bool** GetSysCpu(**int**& nCpuRate) = 0;
22. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
23. /\*函数说明：获取系统内存信息
24. /\*函数参数：nMemTotal->内存总共大小,(MB)；nMemUsed->内存已使用大小,(MB)
25. /\*返回结果：返回结果状态
26. /\*作者时间：Bevis 2014-07-19
27. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
28. **virtual** **bool** GetSysMemory(**int**& nMemTotal,**int**& nMemUsed) = 0;
29. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
30. /\*函数说明：获取系统磁盘信息
31. /\*函数参数：nDiskTotal->磁盘总共大小,(MB)；nDiskUsed->磁盘已使用大小,(MB)
32. /\*返回结果：返回结果状态
33. /\*作者时间：Bevis 2014-07-19
34. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
35. **virtual** **bool** GetSysDisk(**int**& nDiskTotal,**int**& nDiskUsed) = 0;
36. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
37. /\*函数说明：遍历系统所有进程
38. /\*函数参数：无
39. /\*返回结果：进程PID和进程名称的映射集合
40. /\*作者时间：Bevis 2014-07-19
41. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
42. **virtual** QMap<**int**,QString> GetAllProcess() = 0;
43. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
44. /\*函数说明：获取单个进程的内存使用率
45. /\*函数参数：nPid->进程唯一标识号；nProcessMemRate->进程的内存使用率
46. /\*返回结果：返回结果状态
47. /\*作者时间：Bevis 2014-07-19
48. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
49. **virtual** **bool** GetProcessMemory(**int** nPid,**int**& nProcessMemRate) = 0;
50. };

由于时间有限，暂时只实现了WINDOWS平台，后续我会补上LINUX的实现....微笑微笑微笑

windows的实现类头文件定义如下：

**[cpp]** [view plain](https://blog.csdn.net/yhd619/article/details/37988935) [copy](https://blog.csdn.net/yhd619/article/details/37988935)

1. #pragma once
2. #include <QDebug>
3. #include <QString>
4. #include "ISysInfo.h"
5. #include <atlconv.h>
6. #include <string.h>
8. //在WINDOWS平台上实现
9. **class** ISysWin: **public** ISysInfo
10. {
11. **public**:
12. ISysWin(**void**);
13. **virtual** ~ISysWin(**void**);
15. **bool** GetSysCpu(**int**& nCpuRate) ;
17. **bool** GetSysMemory(**int**& nMemTotal,**int**& nMemUsed);
19. **bool** GetSysDisk(**int**& nDiskTotal,**int**& nDiskUsed) ;
21. QMap<**int**,QString> GetAllProcess() ;
23. **bool** GetProcessMemory(**int** nPid,**int**& nProcessMemRate) ;
25. **protected**:
26. //时间转换
27. **\_\_int64** Filetime2Int64(**const** FILETIME\* ftime);
28. //两个时间相减运算
29. **\_\_int64** CompareFileTime(FILETIME preTime,FILETIME nowTime);
30. };

windows的实现类cpp文件定义如下：

**[cpp]** [view plain](https://blog.csdn.net/yhd619/article/details/37988935) [copy](https://blog.csdn.net/yhd619/article/details/37988935)

1. #include "ISysWin.h"

4. ISysWin::ISysWin(**void**)
5. {
6. }

9. ISysWin::~ISysWin(**void**)
10. {
11. }
13. **\_\_int64** ISysWin::Filetime2Int64(**const** FILETIME\* ftime)
14. {
15. LARGE\_INTEGER li;
16. li.LowPart = ftime->dwLowDateTime;
17. li.HighPart = ftime->dwHighDateTime;
18. **return** li.QuadPart;
19. }
21. **\_\_int64** ISysWin::CompareFileTime(FILETIME preTime,FILETIME nowTime)
22. {
23. **return** **this**->Filetime2Int64(&nowTime) - **this**->Filetime2Int64(&preTime);
24. }
26. //将单字节char\*转化为宽字节wchar\_t\*
27. **wchar\_t**\* AnsiToUnicode( **const** **char**\* szStr )
28. {
29. **int** nLen = MultiByteToWideChar( CP\_ACP, MB\_PRECOMPOSED, szStr, -1, NULL, 0 );
30. **if** (nLen == 0)
31. {
32. **return** NULL;
33. }
34. **wchar\_t**\* pResult = **new** **wchar\_t**[nLen];
35. MultiByteToWideChar( CP\_ACP, MB\_PRECOMPOSED, szStr, -1, pResult, nLen );
36. **return** pResult;
37. }
39. //将宽字节wchar\_t\*转化为单字节char\*
40. **inline** **char**\* UnicodeToAnsi( **const** **wchar\_t**\* szStr )
41. {
42. **int** nLen = WideCharToMultiByte( CP\_ACP, 0, szStr, -1, NULL, 0, NULL, NULL );
43. **if** (nLen == 0)
44. {
45. **return** NULL;
46. }
47. **char**\* pResult = **new** **char**[nLen];
48. WideCharToMultiByte( CP\_ACP, 0, szStr, -1, pResult, nLen, NULL, NULL );
49. **return** pResult;
50. }
52. **bool** ISysWin::GetSysCpu(**int**& nCpuRate)
53. {
54. **HANDLE** hEvent;
55. **bool** res;
56. **static** FILETIME preIdleTime;
57. **static** FILETIME preKernelTime;
58. **static** FILETIME preUserTime;
60. FILETIME idleTime;
61. FILETIME kernelTime;
62. FILETIME userTime;
64. res = GetSystemTimes(&idleTime,&kernelTime,&userTime);
66. preIdleTime = idleTime;
67. preKernelTime = kernelTime;
68. preUserTime = userTime;
70. hEvent = CreateEvent(NULL,FALSE,FALSE,NULL);//初始值为nonsignaled
72. WaitForSingleObject(hEvent,500);//等待500毫秒
74. res = GetSystemTimes(&idleTime,&kernelTime,&userTime);
76. **int** idle = CompareFileTime(preIdleTime,idleTime);
77. **int** kernel = CompareFileTime(preKernelTime,kernelTime);
78. **int** user = CompareFileTime(preUserTime,userTime);
80. nCpuRate =(**int**)ceil( 100.0\*( kernel + user - idle ) / ( kernel + user ) );
82. **return** res;
83. }
85. **bool** ISysWin::GetSysMemory(**int**& nMemTotal,**int**& nMemUsed)
86. {
87. MEMORYSTATUSEX memsStat;
88. memsStat.dwLength = **sizeof**(memsStat);
89. **if**(!GlobalMemoryStatusEx(&memsStat))//如果获取系统内存信息不成功，就直接返回
90. {
91. nMemTotal = -1;
92. nMemUsed  = -1;
93. **return** **false**;
94. }
95. **int** nMemFree = memsStat.ullAvailPhys/( 1024.0\*1024.0 );
96. nMemTotal = memsStat.ullTotalPhys/( 1024.0\*1024.0 );
97. nMemUsed= nMemTotal- nMemFree;
98. **return** **true**;
99. }
101. **bool** ISysWin::GetSysDisk(**int**& nDiskTotal,**int**& nDiskUsed)
102. {
103. **static** **char** path[\_MAX\_PATH];//存储当前系统存在的盘符
104. **int** curdrive = \_getdrive();
105. unsigned **long** lFreeAll = 0UL;
106. unsigned **long** lTotalAll = 0UL;
107. **for**(**int** drive = 1; drive <= 26; drive++ )//遍历所有盘符
108. {
109. **if**( !\_chdrive( drive ) )
110. {
111. sprintf(path, "%c:\\", drive + 'A' - 1 );
112. ULARGE\_INTEGER caller, total, free;
113. **WCHAR** wszClassName[\_MAX\_PATH];
114. memset(wszClassName,0,**sizeof**(wszClassName));
115. MultiByteToWideChar(CP\_ACP,0,path,strlen(path)+1,wszClassName,
116. **sizeof**(wszClassName)/**sizeof**(wszClassName[0]));
117. **if** (GetDiskFreeSpaceEx(wszClassName, &caller, &total, &free) == 0)
118. {
119. qDebug()<<"GetDiskFreeSpaceEx Filed!";
120. **return** **false**;
121. }
123. **double** dTepFree = free.QuadPart/( 1024.0\*1024.0 );
124. **double** dTepTotal = total.QuadPart/( 1024.0\*1024.0 );
125. //qDebug()<<"Get Windows Disk Information:"<<path<<"--free:"<<dTepFree<<"--total:"<<dTepTotal;
126. lFreeAll += (unsigned **long**)ceil(dTepFree);
127. lTotalAll += (unsigned **long**)ceil(dTepTotal);
128. }
129. }
130. nDiskUsed = lFreeAll;
131. nDiskTotal= lTotalAll;
132. **return** **true**;
133. }
135. QMap<**int**,QString> ISysWin::GetAllProcess()
136. {
137. **HANDLE** hProcessSnap;    //进程快照的句柄
138. **HANDLE** hProcess;    //用于获取进程的句柄
139. PROCESSENTRY32 pe32;//进程信息的结构体
140. QMap<**int**,QString> pidMap;//保存所有进程的PID
141. // 获取系统进程信息的快照
142. hProcessSnap = CreateToolhelp32Snapshot( TH32CS\_SNAPPROCESS, 0 );
143. **if**( hProcessSnap == INVALID\_HANDLE\_VALUE )
144. {
145. qDebug()<<"CreateToolhelp32Snapshot Failed!";// 打印错误信息
146. **if**(NULL !=hProcessSnap)
147. {
148. CloseHandle( hProcessSnap );          // 关闭进程快照信息
149. hProcessSnap = NULL;
150. }
151. **return** pidMap;
152. }
153. // 在使用之前设置PROCESSENTRY32结构体的初始大小值,如果不初始化dwSize, Process32First 调用会失败.
154. pe32.dwSize = **sizeof**( PROCESSENTRY32 );
155. **if**( !Process32First( hProcessSnap, &pe32 ) )// 开始获取第一个进程的信息，如果获取失败就返回
156. {
157. qDebug()<<"Process32First Failed!"; // 打印错误信息
158. **if**(NULL !=hProcessSnap)
159. {
160. CloseHandle( hProcessSnap );          // 关闭进程快照信息
161. hProcessSnap = NULL;
162. }
163. **return** pidMap;
164. }
165. //开始遍历所有进程
166. pidMap.clear();
167. **do**
168. {
169. //加入PID
170. **if**(!pidMap.contains((**int**)pe32.th32ProcessID))
171. {
172. QString strProcessName =UnicodeToAnsi(pe32.szExeFile);
173. pidMap.insert((**int**)pe32.th32ProcessID,strProcessName);
174. }
176. //当然还可以获取到很多其他信息，例如进程名字(szExeFile[MAX\_PATH])、父进程PPID(th32ParentProcessID)。。。
177. /\* 附上该结构体信息
178. typedef struct tagPROCESSENTRY32 {
179. DWORD     dwSize;
180. DWORD     cntUsage;
181. DWORD     th32ProcessID;
182. ULONG\_PTR th32DefaultHeapID;
183. DWORD     th32ModuleID;
184. DWORD     cntThreads;
185. DWORD     th32ParentProcessID;
186. LONG      pcPriClassBase;
187. DWORD     dwFlags;
188. TCHAR     szExeFile[MAX\_PATH];
189. } PROCESSENTRY32, \*PPROCESSENTRY32;
190. \*/
191. }
192. **while**( Process32Next( hProcessSnap, &pe32 ) );// 获取下一个进程的信息
193. **if**(NULL !=hProcessSnap)//最后关闭快照句柄
194. {
195. CloseHandle( hProcessSnap );
196. hProcessSnap = NULL;
197. }
198. **return** pidMap;
199. }
201. **bool** ISysWin::GetProcessMemory(**int** nPid,**int**& nProcessMemRate)
202. {
203. **HANDLE** hProcess;//该线程的句柄
204. PROCESS\_MEMORY\_COUNTERS pmc;//该线程的内存信息结构体
205. hProcess = OpenProcess( PROCESS\_ALL\_ACCESS ,FALSE, nPid );//利用最大权限打开该线程并获得句柄
206. **if** (NULL == hProcess)
207. {
208. qDebug()<<"OpenProcess Failed!";
209. **return** **false**;
210. }
211. **if** ( !GetProcessMemoryInfo( hProcess, &pmc, **sizeof**(pmc)) )
212. {
213. qDebug()<<"GetProcessMemoryInfo Failed!";
214. **return** **false**;
215. }
217. **int** nMemTotal = 0;
218. **int** nMemUsed = 0;
219. **this**->GetSysMemory(nMemTotal,nMemUsed);
220. **double** tep = ((**int**)pmc.WorkingSetSize\*1.0);
221. nProcessMemRate = (**int**)ceil(100\*(tep/(1024.0\*1024.0)) / nMemTotal) ;
223. CloseHandle(hProcess);
225. **return** **true**;
226. }

最后当然是我的测试类了，

**[cpp]** [view plain](https://blog.csdn.net/yhd619/article/details/37988935) [copy](https://blog.csdn.net/yhd619/article/details/37988935)

1. #include <QtCore/QCoreApplication>
2. #include "ISysInfo.h"
3. #include "ISysWin.h"
4. #include <Windows.h>
5. **int** main(**int** argc, **char** \*argv[])
6. {
7. QCoreApplication a(argc, argv);
8. ISysInfo\* pISys = **new** ISysWin();//调用WINDOWS实现的子类
9. **while** (1)
10. {
11. **int** nCpuRate = -1;
12. **int** nMemTotal = -1;
13. **int** nMemUsed = -1;
14. **int** nDiskTotal = -1;
15. **int** nDiskUsed = -1;
16. **int** nProcessMemRate = -1;
17. QMap<**int**,QString> pidMap;
19. pISys->GetSysCpu(nCpuRate);
20. qDebug()<<"CPU Rate:"<<nCpuRate<<"%";
22. pISys->GetSysMemory(nMemTotal,nMemUsed);
23. qDebug()<<"Mem Total:"<<nMemTotal<<"\t Mem Used:"<<nMemUsed;
25. pISys->GetSysDisk(nDiskTotal,nDiskUsed);
26. qDebug()<<"Disk Total:"<<(nDiskTotal/1024.0)<<"GB \t Disk Used:"<<(nDiskUsed/1024.0)<<"GB";
28. pISys->GetProcessMemory(6472,nProcessMemRate);
29. qDebug()<<"PID:6472;\t Mem Rate:"<<nProcessMemRate<<"%";
31. pidMap = pISys->GetAllProcess();
32. qDebug()<<"Process Number:"<<pidMap.size();
34. qDebug()<<"-------------------------------------------------------------------";
35. Sleep(1000);
36. }
37. **delete** pISys;
38. **return** a.exec();
39. }<span style="color:#ff0000;">
40. </span>

后期我会继续实现linux下的代码，请继续关注！！！