文章
姚 鑫 · 六月 29, 2022 阅读大约需 10 分钟

# 第十三章 信号（三）- 示例演示

# 第十三章 信号（三）- 示例演示

## 运行示例

Main、Producer 和 Consumer 这三个类中的每一个都有自己的 Run 方法，最好在各自的终端窗口中运行它们。每次运行时，它都会显示它为日志生成的消息。一旦用户通过提供它正在等待的输入来响应 Main 类，Main 的 Run 方法将终止删除信号量。然后，用户可以通过键入命令查看所有进程的合并日志文件的显示

```
  Do ##class(Semaphore.Util).ShowLog()
```

注意：以下所有示例都假定所有类都已在 “ USER ” 命名空间中编译。

### 示例 1 - 创建和删除信号量

最简单的例子演示了信号量的创建和销毁。它使用 Semaphore.Main 类。请执行下列操作：

1. 打开一个终端窗口。
2. 输入命令——

```
Do ##class(Semaphore.Main).Run()
```

3. 该方法创建信号量。如果成功，将看到消息 “ 输入任何字符以终止运行方法 ” 。按下 Enter 键。该方法显示信号量的初始化值，将其删除，然后退出。
4. 通过发出命令显示日志文件

```
Do ##class(Semaphore.Util).ShowLog()
```

按照上述步骤在终端窗口中显示的消息示例如下

```
??????

DHC-APP>Do ##class(Semaphore.Main).Run()
(1) Main Started
(2) New semaphore
(3) Created: "Counter"; Value = 0; Id = 0x0x10000
(4) Semaphore create result: 1
(5) Enter any character to terminate Run method
(6) Final value = 0
(7) Semaphore delete status: 1
(8) Main Finished
(9) Closing Semaphore: Id = 0x10000
```

```
?????????
DHC-APP>  Do ##class(Semaphore.Util).ShowLog()
Message Log: Entries = 9

#    $JOB   Sender       Message
1)   24888  Main:        Main Started
2)   24888  Counter:     New semaphore
3)   24888  Counter:     Created: "Counter"; Value = 0; Id = 0x0x10000
4)   24888  Main:        Semaphore create result: 1
5)   24888  Main:        Enter any character to terminate Run method
6)   24888  Main:        Final value = 0
7)   24888  Main:        Semaphore delete status: 1
8)   24888  Main:        Main Finished
9)   24888  Counter:     Closing Semaphore: Id = 0x10000
```

## 示例 2——创建信号量并连续递增它

这个例子展示了生产者在工作，以及从两个进程中捕获日志消息。
1. 打开两个单独的终端窗口。称它们为 “ A ” 和 “ B ” 。
2. 在窗口 A 中，键入以下命令，但不要在末尾键入 ENTER 键 -

```
Do ##class(Semaphore.Main).Run()
```

3. 在窗口 B 中，键入以下命令，但同样，不要在命令末尾键入 ENTER 键 -

```
Do ##class(Semaphore.Producer).Run()
```

4. 现在，在窗口 A 中，按 ENTER 键。然后在窗口 B 中，按 ENTER 键。这将启动两个类并行执行。他们各自的消息显示在他们自己的窗口中。
5. Producer 进程完成后，关闭 B 窗口。
6. 在 A 窗口中，按 ENTER 键以完成 Main 类。然后，使用以下命令显示日志 -

```
Do ##class(Semaphore.Util).ShowLog()
```

对于此示例，以下是输出

A 窗口

```
DHC-APP>Do ##class(Semaphore.Main).Run()
(1) Main Started
(2) New semaphore
(3) Created: "Counter"; Value = 0; Id = 0x0x20001
(4) Semaphore create result: 1
(5) Enter any character to terminate Run method
(17) Final value = 30
(18) Semaphore delete status: 1
(19) Main Finished
(20) Closing Semaphore: Id = 0x20001
```

B 窗口

```
DHC-APP>Do ##class(Semaphore.Producer).Run()
(6) Producer.1 Started
(7) New semaphore
(8) Open Id = 0x20001
(9) Increment 0x20001 = 0 by 5 wait 10 sec
(10) Increment 0x20001 = 5 by 5 wait 4 sec
(11) Increment 0x20001 = 10 by 3 wait 1 sec
(12) Increment 0x20001 = 13 by 5 wait 9 sec
(13) Increment 0x20001 = 18 by 5 wait 8 sec
(14) Increment 0x20001 = 23 by 4 wait 2 sec
(15) Increment 0x20001 = 27 by 1 wait 8 sec
(16) Increment 0x20001 = 28 by 2 wait 5 sec
(21) Producer.1 Finished
(22) Closing Semaphore: Id = 0x20001
```

日志显示

```
DHC-APP>Do ##class(Semaphore.Util).ShowLog()
Message Log: Entries = 22

#    $JOB   Sender       Message
1)   21080  Main:        Main Started
2)   21080  Counter:     New semaphore
3)   21080  Counter:     Created: "Counter"; Value = 0; Id = 0x0x20001
4)   21080  Main:        Semaphore create result: 1
5)   21080  Main:        Enter any character to terminate Run method
6)   27724  Producer.1:  Producer.1 Started
7)   27724  Counter:     New semaphore
8)   27724  Producer.1:  Open Id = 0x20001
9)   27724  Producer.1:  Increment 0x20001 = 0 by 5 wait 10 sec
10)  27724  Producer.1:  Increment 0x20001 = 5 by 5 wait 4 sec
11)  27724  Producer.1:  Increment 0x20001 = 10 by 3 wait 1 sec
12)  27724  Producer.1:  Increment 0x20001 = 13 by 5 wait 9 sec
13)  27724  Producer.1:  Increment 0x20001 = 18 by 5 wait 8 sec
14)  27724  Producer.1:  Increment 0x20001 = 23 by 4 wait 2 sec
15)  27724  Producer.1:  Increment 0x20001 = 27 by 1 wait 8 sec
16)  27724  Producer.1:  Increment 0x20001 = 28 by 2 wait 5 sec
17)  21080  Main:        Final value = 30
18)  21080  Main:        Semaphore delete status: 1
19)  21080  Main:        Main Finished
20)  21080  Counter:     Closing Semaphore: Id = 0x20001
21)  27724  Producer.1:  Producer.1 Finished
22)  27724  Counter:     Closing Semaphore: Id = 0x20001
```

## 示例 3 - 同时运行所有三个进程

此示例显示尝试以连贯的方式增加和减少相同的信号量。它使用所有三个主要类。

1. 打开三个单独的终端窗口。称它们为“ A ”、“ B ”和“ C ”。
2. 在窗口 A 中，键入以下命令，但最后不要按 ENTER 键

```
Do ##class(Semaphore.Main).Run()
```

3. 在窗口 B 中，键入以下命令，但同样，不要在命令末尾按 ENTER 键 -

```
Do ##class(Semaphore.Producer).Run()
```

4. 在窗口 C 中，键入以下命令，但同样，不要在命令末尾按 ENTER 键 -

```
Do ##class(Semaphore.Consumer).Run()
```

5. 从窗口 A 开始，访问每个窗口并键入 ENTER 键。这将启动 Main 类，然后是其他两个类。如前所述，每个进程都会在自己的窗口中显示其日志消息。
6. 当Producer 和Consumer 进程都完成后，关闭B 窗口和C 窗口。
7. 在 A 窗口中，按 ENTER 键以完成 Main 类。然后，使用以下命令显示日志

```
Do ##class(Semaphore.Util).ShowLog()
```

运行此示例会产生类似于以下内容的输出：

窗口 A

```
DHC-APP>Do ##class(Semaphore.Main).Run()
(1) Main Started
(2) New semaphore
(3) Created: "Counter"; Value = 0; Id = 0x0x40003
(4) Semaphore create result: 1
(5) Enter any character to terminate Run method
<ENTER>
(64) Final value = 0
(65) Semaphore delete status: 1
(66) Main Finished
(67) Closing Semaphore: Id = 0x40003
```

窗口B

```
DHC-APP>Do ##class(Semaphore.Producer).Run()
(6) Producer.1 Started
(7) New semaphore
(8) Open Id = 0x40003
(9) Increment 0x40003 = 0 by 5 wait 8 sec
(20) Increment 0x40003 = 0 by 4 wait 4 sec
(25) Increment 0x40003 = 0 by 3 wait 1 sec
(29) Increment 0x40003 = 0 by 2 wait 10 sec
(36) Increment 0x40003 = 0 by 4 wait 3 sec
(40) Increment 0x40003 = 0 by 5 wait 5 sec
(52) Increment 0x40003 = 0 by 5 wait 6 sec
(58) Increment 0x40003 = 0 by 2 wait 2 sec
(62) Producer.1 Finished
(63) Closing Semaphore: Id = 0x40003
```

窗口C

```
DHC-APP>Do ##class(Semaphore.Consumer).Run()
(10) Consumer.1 Started
(11) New semaphore
```

```
(12) Consumer: Open Id = 0x40003
(13) Decrement 0x40003 = 5 by 1 wait 10 sec
(14) WaitCompleted: 0x40003; Amt = 1
(15) Granted
(16) Decrement 0x40003 = 4 by 5 wait 2 sec
(17) WaitCompleted: 0x40003; Amt = 4
(18) Granted
(19) Decrement 0x40003 = 0 by 5 wait 8 sec
(21) WaitCompleted: 0x40003; Amt = 4
(22) Granted
(23) Decrement 0x40003 = 0 by 5 wait 6 sec
(25) WaitCompleted: 0x40003; Amt = 3
(26) Granted
(27) Decrement 0x40003 = 0 by 3 wait 1 sec
(28) Timeout
(30) Decrement 0x40003 = 0 by 4 wait 4 sec
(31) WaitCompleted: 0x40003; Amt = 2
(32) Granted
(33) Decrement 0x40003 = 0 by 2 wait 7 sec
(34) Timeout
(35) Decrement 0x40003 = 0 by 4 wait 9 sec
(37) WaitCompleted: 0x40003; Amt = 4
(38) Granted
(39) Decrement 0x40003 = 0 by 2 wait 5 sec
(41) WaitCompleted: 0x40003; Amt = 2
(42) Granted
(43) Decrement 0x40003 = 3 by 1 wait 3 sec
(44) WaitCompleted: 0x40003; Amt = 1
(45) Granted
(46) Decrement 0x40003 = 2 by 2 wait 10 sec
(47) WaitCompleted: 0x40003; Amt = 2
(48) Granted
(49) Decrement 0x40003 = 0 by 2 wait 4 sec
(50) Timeout
(51) Decrement 0x40003 = 0 by 3 wait 4 sec
(53) WaitCompleted: 0x40003; Amt = 5
(54) Granted
(55) Decrement 0x40003 = 0 by 1 wait 1 sec
(56) Timeout
(57) Decrement 0x40003 = 0 by 3 wait 7 sec
(59) WaitCompleted: 0x40003; Amt = 2
(60) Granted
(61) Consumer.1 Finished
```

日志显示

```
DHC-APP>Do ##class(Semaphore.Util).ShowLog()
Message Log: Entries = 67

#    $JOB   Sender       Message
1)   6412   Main:        Main Started
2)   6412   Counter:     New semaphore
3)   6412   Counter:     Created: "Counter"; Value = 0; Id = 0x0x40003
4)   6412   Main:        Semaphore create result: 1
5)   6412   Main:        Enter any character to terminate Run method
6)   22236  Producer.1:  Producer.1 Started
```

```
7)   22236  Counter:     New semaphore
8)   22236  Producer.1:  Open Id = 0x40003
9)   22236  Producer.1:  Increment 0x40003 = 0 by 5 wait 8 sec
10)  20224  Consumer.1:  Consumer.1 Started
11)  20224  Counter:     New semaphore
12)  20224  Consumer.1:  Consumer: Open Id = 0x40003
13)  20224  Consumer.1:  Decrement 0x40003 = 5 by 1 wait 10 sec
14)  20224  Counter:     WaitCompleted: 0x40003; Amt = 1
15)  20224  Consumer.1:  Granted
16)  20224  Consumer.1:  Decrement 0x40003 = 4 by 5 wait 2 sec
17)  20224  Counter:     WaitCompleted: 0x40003; Amt = 4
18)  20224  Consumer.1:  Granted
19)  20224  Consumer.1:  Decrement 0x40003 = 0 by 5 wait 8 sec
20)  22236  Producer.1:  Increment 0x40003 = 0 by 4 wait 4 sec
21)  20224  Counter:     WaitCompleted: 0x40003; Amt = 4
22)  20224  Consumer.1:  Granted
23)  20224  Consumer.1:  Decrement 0x40003 = 0 by 5 wait 6 sec
24)  22236  Producer.1:  Increment 0x40003 = 0 by 3 wait 1 sec
25)  20224  Counter:     WaitCompleted: 0x40003; Amt = 3
26)  20224  Consumer.1:  Granted
27)  20224  Consumer.1:  Decrement 0x40003 = 0 by 3 wait 1 sec
28)  20224  Consumer.1:  Timeout
29)  22236  Producer.1:  Increment 0x40003 = 0 by 2 wait 10 sec
30)  20224  Consumer.1:  Decrement 0x40003 = 0 by 4 wait 4 sec
31)  20224  Counter:     WaitCompleted: 0x40003; Amt = 2
32)  20224  Consumer.1:  Granted
33)  20224  Consumer.1:  Decrement 0x40003 = 0 by 2 wait 7 sec
34)  20224  Consumer.1:  Timeout
35)  20224  Consumer.1:  Decrement 0x40003 = 0 by 4 wait 9 sec
36)  22236  Producer.1:  Increment 0x40003 = 0 by 4 wait 3 sec
37)  20224  Counter:     WaitCompleted: 0x40003; Amt = 4
38)  20224  Consumer.1:  Granted
39)  20224  Consumer.1:  Decrement 0x40003 = 0 by 2 wait 5 sec
40)  22236  Producer.1:  Increment 0x40003 = 0 by 5 wait 5 sec
41)  20224  Counter:     WaitCompleted: 0x40003; Amt = 2
42)  20224  Consumer.1:  Granted
43)  20224  Consumer.1:  Decrement 0x40003 = 3 by 1 wait 3 sec
44)  20224  Counter:     WaitCompleted: 0x40003; Amt = 1
45)  20224  Consumer.1:  Granted
46)  20224  Consumer.1:  Decrement 0x40003 = 2 by 2 wait 10 sec
47)  20224  Counter:     WaitCompleted: 0x40003; Amt = 2
48)  20224  Consumer.1:  Granted
49)  20224  Consumer.1:  Decrement 0x40003 = 0 by 2 wait 4 sec
50)  20224  Consumer.1:  Timeout
51)  20224  Consumer.1:  Decrement 0x40003 = 0 by 3 wait 4 sec
52)  22236  Producer.1:  Increment 0x40003 = 0 by 5 wait 6 sec
53)  20224  Counter:     WaitCompleted: 0x40003; Amt = 5
54)  20224  Consumer.1:  Granted
55)  20224  Consumer.1:  Decrement 0x40003 = 0 by 1 wait 1 sec
56)  20224  Consumer.1:  Timeout
57)  20224  Consumer.1:  Decrement 0x40003 = 0 by 3 wait 7 sec
58)  22236  Producer.1:  Increment 0x40003 = 0 by 2 wait 2 sec
59)  20224  Counter:     WaitCompleted: 0x40003; Amt = 2
60)  20224  Consumer.1:  Granted
61)  20224  Consumer.1:  Consumer.1 Finished
62)  22236  Producer.1:  Producer.1 Finished
63)  22236  Counter:     Closing Semaphore: Id = 0x40003
64)  6412   Main:        Final value = 0
65)  6412   Main:        Semaphore delete status: 1
```

```
66)  6412   Main:        Main Finished
67)  6412   Counter:     Closing Semaphore: Id = 0x40003
```

## 其他变量

此示例的其他变量是可能的。虽然一次只能运行一个 Semaphore.Main，但在其他窗口中执行的生产者或消费者的数量没有限制。鼓励用户在各种场景中尝试不同数量的消费者和生产者，例如

- 运行三个消费者和一个生产者，这样信号量就会有更大的“ 竞争 ”。运气好的话，日志会显示两个或多个消费者发出了减少信号量的请求，并且都成功了，因为信号量值大到足以满足两个请求的部分或全部。
- 还可以使用这些类来演示删除信号量时其他进程中发生的情况。为此，在 Producers 或 Consumers 运行时，切换到 Main 类正在运行的窗口，然后按 ENTER。在完成处理过程中，Main 类将删除信号量，Producer 或 Consumer 的 OREF 将不再有效。下次尝试使用将产生错误。
- 通过将信号量的名称更改为看起来像全局名称的名称，可以将信号量映射到例如 ECP 系统上的不同实例。

#SQL #Caché

---

**源 URL:** https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%B8%89%E7%AB%A0-%E4%BF%A1%E5%8F%B7%EF%BC%88%E4%B8%89%EF%BC%89-%E7%A4%BA%E4%BE%8B%E6%BC%94%E7%A4%BA