

# Multithreading Application in Win32

## Chapter 3\_Hurry Up and Wait

Eung-Sang Kim

*Kim-0103@mmlab.net*

[Multimedia Lab.](#)

Dept. of Electrical and Computer Eng.

University of Seoul

Seoul, Korea

# 0. Document History

---

- Presented by Eung-Sang Kim[2001. 12. 24]
- Added by hiver [2003. 01. 03]

# 1. Contents

---

1. Busy Waiting
2. Performance Monitors
3. Waiting for a Thread to Exit
4. Signaled Objects
5. Waiting for Multiple Objects
6. Waiting in a GUI Program
7. Summary

## 2. Main Objectives

---

- How the processor gets **shared** system resources.
- How programs can **reduce their impact** on system resources.
- Introduce the performance **monitoring tools**.
- Introduces **signaled objects** and the various **Wait,,()** functions.

# 3.1 Busy Waiting

---

- Two techniques to wait in Chapter 2
  - **Sleep( )** function
    - suspends the execution of the current thread for the specified interval. ( )
    - Sleep(0) : thread relinquish the remainder of its time slice to any other thread of equal priority.
    - Sleep(INFINITE) : cause an infinite delay
  - Create a **busy loop** (= busy waits)
    - Use [GetExitCodeThread\(\)](#) function.
    - But it has the significant drawback that it wastes a lot of CPU time.

## 3.1 Busy Waiting (con't)

---

- Look at the Listing 3-1. BUSYWAIT.C
  - Calculates **PI** twice and displays the amount of time it takes.
    - First run : use normal function call
    - Second run : use CreateThread() function
      - ~ took **almost twice as long to run** as the normal function.
      - ~ caused by preemptive multitasking.
      - The primary thread uses its processor time to **frantically check** the return value over and over again.

## 3.2 Performance Monitors

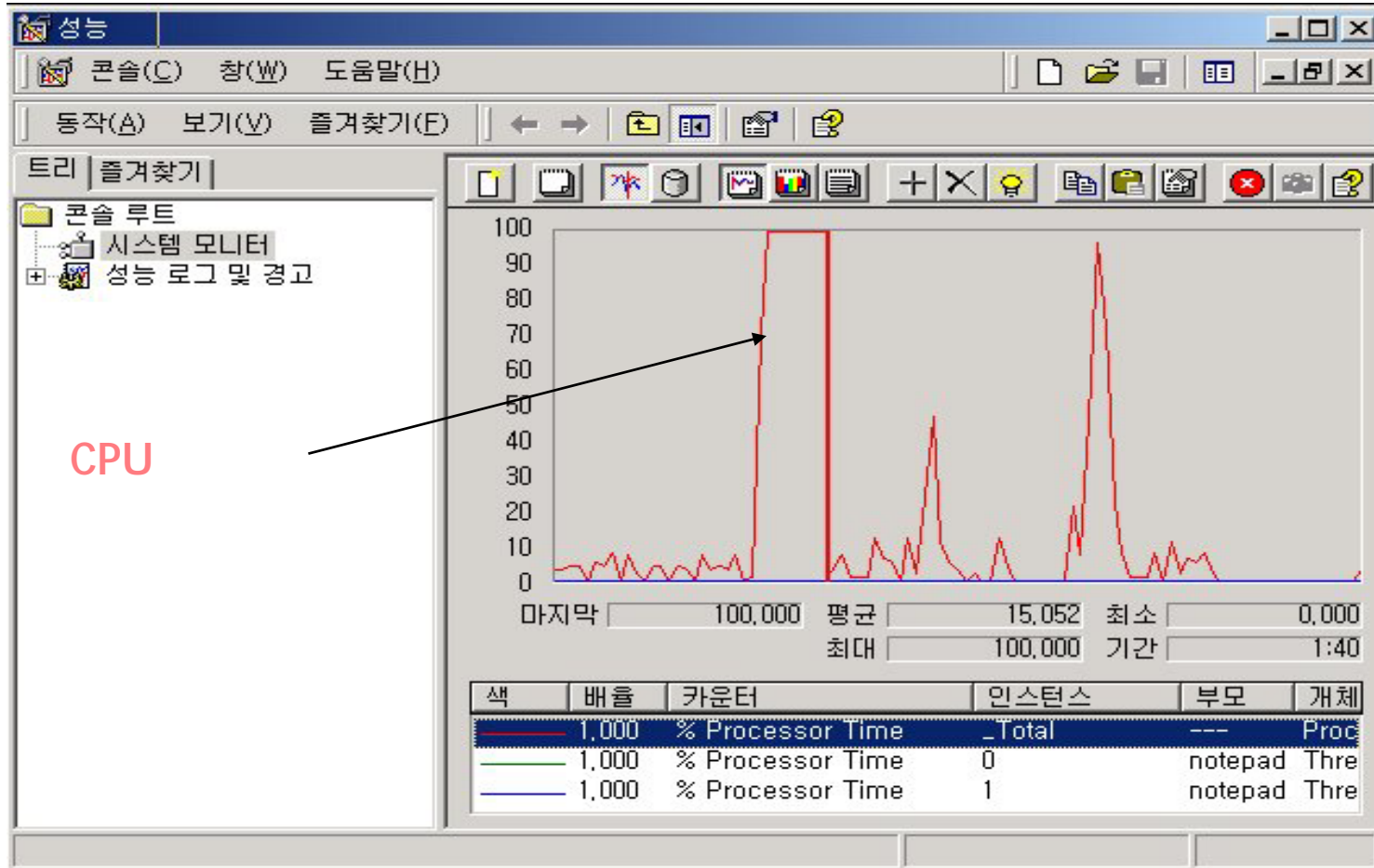
---

- Windows NT and Windows 95 provide tools that can warn you when your application is using excessive CPU time.
- Important for verifying performance in any Win32 application
- Invaluable for developing multithreaded applications.

# 3.2 Performance Monitors

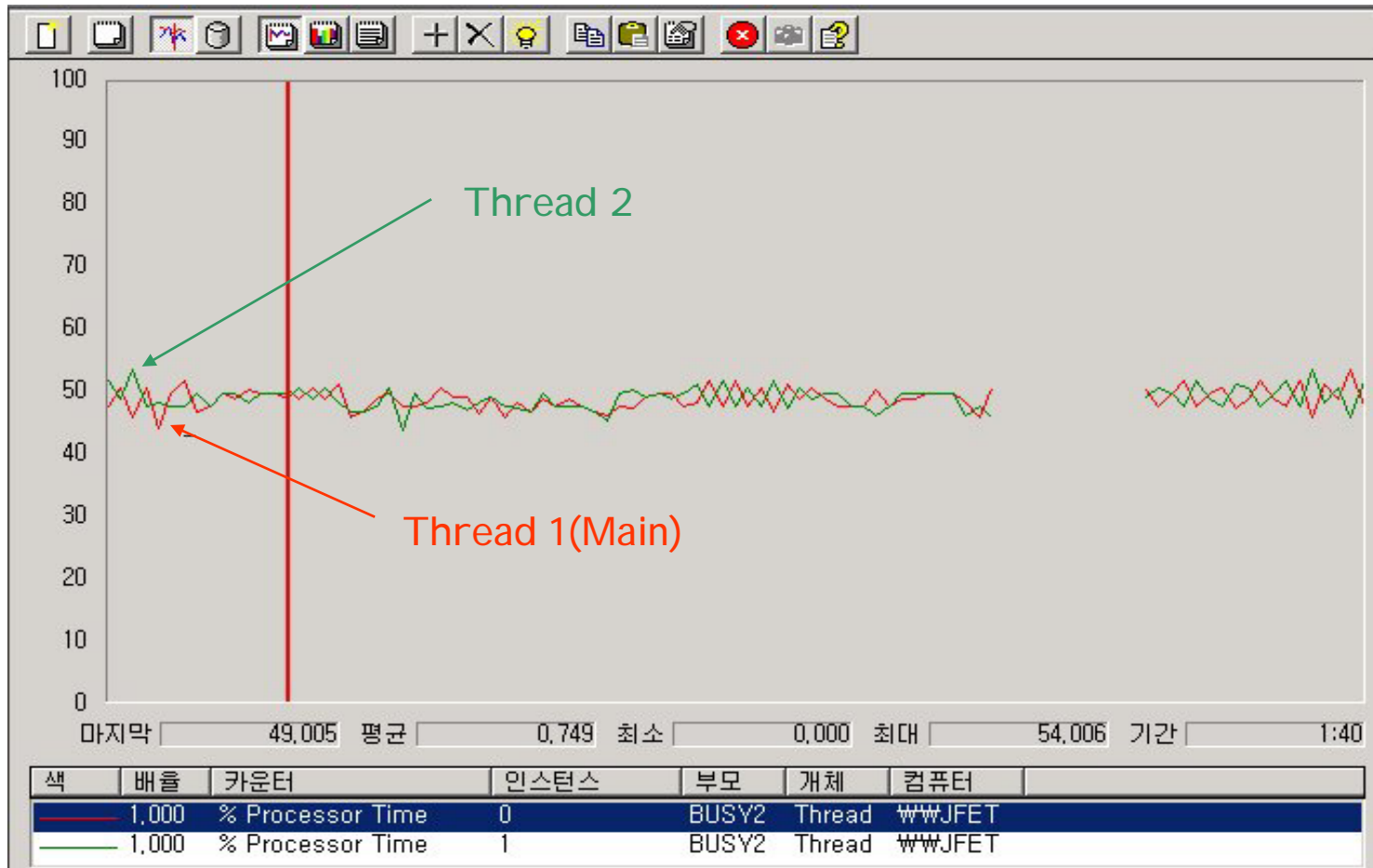
- Windows NT Performance Monitor

-                   ->                   -> perfmon.exe



## 3.2 Performance Monitors

- Run BUSY2.C, then change to Performance Monitor



## 3.3 Waiting for a Thread to Exit

---

- Using busy loops is bad idea.
  - Because of Not very productive....
- What we really need is a **version of the Sleep( )** call that wakes up when a certain thread exits instead of after a certain amount of time has elapsed.

## 3.3 Waiting for a Thread to Exit(con't)

- Use the `WaitForSingleObject()` function

```
DWORD WaitForSingleObject(  
    HANDLE hHandle,          // handle to a kernel object  
    DWORD dwMilliseconds // time-out interval  
);
```

- Ask the O/S to put Thread1 to sleep until either :
  - Thread2 exits.
  - The amount of time (time-out interval) by goes by.
- Return value
  - `WAIT_OBJECT_0` : the object becomes signaled
  - `WAIT_TIMEOUT` : time-out elapses
  - `WAIT_ABANDONED`: thread exits without releasing the mutex(in chap. 4)
  - `WAIT_FAILED` : the fuction failed

busy loop can be replaces with this single line of code.

```
-> WaitForSingleObject(hThrd,INFINITE);
```

## 3.4 Signaled Objects

---

- Kernel objects that can be used with `WaitForSingleObject()` have two states.
  - **Signaled**
  - **Nonsignaled**
- `WaitForSingleObject()` wakes up when an object becomes signaled.
- In the case of thread
  - Nonsignaled : the thread is running
  - Signaled : the thread terminates

## 3.4 Signaled Objects

- Meaning of signaled for kernel objects

object	Description
Thread	Signaled : thread terminates Nonsignaled : thread is running
Process	Signaled : process terminates Nonsignaled : process is running
<u>Mutex</u>	Signaled : Not owned by any thread Nonsignaled : when it is owned
<u>Semaphore</u>	Signaled : count is greater than zero Nonsignaled : count is 0

## 3.5 Waiting for Multiple Objects

---

- Look at a **TASKQUES.C**
  - Use most three threads to do six tasks.
- The task doing work is simulated by calling `Sleep()` to wait around for few seconds.
- Whenever a thread exits, a new thread will be created to do the next task.

## 3.5 Waiting for Multiple Objects (con't)

---

- **BUT**... **TASKQUES.C** is extremely **inefficient**, because it assumes that the thread will exit in the order that they were started.
- Look at a **TASKQUEM.C**
  - The Win32 call `WaitForMultipleObject()` allows you to wait for more than one object at the same time.
  - The function return when one of the following occurs :
    - Either any one or all of the specified objects are in the signaled state.
    - The time-out interval elapses.

## 3.5 Waiting for Multiple Objects (con't)

- **WaitForMultipleObjects()**

```
DWORD WaitForMultipleObjects(  
    DWORD          nCount, // number of handles in array  
    CONST HANDLE  *lpHandles, // pointer to an array of  
object-          handles  
    BOOL          bWaitAll, // wait option  
    DWORD        dwMilliseconds // time-out interval  
);
```

- Return Value
  - WAIT\_TIMEOUT : the function times out
  - WAIT\_OBJECT\_0 : bWaitAll is true
  - Return value - WAIT\_OBJECT\_0 : which object was signaled
  - WAIT\_ABANDONED\_0  
    ~WAIT\_ABANDONED\_0+nCount -1 (on mutex)
  - WAIT\_FAILED : the function failed

## 3.6 Waiting in a GUI Program

---

- The standard message loop in Windows program looks like this:

```
while(GetMessage(&msg, NULL,0, 0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

- `GetMessage()` is like a specialized version of `WaitForSingleObject()` that waits for message instead of waiting for kernel object

## 3.6 Waiting in a GUI Program

---

- GetMessage( ) is heart of Win16 cooperative multitasking
  - GetMessage() does not return until a message show up in program's queue.
  - Meanwhile, Windows is free to give the processor to another application.
- Problem
  - If you are waiting in a call to WaitForSingleObject() or WaitForMultipleObject(), there is no way you can return to message loop
- To slove this problem
  - Use a call named MsgWaitForMultipleObjects()

## 3.7 Summary

---

1. Looked at what signaled object
2. How to wait for one or several in a worker thread or in a GUI thread.
3. How to use the Performance Monitor in Windows NT