doc/VMM

COLLABORATORS			
	TITLE : doc/VMM		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

doc/VMM

1.1 VMM/doc/VMM.guide

```
                                                VMM
              (Virtual Memory Manager for Amigas with MMU)
                            User's Guide
                            Version 3.1
                     $Date: 95/05/23 10:22:25 $
                        written by Martin Apel
                     email: apel@physik.uni-kl.de


CONTENTS

  0.
                     Shareware notice
                      1.
                     Introduction
                      2.
                     Requirements
                      2.
                     Installation
                      3.
                     Changes
                      4.
                     The preferences window
                      5.
                     68030 vs. 68040
                      6.
                     Problems
                      7.
                     Troubleshooting
                      8.
                     Technical description
                      9.
                     VMM.library
                     10.
                     External programs
                     11.
                     Known bugs
                     12.
```

1.2 VMM/doc/VMM.guide/COPYRIGHT

I have decided to make VMM shareware. I have invested about one ↩
and a
half year in developing VMM and it seems there are quite a lot of people really using it. So I decided towards the following policy: You are allowed to test VMM for thirty days to find out if it suits your needs. If you actually use it you are requested to register. The registration fee is 30 DM, 20 US-$ or equivalent. There is no different registered version, this version of VMM is fully functional. You can send cash, euro-cheques or transfer money to my bank account. See
Miscellaneous
for my address and bank account. As a service for registered users I will notify you about new versions, if you supply an e-mail address.

IMPORTANT NOTICE: This program is copyrighted by Martin Apel, but can be freely distributed, provided that the following rules are respected:

- No change is made to the program nor to the accompanying documentation
- The package is always distributed in its complete form.
- Every form of distribution is allowed and encouraged, but no fee can be charged for this program except for, possibly, the cost of magnetic media and/or disk duplication and shipping.
- Inclusion in software libraries such as Fish Disks is allowed, provided the fees charged for these disks are comparable with those charged by Fred Fish.
- The program cannot be distributed in any commercial product without the written consent of the author.

By copying, distributing and/or using the program you indicate your acceptance of the above rules.

1.3 VMM/doc/VMM.guide/INTRODUCTION

1. INTRODUCTION

Even on the A4000 equipped with 6 MB sometimes I longed for more memory or, as an alternative, for virtual memory. As the 68040 contains an MMU and I was interested in learning how it works, I decided to write a virtual memory manager for the Amiga myself. It

emulates up to 128 MB in a user selectable amount of physical RAM. From version 3.0 VMM supports the 68030, 68040 and additionally the 68020 provided you also have an external MMU 68851. Paging can be done either to a partition , a normal file or to a so-called pseudo-partition, which combines the speed of a partition with the flexibility of a file. See
Changes from V3.0 to V3.1
for a description of new features.

What does virtual memory mean ?

In a virtual memory environment the processor needs to be able to translate each address it operates on into a physical address. This translation is carried out in hardware by the memory management unit (MMU) for each memory access. Physical memory is split into page frames of equal size, VMM uses either 4 or 8 KB. It is possible for a page to be resident in memory or to be swapped out to harddisk. Whenever the processor tries to access a page that is swapped out, a pagefault occurs. VMM suspends the task causing the pagefault, fetches the corresponding page from disk and installs it somewhere in physical memory. When the page is in physical memory the task is allowed to resume. This process is carried out transparently, i.e. the task causing the pagefault has no knowledge of what happened. The only visible thing is that memory access time for that access has increased significantly. For further information how virtual memory works in general, see one of the following books:

Operating systems - Design and implementation
Andrew Tanenbaum
Prentice Hall

Operating system concepts
Silberschatz, Galvin
Addison Wesley

How does VMM achieve virtual memory management on the Amiga ?

Unfortunately the Amiga operating system has not been written to support virtual memory, so this program is kind of a hack. I tried to keep everything as system-friendly as possible, but there are certain situations when VMM might crash the system. This is not VMM's fault, but the thoughtlessness of the Amiga's developers regarding virtual memory.
VMM installs a standard memory list in ExecBase, so virtual memory will be handled just like other memory. VM is allocated only when the MEMF_PUBLIC flag in the allocation is not set. Otherwise system data such as task control blocks and IORequests might be paged out, which would lead to failure. From V3.0 code can be put into virtual memory as well. From V1.3 this mechanism has been extended to make non-behaving programs work together with VMM.

1.4 VMM/doc/VMM.guide/REQUIREMENTS

VMM needs a working MMU in your system. Currently the following configurations are supported: 68040, 68030 and 68020+68851. Make sure that you have a processor with an MMU in it, an 68EC030 or 68EC040 will not work. For VMM there is no way to find out if you have a processor equipped with an MMU or not, because Motorola sells those processors as EC types which failed their MMU test. So the MMU on your chip might be nearly fully functional but does not work correctly under all circumstances.
You should also have a harddisk although you could theoretically page to some other block-oriented device as well. Additionally you need at least 2 MB of RAM.
On the software side VMM requires at least OS2.0 to run. With OS2.1 you can benefit from localization features. For the user interface at least MUI 2.3 (V10) is needed.

1.5 VMM/doc/VMM.guide/INSTALLATION

2. INSTALLATION

There are two ways to install VMM on your system. There's an installer script provided which does all the necessary setup for you. Simply click on the icon from Workbench and it will guide you through the installation step by step. I strongly recommend using the installer script, because VMM has grown in size. If you would like to install the french version you currently have to install the english version first and install the catalog files by hand.
If you don't like to use the installer script, you can install VMM manually. You have to execute the following steps to do this:
There are two versions of the main program, one for a pagesize of 4 KB and one for 8 KB. You have to decide, which pagesize you want to use for paging. Generally a pagesize of 4 KB would be appropriate, but there might be cases (e.g. on some '040 cards for the A2000), which only work with 8K pages due to the MMU setup of special processor cards. If there's an MMU table already installed, you should use the size used by that table. Otherwise the system will probably crash. Use the "ShowPageSize" program to find out which pagesize is possible for your system. When you have chosen which one to use, copy the corresponding executable into the L: directory and rename it to "VMM-Handler". Put VMM somewhere into your path or into WBStartup.
If you have previously used VMM you have to convert your old configuration file to the new config file format. This is done by starting 'ConvVMMCfg' once, which can be found in the 'tools' drawer.
The developer files can be copied whereever you like or where your compiler expects them.
The first time you use VMM, you have to specify the parameters used for paging. This is done by simply clicking onto VMM's icon from Workbench or by starting VMM from the CLI. A sample configuration file "VMM.prefs" is provided which you can use as a starting configuration.
After you have set up your initial environment you may change the tooltypes of VMM to "CX_POPUP=NO" to prevent VMM from displaying its window on startup. During runtime you can always invoke the window by

either pressing VMM's hotkey (default: ralt rshift v) or by starting VMM a second time.
The following parameters can be specified when starting VMM. They can be entered from the CLI or as tooltypes from Workbench.

CX_POPUP=YES|NO	
CX_PRIORITY=<n>	
CX_POPKEY=<hotkey definition>	
SETTINGS=<filename>	provide another preferences file
QUIT	This makes VMM quit if it is running
FORCE	This stops VMM from asking whether to overwrite the paging partition on startup. Useful if you use the same partition for VMM and e.g. LINUX.

1.6 VMM/doc/VMM.guide/CHANGES

3. CHANGES

Changes from V1.0 to V1.1

Changes from V1.1 to V1.2

Changes from V1.2 to V1.3

Changes from V1.3 to V2.0

Changes from V2.0 to V2.1

Changes from V2.1 to V3.0

Changes from V3.0 to V3.1

1.7 VMM/doc/VMM.guide/To_1_1

There were quite a few changes / enhancements implemented since the release of V1.0.

- There was a bug, which caused VMM to page to the wrong disk on systems with hard disks with unit numbers other than zero.

- A dynamic memory allocation policy for page-frames is now implemented. This causes VMM to allocate a new page-frame on each pagefault if possible, thus reducing disk access. When memory is needed for other purposes, VMM will free its buffers and return memory to the public memory pool.

- Paging to a disk file is implemented now. Unfortunately it is quite slow due to the overhead of the AmigaDOS filing system.

- VMM should now run on all machines with a genuine 68040 processor without problems. It installs its own MMU table if necessary.

- The statistics window is font-sensitive now and gives more information about the paging process. Additionally a new program "VMMStat" is included, so you don't have to keep the statistics window open all the time.

- Up to 64 MB are now available for virtual memory.

- Two different program versions for pagesizes of 4 and 8 KB are available now.

- Disk access time is reduced by minimizing head motion of the paging device.

1.8 VMM/doc/VMM.guide/To_1_2

Changes from V1.1 to V1.2:

- In V1.1 tasks were forbidden to allocate VM from a forbidden section for security reasons. This has been changed back to the behaviour of V1.0 because of problems with AdPro.

- A small program "ShowPageSize" has been added to determine which pagesize is possible on your system.

- Fixed a bug which caused strange errors when using a page-file with its path longer than 20 characters, or paging to a partition with the partition name longer than 20 characters. The path of the page-file can now be up to 80 characters, the name of the paging partition up to 40 characters.

- Maximum VM size has been increased to 128 MB as requested.

- A special library for giving VM only to dedicated programs has been added. It includes functions such as AllocVMem , FreeVMem and AvailVMem. See vmm_lib.doc for details.

1.9 VMM/doc/VMM.guide/To_1_3

Changes from V1.2 to V1.3:

- FreeMem now marks the freed pages as empty resulting in less paging on freeing memory. Unfortunately this breaks tools such as MungWall which writes to memory it hasn't allocated.

- Fixed a bug which caused strange behaviour during disk IO using multiple units sharing the same device. This was probably responsible for some strange misbehaviours when rendering text and icons.

- The statistics window is now "zoomable" to only the title bar indicating the amount of free VM. Position and initial status of the statistics window are configurable in VMMPrefs.

- The preferences are now changeable while VMM is running. All parameters except the paging device/file, the page-file size and the position of the statistics window will be immediately changed by VMM.

- An advanced section for memory allocation has been added to VMMPrefs. You can now determine the minimum size for VM allocations for PUBLIC and non-PUBLIC requests separately.

- Reduced VMM's usage of signals on behalf of other tasks. There were problems with tasks that had all their signals already allocated.

- The Wait function had to be patched in order to avoid problems with tasks which have a stack in VM.

- You can now exit VMM even if there's VM still allocated. In this case VMM will try to page in all this memory and set up the MMU tables accordingly, after which it will quit.

- Now writes out modified pages before it needs to, resulting in better average pagefault service times.

- Added a reset handler which inhibits a reset until paging currently going on has finished. This prevents the validate procedure after reset if you are paging to a file.

- Fixed a bug which caused VMM to hang when writing the first page to DMA-driven harddisks.

- Some minor changes and cleanups.

1.10 VMM/doc/VMM.guide/To_2_0

Changes from V1.3 to V2.0:

- VMM runs on the 68030. Consequently VMM40 has been renamed to VMM.

- Implemented so-called pseudo-partition, which look like a file but can be accessed with the speed of a partition.

- VMM is a commodity in V2.0 using a hotkey to display its GUI. Consequently the setup of V2.0 has changed a bit. VMM40 has been moved to L:VMM-Handler and VMM40Prefs is now simply called VMM. The VMM: assign and the StartVMM program have become obsolete.

- Hopefully fixed bugs having to do with paging to DMA devices (Patched CachePreDMA and CachePostDMA)

- The GUI has changed a bit to make room for some additionally needed gadgets.

- VMM now patches Workbench's title bar to include the amount of free VM. There's a configuration button to enable or disable this feature.

- Better exit handling if there's still VM allocated.

- Miscellaneous minor changes and bug fixes

1.11 VMM/doc/VMM.guide/To_2_1

Changes from V2.0 to V2.1:

- AvailMem now returns amount of free public memory if task is not permitted to use virtual memory.

- Fixed a bug in the startup code which caused VMM to crash if the preferences file wasn't found. Now a requester is displayed.

- Fixed a bug in the Installer script which caused the default configuration file not to be copied.

- Implemented a write buffer to write multiple pages to disk. Although this involves copying the pages to the buffer it generally results in a quite noticeable speed improvement.

- Deleted the memory option for paging to the largest available chunk since no-one ever seems to use it.

- Paging to a file is speeded up significantly by using additional buffers for the filesystem. FFS and OFS are very inefficient on seeking on long files. VMM installs as many buffers as are needed to keep the file list blocks of the paging file in memory. Other file systems such as the MSDOS filesystem don't need this.

- FreeMem now marks the freed pages as unused instead of invalid. Previously subsequent accesses to that range of memory resulted in a page-fault without any disk access but the overhead for page-fault handling.

- Corrected font calculation for statistics window.

- Improved error handling.

- Fixed a hard to find bug which could cause spurious crashes on 68030 systems with a 68882. I underestimated the amount of data which is pushed onto the stack during a context switch if the FPU is busy.

- Added support for external statistics displays. You can now write your own (possibly graphical) statistics output for VMM.

- Worked around a DOS bug which caused failure to use pseudo-partitions on partitions with the same volume and device name.

1.12 VMM/doc/VMM.guide/To_3_0

Changes from V2.1 to V3.0:

- Implemented code paging, i.e. you can put program code into VM and swap it out just as usual memory.

- Implemented memory tracking. It is now possible to determine how much virtual memory is used by each task.

- Implemented another mode for specification of paging memory: The 'restricted dynamic mode' works like dynamic mode but you can specify lower and upper bounds for paging memory.

- I wrote a completely new user interface using MUI. This also supports localization.

- Hopefully fixed problems with some processor cards (Better handling of transparent translation registers).

- Additionally implemented a 'custom MMU setup' which should make all those nasty boards work which previously had problems with VMM's MMU setup.

- VMM can now be controlled using ARexx.

- VMM now supports machines equipped with a 68020 + 68851 combination.

- Fixed a bug which caused VMM to display a message "Not enough memory" when enabling statistics while VMM was running.

- Fixed a race condition which could cause VMM to crash when it received a message from an external statistics program during program exit time.

- The format of the configuration file has changed to a binary description. There is a program provided to convert the old style prefs file into the new representation.

1.13 VMM/doc/VMM.guide/To_3_1

Changes from V3.0 to V3.1:

- Fixed a bug which caused VMM not to work under OS2.0. An auto-OpenLibrary 'feature' of libnix which was unknown to me caused the startup code to open locale.library and refuse to run if it was not there. This problem has been eliminated by simply disabling the auto-OpenLibrary feature.

- Added a new keyword 'FORCE' to the VMM tooltypes. If specified in the icon or on the commandline, VMM will not ask you if it may overwrite a partition or file not previously used by VMM, but will simply do so. This is very useful for people who use the same paging partition for VMM and e.g. LINUX.

- Raised the VM limit from 128 MB to 512 MB after receiving multiple requests to do so. Additionally the usage of physical memory for MMU tables has decreased a bit under certain circumstances.

- Added a FAST ROM option to VMM. Since on some machines there are problems with tools such as CPU which map the ROM to Fast memory and it wasn't much work I added this one.

- VMM is now able to create a pseudo-partition on a DC-FFS disk.

- Added an empty-page-collector which causes unused pages to be freed more often. This reduces disk accesses due to pagefaults up to 30%.

- Fixed a bug in the reset code which caused VMM to generate a GURU on some machines when pressing reset.

- Fixed a minor bug which caused VMM to try to free memory again and again, although there was no memory to free. Mostly happened when the system was short of CHIP memory although VMM did not use any CHIP memory.

- Changed a bit in the MMU mapping when the VMM_MMU.config file is used. Now memory for a pagetable is allocated on the first access to an address mapped by this table. Previously VMM could not start on some machines because their processor boards mapped the whole 4 GB range needing 4 MB just for pagetables.

- Modified the memory tracking such that the loadfile is displayed as the owner of its code and not the loading task (most important for libraries and devices).

- Found a bug which I have long been searching for which produced obscure irreproducible crashes. The bug is contained in ramlib in conjunction with using semaphores. I have developed a patch for ramlib (which is not very system-conforming) which fixes this bug.

- Fixed a minor bug which caused VMM to incorrectly accept partitions with a blocksize other than 512 bytes for pseudo-partitions.

- Worked around a bug in GCC which caused VMM to always use the default settings for code paging for the compiler passes of GCC.

1.14 VMM/doc/VMM.guide/VMMPREFS

4. The preferences window

In order to enter all the settings that are needed for VMM to work, there's a graphical user interface using MUI. You can change all parameters of VMM while it is running except for a few, where it would be hard and unnecessary (in my opinion) to change them during runtime.

There are three major parts in the configuration of VMM:

Tasks / Programs

Memory settings

Miscellaneous

The three buttons at the lower edge of the window cause the named actions. Selecting 'Cancel' causes VMM to start without changing the parameters. If you want to quit VMM, use the 'Quit' item from the menu or start VMM a second time with the 'QUIT' parameter.

1.15 VMM/doc/VMM.guide/Tasks_Gadget

Tasks / Programs list:

VMM keeps a list of tasks/programs which require special attention when using VM and a default setting. You can either disable virtual memory by default and enter the programs which should use VM or the other way round. The name you enter can either be a task name, the name of a load file (without path), or even a normal AmigaDOS pattern to specify which entity is meant to be run with or without VM. For each entry in the list there are two cases which have to be considered:

1. Code paging: This determines if the program code for a file is put into VM or not.
2. Data paging: For data paging there are three possibilities: Use VM for requests with PUBLIC-bit unset, do never use VM or the advanced options.

The advanced section has been implemented because there are programs which either always allocate memory with the public flag set or they allocate memory with the PUBLIC flag unset, where it must be set. To make these programs run together with VMM, you can tell VMM the minimum sizes to go into VM with the PUBLIC flag set or unset. If you enter a value into one of the string gadgets all requests with sizes larger than the given size will be allocated in VM, e.g. a value of 0 makes all allocations with the corresponding flag settings go into VM. The special value -1 will tell VMM that no allocation should go into VM for the corresponding flag setting. This is a bit difficult to explain, so I hope the following examples will make things clearer.

Example 1: A program allocates all its memory with the PUBLIC flag set and thus makes no use of VM.
Solution: Set the "Min public VM allocation" gadget to some value e.g. 200. You will have to try the actual value. If you enter 200, all requests smaller than 200 bytes with the PUBLIC flag set will go into PUBLIC memory, whereas all larger ones will go into VM. Enter 0 into the "Min non-public VM allocation" gadget to indicate all allocations with the PUBLIC flag unset should use VM.

Example 2: A program allocates system structures with the PUBLIC flag unset leading to crashes with previous versions of VMM.

```
Solution:  Set  the "Min non-PUBLIC VM allocation" gadget to e.g.  200
           and  the "Min PUBLIC VM allocation" gadget to -1.  This way
           system  structures,  which are mostly well below 200 bytes,
           always go into PUBLIC memory.

The  "Use VM" state corresponds to a -1/0 setting of the "Min PUBLIC"/
"Min  non-PUBLIC"  gadgets.   The  "Don't use VM" state corresponds to
a setting of -1/-1.

If  a  name matches two or more names in the task list the first entry
is used.  If it doesn't match any entry the default settings are used.
This   way   you   can   specify   that  e.g.  all  libraries  except
muimaster.library  should  not use VM for their code by first entering
'muimaster.library'  with  code paging turned on and '#?.library' with
code paging turned off.

If you have problems finding out the necessary settings for a task you
can  use
                memory tracking
                 to  find  the name of each task using VM and
amount of virtual memory it uses.

The additional buttons to add, remove or sort entries should be clear.
```

1.16 VMM/doc/VMM.guide/Memory_Settings

```
                This section determines the paging device and the configuration of
memory used for paging.


                 Memory allocation

                 Memory Type

                 Write buffer

                 VM Priority

                 Swap medium

                 Swap file size
```

1.17 VMM/doc/VMM.guide/MemType_Gadget

```
Memory allocation for pageframes:

There are three policies for memory allocation:

- Fixed size: VMM  allocates  as  much  memory as given by the memory
              size gadget on startup.
```

- Dynamic: VMM allocates and frees memory for page frames as required during run-time.

- Restricted dynamic: This works identically to the dynamic option described above except that there will never be less physical memory used for paging than the value specified by the minimum memory slider and never more than specified by the maximum memory slider.

1.18 VMM/doc/VMM.guide/MemFlags_Gadget

Memory type for pageframes:

This gadget determines the type of memory to be used for paging. Either FAST, CHIP or ANY can be selected. Normally CHIP memory is inhibited from being cached; also on the A4000 pageframes lying in CHIP RAM can't be cached. However on the A2000 with a 68040 card it might be possible that pageframes located in CHIP memory are cached, making it valuable to use CHIP memory as a paging buffer to acquire more cacheable RAM.

1.19 VMM/doc/VMM.guide/WriteBuffer_Gadget

Write buffer:

From V2.1 VMM uses a write buffer in order to be able to write out several pages in one go. This results in a considerable speedup because it reduces disk seek times and overhead for I/O handling.
With the write buffer slider the amount of memory dedicated to buffering pages going out to disk can be determined. Setting this to 0 disables write buffering. Specifying an amount of 100 to 200 K for this should suffice for nearly all situations.

1.20 VMM/doc/VMM.guide/MemPri_Gadget

VM priority:

Here you can select, when VM will be allocated. Exec searches through all memory lists in a priority order with normal FAST memory usually at priority 30, CHIP at -10. So if you want to allocate VM first, you have to select a value larger than 30; if you want to use normal fast memory first, but VM before CHIP memory, you have to select a value between -10 and 30. Dynamic memory allocation works best with VM priority being the highest in the system. Otherwise all physical memory will already be taken and VMM has to page all virtual memory into a small buffer, because it cannot enlarge it anymore.

1.21 VMM/doc/VMM.guide/SwapMedium_Gadget

Swap medium:

There are three possible mediums where to page to:

- Paging to a partition: When clicking on the popup gadget you will be asked for the partition you want to use for paging. If you select a partition for paging for the first time, VMM will ask you if you really want to overwrite that partition, so you need not worry about destroying the wrong partition.

- Paging to a file: When clicking on the popup gadget you will be asked for a filename used for paging.

- Paging to a so-called pseudo-partition: Pseudo-partitions are files which use a contiguous block on your harddisk. As such they can be listed, viewed and deleted just like normal files. VMM creates this kind of file for you and can access it using device commands yielding the speed of paging onto a partition while still using a file. If the file already exists VMM checks its consistency upon startup so you need not worry about VMM overwriting part of your partition.
  Pseudo-partitions currently work only on FFS volumes with a blocksize of 512 bytes. Because pseudo-partitions need contiguous blocks on your harddisk and FFS stores the root block in the middle of a partition, a pseudo-partition cannot be larger than half the size of the volume it resides on. A sentence of warning has to be spoken here: This is a very DANGEROUS option. If there is a bug in the code creating and maintaining the pseudo-partition, VMM has the potential of destroying your partition. However, I've thoroughly tested this feature and never had any problems with it.

This parameter and the following page file size will not be updated until the next start of VMM. It would be very hard and unnecessary to switch page files during runtime.

1.22 VMM/doc/VMM.guide/FileSize_Gadget

Filesize:

This gadget determines the size of the paging file or pseudo partition.

This parameter and the previous swap medium will not be updated until the next start of VMM. It would be very hard and unnecessary to switch page files during runtime.

1.23 VMM/doc/VMM.guide/Misc_Settings

Statistics:

VMM can keep a statistics window to inform you about parameter like the number of pagefaults, the amount of virtual memory used etc. This can be turned on or off, turned into a simple titlebar and positioned.

Cache Zorro II RAM:

On some Amigas with a 68040 card you have to disable the caching of RAM in the range of 0 to $00ffffff (Zorro II addressing range). This setting is ignored if the VMM_MMU.config file generated by ReadMMUConfig is used.

Show VM in WB title:

After receiving many requests to patch the Workbench title bar in order to reflect the amount of free virtual memory I finally implemented this. As this patch is a bad hack (patches SetWindowTitles() and compares the string title to "Amiga Workbench" or "AmigaOS 3.1") I decided to make it configurable. That's what this button is for.

Memory tracking:

If this option is turned on, VMM tracks each allocation of virtual memory. It remembers the size of the allocation and the name of the task that allocated it. With the VMMUsage program you can find out, how much virtual memory a certain task uses. The overhead for this option is rather small, so you will probably not even notice a slowdown if this is enabled. However this is mainly intended for finding out tasks which don't work together with VMM without special care.
If you change the settings of this option while VMM is running all requests will be tracked while this option is on. They will not be removed from the list when you turn this option off.

Fast ROM:

Turning on this option will cause VMM to copy the ROM to FAST RAM and adjust the MMU tables such that the ROM addresses are mapped to this range. This greatly speeds up ROM accesses on some machines. This has been added since some tools which do this as well generate strange MMU settings which VMM is not able to handle.
This setting is ignored if the VMM_MMU.config file generated by ReadMMUConfig is used.

Minimum VM allocation

To speed up the AllocMem routine which is heavily used by the whole system, VMM doesn't allocate VM for requests smaller than the size given here. This saves AllocMem the overhead of looking into a hash table to find out if the current task is allowed to use VM. Setting this to zero will effectively cause VMM to look into its hash table on every AllocMem. A value of 100 to 200 is reasonable.

Hotkeys:

There are two additional hotkeys to enable or disable allocation of virtual memory during runtime. The hotkey descriptions have to follow the commodities rules for hotkey descriptions. The popup hotkey can only be specified via commandline or tooltypes (according to commodities setup).

1.24 VMM/doc/VMM.guide/PROC_DIFFS

5. Processor dependencies

There are certain differences in the 68030 and 68040 implementation. First, on the 68030 VMM (nearly) always installs its own MMU table. This is because most 68030 machines either don't use the MMU at all or they have it set to a different pagesize. This means that Enforcer does not work on the 68030 together with VMM. It uses a pagesize of 256 bytes. From V2.1 I have added a small hack which causes the machine to reload the kickfile after a crash if you are using a softkicked A3000. This feature is not documented anywhere but has been proposed by Mike Sinz.
The 68020+68851 setup is nearly the same as on the 68030 with only minor differences.

1.25 VMM/doc/VMM.guide/PROBLEMS

6. PROBLEMS

Commodore defined the MEMF_PUBLIC flag for the AllocMem function a long time ago, when no-one knew what this would mean in the future. The result is that either people allocated all their memory with the MEMF_PUBLIC flag set or they ignored it and never set it at all. The first way doesn't hurt VMM, but it prevents the corresponding program from using virtual memory. The second alternative is worse. There are a lot of programs, which allocate messages or IORequests on the stack, which might produce a failure in a virtual memory environment. Such programs can be forbidden to use virtual memory using the preferences program, otherwise spurious crashes may result. If you are considering writing programs which might benefit from virtual memory, you should read the file "VMProgGuideline" to see what should be avoided in a virtual memory environment.
Caching programs such as FastCache or PowerCache should be disabled from using virtual memory (doesn't make sense to give virtual memory to a caching program, does it?). The same applies to all programs that patch the BeginIO vector of the paging device. When using disk caching programs, the file system tasks probably have to be disabled from using VM, too.
Programs which use the Access Fault Trap Vector such as Enforcer have to be run before VMM, otherwise Enforcer will flag all pagefaults as

invalid memory references.
Code paging should not be used for programs which either contain input or interrupt handlers. An example for this is 'ixemul.library' which cannot be put into VM.

1.26 VMM/doc/VMM.guide/TROUBLESHOOTING

7. TROUBLESHOOTING

Question: Program "X" doesn't use virtual memory. Why?
Answer: "X" might always allocate memory with the MEMF_PUBLIC flag set. Use the "advanced" section to enable this program to use VM.

Question: VMM crashes with my configuration.
Answer: There are two possibilities why this is happening. If VMM crashes immediately upon startup or upon the first harddisk access, it is probably a MMU setup problem. See
MMU setup
on how to cure this. The other possibility is that ↩
there's
a program running which doesn't work correctly with VM. If this is the case, do the following:
Set the default allocation to "Don't use VM" and move the memory slider to a low position, so there are many pagefaults when VMM runs. Then enable virtual memory for each task in your system one after another. This way you can see, when the system crashes first. You can disable VM for the task producing the crashes then. Additionally you can use the
Memory tracking
feature to determine all tasks
using VM.

Question: VMM sometimes hangs when accessing a partition using the same device as the paging partition. Why?
Answer: You probably have a harddisk using DMA transfers with the mask parameter for the filesystem set wrong. You have to use either HDToolbox or edit your mountlist to change the value of the mask parameter for all partitions on the paging device. If you don't know what the mask parameter means, simply set it to 0xffffffe, limiting transfers to the lower 16 MB. This error should not occur anymore from V2.0. If it still does, please send me a mail.

Question: My disk-caching program doesn't work even if disabled from using VM. Why?
Answer: Some programs patch code executed by other tasks. E.g. DynamiCache patches the BeginIO vector of cached devices, so that all memory allocation for cache buffers is done not by DynamiCache, but by the filesystem task. If this is a problem, the tasks executing the patched code (here: DH0, DH1,...) have to be disabled from using VM. For a problem regarding the copyback cache of PowerCache see section

Known bugs
.

Question: VMM crashes the machine when pressing either of the 'Save', 'Use' or 'Cancel' button in the GUI.
Answer: You might not have a working MMU.
Unfortunately there is no way to find out if your 68030/68040 contains a working MMU except looking at the chip and watching out for an 'EC' sign. If your processor contains those two letters somewhere in its name (e.g. 68EC030) the MMU in there is not fully functional and VMM will not run on your machine. This can happen because Motorola sells 68030/68040 processors as EC types if they failed their MMU test. This might indicate only a small defect but cannot be detected with quick tests. This may cause SysInfo to report a working MMU.
If your machine contains a working MMU it might be an MMU setup problem. See
MMU setup
.

Question: What are the pros and cons of 4K and 8K pages?
Answer: In general you should follow the recommendations of the installer script which pagesize to use. Anyway VMM might work with the other pagesize as well.
With 4K pages the pagefault-rate is lower on standard applications than with 8K pages. If you are doing a lot of image processing with AdPro or the like, 8K pages might be faster. This generally applies to programs which access the memory in a linear fashion.

1.27 VMM/doc/VMM.guide/TECHNICAL_DES

8. TECHNICAL DESCRIPTION

VMM consists of three tasks, enabling the statistics window invokes a fourth. The first one is the VM_Manager, which takes care of deciding which task is allowed to use VM, initializing everything and so on. The PageHandler does the paging, when a pagefault has occurred. The Prepager causes pages to be locked in memory, when IO has to be carried out to or from VM using the paging device. Most of the effort has not been invested into getting the paging to run, but to make the system use VM correctly under all circumstances. Unfortunately Commodore hasn't invested much thought into the definition of the MEMF_PUBLIC flag, so quite a few system functions had to be patched. I have tried to keep VMM as system-friendly as possible, but I had to make a few assumptions, which are not documented. The worst one is that I had to patch the Switch function, which changes a task from running state into the ready or wait state. This means that VMM might not run in future versions of the OS, though I don't think Commodore will be changing much in such low level code.
Commodore also didn't state anything about using non-public memory inside Forbid/Permit or Disable/Enable. Causing a pagefault inside a forbidden/disabled section is dangerous, because paging results in task switching. Memory which should be freed inside such a section is

freed when this section is completed.
Currently the number of faults in progress plus the number of tasks using VM for their stack must not be more than 20 at any moment.

The VM_Manager process

The VM_Manager starts up all other tasks and initializes most data. It also handles the quit request by the user. Each time AllocMem is called, VMM has to decide whether the requesting process is allowed to use VM. The first time a task/process allocates memory, a message is sent to the VM_Manager, which decides if usage of VM is permitted. Subsequent requests by the same task are dispatched quickly using a hash table. The manager process also handles all messages generated by pressing a hotkey or using the Exchange program.

The PageHandler task

All paging is handled asynchronously by the page handler. When a fault occurs, the parameters for such a fault are put into a so-called trap-struct by the trap-handling code. This structure is then sent to the pagehandler. The pagehandler chooses a page to be evicted and possibly writes out a modified page first. The so-called second-chance algorithm is used for choosing a page to be evicted. When this process is finished, the required page is read in. When the read has been successfully completed, the faulting task is signalled to continue its computation. During IO to the paging device other tasks are permitted to run and eventually cause other pagefaults, which are handled immediately.
VMM uses a write buffer for pages which are to be written to disk. It collects pages until the write buffer is full and writes them to disk in one go. This dramatically decreases disk access times for paging.

The Prepager task

In the process of writing VMM I have detected some cases, in which IO to the paging device is requested, e.g. to another partition living on the same physical device as the paging partition. As the paging device itself must never block for a pagefault, this has to be prevented. All IO to the paging device (except that from the pagehandler) is examined for usage of virtual memory. If it uses VM, the corresponding pages are copied to a temporary buffer, the IO is carried out using that buffer, the pages are copied back if needed and the process requesting IO is permitted to continue.

The statistics task

The statistics task is only created, if statistics are enabled. Every second it prints a few lines about usage of virtual memory and the number of pagefaults occurred so far.

Patches to system functions

The following Exec functions are patched by VMM: Switch, AddTask, Wait, AllocMem, FreeMem, AvailMem, CachePreDMA, CachePostDMA, LoadSeg

and NewLoadSeg.
On 68030 system ColdReboot is also patched to restore the original MMU table before reset. Additionally the BeginIO function of the paging device is patched. The Switch, Wait and AddTask functions had to be patched because the stack may be in virtual memory. To prevent pagefaults while in supervisor mode (task-switching), the stack is replaced by a temporary stack large enough to contain all registers pushed onto the stack during a context switch. When the task is re-launched, the original stack is used again.
If you have enabled the display of virtual memory in the Workbench titlebar SetWindowTitles is also patched.

MMU setup

1.28 VMM/doc/VMM.guide/MMU_SETUP

The MMU setup used by VMM

Whenever possible VMM tries to use an already installed MMU table. There are certain cases when this is not possible such as a different pagesize than the one VMM uses or usage of certain hardware registers. (For experts: These are the so-called transparent translation registers) Some boards such as GVP's GForce '040 board, the FusionForty board and some of GVP's harddisk controllers use address ranges defined as reserved by Commodore. There is no way to find out these kind of things.
In order to make VMM work together with those boards, I have created a special way to tell VMM about its MMU setup. There's a tool called 'ReadMMUConfig' which can read the current MMU setup for the whole 4 GB address range. This information is written to a file and read by VMM upon startup. The file generated by this tool contains one line for each contiguously mapped chunk of memory. VMM generates an equal map, which does not use those special registers and the right pagesize.
From V3.1 I have made this mapping dynamic, i.e. the memory for the pagetables is only allocated after an address is first used. This should not generate any problems like in earlier versions. If it still does you might have to edit the VMM_MMU.config file by hand so following is a description of what each entry means:

Each line contains the following four entries:

log. start address | phys. start address | blocklength | flags

This means a block starting at 'log. start address' maps to 'phys. start address' for 'blocklength' bytes. The flags for this region are the following:

Bit 0 must always be set.
Bit 1 must always be cleared.
Bit 2 must be set if this region should be write-protected.
Bit 3 and 4 are ignored.

Bit 5 and 6 reflect the caching mode as follows:

Bit 6	Bit 5	68040	68030 and 68851
0	0	Cacheable, write-through	Cacheable
0	1	Cacheable, copyback	Cacheable
1	0	Noncacheable, serialized	Noncacheable
1	1	Noncacheable	Noncacheable

If you cannot get it to work by modifying this file yourself, send me a mail with the output of the 'ReadMMUConfig' file, the output of ShowConfig or similar and I will try to assist you in setting up an MMU table which will work.

1.29 VMM/doc/VMM.guide/VMM_LIBRARY

9. VMM.LIBRARY

A library has been added to the VMM distribution as requested by some people. It contains functions for AllocVMem, FreeVMem, AvailVMem, AllocVVec and FreeVVec. When the library is first loaded, the VMM pagehandler is automatically installed. Only programs using the function calls of vmm.library will get virtual memory. If you wish to use vmm.library and the standard way of getting VM, you have to use VMM just like before. Currently the pagehandler is not terminated when the library is expunged. A doc file for the library functions in Autodoc format is included.

1.30 VMM/doc/VMM.guide/EXT_PROGS

10. EXTERNAL PROGRAMS

From V2.1 it is possible to write your own statistics display program. You can add graphical output and additional computations such as average page fault rate and so on. There's an include file describing the message structure you have to send to VMM to acquire status information. See that include file for further information. If you write a nice program you can send it to me and I might include it in the next release.
From V3.0 the usage statistics can also be retrieved by an external program. So if you don't like VMMUsage, or you would like to have a combined display for statistics and VM usage you can write your own program. The way to retrieve the necessary information is depicted in the same include file as the statistics info.

1.31 VMM/doc/VMM.guide/KNOWN_BUGS

11. KNOWN BUGS

As far as I know there are no major bugs in VMM. However there is one

minor bug which still has to be corrected. If a task is removed via RemTask from another task and it uses virtual memory for its stack, the TrapStruct is currently not released. Commodore recommends not to call RemTask on another task and it isn't done very often in real programs. One possibility is to prevent the task that allocates the stack for such programs from using virtual memory.

It can happen that VMM cannot startup due to lack of memory. In this case it should bring up a requester stating this fact. If the requester cannot be created, there will be no message at all indicating the reason of failure. This is due to a bug in EasyRequestArgs, as there is no way to find out if the requester was successful or not. In future OS versions (if there will be any) this should produce a recoverable alert.

There's a strange phenomenon in conjunction with console.device. When you open a new CLI window, sometimes the cursor remains ghosted although the window is activated. Activating another window and then again the new console window will activate the cursor. The 'CON' task seems to allocate its stack in VM and that produces the problem. If you disable 'CON' from using VM this problems disappears.

There's a bug in conjunction with the copyback mode of PowerCache. It causes file contents to be garbled. This is rather a bug in PowerCache than in VMM. In the current version of PowerCache (37.115) there's are two choices: either you have to turn off the copyback cache of PowerCache or you have to set the mask (in HDToolbox) for all partitions on the paging device to 0xffffffff (limits accesses to lower 256 MB). I hope this will be fixed in future versions of PowerCache.

There's a strange phenomenon in conjunction with Workbench. If you try to copy a large file (min. 10 MB) to RAM: using the Workbench it will fail for lack of memory even if your paging partition or file is large enough. If you do the same from the CLI all works fine. This is a bug of Workbench. I have tried this without VMM running but with 16 MB Fast RAM in my machine and the same error turns up.

VMM patches the LoadSeg library vector to accomplish code paging. Unfortunately this patch cannot simply call the original LoadSeg function when it is done. This means that patches to this function like e.g. SegTracker applied before VMM is started are not executed for programs which are loaded into virtual memory. This has become necessary because the OS often does not call its own functions via the official (patchable) vectors.

1.32 VMM/doc/VMM.guide/BUG_REPORTING

12. BUG REPORTING

If you wish to report a bug or propose an enhancement to VMM please use the bug-report form which can be found in this archive. I frequently cannot answer questions because I don't know which version of VMM people are referring to, which hardware they use and so on. This is the reason the bug-report form is important for me.
You can find my address under

Miscellaneous
.

1.33 VMM/doc/VMM.guide/ACKNOWLEDGMENTS

13. ACKNOWLEDGMENTS

I would like to thank the following people for ideas, improvements and beta-testing of VMM (in alphabetical order).

Michael Berg (Danish translation)
Wayne Cole
Torsten Ebeling
Markus Eiblmeier
Sven Fischer
Frank Grimm
Magnus Holmgren (Swedish translation)
Robert Kiehne
Jeff Koons
Steve Koren
Andree Maedl
Manfred Matzinger
Barry McConnell
Marco Musso (Italian translation)
Paul Ney
Stefan Odendahl
Hans Otten
Volker Rudolph (for helping me with the 68030 port)
Stefan Schmidt
Torsten Stolpmann
Erno Tuomainen
Emmanuel Vacher (French translation)
Nikola Vukovljak
Alg Inge Wang
Juergen Zimmermann

I also want to thank all those people who sent me mail with wishes and bug reports. Without them VMM wouldn't work as well as it does.

1.34 VMM/doc/VMM.guide/MISCELLANEOUS

14. MISCELLANEOUS

I would be glad to hear from you, if VMM works on your machine, what programs have difficulties in running with VMM. If you report a bug PLEASE use the bug-report form which can be found in this archive. Because most bugs seem to be very hardware dependent I need your configuration data to find out what happens.

email: apel@physik.uni-kl.de

snail-mail: Martin Apel
Gerhart-Hauptmann-Str.5
67663 Kaiserslautern
Germany

phone: 0631 / 24257

bank account: 145 009 494
at: Sparkasse Bonn
BLZ 380 500 00