Funcion CreateProcess

La función CreateProcess crea un nuevo proceso, que se ejecuta independientemente del proceso de creación. Sin embargo, para simplificar, la relación se conoce como una relación padre-hijo.

El nuevo proceso se ejecuta en el contexto de seguridad del proceso de llamada. Si el proceso de llamada se hace pasar por otro usuario, el nuevo proceso usa el token para el proceso de llamada, no el token de suplantación.

Para ejecutar el nuevo proceso en el contexto de seguridad del usuario representado por el token de suplantación, use la función CreateProcessAsUser o CreateProcessWithLogonW

Sintaxis

```
BOOL WINAPI CreateProcess(
  _In_opt_    LPCTSTR               lpApplicationName,
  _Inout_opt_ LPTSTR                lpCommandLine,
  _In_opt_    LPSECURITY_ATTRIBUTES lpProcessAttributes,
  _In_opt_    LPSECURITY_ATTRIBUTES lpThreadAttributes,
  _In_        BOOL                  bInheritHandles,
  _In_        DWORD                 dwCreationFlags,
  _In_opt_    LPVOID                lpEnvironment,
  _In_opt_    LPCTSTR               lpCurrentDirectory,
  _In_        LPSTARTUPINFO         lpStartupInfo,
  _Out_       LPPROCESS_INFORMATION lpProcessInformation
);
```

Parametros

lpApplicationName [en, opcional]

Este módulo puede ser una aplicación basada en Windows. Puede ser algún otro tipo de módulo (por ejemplo, MS-DOS u OS / 2) si el subsistema apropiado está disponible en la computadora local.

La cadena puede especificar la ruta completa y el nombre de archivo del módulo a ejecutar o puede especificar un nombre parcial. En el caso de un nombre parcial, la función usa la unidad actual y el directorio actual para completar la especificación. La función no usará la ruta de búsqueda. Este parámetro debe incluir la extensión del nombre de archivo; no se asume ninguna extensión por defecto.

Este parámetro puede ser NULL. En ese caso, el nombre del módulo debe ser el primer token blanco delimitado por espacios en la cadena **lpCommandLine**. Si está utilizando un nombre de archivo largo que contiene un espacio, use cadenas entre comillas para indicar dónde termina el nombre del archivo y comienzan los argumentos; de lo contrario, el nombre del archivo es ambiguo. Por ejemplo, considere la cadena "c: \ archivos de programa \ subdirectorio \ nombre de programa". Esta cadena se puede interpretar de varias maneras. El sistema intenta interpretar las posibilidades en el siguiente orden:

Funcion CreateProcess

c: \ Archivos de programa.exe \ subdirectorio \ nombre de programa

c: \ archivos de programa \ subd.exe dir \ nombre del programa

c: \ archivos de programa \ subdirectorio \ program.exe

c: \ archivos de programa \ subdirectorio \ program name.exe

Si el módulo ejecutable es una aplicación de 16 bits, **lpApplicationName** debe ser NULL, y la cadena apuntada por **lpCommandLine** debe especificar el módulo ejecutable así como sus argumentos.

Para ejecutar un archivo por lotes, debe iniciar el intérprete de comandos; establezca **lpApplicationName** en cmd.exe y establezca **lpCommandLine** en los siguientes argumentos: / c más el nombre del archivo por lotes.

lpCommandLine [in, out, optional]

La línea de comando a ser ejecutada. La longitud máxima de esta cadena es 32.768 caracteres, incluido el carácter nulo de terminación Unicode. Si **lpApplicationName** es NULL, la porción del nombre del módulo de **lpCommandLine** está limitada a MAX_PATH caracteres.

La versión Unicode de esta función, CreateProcessW, puede modificar el contenido de esta cadena. Por lo tanto, este parámetro no puede ser un puntero a la memoria de solo lectura (como una variable const o una cadena literal). Si este parámetro es una cadena constante, la función puede causar una violación de acceso.

El parámetro lpCommandLine puede ser NULL. En ese caso, la función usa la cadena apuntada por lpApplicationName como la línea de comando.

lpProcessAttributes [en, opcional]

Un puntero a una estructura SECURITY_ATTRIBUTES que determina si el proceso devuelto al nuevo objeto de proceso puede ser heredado por procesos secundarios. Si lpProcessAttributes es NULL, el identificador no se puede heredar.

El miembro lpSecurityDescriptor de la estructura especifica un descriptor de seguridad para el nuevo proceso. Si lpProcessAttributes es NULL o lpSecurityDescriptor es NULL, el proceso obtiene un descriptor de seguridad predeterminado. Las ACL en el descriptor de seguridad predeterminado para un proceso provienen del token primario del creador.

lpThreadAttributes [en, opcional]

Un puntero a una estructura SECURITY_ATTRIBUTES que determina si los procesos hijo pueden heredar el identificador devuelto al nuevo objeto thread. Si lpThreadAttributes es NULL, el identificador no se puede heredar.

bherithandles [in]

Si este parámetro es TRUE, el nuevo proceso heredará cada identificador heredable en el proceso de llamada. Si el parámetro es FALSE, los identificadores no se heredan. Tenga

en cuenta que los identificadores heredados tienen el mismo valor y derechos de acceso que los identificadores originales.

Servicios de Terminal Server: no puede heredar identificadores en todas las sesiones. Además, si este parámetro es VERDADERO, debe crear el proceso en la misma sesión que la persona que llama.

Procesos de luz protegida (PPL): la herencia de identificador genérico se bloquea cuando un proceso de PPL crea un proceso que no es de PPL, ya que PROCESS_DUP_HANDLE no está permitido desde un proceso que no es de PPL hasta un proceso de PPL. Consulte Seguridad de proceso y derechos de acceso

dwCreationFlags [en]

Las banderas que controlan la clase de prioridad y la creación del proceso. Para obtener una lista de valores, consulte Indicadores de creación de procesos.

Este parámetro también controla la clase de prioridad del nuevo proceso, que se usa para determinar las prioridades de programación de los hilos del proceso. Para obtener una lista de valores, vea GetPriorityClass. Si no se especifica ninguno de los indicadores de clase de prioridad, la clase de prioridad tiene como valor predeterminado NORMAL_PRIORITY_CLASS a menos que la clase de prioridad del proceso de creación sea IDLE_PRIORITY_CLASS o BELOW_NORMAL_PRIORITY_CLASS. En este caso, el proceso hijo recibe la clase de prioridad predeterminada del proceso de llamada.

ipEnvironment [en, opcional]

Un puntero al bloque de entorno para el nuevo proceso. Si este parámetro es NULL, el nuevo proceso utiliza el entorno del proceso de llamada.

Un bloque de entorno consiste en un bloque terminado en nulo de cadenas terminadas en nulo. Cada cadena está en la siguiente forma:

nombre = valor \ 0

Como el signo igual se usa como separador, no se debe usar en el nombre de una variable de entorno.

Un bloque de entorno puede contener caracteres Unicode o ANSI. Si el bloque de entorno señalado por lpEnvironment contiene caracteres Unicode, asegúrese de que dwCreationFlags incluya CREATE_UNICODE_ENVIRONMENT. Si este parámetro es NULL y el bloque de entorno del proceso principal contiene caracteres Unicode, también debe asegurarse de que dwCreationFlags incluya CREATE_UNICODE_ENVIRONMENT.

La versión ANSI de esta función, CreateProcessA falla si el tamaño total del bloque de entorno para el proceso supera los 32.767 caracteres.

Tenga en cuenta que un bloque de entorno ANSI termina en dos bytes cero: uno para la última cadena y otro para terminar el bloque. Un bloque de entorno Unicode termina en cuatro bytes cero: dos para la última cadena, dos más para terminar el bloque.

Funcion CreateProcess

lpCurrentDirectory [en, opcional]

La ruta completa al directorio actual para el proceso. La cadena también puede especificar una ruta UNC.

Si este parámetro es NULL, el nuevo proceso tendrá la misma unidad y directorio actuales que el proceso de llamada. (Esta función se proporciona principalmente para las shells que necesitan iniciar una aplicación y especificar su unidad inicial y su directorio de trabajo).

Ejemplo:

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

void _tmain( int argc, TCHAR *argv[] )
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;

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

    if( argc != 2 )
    {
        printf("Usage: %s [cmdline]\n", argv[0]);
        return;
    }

    // Start the child process.
    if( !CreateProcess( NULL,   // No module name (use command line)
        argv[1],        // Command line
        NULL,           // Process handle not inheritable
        NULL,           // Thread handle not inheritable
        FALSE,          // Set handle inheritance to FALSE
        0,              // No creation flags
        NULL,           // Use parent's environment block
        NULL,           // Use parent's starting directory
        &si,            // Pointer to STARTUPINFO structure
        &pi )           // Pointer to PROCESS_INFORMATION structure
    )
    {
        printf( "CreateProcess failed (%d).\n", GetLastError() );
        return;
    }

    // Wait until child process exits.
    WaitForSingleObject( pi.hProcess, INFINITE );

    // Close process and thread handles.
    CloseHandle( pi.hProcess );
```

Funcion CreateProcess

```
    CloseHandle( pi.hThread );
}
```

OBSERVACIONES:

La problemática en los sistemas operativos (Windows 8 y Windows 10puede ser resuelta de la siguiente manera: colocando un NULL en el parámetro lpApplicationName ya que el nombre de modulo (limitado por un espacio en negro) debe ser el primer token blanco, en donde será colocado la dirección del programa que queremos ejecutar.

El parámetro bherithandles en estas circunstancias es tomado como a causa de que los indicadores que pudiera heredar no están presentes en todas las sesiones, además si este fuera TRUE sería necesario crear el proceso en la misma sesión.

.