BBj Memory And Process Management

By Jim Douglas

Traditional BBx® memory management is based on the concept of a fixed-size workspace. Programs and data are allocated within a workspace defined by the -m command line parameter or a START command. For example:

```
pro5 -m1024 menu.bbx
```

This command starts a new PRO/5® process with an initial workspace size of 1024 pages (256 kilobytes), then runs menu.bbx, allocating any required data within that workspace. If there are 30 PRO/5 processes running at the same time, they are independent of each other; nothing is shared.

BBj memory management is both simpler and more complicated. The -m parameter is not needed when starting interpreter sessions, so this is the BBj® version of the above command:

```
bbj menu.bbx
```

This command starts a new BBj client process, then runs menu.bbx. The BBj client process is nothing more than a tiny loader. To understand what happens next, we need to look at BBjServices.

BBjServices

BBj Interpreter sessions run as threads within the BBjServices process. The default memory allocated to BBjServices is 64 megabytes. This setting might need to be increased to allow for many concurrent clients, or for applications that use a large amount of memory.

This program can be used to determine the current memory usage of BBjServices:

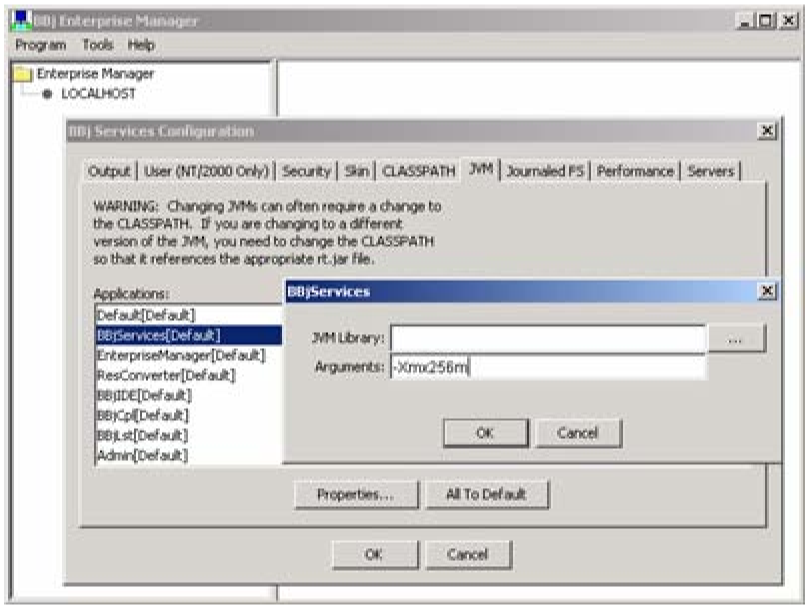
```
0010 GOSUB memory
0020 java.lang.System.gc(); java.lang.System.runFinalization()
0030 java.lang.System.gc(); REM ' For demonstration purposes only!!!
0040 GOSUB memory
0050 STOP
0060 memory:
0070 LET free=java.lang.Runtime.getRuntime().freeMemory()
0080 LET total=java.lang.Runtime.getRuntime().totalMemory()
0090 LET adjustment=64*1024*1024; REM ' maxMemory is -Xmx + 64MB
0100 LET max=java.lang.Runtime.getRuntime().maxMemory()-adjustment
0110 LET used=total-free
0120 PRINT "Used:",used*100/max,"% of",max/1024/1024," MB"
0130 RETURN
0140 END
>run
Used: 19.58% of 64 MB
Used: 7.81% of 64 MB
```

This example displays the amount of memory that is currently in use as a percentage of the maximum memory available to BBjServices. Notice the effect of garbage collection. The first line shows almost 20% of the maximum memory in use. After forcing a garbage collection, memory usage immediately drops to less than 8%. In a production environment, it is never necessary or appropriate to force garbage collection. Java invokes garbage collection as required for efficient memory management; we only force a garbage collection cycle here to get a more accurate report of memory usage.

If BBjServices begins to refuse client connections due to insufficient memory, or if one or more programs regularly report !ERROR=33, then the amount of memory allocated to BBjServices should be increased. To increase BBjServices memory to 256 megabytes, follow these steps:

1. Run Enterprise Manager
2. From the menu system, select Tools, then Configure Local Machine
3. Click on the JVM tab
4. Double-click on the BBjServices line
5. Enter a memory parameter in the Arguments field using Java format: -Xmx256m
6. Click OK to accept the update, close Enterprise Manager, and restart BBjServices for the change to take effect

Increase BBjServices memory to 256 megabytes.



WARNING: Changing JVMs can often require a change to the CLASSPATH. If you are changing to a different version of the JVM, you need to change the CLASSPATH so that it references the appropriate rt.jar file.

Applications:
Default[Default]
BBjServices[Default]
EnterpriseManager[Default]
ResConverter[Default]
BBjIDE[Default]
BBjCpl[Default]
BBjLst[Default]
Admin[Default]

JVM Library:
Arguments: -Xmx256m

To change the amount of memory allocated to thin client sessions, follow the same steps, but select the Default line instead of the BBjServices line.

Program Cache

BBjServices automatically caches the most recently used programs and resource files system-wide. This cache can improve performance significantly, at the cost of increased memory usage.

The cache is configurable and separate from any programs that might be cached using the ADDR verb.

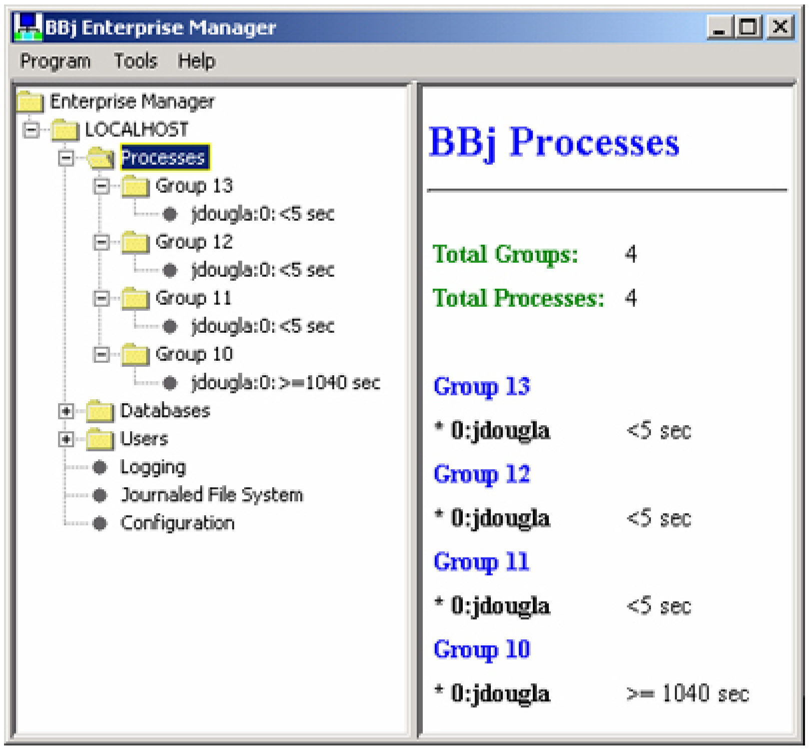
BBj Process Management

Because interpreter sessions run as individual threads within BBjServices, the Enterprise Manager includes a utility that is similar to the Windows "Task Manager" application. This utility can be used to monitor individual threads and, in rare cases, to kill runaway processes. For example, the following BBj program creates an infinite loop that can only be interrupted by killing the process in Enterprise Manager.

```
0010 SETESC 0010; WHILE 1; WEND
```

To kill a runaway process, select the Processes node in Enterprise Manager and look for the task that has been unresponsive for a very long time. In the following example, the process associated with Group 10 has been unresponsive for several minutes:

The process associated with Group 10 has been unresponsive for several minutes.


BBj Enterprise Manager

Program Tools Help

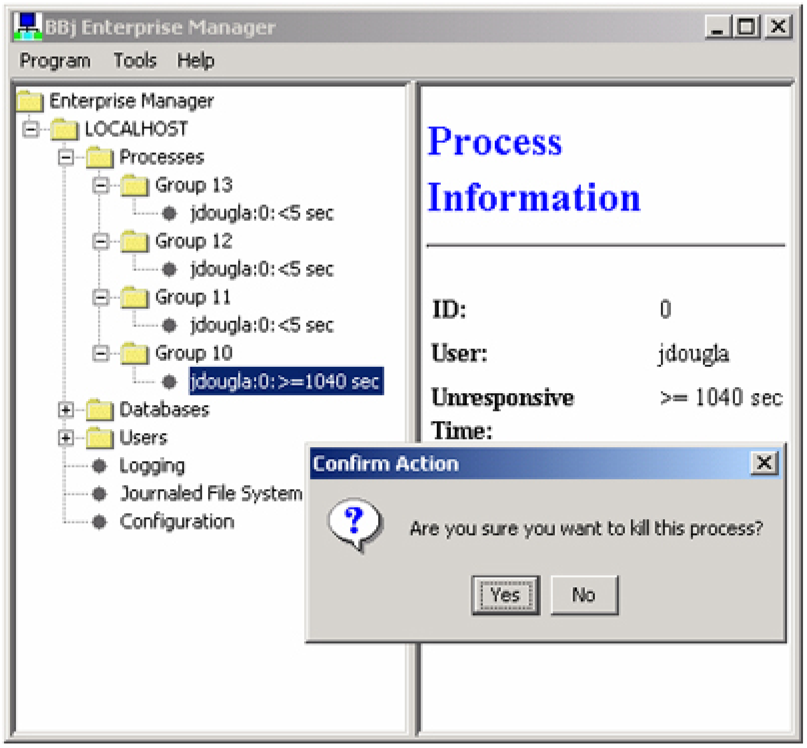
- Enterprise Manager
  - LOCALHOST
    - Processes
      - Group 13
        - jdougla:0:<5 sec
      - Group 12
        - jdougla:0:<5 sec
      - Group 11
        - jdougla:0:<5 sec
      - Group 10
        - jdougla:0:>=1040 sec
    - Databases
    - Users
    - Logging
    - Journaled File System
    - Configuration

BBj Processes

Total Groups:	4
Total Processes:	4
Group 13	
* 0:jdougla	<5 sec
Group 12	
* 0:jdougla	<5 sec
Group 11	
* 0:jdougla	<5 sec
Group 10	
* 0:jdougla	>= 1040 sec

To kill that process, right click on it and click the "Kill Process" button, then confirm that you want to kill the process:

Right click on the process to kill it, then confirm you want to kill the process.


BBj Enterprise Manager

Program Tools Help

- Enterprise Manager
  - LOCALHOST
    - Processes
      - Group 13
        - jdougla:0:<5 sec
      - Group 12
        - jdougla:0:<5 sec
      - Group 11
        - jdougla:0:<5 sec
      - Group 10
        - jdougla:0:>=1040 sec
    - Databases
    - Users
    - Logging
    - Journaled File System
    - Configuration

Process Information

ID:	0
User:	jdougla
Unresponsive Time:	>= 1040 sec

Confirm Action

Are you sure you want to kill this process?

This capability is intended for extreme cases; it's not the recommended way of terminating a BBj session.

Summary

The most important point to remember about BBj memory management is that BBj operates as a tightly integrated system. Most of the work is handled by BBjServices within a single Java Virtual Machine. This design means that memory requirements are set for the system as a whole, not on a per-task basis. It also means that BBj is able to cache and share programs on a system-wide basis - a significant benefit in a typical multi-user environment.