

# Scheduling 5 WINDOWS

martedì 28 ottobre 2025 14:49

Vediamo ora lo scheduling in windows :

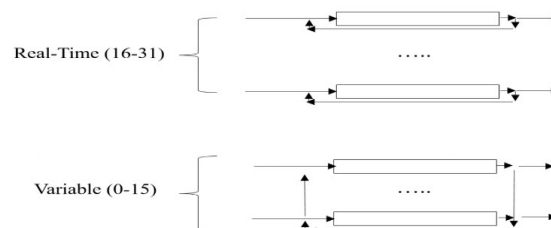
## Caratteristiche

- code multiple distinte in due fasce: *Real-Time* e *Variable*
- un livello di priorità distinto per ciascuna coda (0-15 nella fascia Variable e 16-31 nella fascia Real-Time)
- gestione di tipo Round-Robin nell'ambito di ciascuna coda
- priorità base per i processi
- priorità dinamica entro vincoli per i threads
- prelievo basato su priorità

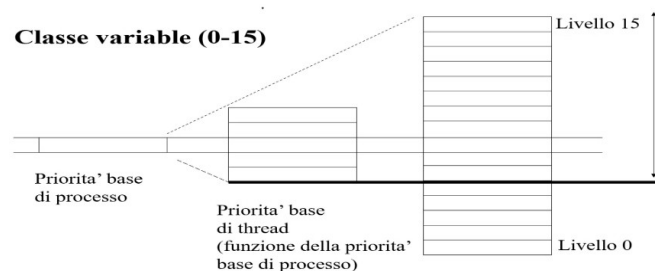
## Passaggio da una coda all'altra (feedback)

- non ammesso nella fascia Real-Time
- ammesso nella fascia Variable (rilascio della CPU allo scadere del quanto provoca diminuzione della priorità, rilascio anticipato provoca incremento)

Quindi in dettaglio :



Quindi se in fascia variable , il thread rientra in ready torna nella stessa fascia di dove era; mentre nella fascia variable è possibile cambiare la priorità ( aumentadola/diminuendola). Quindi focalizziamoci sulla fascia variable:



Nota : la priorità del thread e del processo potrebbe essere diversa. Quindi andiamo a vedere le system call che permettono di assegnare la priorità al processo :

```
BOOL WINAPI SetPriorityClass(  
    _In_ HANDLE hProcess,  
    _In_ DWORD dwPriorityClass  
);
```

Priority	Meaning
ABOVE_NORMAL_PRIORITY_CLASS (0x00000000)	Process that has priority above <b>NORMAL_PRIORITY_CLASS</b> but below <b>HIGH_PRIORITY_CLASS</b> .
BELOW_NORMAL_PRIORITY_CLASS (0x00000001)	Process that has priority above <b>IDLE_PRIORITY_CLASS</b> but below <b>NORMAL_PRIORITY_CLASS</b> .
HIGH_PRIORITY_CLASS (0x00000002)	Process that performs time-critical tasks that must be executed immediately. The threads of the process preempt the threads of normal or idle-priority class processes. An example is the Task List, which must respond quickly when called by the user, regardless of the load on the operating system. Use extreme care when using the high-priority class, because a high-priority class application can use nearly all available CPU time.
IDLE_PRIORITY_CLASS (0x00000003)	Process whose threads run only when the system is idle. The threads of the process are preempted by the threads of any process running in a higher priority class. An example is a screen saver. The idle-priority class is inherited by child processes.
NORMAL_PRIORITY_CLASS (0x00000004)	Process with no special scheduling needs.
PROCESS_MODE_BACKGROUND_BEGIN (0x00000005)	Begin background processing mode. The system lowers the resource scheduling priorities of the process (and its threads) so that it can perform background work without significantly affecting activity in the foreground. This value can be specified only if <i>hProcess</i> is a handle to the current process. The function fails if the process is already in background processing mode. <b>Windows Server 2003 and Windows XP:</b> This value is not supported.
PROCESS_MODE_BACKGROUND_END (0x00000006)	End background processing mode. The system restores the resource scheduling priorities of the process (and its threads) as they were before the process entered background processing mode. This value can be specified only if <i>hProcess</i> is a handle to the current process. The function fails if the process is not in background processing mode. <b>Windows Server 2003 and Windows XP:</b> This value is not supported.
REALTIME_PRIORITY_CLASS (0x00000007)	Process that has the highest possible priority. The threads of the process preempt the threads of all other processes, including operating system processes performing important tasks. For example, a real-time process that executes for more than a very brief interval can cause disk caches not to flush or cause the mouse to be unresponsive.

Mentre per cambiare la priorità ai thread :

```

BOOL WINAPI SetThreadPriority(
    _In_ HANDLE hThread,
    _In_ int nPriority
);

```

Priority	Meaning
THREAD_MODE_BACKGROUND_BEGIN (0x0010000)	Begin background processing mode. The system lowers the resource scheduling priorities of the thread so that it can perform background work without significantly affecting activity in the foreground. This value can be specified only if <i>hThread</i> is a handle to the current thread. The function fails if the thread is already in background processing mode. <b>Windows Server 2003 and Windows XP:</b> This value is not supported.
THREAD_MODE_BACKGROUND_END (0x0020000)	End background processing mode. The system restores the resource scheduling priorities of the thread as they were before the thread entered background processing mode. This value can be specified only if <i>hThread</i> is a handle to the current thread. The function fails if the thread is not in background processing mode. <b>Windows Server 2003 and Windows XP:</b> This value is not supported.
THREAD_PRIORITY_ABOVE_NORMAL 1	Priority 1 point above the priority class.
THREAD_PRIORITY_BELOW_NORMAL -1	Priority 1 point below the priority class.
THREAD_PRIORITY_HIGHEST 2	Priority 2 points above the priority class.
THREAD_PRIORITY_IDLE -15	Base priority of 1 for <b>IDLE_PRIORITY_CLASS</b> , <b>BELOW_NORMAL_PRIORITY_CLASS</b> , <b>NORMAL_PRIORITY_CLASS</b> , <b>ABOVE_NORMAL_PRIORITY_CLASS</b> , or <b>HIGH_PRIORITY_CLASS</b> processes, and a base priority of 16 for <b>REALTIME_PRIORITY_CLASS</b> processes.
THREAD_PRIORITY_LOWEST -2	Priority 2 points below the priority class.
THREAD_PRIORITY_NORMAL 0	Normal priority for the priority class.
THREAD_PRIORITY_TIME_CRITICAL 15	Base priority of 15 for <b>IDLE_PRIORITY_CLASS</b> , <b>BELOW_NORMAL_PRIORITY_CLASS</b> , <b>NORMAL_PRIORITY_CLASS</b> , <b>ABOVE_NORMAL_PRIORITY_CLASS</b> , or <b>HIGH_PRIORITY_CLASS</b> processes, and a base priority of 31 for <b>REALTIME_PRIORITY_CLASS</b> processes.

If the thread has the **REALTIME\_PRIORITY\_CLASS** base class, this parameter can also be -7, -6, -5, -4, -3, 3, 4, 5, or 6. For more information, see [Scheduling Priorities](#).

Dove il parametro *nPriority* assume i seguenti valori , mediante costanti :

Process priority class	Thread priority level	Base priority
IDLE_PRIORITY_CLASS	THREAD_PRIORITY_IDLE	1
	THREAD_PRIORITY_LOWEST	2
	THREAD_PRIORITY_BELOW_NORMAL	3
	THREAD_PRIORITY_NORMAL	4
	THREAD_PRIORITY_ABOVE_NORMAL	5
	THREAD_PRIORITY_HIGHEST	6
	THREAD_PRIORITY_TIME_CRITICAL	15
BELOW_NORMAL_PRIORITY_CLASS	THREAD_PRIORITY_IDLE	1
	THREAD_PRIORITY_LOWEST	4
	THREAD_PRIORITY_BELOW_NORMAL	5
	THREAD_PRIORITY_NORMAL	6
	THREAD_PRIORITY_ABOVE_NORMAL	7
	THREAD_PRIORITY_HIGHEST	8
	THREAD_PRIORITY_TIME_CRITICAL	15
NORMAL_PRIORITY_CLASS	THREAD_PRIORITY_IDLE	1
	THREAD_PRIORITY_LOWEST	6
	THREAD_PRIORITY_BELOW_NORMAL	7
	THREAD_PRIORITY_NORMAL	8
	THREAD_PRIORITY_ABOVE_NORMAL	9
	THREAD_PRIORITY_HIGHEST	10
	THREAD_PRIORITY_TIME_CRITICAL	15
ABOVE_NORMAL_PRIORITY_CLASS	THREAD_PRIORITY_IDLE	1
	THREAD_PRIORITY_LOWEST	8
	THREAD_PRIORITY_BELOW_NORMAL	9
	THREAD_PRIORITY_NORMAL	10
	THREAD_PRIORITY_ABOVE_NORMAL	11
	THREAD_PRIORITY_HIGHEST	12
	THREAD_PRIORITY_TIME_CRITICAL	15
HIGH_PRIORITY_CLASS	THREAD_PRIORITY_IDLE	1
	THREAD_PRIORITY_LOWEST	11
	THREAD_PRIORITY_BELOW_NORMAL	12
	THREAD_PRIORITY_NORMAL	13
	THREAD_PRIORITY_ABOVE_NORMAL	14
	THREAD_PRIORITY_HIGHEST	15
	THREAD_PRIORITY_TIME_CRITICAL	15
REALTIME_PRIORITY_CLASS	THREAD_PRIORITY_IDLE	16
	THREAD_PRIORITY_LOWEST	22
	THREAD_PRIORITY_BELOW_NORMAL	23
	THREAD_PRIORITY_NORMAL	24
	THREAD_PRIORITY_ABOVE_NORMAL	25
	THREAD_PRIORITY_HIGHEST	26
	THREAD_PRIORITY_TIME_CRITICAL	31