第十一课　进程控制

一、进程基本概念

1. 进程是一个容器，包含程序执行需要的代码、数据、资源等信息。
   Windows是多任务操作系统，可以同时执行多个进程。

2. 进程的特点

1) 每个进程都有自己的唯一标识号——PID。
2) 每个进程都有自己的安全属性。
3) 每个进程都有自己的地址空间，
   进程之间无法互相访问对方的地址空间。
4) 每个进程中至少包含一个线程。

二、进程环境(进程上下文)

1. 获取环境块

```
LPTCH GetEnvironmentStrings (void);
```

成功返回进程环境块指针，失败返回NULL。

```
Name1=Value1\0Name2=Value2\0...NameN=ValueN\0\0
```

2. 释放环境块

```
BOOL FreeEnvironmentStrings (
    LPTCH lpszEnvironmentBlock // 进程环境块指针
);
```

成功返回TRUE，失败返回FALSE。

3. 设置环境值

```
BOOL SetEnvironmentVariable (
    LPCTSTR lpName, // 变量名
    LPCTSTR lpValue // 变量值
);
```

成功返回TRUE，失败返回FALSE。

4. 获取环境值

```
DWORD GetEnvironmentVariable (
    LPCTSTR lpName,   // 变量名
    LPTSTR  lpBuffer, // 变量值缓冲区
    DWORD   nSize     // 变量值缓冲区大小
```

```
                                    // 以字符为单位，含结尾空字符
);
```

成功返回变量值字符串长度，失败返回0。

范例：WinEnv

三、进程信息
---

1. 获取进程ID

```
DWORD GetCurrentProcessId (void);
```

返回调用进程ID。不会失败。

2. 获取进程句柄

```
HANDLE GetCurrentProcess (void);
```

返回调用进程伪句柄，用于后续函数调用。
不会失败。

范例：WinProc

四、创建进程
---

1. WinExec

Win16遗留，现在基本不用

2. ShellExecute

Shell操作

3. CreateProcess

现在最多使用

```
BOOL WINAPI CreateProcess (
    LPCTSTR lpApplicationName,    // 可执行程序路径
    LPTSTR  lpCommandLine,        // 命令行参数
    LPSECURITY_ATTRIBUTES lpProcessAttributes,
                                  // 进程安全属性，NULL
    LPSECURITY_ATTRIBUTES lpThreadAttributes,
                                  // 线程安全属性，NULL
    BOOL    bInheritHandles,      // 子进程是否可以继承父进程的可继承句柄
    DWORD   dwCreationFlags,      // 创建方式，0立即启动
    LPVOID  lpEnvironment,        // 子进程环境，
                                  // NULL继承父进程的环境
```

```
    LPCTSTR lpCurrentDirectory,  // 子进程的工作目录，
                                 // NULL继承父进程的工作目录
    LPSTARTUPINFO lpStartupInfo, // 启动信息
    LPPROCESS_INFORMATION lpProcessInformation
                                 // 进程信息(输出)，
                                 // 子进程及其主线程的ID和句柄
);

typedef struct _PROCESS_INFORMATION {
    HANDLE hProcess;    // 子进程句柄
    HANDLE hThread;     // 子进程主线程句柄
    DWORD  dwProcessId; // 子进程ID
    DWORD  dwThreadId;  // 子进程主线程ID
}   PROCESS_INFORMATION, *LPPROCESS_INFORMATION;
```

成功返回TRUE，失败返回FALSE。

范例：WinChild、WinParent

五、退出进程

```
VOID ExitProcess (
    UINT uExitCode // 退出码
);
```

退出本进程。

六、终止进程

```
BOOL TerminateProcess (
    HANDLE hProcess, // 进程句柄
    UINT   uExitCode // 退出码
);
```

成功返回TRUE，失败返回FALSE。

终止其它进程。

七、通过进程ID获取其句柄

```
HANDLE OpenProcess (
    DWORD dwDesiredAccess, // 访问权限，PROCESS_ALL_ACCESS
    BOOL  bInheritHandle,  // 子进程是否可以继承此函数返回的句柄
    DWORD dwProcessId      // 进程ID
);
```

成功返回对应给定PID的进程句柄，失败返回NULL。

范例：WinKill

八、关闭进程句柄

CloseHandle

并非关闭进程，仅释放进程句柄资源。

九、等候进程

```
DWORD WaitForSingleObject (
    HANDLE hHandle,       // 句柄
    DWORD  dwMilliseconds // 等候时间(毫秒)，INFINITE永远等候
);
```

成功返回引起该函数返回的事件码，失败返回WAIT_FAILED(-1)。

引起该函数返回的事件：
    WAIT_OBJECT_0 - 句柄有信号
    WAIT_TIMEOUT - 超出等候时间

阻塞，直到句柄有信号或者超出等候时间才返回。

范例：WinWait