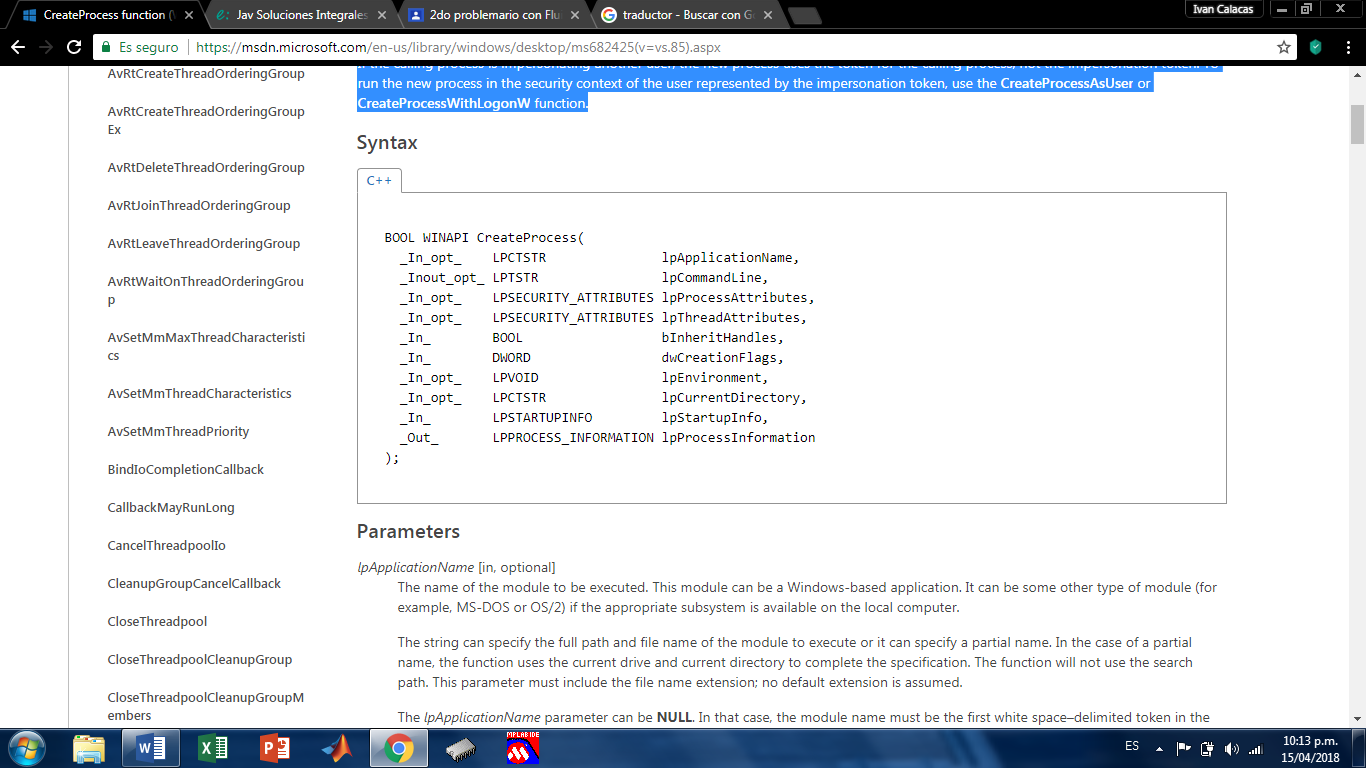
# Función CreateProcess

Crea un nuevo proceso y su hilo principal. 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



# Parámetros

lpApplicationName [entrada, opcional]

El nombre del módulo a ser ejecutado. 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 “STRING” 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.

El parámetro *lpApplicationName* 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:

* 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 [entrada, salida, 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 string). 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.

Si *lpApplicationName* y *lpCommandLine* no son NULL, la cadena terminada en nulo apuntada por *lpApplicationName* especifica el módulo a ejecutar, y la cadena terminada en nulo apuntada por lpCommandLine especifica la línea de comando. El nuevo proceso puede usar *GetCommandLine* para recuperar toda la línea de comando. Los procesos de consola escritos en C pueden usar los argumentos argc y argv para analizar la línea de comando. Como argv [0] es el nombre del módulo, los programadores C generalmente repiten el nombre del módulo como el primer token en la línea de comando.

Si *lpApplicationName* es NULL, el primer token delimitado por espacios en blanco de la línea de comando especifica el nombre del módulo. Si está utilizando un nombre de archivo largo que contiene un espacio, use las cadenas entre comillas para indicar dónde termina el nombre del archivo y comienzan los argumentos (consulte la explicación del parámetro *lpApplicationName*). Si el nombre del archivo no contiene una extensión, se agrega .exe. Por lo tanto, si la extensión de nombre de archivo es .com, este parámetro debe incluir la extensión .com.

Si el nombre del archivo finaliza en un punto (.) Sin extensión, o si el nombre del archivo contiene una ruta, no se agrega .exe. Si el nombre del archivo no contiene una ruta de directorio, el sistema busca el archivo ejecutable en la siguiente secuencia:

1. El directorio desde el que se cargó la aplicación.
2. El directorio actual para el proceso principal.
3. El directorio del sistema de Windows de 32 bits. Use la función *GetSystemDirectory* para obtener la ruta de este directorio.
4. El directorio del sistema de Windows de 16 bits. No hay ninguna función que obtenga la ruta de este directorio, pero se busca. El nombre de este directorio es Sistema.
5. El directorio de Windows. Use la función *GetWindowsDirectory* para obtener la ruta de este directorio.

Los directorios que se enumeran en la variable de entorno PATH. Tenga en cuenta que esta función no busca la ruta por aplicación especificada por la clave de registro App Paths. Para incluir esta ruta por aplicación en la secuencia de búsqueda, use la función ShellExecute.

El sistema agrega un carácter nulo de terminación a la cadena de línea de comandos para separar el nombre del archivo de los argumentos. Esto divide la cadena original en dos cadenas para procesamiento interno.

*lpProcessAttributes [entrada, 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.

*Windows XP: las ACL en el descriptor de seguridad predeterminado para un proceso provienen del token primario o de suplantación del creador. Este comportamiento cambió con Windows XP con SP2 y Windows Server 2003.*

*lpThreadAttributes [entrada, 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.

El miembro *lpSecurityDescriptor* de la estructura especifica un descriptor de seguridad para el hilo principal. Si *lpThreadAttributes* es NULL o *lpSecurityDescriptor* es NULL, el hilo obtiene un descriptor de seguridad predeterminado. Las ACL en el descriptor de seguridad predeterminado para un hilo provienen del token de proceso.

*Windows XP: las ACL en el descriptor de seguridad predeterminado para un hilo provienen del token primario o de suplantación del creador. Este comportamiento cambió con Windows XP con SP2 y Windows Server 2003.*

*bherithandles [entradas]*

Si este parámetro es VERDADERO, 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 de proceso 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.

*dwCreationFlags [entradas]*

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. 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.

*lpEnvironment [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.

*lpCurrentDirectory [entrada, 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).

*lpCurrentDirectory [entrada, 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).

*lpStartupInfo [entrada]*

Un puntero a una estructura STARTUPINFO o STARTUPINFOEX.

# Valor de retorno

Si la función tiene éxito, el valor devuelto es distinto de cero.

Si la función falla, el valor de retorno es cero. Para obtener información extendida sobre el error, llame a GetLastError.

Tenga en cuenta que la función regresa antes de que el proceso haya finalizado la inicialización. Si no se puede ubicar una DLL requerida o no se puede inicializar, el proceso finaliza. Para obtener el estado de finalización de un proceso, llame a GetExitCodeProcess.

# Requerimientos

