实验一 进程控制与描述

1. 实验目的

利用Windows提供的API函数，编写程序，实现进程的创建和终止（如创建写字板进程及终止该进程），加深对操作系统进程概念的理解，观察操作系统进程运行的动态性能，获得包含多进程的应用程序编程经验。

1. 实验内容

编写一个在dos界面下的简单进程控制系统，实现根据用户选项进行进程的创建、终止，并提供进程的状态。

1. 进程的创建和终止。编写一段程序，可以创建一个进程，并终止当前创建的进程。试观察记录程序执行的结果，并分析原因。
2. 利用VC++6.0实现上述程序设计和调试操作，对于进程创建的成功与否、终止进程操作的成功与否提供一定的提示框。
3. 通过阅读和分析实验程序，学习创建进程、观察进程和终止进程的程序设计方法。
4. 实验步骤
5. 程序设计

Windows所创建的每个进程都从调用CreateProcess() API函数开始，该函数的任务是在对象管理器子系统内初始化进程对象，它实际上是运行一个新的可执行程序，这里需要设置其必备的参数。每一进程都以调用ExitProcess() 或TerminateProcess() API函数终止。通常应用程序的框架负责调用 ExitProcess() 函数。终止进程首先是要找到运行进程的句柄，然后终止该进程。对于C++ 运行库来说，这一调用发生在应用程序的main() 函数返回之后。

通过main()函数调用两个函数void startprocess()和void stop\_process()。主要是在main()函数中用while循环和开关语句switch选择序号对其进行控制，可以先创建进程并观察然后在选择序号终止该进程。

1. 具体实现
2. 创建进程

使用windows提供的CreateProcess函数创建一个新的进程和它的主线程，这个新进程运行指定的可执行文件。

函数原型：

BOOL CreateProcess{

LPCTSTR lpApplicationName,//指定要执行的模块,包括可执行代码的exe文件的文件名

LPTSTR lpCommandLine,//指向一个以空结尾的串,该串定义了要执行的命令行

LPSECURITY\_ATTRIBUTES lpProcessAttributes, //指向一个SECURITY\_ATTRIBUTES结构,该结构决定了返回的句柄是否可被子进程继承

LPSECURITY\_ATTRIBUTES lpThreadAttributes, //同上

BOOL bInheritHandles, //表明新进程是否可继承创建者进程的句柄

DWORD dwCreationFlags, //定义控制优先类和进程创建的附加标志

LPVOID lpEnvironment, //指向一个新进程的环境块

LPCTSTR lpCurrentDirectory, //指向一个以空结尾的串,该串定义了子进程的当前驱动器和当前目录

LPSTARTUPINFO lpStartupInfo, //指向一个STARTUPINFO结构,该结构定义了新进程的主窗口将如何显示

LPPROCESS\_INFORMATION lpProcessInformation //!指向进程信息描述结构,该结构接受关于新进程的描述信息

};

1. 终止进程

使用windows提供的TerminateProcess函数终止进程。

函数原型：

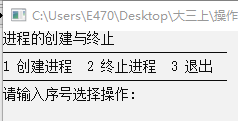
BOOL TerminateProcess{

HANDLE hProcess, //指定要中断进程的句柄

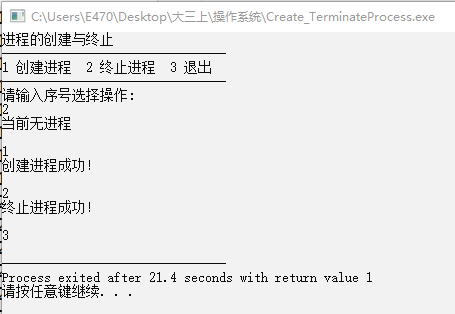
UINT uExitCode //进程及其所有线程的退出代码

};

1. 运行结果
2. 初始界面



1. 创建与终止进程的操作



如果进入主界面时，直接执行序号2终止进程，则没有进程被终止，因为事先没有创建进程，所以句柄为空，无法终止进程。输入1即可打开QQ音乐，输入2即可关闭QQ音乐，输入3结束。

1. 问题及解决方法

问题：对句柄的概念很模糊。

解决方法：查找资料后总结了以下信息：

在程序设计中，句柄是一种特殊的智能指针。当一个应用程序要引用其他系统所管理的内存块或对象时，就要使用句柄。

句柄与普通指针的区别：指针包含的是引用对象的内存地址，而句柄则是由系统所管理的引用标识，该标识可以被系统重新定位到一个内存地址上。这种间接访问对象的模式增强了系统对引用对象的控制。

句柄与进程ID的区别：句柄实际是一个指针，它指向一块包含具体信息数据的内存，可以当做索引，所以进程句柄是当你要访问该进程时取得的，使用完毕必须释放。ID一般是一个标识符，用来区别不同项。一个进程在不同调用时间可能句柄的值不一样，但ID只能有一个。

问题：起初不知道如何创建lpCommandLine。

解决方法：上网搜索后明白lpCommandLine指向的就是要执行的exe文件的绝对路径的字符串，可通过如下方式创建：LPTSTR lpCommandLine=\_tcsdup(TEXT("C://Program Files (x86)//Tencent//QQMusic//QQMusic.exe"));

问题：CreateProcess函数调用失败会使程序出错。

解决方法：写一条CreateProcess函数的判断语句，根据它返回的BOOL值能判断进程是否创建成功：

if(!CreateProcess(NULL,lpCommandLine,NULL,NULL,false,0,NULL,NULL,&si,&pi)){

cout <<"创建进程失败!" <<endl;

exit(1);

}

问题：有时候未关闭运行界面，想要再次运行代码时会报错。

解决方法：运行完一次代码就关闭运行界面。

1. 思考
2. 系统是怎样创建进程的？
3. 打开要在该进程中执行的映像文件
4. 创建Windows执行体进程对象
5. 创建初始线程(栈、堆执行环境初始化及执行线程体对象)
6. 通知Windows子系统新进程已创建
7. 开始执行初始线程
8. 在新进程和线程环境中完成地址空间的初始化(比如加载必须DLL和库），然后开始到进程入口执行
9. 可执行文件加载时进行了那些处理？

首先操作系统判断该文件是否是一个合法的可执行文件，若是则操作系统将按照段表中的指示为可执行程序分配地址空间。

然后进行了将源代码转换为机器可认识的代码的过程。编译程序读取源程序(字符流），对之进行词法和语法的分析，将高级语言指令转换为功能等效的汇编代码，再由汇编程序转換为机器语言，并按操作系统对可执行文件格式的要求链接生成可执行程序。具体处理过程如下：

C源程序一>編译预处理一>编译一>优化程序一>汇编程序一>链接程序一>可执行文件

1. 当首次调用新创建进程时，其入口在那里？

在指向进程的句柄。