Subject: Practice on Operating System Lecture Practice Thread Java
From: IKENOYA Katsutoshi <j05002@ie.u-ryukyu.ac.jp>
Date: Wed, 07 Feb 2007 17:21:57 +0900
To: Shinji KONO <kono@ie.u-ryukyu.ac.jp>

学籍番号：055702B

・修正点

３.スケジューリングの観察
の考察(Java API documentの記述の引用)
を追加しました。

・開発環境
OS : Mac OS X 10.4.8
コンパイラ :gcc version 4.0.1 (Apple Computer, Inc. build 5250)
java version "1.5.0_06"

・実行環境
同上

１.プロセスのコンテキスト切り替えの観察

・ソース

ソースの場所：/home/y05/j05002/OS/report5/processExample/SimpleThread.java

```
package processExample;

public class SimpleThread extends Thread {
省略

public void run() {
while(count < 0 || count-- > 0) {
String msg = "Thread "+name+" "+Integer.toString(a)+" "+Integer.toString(b)
+" "+Integer.toString(c);
System.out.println(msg);
yield();
}
}
}
```

・実行結果

(1)yield()無しの場合

```
[j05002@OS-report5]% java processExample/ThreadCreate
Thread t1 created.
Thread t2 created.
Thread t3 created.
Thread t1 1 2 3
Thread t1 1 2 3
Thread t2 4 5 6
Thread t2 4 5 6
Thread t3 7 8 9
Thread t3 7 8 9
```

(2)yield()有りの場合

```
[j05002@OS-report5]% java processExample/ThreadCreate
```

```
Thread t1 created.
Thread t2 created.
Thread t3 created.
Thread t1 1 2 3
Thread t2 4 5 6
Thread t1 1 2 3
Thread t3 7 8 9
Thread t2 4 5 6
Thread t3 7 8 9
```

yield()によって軽量プロセスの実行順序が変化していることがわかる。

２.スレッドを止める方法

count に-1を指定し、何回ループが実行されるかを調べた。

・実行結果

```
[j05002@OS-report5]% java processExample/ThreadStop
Thread t1 created.
Thread t2 created.
Thread t3 created.
Thread t1 1 2 3
t1:loop count is 1
Thread t1 1 2 3
t1:loop count is 2
Thread t1 1 2 3
t1:loop count is 3
省略
t1:loop count is 116
Thread t1 1 2 3
t1:loop count is 117
Thread t2 4 5 6
t2:loop count is 1
Thread t2 4 5 6
t2:loop count is 2
Thread t2 4 5 6
省略
t2:loop count is 314
Thread t2 4 5 6
t2:loop count is 315
Thread t2 4 5 6
t2:loop count is 316
Thread t3 7 8 9
t3:loop count is 1
Thread t3 7 8 9
t3:loop count is 2
Thread t3 7 8 9
省略
t3:loop count is 77
Thread t1 1 2 3
t1:loop count is 119
Thread t1 1 2 3
t1:loop count is 120
Thread t1 1 2 3
省略
Thread t1 1 2 3
t1:loop count is 400
Thread t1 1 2 3
t1:loop count is 401
Thread t3 7 8 9
t3:loop count is 78
Thread t3 7 8 9
```

```
t3:loop count is 79
Thread t3 7 8 9
t3:loop count is 80
Thread t3 7 8 9
省略
t3:loop count is 583
Thread t3 7 8 9
t3:loop count is 584
Thread t3 7 8 9
t3:loop count is 585
Thread t3 7 8 9
t3:loop count is 586
```

実行結果よりt1は401回、t2は316回、t3は586回ループを実行したことが分かる。

次に、stop() を使わないスレッドの停止方法を実装してみた。

・ソース

ソースの場所：
/home/y05/j05002/OS/report5/processExample/SimpleSaftThread.java
/home/y05/j05002/OS/report5/processExample/ThreadSafeStop.java

- SimpleSaftThread.java

```
package processExample;

public class SimpleSaftThread extends Thread {


省略

public void run() {

while((count < 0 || count-- > 0) && running) {
String msg = "Thread "+name+" "+Integer.toString(a)+" "+Integer.toString(b)
+" "+Integer.toString(c);
System.out.println(msg);
yield();

}
}

public void SafeStop(){
running = false;
}

}
```

- ThreadSafeStop.java

```
package processExample;

public class ThreadSafeStop extends Thread {

省略

public synchronized void sync() throws InterruptedException {
wait(120);

t1.SafeStop();
t2.SafeStop();
t3.SafeStop();
```

```
}
}
```

・実行結果

```
[j05002@OS-report5]% java processExample/ThreadSafeStop
Thread t1 created.
Thread t2 created.
Thread t3 created.
Thread t1 1 2 3
Thread t2 4 5 6
Thread t1 1 2 3
Thread t2 4 5 6
Thread t1 1 2 3
省略
Thread t2 4 5 6
Thread t3 7 8 9
Thread t2 4 5 6
Thread t3 7 8 9
Thread t3 7 8 9
[j05002@OS-report5]%
```

３.スケジューリングの観察

```
t1.setPriority(3);
t2.setPriority(1);
t3.setPriority(2);
```

とした場合

・実行結果

```
[j05002@OS-report5]% java processExample/ThreadCreate
Thread t1 created.
Thread t2 created.
Thread t3 created.
Thread t1 1 2 3
Thread t1 1 2 3
Thread t3 7 8 9
Thread t3 7 8 9
Thread t2 4 5 6
Thread t2 4 5 6
```

Priorityが高いものから実行されているのが分かる。
またJava API Documentの
java.lang.Object
|_java.lang.Thread
のsetPriorityメソッドを見ると、
優先順位の範囲が、
MIN_PRIORITY ～ MAX_PRIORITY
とあった。
java.lang.Threadの定数フィールド値を見てみると
MAX_PRIORITY 10
MIN_PRIORITY 1
とあったので、Priorityの値の範囲は、
1～10でなければならない。