Virtual Memory

Documentation

mchugh@hamlet.uncg.edu

Amiga Magazine
Issue #1, 1993
Disk B

Introduction

DISCLAIMER:

BEFORE I CONTINUE ANY FURTHER I WOULD LIKE TO STRESS THE FACT THAT I AM NOT THE AUTHOR OF THE VIRTUAL MEMORY PROGRAM, NOR DO I CLAIM TO BE. I CLAIM NO RESPONSIBILITY FOR ANY DATA LOSS OR OTHER NON DESIRABLES THAT MAY OCCUR BY USING THE SOFTWARE. ALSO, THE TEXT FILE INCLUDED WITH THE SOFTWARE WAS WRITTEN IN GERMAN AND I AM SURE YOU WILL FIND OUT THAT MY GERMAN IS TERRIBLE SO I APOLOGIZE FOR ANY MISTAKES IN ADVANCE.

PURPOSE:

This document is designed for the non German speaker to install and use the Virtual Memory program. The Virtual Memory program referred to in this manual can be found on the German "Amiga Magazin" cover disk 'B' for issue #1 in 1993. An easier method to access this program would be to download the cover disk from an Aminet site. It has the file name:

1. **/misc/amag/Amiga_9301b.lha**
2. **AMINET_0294:aminet/misc/amag/am9301b.lha**

The second file name is the filename in which it appears on the Aminet CD-ROM (Mar 1994) from Walnut Creek. Why they differ I do not know. On a different note, if you do have a CD-ROM drive then the $19 Aminet CD-ROM is a good investment and also shows your support of the nice people at Walnut Creek.

Some Aminet Sites In The USA:

wuarchive.wustl.edu
wcarchive.cdrom.com
ftp.etsu.edu

Documentation

What Is Virtual Memory?

Virtual Memory allows you to mimic RAM by using your hard drive to store information which would normally be stored in actual RAM chips. The purpose of the virtual memory program is to provide this RAM mimicking service so it appears transparent to normal programs. The main problem with virtual memory is that it is slower than normal RAM. The speed of the virtual memory will be proportional to the speed of your hard drive; the faster the drive, the better performance.

How It Works:

The concept of virtual memory is known as swapping. When a process asks for more memory, usually via the Allocmem(); call, the virtual memory program will intercept this request and perform its magic. What happens is that the swapper takes some memory from RAM that is not currently being used and it writes it out to disk. Then the program returns that memory block in the Allocmem() function call. Thus the program running has no idea whether the RAM is physical, or virtual!!

The whole idea seems very slow at all but the author has gone to the trouble of writing in the program several common and profitable algorithms which speed up the process considerably. Such algorithms are, LRU (Least Recently Used), Second Chance and Touched And Modified.

To leave out the technicalities, these algorithms reduce the amount of disk access (swapping) by calculating whether memory that needs swapping has been changed (Touched & Modified) and also calculating which blocks of memory are rarely used (LRU)

Does It Work?

In short.... Yes, very well. Then again, Virtual memory is no new concept, it has been around for a long time but we have never really seen a good implementation on the Amiga until now. Our parent operating system, UNIX, has been using Virtual memory now for many years and it is probably one of the reasons it is most successful because it can process vast amounts of data and users.

However good the code is, there is usually always a problem which arises. This is also the case with Virtual Memory. Due to the fact that the Amiga and its operating system (AmigaOS) were never designed with Virtual Memory in mind, the third party software has to patch some system functions and do some other little tricks in order to have a successful implementation. Yes, it sometimes crashes!!

Problem sometimes arise due to programs requesting certain kinds of memory. In order to function correctly, programs requesting memory of type MEMF_PUBLIC will receive the virtual memory with no problems, however other memory requests do not happen so gracefully. In the nice textbook examples, the running process will exit gracefully with an "out of memory" error or something similar, but as with real life, this isn't always the case. In the worst case scenario your machine will lock up and you will have a validation error on the partition to which you are swapping. Later in the documentation we will highlight techniques to reduce such events.

Installation

The most common bugs and problems occur when installing Virtual Memory. Of course, unless you are fluent in German then the docs do not help either. This section will explain what you need, what you need to do and how to go about doing it. A somewhat familiarity with the Amiga is required but then again, virtual memory is not for the faint of heart.

Requirements:

In order for virtual memory to work you will need the following:

- An Amiga :-)
- 32bit CPU (68020, 68030, 68040)
- MMU (68851)
- Hard Disk
- 1 - 2 MB Free RAM

The type of Amiga you own does not matter, however, what is "under the hood" will! Virtual Memory needs some form of 32 bit CPU, in other words an accelerated Amiga. These include the 68020, 68030 and 68040 processors. Virtually memory also requires the use of an MMU (Memory Management Unit) which is included with the 68030 and 68040 processors. 68020 users will need the separate MMU which is the 68851. NOTE: Economy versions of the 680x0 family do NOT have an MMU. These are denoted as 680EC20, 680EC30 etc.

The amount of free memory and the amount of free hard disk space will depend totally on how much Virtual Memory you wish to set up and the amount of buffer space you assign.

Installing The Binary Program:

Simply copy the program and it's icon to the directory of choice. Because the binary is so small and useful I have mine installed in the WBStartup drawer so it is executed at boot time. Workbench users can simply drag and drop the icon to its new home, while the shell users will need to refer to the AmigaDOS copy command.

> copy VirtualM#? SYS:WBStartup

Once the binary is set in place you will need to alter the tool types. As such, the Virtual Memory program has no preference files or environment variables. All needed information is stored as tool types in the programs icon (*.info*).

Tool Types:

There are 6 recognized tool types used by Virtual Memory. Each is listed below with an in-depth explanation, recommendation and example:

Tool Type: **SWAPFILE**
Example: **SWAPFILE=DH2:VMem.Swap**
SWAPFILE=DH2:

Swapfile is used to specify the filename or partition to where the images will be swapped. Due to the file naming conventions of AmigaDOS, the program can tell automatically if you want to use a partition or a file.

The best choice here will be to use the partition (if you can spare it). In my case I had a 20MB scratch partition which I used for downloads, printer spool, workspace etc.. I simply split that partition in two giving me a 10MB scratch and a 10MB swap partition. The advantages of using a partition are greatly in your favour. If you're system crashes and you are writing to a swapfile then chances are the entire volume or partition will need to be validated and could result in data loss. If you are using the partition method then in the rare event of a validation error you will not have data loss because all swapping is limited to its own partition. The second advantage is that using a partition rather than a file allows the Virtual Memory program to read and write directly to the disk without having to use AmigaDOS calls to perform file I/O thus resulting in a much, much faster system.

On my system I initialize the partition in my startup-sequence or user-startup to make sure it has no files on it and that it is validated with the following line:

Format Drive DH5: Name Swap FFS NOICONS QUICK

This takes about a second and causes no harm because files are not allowed to reside on the swap partition. If you have files on swap then Virtual Memory will not run when it is started, another reason to leave swap alone is that writing to swap after Virtual Memory is running will corrupt your memory. Even though the File system will report Volume Swap as being 0% full, 100% empty you mustn't use it. Virtual Memory writes directly to the disk, AmigaDOS knows nothing about it.

If you do use a file rather than a partition make sure you are on a work or scratch partition that can *easily* be restored should anything happen.

Tool Type: **FILESIZE**
Example: **FILESIZE=80**
FILESIZE=44

Filesize allows you to specify the size of the swapfile or partition. The filesize is entered in blocks rather than actual Kilobytes or Megabytes.

Virtual Memory specifies a 128K blocksize. Therefore your filesize must be specified in blocks of 128K, with 8 blocks making up 1MB. In the above examples, the first example specifies 10MB of virtual memory (8 blocks * 10MB) while the second example only requests 5½MB.

Of course the amount of virtual memory must be less than or equal to the partition or file size.

Tool Type: **SWAPBUFFER**
Example: **SWAPBUFFER=32**
SWAPBUFFER=16

Swapbuffer allows you to set the amount of real RAM that the Virtual Memory program can use. This memory is used basically as a sort of warehouse where memory that is waiting to be swapped in and out can be stored. The larger the swapbuffer, the better the performance. Like the filesize tool type, swapbuffer uses an 8k blocksize, thus 1 unit = 8k. It is recommended that you assign 256K of real RAM for every 4MB of virtual RAM. So for 10MB of virtual RAM you will require 640K of real RAM or 80 blocks. You will notice that the filesize and swapbuffer units work out to be identical.

Virtual memory will fail if it cannot allocate its buffer. Sometimes it does not always exit gracefully and sometimes it will appear as if everything is OK, until you try and do something like run another application. If this is the case, lower your swapbuffer. I am very low on real RAM once my system comes up so I have a 256K (32 block) buffer set for my 10MB of virtual memory (well below the desired limit) but my system functions well. This will be your major cause of lockup so lower your buffer if you have problems.

Tool Type: **LOOKAHEAD**
Example: **LOOKAHEAD=32**
LOOKAHEAD=64

The lookahead is a cache buffer used to boost system performance. The larger the buffer is the better the results will be. The original documentation indicates no unit size so I am assuming that the values are to be entered in kilobytes. A 16k or 32k buffer should be more than plenty.

Tool Type: **VMEMBASE**
Example: **VMEMBASE=$08000000**

The vmembase tool type allows you to set the memory address where the virtual memory will begin, its base address. If no value is entered then the above base address of 0x08000000 is assumed and used. At this writing I have been unsuccessful in achieving any other base address. *I strongly recommend that you DO NOT even use this tool type.*

Tool Type: **DONOTWAIT**
Example: **DONOTWAIT**

This tool type is the Amiga standard tool type and really has nothing to do with Virtual Memory. It just tells the operating system not to wait until the program finishes running before continuing with other tasks. You will need this tool type only if you execute Virtual Memory from your WBStartup drawer.

All the tool types described in this section with the exception of the DONOTWAIT may be issued from the command line interface using regular notation.

DMA And Masking:

The last topic you will need to concern yourself about is that of Direct Memory Access (DMA) and Masking. Due to the expeditious nature of swapping it is necessary to increase the partition mask for **EVERY partition that is on the physical drive.**

In order to increase your mask value you will need to load HDToolbox which accompanies Workbench 2.04+. By default HDToolbox loads the scsi.device driver, so if you are using another interface you will need to specify it on the command line like so:

> hdtoolbox gvpscsi.device

Once inside click on the hard drive in which Virtual Memory will be running and then choose the "Advanced Options" button to take you to another screen. At this point you will now have to change the default mask value for every partition on that physical drive. Follow the steps below:

1. Highlight the partition in which to change.
2. Enter the new mask value of **0x07FFFFFF** in the mask box, and hit return. *
3. Repeat steps 1 through 2 for EVERY other partition.

* **Note that on some systems HDToolBox will not allow you to enter a value of 0x07FFFFFF as the mask. If this is the case simply enter the mask 0x07FFFFFF and HDToolBox will adjust it accordingly.**

After you have changed all the mask values for your partitions, return to the main menu and select the Save Drive Changes button and answer yes (if it prompts you).

Installation of Virtual Memory is now complete.

Tips And Tricks

Below are tips and tricks that were mentioned throughout the document, but if you are like most users/programmers then you will not have read the whole thing so here are the highlights so to speak!

- **Choose A Partition Rather Than A Swap File**
- **Initialize (FORMAT ... QUICK) Partition At Boot Time**
- **Change Mask For Every Partition On The Drive**
- **Lower BufferSize If Problems Arise**
- **Add The DONOTWAIT tool type if using in WBStartup drawer**

If you have any other troubles or questions then drop me an EMail message and we'll see what we can do.

Miscellaneous

To obtain a copy of this document send E-Mail to me or download it from my BBS at the number below:

EMail: mchugh@hamlet.uncg.edu
BBS: (919) 563-0183

This document is shipped as a PostScript™ image which can be printed by simply copying it to any PostScript™ compatible printer.

Once again I am not the author of Virtual Memory, nor do I receive anything for making this documentation other than the satisfaction of promoting and supporting what I consider to be the best platform available on the market at the current time.

This document is © Kevin A. McHugh 1994 and may be reproduced and transferred by any means possible as long as it is not altered in any shape or form.

Acknowledgments

The author of Virtual Memory, whose name did not appear in the archive, I'm sure I speak for everyone when I say that we appreciate his work.

Antonio in Spain for his help on the DMA masking problem.

Martin Reichelt for translating those big German words.