Multiple Looking-Up Processes

Writer Process

```
.....
down (m);
Write_item;
up (m);
……
```

Looking-up Process

```
….
down(s);
  r:=r+1;
  if r=1 then down(m);
up(s);
Look_up_item;
down(s);
  r:=r-1;
  if r=0 then up(m);
up(s);
……
```


Multiple Looking-Up Processes – Writer precedence

Writer Process

```
.....
down(readtry);
down (m);
Write_item;
up (m);
up(readtry);
……
```

Looking-up Process

```
….
down(readtry);
down(s);
  r:=r+1;
  if r=1 then down(m);
up(s);
Up(readtry);
Look_up_item;
down(s);
  r:=r-1;
  if r=0 then up(m);
up(s);
……
```

Windows Threads

In Windows, new threads are created using the *CreateThread* API function:

```
HANDLE  CreateThread (
  LPSECURITY_ATTRIBUTES   lpAttr,  ,      //Security Attributes
  DWORD   StackSize                       //Stack size
  LPTHREAD_START_ROUTINE   lpFunc,  ,     //Function name
  LPVOID   lpParam ,                      //Argument
  DWORD   Flags   ,                       //Creation Flags
  LPDWORD   lpThreadID   );               //Pointer to thread ID
```


Windows Threads

```
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

DWORD WINAPI Fn(LPVOID param)
{
  while(1){printf("The thread is running.\n\n");}
}

int main()
{
  DWORD ThreadID;
  char c;

  HANDLE ht = CreateThread(NULL,0,Fn,NULL,0,&ThreadID);

  while(c != 'e'){c=getche();}
}
```

Windows Threads

Thread has a base priority level determined by the priority class of its process as well as its relative priority.

Windows has six priority classes to which a process can belong: IDLE. BELOW NORMAL, NORMAL, ABOVE NORMAL, HIGH, and REALTIME. Priorities in all classes except REALTIME are variable.

The values for relative priorities of thread within class include: IDLE, LOWEST, BELOW NORMAL, NORMAL, ABOVE NORMAL, HIGHEST, TIME CRITICAL.

Threads are scheduled in a round-robin fashion at each priority level, and only when there are no executable threads at a higher level will scheduling of threads at a lower level take place.

Windows Threads

Thread Base Priority	Process Priority **Idle**	Process Priority **Below Normal**	Process Priority **Normal**	Process Priority **Above Normal**	Process Priority **High**	**Thread Base Priority**	Process Priority **Realtime**
0						**16**	Idle
1	Idle	Idle	Idle	Idle	Idle	**17**	
2	Lowest					**18**	
3	BelowNormal					**19**	
4	Normal	Lowest				**20**	
5	AboveNormal	BelowNormal				**21**	
6	Highest	Normal	Lowest			**22**	Lowest
7		AboveNormal	BelowNormal			**23**	BelowNormal
8		Highest	Normal	Lowest		**24**	Normal
9			AboveNormal	BelowNormal		**25**	AboveNormal
10			Highest	Normal		**26**	Highest
11				AboveNormal	Lowest	**27**	
12				Highest	BelowNormal	**28**	
13					Normal	**29**	
14					AboveNormal	**30**	
15	TimeCritical	TimeCritical	TimeCritical	TimeCritical	Highest TimeCritical	**31**	TimeCritical

Windows Semaphores

In Windows, semaphores are declared using the API function *CreateSemaphore()*.

HANDLE CreateSemaphore (SecurityAttributes,

InitialCount, MaxCount, SemaphoreID);

down and up operations are performed using the API functions WaitForSingleObject( ) and ReleaseSemaphore ( ).

Windows provides other synchronization objects such as mutexes and critical sections.

Windows Semaphores

```
DWORD WINAPI Fn1(LPVOID param)
{  int i;
   while(1){ gotoxy(10,10); i=i+1;
             printf("Thread 1 %d",i);
           }
}

DWORD WINAPI Fn2(LPVOID param)
{  int i;
   while(1){ gotoxy(10,20); i=i+1;
             printf("Thread 2 %d",i);
           }
}
```

Windows Semaphores

```
main( )
{
 DWORD ThreadID1,ThreadID2;
 char Sem; char c;

 hsem = CreateSemaphore(NULL,1,1,Sem);

 HANDLE ht1 = CreateThread(NULL,0,Fn1,NULL,0,&ThreadID1);
 HANDLE ht2 = CreateThread(NULL,0,Fn2,NULL,0,&ThreadID2);

 while(c != 'e') {c = getche();}
}
```