Process and thread services in Windows

Enrique Arias
Universidad de Castilla–La Mancha

Contents

Contents

Main objectives

- To know how to program in C language the creation of threads and processes, as well as synchronization among threads.
  - To know services related to objects, such as **CloseHandle**.
  - To know the functions to identify threads and processes: **GetCurrentProcessId, GetCurrentTrhreadId**
  - To know how to create threads and processes in Windows: **CreateThread and CreateThread**
  - To know how to synchronize objects: **WaitForSingleObject, WaitForMultipleObjects**

Contents

Before starting

- In order to use the Windows' services it is necessary to include the header **Windows.h**.
- Do you know how to pass parameters through the main program?

```
int main(int argc, char *argv[])
```

- Netbeans environment + gcc compiler or only gcc (Notepad+ for editing)

```
gcc Sourcefile.c -o Exefile.exe
```

Contents

Objects, handles and identifiers

- Which are objects and handles?

https://docs.microsoft.com/es-es/windows/desktop/SysInfo/handles-and-objects

- Once the object is not going to be used anymore, its handle has to be closed using **CloseHandle** service

```
BOOL CloseHandle(HANDLE hObject);
```

Objects, handles and identifiers

- Processes and threads have an unique identifier in the system. To obtain this identifier the following services have be used

DWORD GetCurrentProcessId(void);

https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-getcurrentprocessid

DWORD GetCurrentThreadId(void);

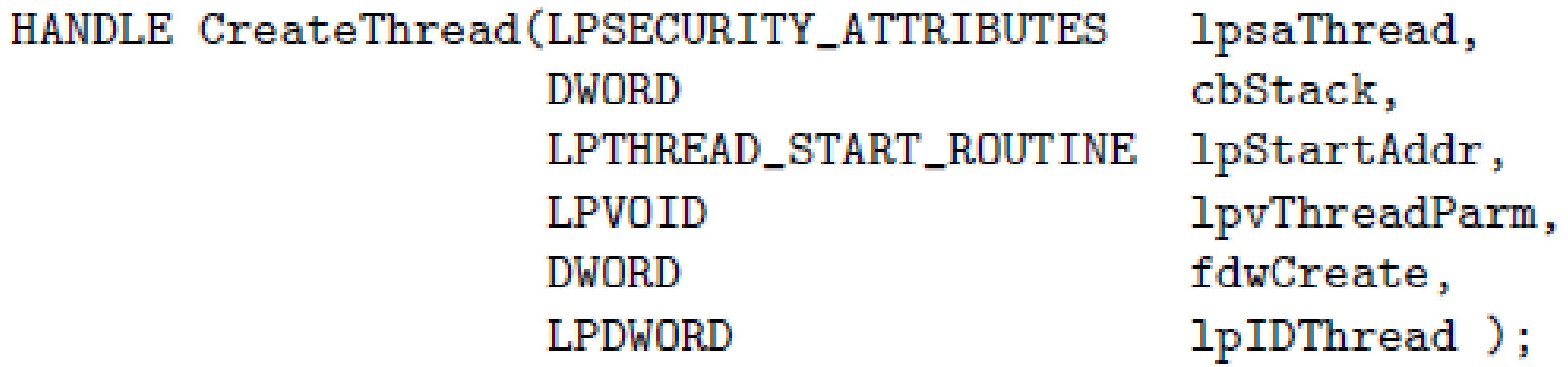
https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-getcurrentthreadid

Contents

Threads creation

- Primary and secondary threads.
- CreateThread

https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createthread



```
HANDLE CreateThread(LPSECURITY_ATTRIBUTES   lpsaThread,
                    DWORD                   cbStack,
                    LPTHREAD_START_ROUTINE  lpStartAddr,
                    LPVOID                  lpvThreadParm,
                    DWORD                   fdwCreate,
                    LPDWORD                 lpIDThread );
```

Threads creation

- CreateThreads
  - lpsaThread: A pointer to a SECURITY_ATTRIBUTES structure that determines whether the returned handle can be inherited by child processes. If is NULL, the handle cannot be inherited.
  - cbStack: The initial size of the stack, in bytes. The system rounds this value to the nearest page. If this parameter is zero, the new thread uses the default size for the executable
  - lpStartAddr: A pointer to the application-defined function to be executed by the thread.
  - lpvThreadParm: A pointer to a variable to be passed to the thread.
  - fdwCreate: The flags that control the creation of the thread.
  - lpIDThread: A pointer to a variable that receives the thread identifier

Contents

Waiting services on objects

- WaitForSingleObject waits until the specified object is in the signaled state or the time-out interval elapses.
- DWORD WaitForSingleObject(HANDLE hObject, DWORD dwTimeout);

https://docs.microsoft.com/en-us/windows/desktop/api/synchapi/nf-synchapi-waitforsingleobject

  - hObject: A handle to the object.
  - dwTimeout: The time-out interval, in milliseconds.
  - Return value: WAIT_OBJECT_0 or WAIT_TIMEOUT if all is ok, if not WAIT_ABANDONED or WAIT_FAILED

Waiting services on objects

- WaitForMultipleObjects waits until one or all of the specified objects are in the signaled state or the time-out interval elapses.
- DWORD WaitForMultipleObjects(HANDLE cObjects, LPHANDLE lpHandles, BOOL bWaitAll, DWORD dwTimeout);

https://docs.microsoft.com/en-us/windows/desktop/api/synchapi/nf-synchapi-waitformultipleobjects

Waiting services on objects

- cObjects: The number of object handles in the array pointed to by *lpHandles*. This parameter cannot be zero.
- LPHANDLE lpHandles: An array of object handles.
- bWaitAll: If this parameter is TRUE, the function returns when the state of all objects in the lpHandles array is signaled. If FALSE, the function returns when the state of any one of the objects is set to signaled.
- dwTimeout: The time-out interval, in milliseconds.
- Return value similar to WaitForSingleObject.

Exercise 1

- Write a program that creates a secondary thread that receives as parameter an integer value N passed on the command line (Windows console). The secondary thread must execute N times a loop where it adds 1 to a local variable initialized to 0. The primary thread must wait until the secondary thread ends and then close the descriptor.

- Locate both the source and executable files on the disk and execute the program from the Windows console.

Exercise 1

- Example of the output
  - I am process 2916
  - Starts primary thread (ID 1716)
  - Starts secondary thread (ID 1136)
  - Final value of variable: 30
  - Ends secondary thread (ID 1136)
  - Ends primary thread (ID 1716)

Exercise 2

- Write a program that creates two threads that execute the same function. Each of the threads must receive as a parameter a different value (1 and 2). Both the primary thread and the two created threads must show on the screen (at the appropriate time) the messages Start thread X and end thread X, where X is primary, 1 or 2. The primary thread must wait until the end of the secondary thread and then close the handles.

- For its part, the secondary threads execute a loop N times where they must modify a shared variable whose initial value is 0: thread 1 increases the variable by one unit and thread 2 also decrements it by one unit. After each modification of the variable each thread must show a message indicating who it is (1 or 2), what operation it has performed (increase, decrease) and the value of the variable after the operation.

Exercise 2

- Example of output
  - I am the process 2824
  - Starts primary thread (ID 2756)
  - Starts thread 2 (ID 2772)
  - Starts thread 1 (ID 2076)
  - Thread 2, decreases, -1
  - Thread 2, decreases, -2
  - Ends thread 2 (ID 2772)
  - Thread 1, increases, -1
  - Thread 1, increases, 0
  - Ends thread 1 (ID 2076)
  - Ends primary thread (ID 2756)

Contents

Process creation

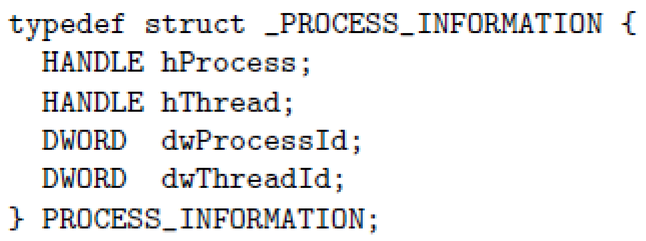
- CreateProcess creates a new process and its primary thread.

```
BOOL CreateProcess(LPCTSTR                  lpszImageName,
                   LPCTSTR                  lpszCommandLine,
                   LPSECURITY_ATTRIBUTES    lpsaProcess,
                   LPSECURITY_ATTRIBUTES    lpsaThread,
                   BOOL                     fInheritHandles,
                   DWORD                    fdwCreate,
                   LPVOID                   lpvEnvironment,
                   LPTSTR                   lpszCurDir,
                   LPSTARTUPINFO            lpsiStartInfo,
                   LPPROCESS_INFORMATION    lppiProcInfo) ;
```

https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createprocessa

Process creation

- CreateProcess
  - lpszImageName: The string can specify the full path and file name of the module to execute
  - lpszCommandLine:The command line to be executed.
  - lpsaProcess: A pointer to a SECURITY_ATTRIBUTES structure that determines whether the returned handle to the new process object can be inherited by child processes. Could be NULL.
  - lpsaThread: A pointer to a SECURITY_ATTRIBUTES structure that determines whether the returned handle to the new thread object can be inherited by child processes. Could be NULL.
  - fInheritHandles: If this parameter is TRUE, each inheritable handle in the calling process is inherited by the new process. If the parameter is FALSE, the handles are not inherited.
  - fdwCreate: The flags that control the priority class and the creation of the process.
  - lpvEnvironment: A pointer to the environment block for the new process.
  - lpszCurDir: The full path to the current directory for the process
  - lpsiStarInfo: A pointer to a STARTUPINFO or STARTUPINFOEX structure.
  - lppiProcInfo: A pointer to a PROCESS_INFORMATION structure that receives identification information about the new process.



```
typedef struct _PROCESS_INFORMATION {
  HANDLE hProcess;
  HANDLE hThread;
  DWORD  dwProcessId;
  DWORD  dwThreadId;
} PROCESS_INFORMATION;
```

Exercise 3

- Write a program that creates two processes and wait until both end. One of the processes must invoke the program of exercise 1 and the other that of exercise 2.
- Example of output
  - -->I am process 1656
  - -->Created process 2916
  - -->Created primary thread 1716
  - ... Here the information of exercise 1 ...
  - -->Created the process 2824.
  - -->Created the primary thead 2756
  - ... Here the information of exercise 2 ...
  - -->Ends process 2916
  - -->Ends process 2824

Process and thread services in Windows

Enrique Arias
Universidad de Castilla–La Mancha