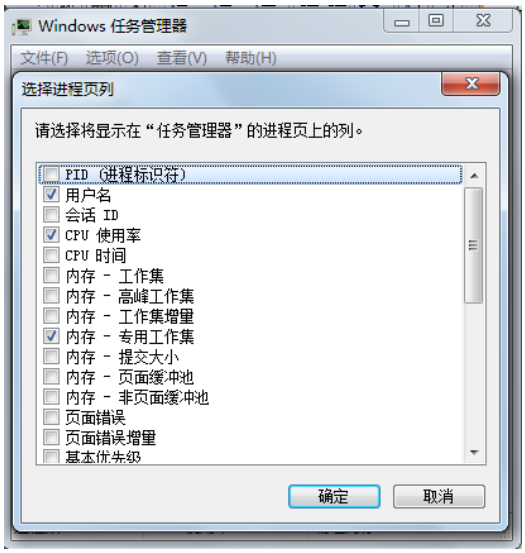
第二章　实验一

Windows 基本进程管理

1. 观察任务管理器

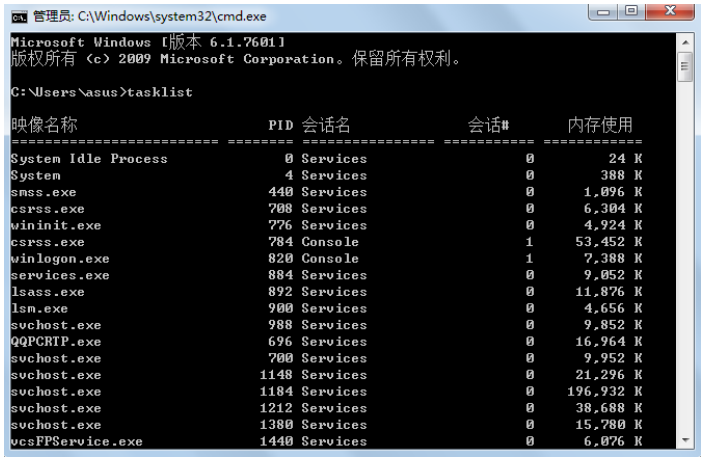
- 进入 Windows 操作系统。
- 按 Ctrl+Alt+Del 调查任务管理器、
- 单击“查看”--“选择列”，查看进程的 PID、CPU 使用时间、内存使用情况、进程的优先级等。



- 单击“性能”，查看 CPU 使用情况、内存使用情况

2. 通过命令观察进程情况

- 在 Windows 中单击“开始”--“运行”，输入 cmd，进入“命令提示符下”
- 输入 tasklist 后显示如下结果



- 输入 tasklist /?寻找帮助
- 关闭进程命令：taskkill　/PID　208　/T

常见 MS-DOS 联机命令（参考教材 P97-99）：

- 格式如下：

  Command　arg1,arg2,…,argn, [option 1,…option k]

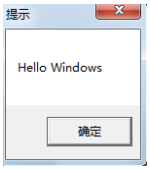
- 分类：
  - ✧ 目录操作类 : dir md cd ls mkdir
  - ✧ 文件操作类 : copy del type cp rm chmod　chown
  - ✧ 磁盘操作类: diskcopy　diskcomp format
  - ✧ 系统访问类: cls clear

3. 简单 Windows 的图形用户接口(GUI)编程

- 使用 Micrsoft Visual Studio C++ 6.0 或 CodeBlocks 编程程序 2_1_GUI.cpp。
- 输入如下程序代码：

```
#include <windows.h>
#include <stdio.h>
//告诉连接器与包括 MessageBox API 函数的 user32 库进行连接
#pragma comment(lib, "user32.lib")
int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow)
{
    ::MessageBox(
        NULL,                 //没有父窗口
        "Hello Windows",    //消息框中的文本显示
        "提示",               //消息框标题
        MB_OK);               //其中只有一个 OK 按钮
    //返回 0 以便通知系统不进入消息循环
    return 0;
}
```

运行结果：



入口函数 在 C/C++中，有一个如下格式的函数： int main(int argc, _TCHAR* argv[]) { return 0; } 该函数是 C/C++中的入口函数，而在 WindowsAPI 中也有自己的入口函数，该函数的格式为： int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LP***）

其中 APIENTRY 表示该函数为一个被系统调用的函数，WinMain 即为 Win32API 的入口函数名。WinMain 函数带有四个参数：

1. HINSTANCE hInstance---系统为本应用程序分配的一个唯一的实例句柄。由于 Windows 系统中存在各种各样的进程，为了便于管理，系统在每一个应用程序启动的时候都会为该应用程序分配一个唯一的数值，这个值即为该应用程序的实例句柄。

2. HINSTANCE hPrevInstance--这个参数只是为了保持与 16 位 Windows 的应用程序的兼容性。设为 NULL 即可。

3. LPTSTR lpCmdLine---命令行字符串数组指针。我们可以在启动一个程序的同时将需要传入到应用程序的数据通过这个参数传入到应用程序当中。

4. int nCmdShow---窗口的显示方式。可以根据传入不同的整数值来决定窗口以什么样的方式显示（当然需要在应用程序中利用该参数作为窗口的显示模式）。

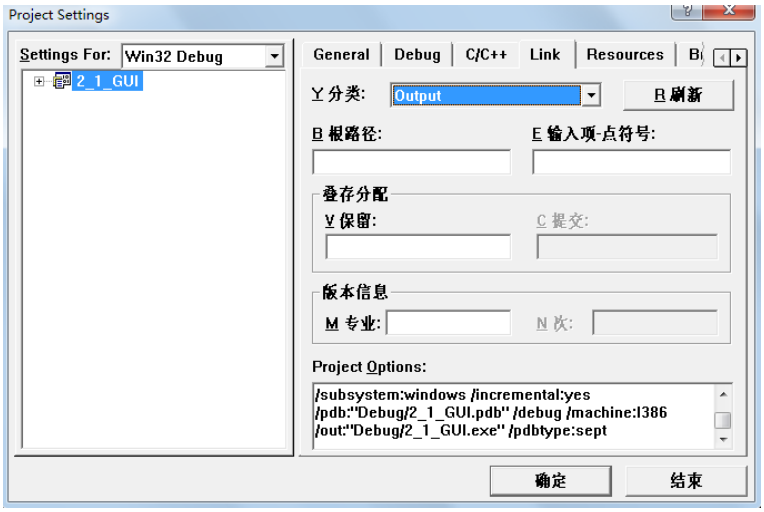
使用 Micrsoft Visual Studio C++ 6.0 调试出现如下错误：

出现问题：提示 LIBCD.lib(crt0.obj) : error LNK2001: unresolved external symbol _main

原因：Windows 项目要使用 Windows 子系统，而不是 Console，可以这样设置:

解决办法：[Project] --> [Settings] --> 选择"Link"属性页，

在 Project Options 中将/subsystem:console 改成/subsystem:windows



4. 确定进程的优先级

- 句柄实际是一个指针，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。
- 使用 Micrsoft Visual Studio C++ 6.0 或 CodeBlocks 编程程序 2_2_Priority.cpp。
- 输入如下程序代码：

```
#include <windows.h>
#include <iostream>
#include <stdio.h>
```

```
//确定自己的优先权
int main()
{
	//获得当前进程的句柄
	HANDLE hProcessThis =::GetCurrentProcess();

	//请求内核提供该进程所属的优先权类
	DWORD dwPriority =::GetPriorityClass(hProcessThis);

	//发出信息，为用户描述该类
	std::cout<< "Current procss priority:";
	switch(dwPriority)
	{
		case HIGH_PRIORITY_CLASS:
			std::cout<<"High";
			break;
		case NORMAL_PRIORITY_CLASS:
			std::cout<<"Normal";
			break;
		case IDLE_PRIORITY_CLASS:
			std::cout<<"Idle";
			break;
		case REALTIME_PRIORITY_CLASS:
			std::cout<<"Realtime";
			break;
		default:
			std::cout<<"<unknow>";
			break;
	}

	std::cout<<std::endl;
	getchar();
}
```

运行结果：