# 学习大纲

|  |
| --- |
|  |

# API

## CreateToolelp32Snapshot函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 获取指定进程以及这些进程使用的堆、模块和线程的快照。 语法 C++复制  HANDLE CreateToolhelp32Snapshot(  [in] DWORD dwFlags,  [in] DWORD th32ProcessID  ); 参数 [in] dwFlags  要包含在快照中的系统部分。 此参数可使用以下一个或多个值。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **TH32CS\_INHERIT**  0x80000000 | 指示快照句柄是可继承的。 | | **TH32CS\_SNAPALL** | 包括系统中的所有进程和线程，以及 *th32ProcessID* 中指定的进程的堆和模块。 等效于指定使用 OR 操作 (“|”组合的**TH32CS\_SNAPHEAPLIST、TH32CS\_SNAPMODULE**、**TH32CS\_SNAPPROCESS和TH32CS\_SNAPTHREAD**值) 。 | | **TH32CS\_SNAPHEAPLIST**  0x00000001 | 包括快照 *th32ProcessID* 中指定的进程的所有堆。 若要枚举堆，请参阅 [Heap32ListFirst](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-heap32listfirst)。 | | **TH32CS\_SNAPMODULE**  0x00000008 | 包括快照 *th32ProcessID* 中指定的进程的所有模块。 若要枚举模块，请参阅 [Module32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-module32first)。 如果函数失败并 **出现ERROR\_BAD\_LENGTH**，请重试该函数，直到成功。  **64 位 Windows：**在 32 位进程中使用此标志包括 *th32ProcessID 中指定的进程的 32* 位模块，而在 64 位进程中使用它包括 64 位模块。 若要从 64 位进程包括 *th32ProcessID 中指定的进程的 32* 位模块，请使用 **TH32CS\_SNAPMODULE32** 标志。 | | **TH32CS\_SNAPMODULE32**  0x00000010 | 从 64 位进程调用时，包括快照中 *th32ProcessID* 中指定的进程的所有 32 位模块。 此标志可以与 **TH32CS\_SNAPMODULE** 或 **TH32CS\_SNAPALL**结合使用。 如果函数失败并 **出现ERROR\_BAD\_LENGTH**，请重试该函数，直到成功。 | | **TH32CS\_SNAPPROCESS**  0x00000002 | 包括系统中快照中的所有进程。 若要枚举进程，请参阅 [Process32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32first)。 | | **TH32CS\_SNAPTHREAD**  0x00000004 | 包括快照系统中的所有线程。 若要枚举线程，请参阅 [Thread32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-thread32first)。  若要标识属于特定进程的线程，请在枚举线程时将其进程标识符与 [THREADENTRY32](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/ns-tlhelp32-threadentry32) 结构的 **th32OwnerProcessID** 成员进行比较。 |   [in] th32ProcessID  要包含在快照中的进程的进程标识符。 此参数可以为零以指示当前进程。 指定 **TH32CS\_SNAPHEAPLIST**、 **TH32CS\_SNAPMODULE**、 **TH32CS\_SNAPMODULE32**或 **TH32CS\_SNAPALL** 值时，使用此参数。 否则，将忽略它，并且所有进程都包含在快照中。  如果指定的进程是空闲进程或 CSRSS 进程之一，则此函数将失败，并且最后一个错误代码 **ERROR\_ACCESS\_DENIED** ，因为它们的访问限制阻止用户级代码打开它们。  如果指定的进程是 64 位进程，调用方是 32 位进程，则此函数将失败，最后一个错误代码 **ERROR\_PARTIAL\_COPY** (299) 。 返回值 如果函数成功，它将返回指定快照的打开句柄。  如果函数失败，它将返回 **INVALID\_HANDLE\_VALUE**。 要获得更多的错误信息，请调用 GetLastError。 可能的错误代码包括 **ERROR\_BAD\_LENGTH**。 注解 此函数采用的快照由其他工具帮助函数检查，以提供其结果。 对快照的访问是只读的。 快照句柄充当对象句柄，并受制于有关其有效进程和线程的相同规则。  若要枚举所有进程的堆或模块状态， **请指定TH32CS\_SNAPALL** 并将 *th32ProcessID* 设置为零。 然后，对于快照中的每个附加进程，再次调用 **CreateToolhelp32Snapshot**，并指定其进程标识符以及**TH32CS\_SNAPHEAPLIST**或**TH32\_SNAPMODULE**值。  为当前进程以外的进程创建包含堆和模块的快照时， **CreateToolhelp32Snapshot** 函数可能会因各种原因而失败或返回不正确的信息。 例如，如果目标进程中的加载程序数据表已损坏或未初始化，或者模块列表在函数调用期间由于加载或卸载 DLL 而发生更改，则函数可能会失败 **并出现ERROR\_BAD\_LENGTH** 或其他错误代码。 确保目标进程未在挂起状态下启动，然后再次尝试调用函数。 如果使用 **TH32CS\_SNAPMODULE或** TH32CS\_SNAPMODULE32 调用函数时ERROR\_BAD\_LENGTH **失败，请**再次调用函数，直到成功。  **TH32CS\_SNAPMODULE**和**TH32CS\_SNAPMODULE32**标志不会检索使用**LOAD\_LIBRARY\_AS\_DATAFILE**或类似标志加载的模块的句柄。 有关详细信息，请参阅 [LoadLibraryEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/libloaderapi/nf-libloaderapi-loadlibraryexa)。  若要销毁快照，请使用 [CloseHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/handleapi/nf-handleapi-closehandle) 函数。  请注意，可以使用 [QueryFullProcessImageName](https://learn.microsoft.com/zh-cn/windows/desktop/api/winbase/nf-winbase-queryfullprocessimagenamea) 函数从 32 位进程中检索 32 位和 64 位进程的可执行映像的全名。 示例 有关示例，请参阅 [拍摄快照和查看进程](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/taking-a-snapshot-and-viewing-processes)。 要求  |  |  | | --- | --- | | **标头** | tlhelp32.h | | **Library** | Kernel32.lib | | **DLL** | Kernel32.dll |  另请参阅 [CloseHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/handleapi/nf-handleapi-closehandle)  [Heap32ListFirst](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-heap32listfirst)  [Module32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-module32first)  [Process32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32first)  [系统的快照](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/snapshots-of-the-system)  [Thread32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-thread32first)  [工具帮助函数](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/tool-help-functions) |

## 这个函数可以创建堆的快照，进程的快照、线程的快照

## Process32First函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 检索有关系统快照中遇到的第一个进程的信息。 语法 C++复制  BOOL Process32First(  [in] HANDLE hSnapshot,  [in, out] LPPROCESSENTRY32 lppe  ); 参数 [in] hSnapshot  快照的句柄，该句柄是从上次调用 [CreateToolhelp32Snapshot](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-createtoolhelp32snapshot) 函数返回的。  [in, out] lppe  指向 [PROCESSENTRY32](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/ns-tlhelp32-processentry32) 结构的指针。 它包含进程信息，例如可执行文件的名称、进程标识符和父进程的进程标识符。 返回值 如果进程列表的第一个条目已复制到缓冲区，则返回 **TRUE** ，否则返回 **FALSE** 。 如果不存在任何进程或**快照**不包含进程信息，[则 GetLastError](https://learn.microsoft.com/zh-cn/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) 函数将返回ERROR\_NO\_MORE\_FILES错误值。 注解 调用应用程序必须将 [PROCESSENTRY32](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/ns-tlhelp32-processentry32) 的 **dwSize** 成员设置为 结构的大小（以字节为单位）。  若要检索同一快照中记录的其他进程的相关信息，请使用 [Process32Next](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32next) 函数。 示例 有关示例，请参阅 [拍摄快照和查看进程](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/taking-a-snapshot-and-viewing-processes)。 要求  |  |  | | --- | --- | | **标头** | tlhelp32.h | | **Library** | Kernel32.lib | | **DLL** | Kernel32.dll |  另请参阅 [CreateToolhelp32Snapshot](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-createtoolhelp32snapshot)  [PROCESSENTRY32](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/ns-tlhelp32-processentry32)  [流程步行](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/process-walking)  [Process32Next](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32next)  [工具帮助函数](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/tool-help-functions) |

## Process32Next函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 检索有关系统快照中记录的下一个进程的信息。 语法 C++复制  BOOL Process32Next(  [in] HANDLE hSnapshot,  [out] LPPROCESSENTRY32 lppe  ); 参数 [in] hSnapshot  从对 [CreateToolhelp32Snapshot](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-createtoolhelp32snapshot) 函数的上一次调用返回的快照句柄。  [out] lppe  指向 [PROCESSENTRY32](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/ns-tlhelp32-processentry32) 结构的指针。 返回值 如果进程列表的下一个条目已复制到缓冲区，则返回 **TRUE** ;否则返回 **FALSE** 。 如果不存在任何进程或**快照**不包含进程信息，[则 GetLastError](https://learn.microsoft.com/zh-cn/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror) 函数将返回ERROR\_NO\_MORE\_FILES错误值。 注解 若要检索有关快照中记录的第一个进程的信息，请使用 [Process32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32first) 函数。 示例 有关示例，请参阅 [拍摄快照和查看进程](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/taking-a-snapshot-and-viewing-processes)。 要求  |  |  | | --- | --- | | **标头** | tlhelp32.h | | **Library** | Kernel32.lib | | **DLL** | Kernel32.dll |  另请参阅 [CreateToolhelp32Snapshot](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-createtoolhelp32snapshot)  [PROCESSENTRY32](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/ns-tlhelp32-processentry32)  [进程步行](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/process-walking)  [Process32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32first)  [工具帮助函数](https://learn.microsoft.com/zh-cn/windows/desktop/ToolHelp/tool-help-functions) |

## PROCESSENTRY32结构体

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 描述采用快照时驻留在系统地址空间中的进程的列表中的条目。 语法 C++复制  typedef struct tagPROCESSENTRY32 {  DWORD dwSize;  DWORD cntUsage;  DWORD th32ProcessID;  ULONG\_PTR th32DefaultHeapID;  DWORD th32ModuleID;  DWORD cntThreads;  DWORD th32ParentProcessID;  LONG pcPriClassBase;  DWORD dwFlags;  CHAR szExeFile[MAX\_PATH];  } PROCESSENTRY32; 成员 dwSize  结构大小（以字节为单位）。 在调用 [Process32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32first) 函数之前，请将此成员设置为 sizeof(PROCESSENTRY32)。 如果未初始化 **dwSize**， **则 Process32First** 将失败。  cntUsage  此成员不再使用，并且始终设置为零。  th32ProcessID  进程标识符。  th32DefaultHeapID  此成员不再使用，并且始终设置为零。  th32ModuleID  此成员不再使用，并且始终设置为零。  cntThreads  进程启动的执行线程数。  th32ParentProcessID  创建此进程的进程的标识符 (其父进程) 。  pcPriClassBase  此进程创建的任何线程的基本优先级。  dwFlags  此成员不再使用，并且始终设置为零。  szExeFile[MAX\_PATH]  进程的可执行文件的名称。 若要检索可执行文件的完整路径，请调用 [Module32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-module32first) 函数并检查返回的 [MODULEENTRY32](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/ns-tlhelp32-moduleentry32) 结构的 **szExePath** 成员。 但是，如果调用进程是 32 位进程，则必须调用 [QueryFullProcessImageName](https://learn.microsoft.com/zh-cn/windows/desktop/api/winbase/nf-winbase-queryfullprocessimagenamea) 函数来检索 64 位进程的可执行文件的完整路径。 要求 展开表   | **要求** | **值** | | --- | --- | | **最低受支持的客户端** | Windows XP [仅限桌面应用] | | **最低受支持的服务器** | Windows Server 2003 [仅限桌面应用] | | **标头** | tlhelp32.h |  另请参阅 [Process32First](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32first)  [Process32Next](https://learn.microsoft.com/zh-cn/windows/desktop/api/tlhelp32/nf-tlhelp32-process32next) |

# 演练

## 枚举系统进程有上面四种方法，这一节学习第一种，通过快照进行枚举

## 1.新建一个cpp常规空项目，取名Lesson62-enum-proc-snapshot,然后新建一个cpp文件，取名enum-proc-snapshot，然后添加下面的代码

|  |
| --- |
|  |
| #include<Windows.h>  #include<TlHelp32.h>  #include<stdio.h>  int main()  {    HANDLE hpSnapshot;  PROCESSENTRY32 pe32 = {0};  pe32.dwSize = sizeof(PROCESSENTRY32);  //这个函数可以创建堆，进行和线程的快照，我们这里创建进程的快照  hpSnapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);  printf("运行中的进程快照\n");  if(INVALID\_HANDLE\_VALUE == hpSnapshot)  {  printf("创建进程快照失败\n");  return -1;  }  if(!Process32First(hpSnapshot,&pe32))  {  printf("获取第一个进程失败\n");  return -2;  }  do  {  //输出的时候一定要主要格式符号不要写错，否则程序会崩溃  printf("Process Name: %s\n",pe32.szExeFile);  printf("Process ID: %d\t",pe32.th32ProcessID);  printf("Parent Process id: %d\t",pe32.th32ParentProcessID);  printf("Process Thread Count: %d\n",pe32.cntThreads);  printf("\n",pe32.cntThreads);    }while(Process32Next(hpSnapshot,&pe32));  system("pause");  CloseHandle(hpSnapshot);  return 0;  } |

### 运行效果

|  |
| --- |
|  |

### 一个妥妥的命令行简易任务查看器就做好了

## 老师的版本

|  |
| --- |
|  |

### 注意，使用PROCESSENTRY32结构体变量之前需要先把所有成员设置为0，然后设置他的dwSize属性为结构体的大小，否则程序没有任何消息输出

# 扩展，可以尝试使用win32对话框程序或者MFC对话框程序来练习上面的API，使用列表框来显示进程信息

# 这一节学习到此为止，上面上面已经给出