1106 Memory Management Guide

Revison: 3.1 *Last Date Modified: 04-October-2002*

This Technical Support Guide is intended to help you understand and prevent 'Out of Memory' errors in MATLAB.

Section 1: How does an operating system manage memory?

A computer's memory must accommodate both the Operating System's processes and user processes. The following outline describes what happens when a user starts a program such as MATLAB.

1. User starts the program.

2. The operating system creates an address space for the program to run in, which involves setting up approprate page tables.

3. The program's text and data segments are mapped from the executable file on disk to the virtual address space, the former being mapped as read-only,

the latter being mapped as read-write.

4. The stack and heap are initialized.

5. Control is transferred to the program.

6. The CPU tries to access the first instruction. Since it is not in memory, a page fault is generated. This causes the operating system to intervene and allocate one or more pages of physical memory and to copy the code in from the file, hence the name "Demand Paged Virtual Memory."

7. The CPU then returns control to the program, which continues executing until it encounters another page fault. At this point, step 6 is repeated. If there is no free physical memory, the operating system will have to either discard (if read-only) or swap out (write data to swap file if read-write) pages before they can be reused. Typically an LRU (least recently used) algorithm is used for deciding which page(s) get tossed.

8. This pattern continues for the life of the program.

For more details on this subject, please read the remainder of this section. Key words are in **bold** for easy reading.

When a user executes a program, the operating system creates an **address space** for it to run in. This address space will include the instructions for the program itself, as well as any data it requires. In a system with memory protection, each **process** is restricted by the operating system to accessing only the memory in its own address space. However, the combined program memory requirements often exceed the system's amount of **physical memory (RAM)** installed on a computer. So, modern operating system’s use a portion of the hard disk called a **swap file** to extend the amount of available memory. This technique, called **virtual memory**, treats physical memory as a cache of the most recently used data. In a virtual memory system, memory is divided into units called **pages** (a page is typically 4-8Kb in size). The set of addresses that identify locations in virtual memory is called the **virtual address space**. Each process is allocated a virtual address space. The virtual address space can range from 0-4GB on a 32-bit architecture. Figure 1 illustrates the virtual memory

addressing scheme.

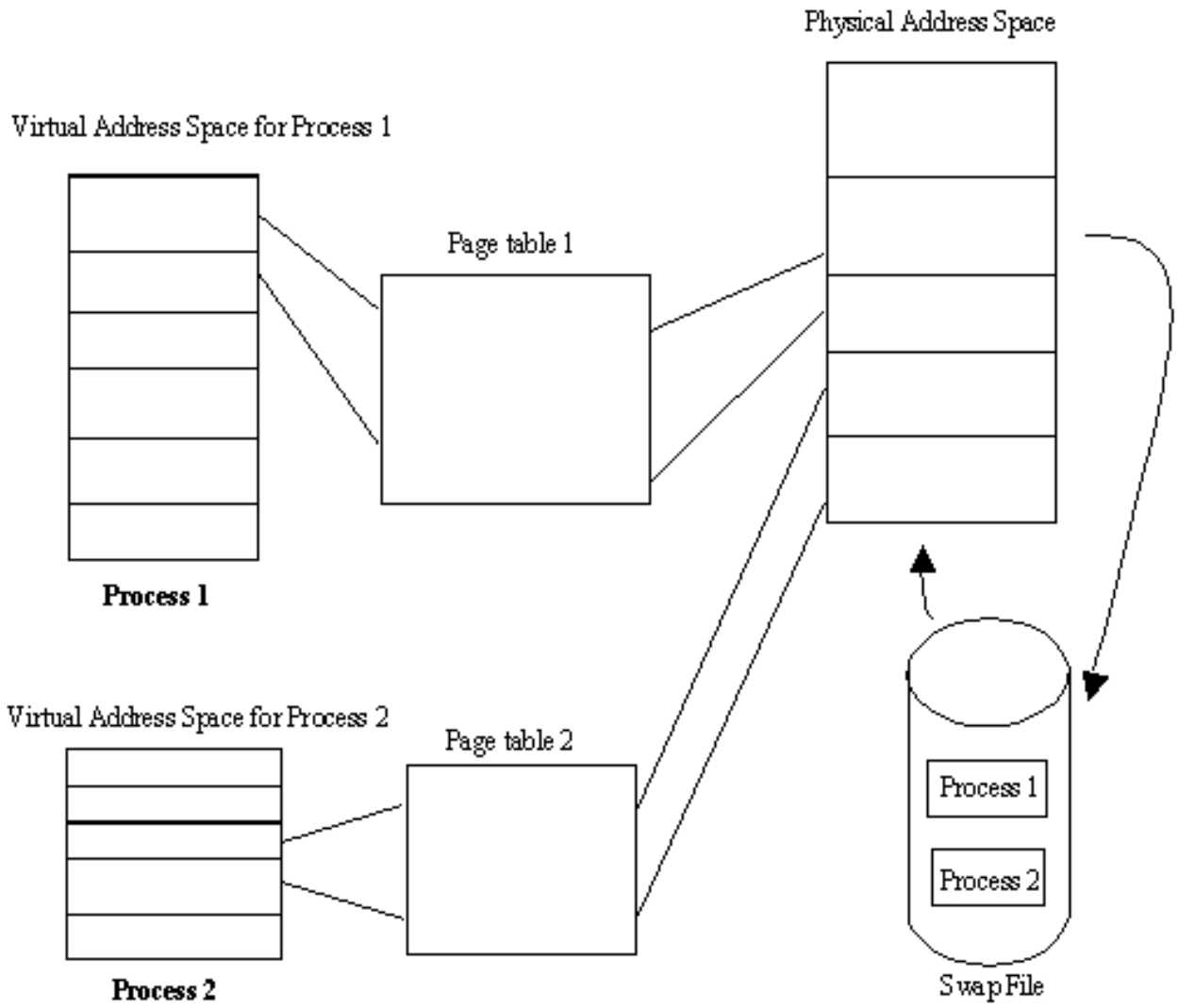

Figure 1. Virtual memory addressing

A process's address space contains the set of instructions and data that is mapped into virtual memory when a program is executed. Virtual memory addresses are translated into physical memory addresses through the use of a look-up table, called a **page table**. In addition to mapping the entire process into virtual memory, a subset of pages is also mapped into physical memory. As each instruction in the process executes, it is either found in physical memory or is not found (called a **page fault**). When a page fault occurs, the page that is needed must be moved from the hard disk into physical memory before the instruction can be executed. To make room for the new page, the operating system may need to decide which page to move out of physical memory. This is called **swapping**. A page fault is time-consuming because retrieving data from the hard

disk is orders of magnitude slower than obtaining it directly from the physical memory. Operating systems attempt to minimize the number of page faults by swapping multiple pages at once. This becomes a trade-off between operating system overhead and the time saved by minimizing page faults, which is affected by the size of each process's working set.

The operating system takes care of swapping and translating from virtual address space to physical address space. This means that the developer has a flat address space at his disposal. With a virtual memory scheme, the amount of memory available to a developer is seemingly limited only by the amount of hard drive space. In essence, virtual memory has much more memory available than physical memory alone because it removes the restriction that an entire process must be loaded into memory at one time.

Section 2: How do I set the swap space for my operating system?

How you set the swap space for your computer depends on the operating system that you are running on.

On UNIX, information about swap space can be procured by typing `pstat -s` at the UNIX command prompt. For detailed information on changing swap space, please ask your system administrator.

On Linux, swap space can be changed by using the `mkswap` and `swapon` commands. For more information on the above commands, type `man command_name` at the Linux prompt.

In Windows, you can set the amount of virtual memory available using the steps outlined below.

For Windows NT:

1. Right-click on ‘My Computer’ icon and select ‘Properties’.
2. Select the ‘Performance’ tab and click on the ‘Change’ button to increase the virtual memory.

For Windows 2000:

1. Right-click on ‘My Computer’ icon and select ‘Properties’.
2. Select the ‘Advanced’ tab and choose ‘Performance Options’.
3. Click on the Change button to increase the virtual memory.

For Windows 95, 98, and Me:

1. Right-click on ‘My Computer’ icon and select ‘Properties’.
2. Select the ‘Performance’ tab and click the ‘Change’ button to modify the virtual memory.

Section 3: Why do I get ‘Out of Memory’ errors in MATLAB?

From Section 1 it appears that virtual memory is seemingly unlimited. So why do people get out of memory errors in MATLAB? Most systems today are 32-bit architectures, meaning that the length of an instruction or number that can be sent to the processor can be at most 32 bits long, which implies that the memory addresses can be a maximum of 32 bits long. This results in a maximum of 2^{32} possible memory addresses, which is equivalent to 4 gigabytes (GB) of addressable memory. Therefore, under 32 bit architectures, the seemingly unlimited virtual memory space is actually limited to approximately 4 GB.

The Windows operating system further reduces the theoretical limit down to 2 GB of virtual memory due to a design decision to reserve the upper 2 GB of address space. On machines running Windows 2000 Advanced Server or Windows 2000 Datacenter Server, the amount of virtual memory space reserved by the operating system can be reduced by using the /3GB switch in the boot.ini file. More documentation on this option can be found at the following URL:

http://support.microsoft.com/support/kb/articles/Q291/9/88.ASP

Most 32-bit UNIX operating systems also reserve the upper 2 GB of address space, thereby limiting the virtual memory available to 2 GB. If you need to find out exactly how your operating system reserves virtual memory, we suggest you contact the operating system's vendor.

When you launch MATLAB, approximately 0.5 GB of virtual address space is used to load the heap, stack, DLLs, and other operating system services. For Windows, this limits the net available virtual address space to approximately 1.5 GB to store variables and other data needed by MATLAB.

In MATLAB 6.x, a Java user interface was added using the Java Virtual Machine (JVM). The JVM consumes additional memory, so the net memory available to the user is further reduced in MATLAB 6.x. As described in the next section, this user interface can be disabled to help avoid 'Out of Memory' errors.

The second major cause of 'Out of Memory' errors is fragmentation. When virtual memory is used and freed during normal operation of MATLAB, memory becomes fragmented. This means that the amount of total free memory may be greater than the amount of contiguous free memory. Since matrices in MATLAB must be stored in a contiguous block of virtual memory addresses, the largest matrix that can be created at a particular point in time is limited by the amount of contiguous free virtual address space.

You can see the memory status of any Windows system along with the largest contiguous block of memory available with one of the following methods.

For MATLAB 6.0 (R12) and 6.1 (R12.1):

1. Download memdump.dll.
2. Place it on your MATLAB path.
3. Type `memdump` at the MATLAB prompt.

For MATLAB 6.5 (R13):

This functionality is built-in to MATLAB R13. You can use the following command:

```
system_dependent DumpMem
```

to see the memory status.

NOTE: For information regarding the output of the above commands, please see Solution 31706: What additional information is available about the memdump.dll file included with Technical Note 1106?

Section 4: How do I avoid 'Out of Memory' errors in MATLAB?

There are several tactics that you can apply to optimize the use of memory and to avoid problems caused by fragmentation.

1. Pre-allocate your variables using the ZEROS function or other similar functions. Each variable in MATLAB must be stored in one contiguous block of memory addresses. If memory for a large variable is not allocated at the beginning of the program, then the memory around it may be allocated, restricting its space to expand and resulting in out of memory errors. Please refer to Solution 26623 for more information on pre-allocating large variables.

2. Set the swap space on your system to the maximum that can be utilized by your operating system. For more information on setting swap space, please refer to Section 2 of this document.

3. Use the CLEAR, PACK, and QUIT commands to keep memory as free and un-fragmented as possible. Section 5 of this document provides specific instructions on using these commands.

4. If possible, break large matrices into several smaller matrices so that less memory is used at any one time.

5. Run memory intensive tasks in “-nojvm” mode. In MATLAB 6.x (R12.x), we adopted a Java interface to MATLAB that makes many tasks easier and more intuitive. However, the Java Virtual Machine (JVM) creates Java class objects and loads additional DLL files that consume a significant amount of virtual memory. In addition, MATLAB memory management commands such as PACK are unable to compress memory locations occupied by the JVM. In essence, the JVM and MATLAB compete for the same virtual address space. Running MATLAB in “-nojvm” mode disables the JVM and allows MATLAB to have more address space available. This means that the new features introduced with the JVM cannot be used in this mode. We suggest doing development in normal mode and using “-nojvm” mode for processing tasks that produce ‘Out of Memory’ errors in normal mode.

   To run in “-nojvm” mode on Windows:

   A. Right-click on the shortcut that you use to start MATLAB and go to “Properties.”

   B. In the target path, append “–nojvm” after $MATLAB/bin/win32/matlab.exe (where $MATLAB is

your root MATLAB directory).

C. Click “OK” and start MATLAB using this shortcut.

For UNIX or Linux, append “ –nojvm” to the command that you use to start MATLAB.

This runs MATLAB without the Java interface and reduces the competition for virtual address space.

Section 5: How do I defragment the MATLAB workspace memory?

There are five functions that you can use to free and defragment MATLAB workspace memory. They are listed in the table below. Click on the each link to see the documentation for that command.

clear	Removes variables from memory
pack	Writes existing variables off to disk, then reloads them contiguously
quit	Exits MATLAB and returns all allocated memory to the system
save	Selectively stores variables to disk
load	Reloads a data file saved with the SAVE command

You can use the SAVE and LOAD commands in conjunction with the QUIT command to free memory by:

- Saving any needed variables with the SAVE command
- Quitting MATLAB to free all memory allocated to MATLAB
- Starting a new MATLAB session and loading the saved variables back into the clean MATLAB workspace

For additional information on the five commands mentioned above, please type `help <functionname>` in the MATLAB command window where

<functionname> refers to one of the five commands.

NOTE: There are some limitations to these commands. MATLAB relies on C runtime library calls (`malloc`, `calloc`, and `free`) to allocate and de-allocate memory. These commands rely on the operating system to properly manage memory. Depending on the operating system, these commands may be handled differently. Also, these commands do not affect the Java class objects created by the Java Virtual Machine (JVM) used in MATLAB 6.x.

Did this information help? Yes ○ No ○ Didn't Apply ○

Is the level of technical detail appropriate? Yes ○ Too Much ○ Not Enough ○

What did you expect to find on this page, that you want us to consider adding?

Additional Comments:

SUBMIT

---

related topics:

Demos | Search | Contact Support | Consulting | News & Notes | Usability