<https://blog.csdn.net/caocg504/article/details/2453490>

自从出现windows系统以来，各种各样的软件层出不穷。购买或者免费下载以后，轻轻点击Install或者Setup以后，稍作些许简单配置就完成安装的整个过程。显然这给用户带来了许多方便，但对用户来说这些软件似乎是个"黑匣子"，展现在用户面前的仅仅是华丽的外表和简洁的操作，至于软件运行过程中调用了哪些文件和模块文件就无法知道了。或许真正希望了解这些的用户并不是很多，甚至对某些用户来说是不必要的，但是对于一个系统安全管理员来说这是一件很重要的事情。本文重点阐述如何获取系统的所有进程以及进程运行过程中所调用的模块文件(象EXE、DLL文件)，实际是对Windows系统所提供的任务管理器进一步完善，开发工具采用Delphi语言。    
  
  
一、函数介绍   
  
  
在Windows系统中动态链接库kernel32.dll提供了获取和处理系统进程的许多接口函数，Delphi语言把这些函数接口封装到Tlhelp32.pas中，供Delphi用户开发过程调用。要详细了解相关知识可以查阅Tlhelp32.pas原文件和Windows SDK提供的帮助文件。其中同本文涉及的接口函数主要有CreateToolhelp32Snapshot、process32first、process32next、module32first、module32next五个函数以及TMODULEENTRY32、TPROCESSENTRY32两个数据结构。   
  
  
1、CreateToolhelp32Snapshot   
  
该函数是要实现上述目的最核心的一个函数，它可以获取系统运行进程(Process)列表、线程(Thread)列表和指定运行进程的堆(Heap)列表、调用模块(Module)列表。如果函数运行成功将返回一个非零"Snapshot"句柄，通过该句柄调用相关WinAPI函数就可以实现上述目的，其函数格式为：   
  
  
HANDLE WINAPI CreateToolhelp32Snapshot(DWORD dwFlags,   
DWORD th32ProcessID);   
  
  
参数说明：   
  
dwFlags参数:对函数建立的"Snapshot"所包含的列表类型，可选项包括：   
  
TH32CS\_SNAPHEAPLIST：所创建的Snapshot包含堆列表   
  
TH32CS\_SNAPMODULE ：所创建的Snapshot包含调用模块列表   
  
TH32CS\_SNAPTHREAD ：所创建的Snapshot包含线程列表；   
  
TH32CS\_SNAPPROCESS ：所创建的Snapshot包含进程列表；   
  
TH32CS\_SNAPALL ：所创建的Snapshot包含上述所有列表；   
  
th32ProcessID参数：进程句柄参数，可以为零表示当前进程，该参数只对dwFlags包含TH32CS\_SNAPMODULE、TH32CS\_SNAPHEAPLIST可选项时起作用。当dwFlags为TH32CS\_SNAPPROCESS，th32ProcessID为零时函数得到系统的所有进程列表。   
  
  
2、Process32First、Process32Next、Module32First、Module32Next   
  
这四个函数都是对"Snapshot"所包含的列表进行息获取，根据函数字面的英文意义，不难理解各函数的含义和区别，四个函数的格式分别为：   
  
  
BOOL WINAPI Process32First(HANDLE hSnapshot,   
LPPROCESSENTRY32 lppe);   
  
BOOL WINAPI Process32Next(HANDLE hSnapshot,  
 LPPROCESSENTRY32 lppe);   
  
BOOL WINAPI Module32First(HANDLE hSnapshot,  
 LPMODULEENTRY32 lpme);   
  
BOOL WINAPI Module32Next(HANDLE hSnapshot,   
LPMODULEENTRY32 lpme);   
  
  
3、TMODULEENTRY32、TPROCESSENTRY32   
  
  
这两个数据结构中TPROCESSENTRY32是在Process32First、Process32Next两个函数所用到的数据结构，TMODULEENTRY32是在Module32First、Module32Next所用到的数据结构，两个数据结构分别如下：   
  
typedef struct tagPROCESSENTRY32 {    
  
DWORD dwSize;    
  
DWORD cntUsage;    
  
DWORD th32ProcessID; //进程句柄   
  
DWORD th32DefaultHeapID;    
  
DWORD th32ModuleID;    
  
DWORD cntThreads;    
  
DWORD th32ParentProcessID;    
  
LONG pcPriClassBase;    
  
DWORD dwFlags;    
  
char szExeFile[MAX\_PATH];    
  
} PROCESSENTRY32;   
  
typedef struct tagMODULEENTRY32 {    
  
DWORD dwSize;    
  
DWORD th32ModuleID;    
  
DWORD th32ProcessID;    
  
DWORD GlblcntUsage;    
  
DWORD ProccntUsage;    
  
BYTE \* modBaseAddr;    
  
DWORD modBaseSize;    
  
HMODULE hModule;    
  
char szModule[MAX\_MODULE\_NAME32 + 1];    
  
char szExePath[MAX\_PATH]; //调用模块的含路径文件名   
  
} MODULEENTRY32;   
  
  
在使用上面两个数据结构要特别强调一点，那就是函数使用这两个数据结构的变量时要先设置dwSize的值，分别用sizeof(TPROCESSENTRY32)和sizeof(TMODULEENTRY32)。   
  
  
由于篇幅有限以上所提到的函数和数据结构可以查看Windows SDK帮助文件获取更详细的信息。 

二、实现原理   
  
  
要实现获得系统的所有运行进程和每个运行进程所调用模块的信息，实际上只要使用两重循环，外循环获取系统的所有进程列表，内循环获取每个进程所调用模块列表。用以下四组API调用实现:   
  
  
1、创建系统的所有进程列表   
  
ProcessList:= CreateToolhelp32Snapshot  
(TH32CS\_SNAPPROCESS,0);   
  
  
2、提取进程列表项信息存储在TPROCESSENTRY32 pe中   
  
Process32First(ProcessList,pe)   
  
Process32Next(ProcessList,pe)   
  
  
3、创建指定进程所有调用模块列表   
  
ModuleList:=CreateToolhelp32Snapshot(TH32CS\_SNAPMODULE,   
  
pe.processID);   
  
  
4、提取调用模块列表项信息存储在TMODULEENTRY32 pm中   
  
Module32First(ModuleList,pm);   
  
Module32Next(ModuleList,pm);   
三、核心源码   
  
  
有了以上知识以后，我们就可以很容易地实现获取系统的所有进程以及各进程运行过程所调用的模块文件了。具体步骤如下：   
  
  
1、运行Delphi，进入Delphi的IDE开发环境；   
  
2、新建Application；   
  
3、在默认的Form1中添加Treeview和Button控件；   
  
4、设置Button的OnClick属性为ProcessEnum；   
  
5、把以下代码输入ProcessEnum过程中；   
  
6、编译运行；   
  
  
核心代码如下所示：   
  
uses Tlhelp32;   
  
procedure TForm1.ProcessEnum(Sender: TObject);   
  
var   
  
ProcessList :Thandle;   
  
pe :TPROCESSENTRY32;   
  
node :TTreenode;   
  
processnumber :integer;   
  
procedure ModuleEnum(processid:Dword);   
  
var   
  
ModuleList :Thandle;   
  
pm :TMODULEENTRY32;   
  
begin   
  
ModuleList:=CreateToolhelp32Snapshot  
(TH32CS\_SNAPMODULE,processID);   
  
pm.dwSize:=sizeof(TMODULEENTRY32);   
  
if module32first(ModuleList,pm)   
  
then begin   
  
treeview1.Items.addchild(node,pm.szexepath);   
  
while module32next(ModuleList,pm) do   
  
treeview1.items.addchild(node,pm.szexepath);   
  
end;   
  
closehandle(ModuleList);   
  
end; // ModuleEnum    
  
begin // ProcessEnum   
  
processnumber:=0;   
  
treeview1.Items.Clear;   
  
ProcessList:=CreateToolhelp32Snapshot  
(TH32CS\_SNAPPROCESS,0);   
  
pe.dwSize:=sizeof(TPROCESSENTRY32);   
  
if process32first(ProcessList,pe) then   
  
begin   
  
node:=treeview1.Items.Add(nil,pe.szexefile);   
  
ModuleEnum(pe.th32ProcessID);   
  
inc(processnumber);   
  
while process32next(ProcessList,pe) do   
  
begin   
  
node:=treeview1.Items.Add(nil,pe.szexefile);   
  
ModuleEnum(pe.th32ProcessID);   
  
inc(processnumber);   
  
end;   
  
end;   
  
edit1.text:=’系统进程：’+inttostr(processnumber);   
  
closehandle(ProcessList);   
  
end;   
四、运行结果