实验 3 认识 Windows 有关进程控制的系统调用，编写程序了解系统调用的用法

目标：教材 3.4 节针对 Unix/Linux 操作系统讲解了常用的进程控制系统调用。本实验调研 Windows 操作系统中相关进程控制系统调用及其正确使用方法。

在 VC 或其它开发环境下，运行如下程序：

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
int main()
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;

    //allocate memory
    ZeroMemory(&si, sizeof(si));
    si.cb=sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

    //create child process
    if(!CreateProcess(NULL,       //use command line
                      "C:\\windows\\system32\\mspaint.exe", //command line
                      NULL,    //don't inherit process handler
                      NULL,    //don't inherit thread handler
                      FALSE,    //disable handle inheritance
                      0,    //no creation flags
                      NULL, //use parent's environment block
                      NULL,    //use parent's existing directory
                      &si,
                      &pi))
    {
        fprintf(stderr, "Create process failed");
        return -1;
    }
    //Parent will wait for the child to complete
    WaitForSingleObject(pi.hProcess, INFINITE);
    printf("Child Complete");

    //close handles. Notice: CloseHandle() only decrease the reference counter of handles
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
}
```

其中：

- CreateProcess()：创建进程的系统调用；
- WaitForSingleObject()：等待一个内核对象（如进程）终止

回答问题：

1、 创建的子进程执行哪个可执行文件镜像？
2、 程序中哪条语句实现了父进程等待子进程结束？
3、 如果父进程在子进程终止前终止，会不会出现子进程随父进程结束而终止的情况？
4、 CloseHandle()是否能够终止子进程？
5、 根据上面四个问题的回答，分析 Unix 和 Windows 进程管理的异同（着重从父子进程的关系入手）。

进一步拓展：

1、 使用 Process Explorer 观察上述父子进程，获取其 pid、优先级和 CPU 使用率等信息。
2、 在例子程序的基础上，编写一个父、子进程竞争标准输出的程序。父进程向标准输出连续写入 10 个字符“a”，子进程向标准输出连续写入 10 个字符“b”。试观察每次输出结果是否一致？试解释结果的不确定性产生的原因？如果要求父进程先输出、子进程后输出，如何实现这样的顺序需求？