doc/VMM

COLLABORATORS			
	TITLE : doc/VMM		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 18, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

doc/VMM

1.1 VMM/doc/VMM.guide

```
                                                VMM
            (Virtual Memory Manager für Amigas mit MMU)
                        Benutzeranleitung
                           Version 3.3a
                  $Date: 96/07/24 20:29:26 $
                         von Martin Apel
                     email: apel@tecmath.de


CONTENTS

  0.
                 Shareware-Hinweis
                  1.
                 Einführung
                  2.
                 Voraussetzungen
                  3.
                 Installation
                  4.
                 Änderungen
                  5.
                 Das Einstellungssfenster
                  6.
                 Prozessorunterschiede
                  7.
                 Schwierigkeiten
                  8.
                 Häufig auftretende Probleme
                  9.
                 Technische Beschreibung
                 10.
                 VMM.library
                 11.
                 Externe Programme
                 12.
                 Bekannte Fehler
                 13.
```

1.2 VMM/doc/VMM.guide/COPYRIGHT

Ich habe entschieden, VMM zu Shareware zu machen. Ich habe ↩
ungefähr
zwei Jahre mit der Entwicklung von VMM verbracht und es gibt scheinbar eine Menge Leute, die es verwenden. Daher habe ich mich für die folgende Vereinbarung entschieden: Sie können VMM für 30 Tage testen, um herauszufinden, ob es für Ihre Ansprüche einsetzbar ist. Wenn Sie VMM dauerhaft einsetzen, bitte ich Sie, VMM zu registrieren. Die Registrierungsgebühr beträgt 30 DM, 20 US-$ oder den gleichen Betrag in anderer Währung. Es gibt keine extra registrierte Version, die vorliegende Version von VMM beinhaltet alle Fähigkeiten. Sie können mir Bargeld schicken oder eine Überweisung auf mein Konto durchführen. Meine Adresse und Bankverbindung finden Sie unter
Verschiedenes
.
Registrierte Benutzer werden von mir bei jeder neuen Version benachrichtigt, falls mir eine E-Mail Adresse von Ihnen zur Verfügung steht.

WIHTIG: Das Urheberrecht dieses Programms liegt bei Martin Apel, es kann jedoch frei verteilt werden, vorausgesetzt, die folgenden Regeln werden beachtet:

- Weder das Programm noch die beiliegende Dokumentation dürfen in irgendeiner Weise verändert werden.
- Dieses Archiv darf nur in vollem Umfang weitergegeben werden.
- Eine Verteilung von VMM darf auf beliebigem Wege erfolgen, es darf jedoch nur eine geringe Gebühr für Kopieren, Medienkosten und Versand erhoben werden.
- Einfügen in Software-Bibliotheken wie Fish Disks ist erlaubt, falls die Gebühren denen für Fish-Disks entsprechen.
- Das Programm darf nicht ohne die schriftliche Zustimmung des Authors kommerziell vertrieben werden.

Durch Kopieren, Verteilung und Benutzung des Programms erklärt sich der Benutzer mit den obigen Regeln einverstanden.

1.3 VMM/doc/VMM.guide/INTRODUCTION

1. EINFÜHRUNG

Sogar auf dem A4000 mit 6 MB habe ich mich manchmal nach mehr Speicher gesehnt, oder alternativ nach virtuellem Speicher. Da der 68040 eine

MMU enthält und ich daran interessiert war, wie sie funktioniert, habe ich selbst einen virtuellen Speichermanager geschrieben. Dieser emuliert bis zu 512 MB virtuellen Speicher in einer vom Benutzer konfigurierbaren Menge physikalischen Speichers. Ab Version V3.3 unterstützt VMM alle 680x0 Prozessoren mit einer MMU, selbst den 68060. Das Auslagern von Seiten (Paging) kann auf eine Partition, eine Datei oder eine sogenannte Pseudo-Partition erfolgen, die die Geschwindigkeit einer eigenen Partition mit der Flexibilität einer Datei verknüpft. Siehe

Änderungen von V3.2 zu V3.3
für eine Beschreibung neuer Funktionen.

Was bedeutet virtueller Speicher?

In einem System mit virtuellem Speicher muß der Prozessor jede Adresse, die er verwendet, erst in eine physikalische Adresse übersetzen. Diese Übersetzung wird in Hardware von der Speicherverwaltungseinheit (MMU) für jeden Speicherzugriff durchgeführt. Der physikalische Speicher wird in sogenannte Kacheln gleicher Größe aufgeteilt; VMM verwendet eine Seitengröße von 4 oder 8 KB. Eine Seite kann sich entweder im Hauptspeicher oder auf der Festplatte befinden. Wenn ein Zugriff auf eine ausgelagerte Seite erfolgt, wird ein Seitenfehler ausgelöst. VMM hält den Prozeß, der den Seitenfehler verursacht hat, an und lädt die benötigte Seite irgendwo in den Hauptspeicher. Danach kann der Prozeß mit seiner Arbeit fortfahren. Von diesem Vorgang merkt der Prozeß nichts, außer daß dieser eine Speicherzugriff erheblich länger als normal dauert. Für genauere Informationen zum Thema virtuelle Speicherverwaltung seien folgenden Bücher empfohlen (Englisch):

Operating systems - Design and implementation
Andrew Tanenbaum
Prentice Hall

Operating system concepts
Silberschatz, Galvin
Addison Wesley

Wie stellt VMM virtuellen Speicher auf dem Amiga bereit?

Leider unterstützt das Amiga-Betriebssystem keinen virtuellen Speicher, daher ist VMM in gewisser Weise ein Hack. Ich habe versucht, alles so systemkonform wie möglich zu halten, aber es gibt bestimmte Situationen, in denen VMM einen Systemcrash hervorruft. Dies liegt nicht an VMM, sondern an der Gedankenlosigkeit der Amiga-Entwickler bzgl. virtuellen Speichers.
VMM installiert eine normale Speicherliste in ExecBase, sodaß virtueller Speicher genauso wie normaler Speicher behandelt wird. Es wird nur dann virtueller Speicher allokiert, wenn das MEMF_PUBLIC bei einer Anforderung nicht gesetzt ist. Sonst könnte es passieren, daß Systemdaten wie Taskkontrollblöcke oder IORequests ausgelagert würden, was zu einem Absturz führen würde. Zusätzlich gibt es einen Mechanismus für Programme, die unter normalen Umständen mit VMM abstürzen würden. Ab V3.0 kann auch Programmcode in den virtuellen Speicher verlegt werden.

1.4 VMM/doc/VMM.guide/REQUIREMENTS

2. VORAUSSETZUNGEN

VMM benötigt eine funktionierende MMU. Stellen Sie sicher, daß Ihr Prozessor eine MMU enthält, ein EC-Typ des Prozessors wird nicht mit VMM zusammenarbeiten. Es gibt leider keinen zuverlässigen Weg herauszufinden, ob eine funktionsfähige MMU vorhanden ist, da Motorola ihre Prozessoren mit Defekten in der MMU als EC-Typen verkauft. Es kann also passieren, daß die MMU auf den ersten Blick funktioniert, aber unter bestimmten Bedingungen Fehler besitzt.
Außerdem sollten Sie eine Festplatte besitzen, obwohl es theoretisch möglich ist, auf jedes beliebige blockorientierte Gerät auszulagern. Es sollten auch mindesten 2 MB Hauptspeicher vorhanden sein.
Als Betriebssystem benötigt VMM mindestens Version 2.0. Ab V2.1 können Sie auch die Vorteile verschiedener Sprachunterstützung genießen. Für die Benutzerschnittstelle wird mindesten MUI 2.3 (V10) benötigt.

1.5 VMM/doc/VMM.guide/INSTALLATION

3. INSTALLATION

Zur Installation gibt es ein Installer-Skript, das die nötige Arbeit durchführt. Klicken Sie auf das Icon auf der Workbench und Sie werden durch den Installationsvorgang geführt. Die Installation fragt Sie nach der Seitengröße, die Sie verwenden wollen. Normalerweise ist eine Seitengröße von 4 KB zu empfehlen, es gibt jedoch einige Fälle, in denen nur die 8 KB-Version bei Ihnen korrekt arbeitet (z.B. bestimmte 68040-Karten für den A2000). Eine Seitengröße von 8 KB kann schneller sein, falls Sie viel mit Bildbearbeitungsprogrammen arbeiten.
Wenn Sie das erste Mal VMM benutzen, müssen Sie einige Parameter in der Benutzeroberfläche einstellen. Dies können Sie tun, indem Sie VMM von der Workbench oder vom CLI aus starten. Eine Standardkonfiguration ist vorhanden.
Nachdem Sie Ihre Einstellungen durchgeführt haben, können Sie die Tooltypes im Icon zu "CX_POPUP=NO" ändern, damit VMM nicht bei jedem Start sein Fenster öffnet. Zur Laufzeit können Sie dieses Fenster jedoch jederzeit über den Hotkey (Standard: ralt rshift v) öffnen. Alternativ können Sie VMM einfach ein zweites Mal starten. Die folgenden Parameter können als Tooltypes oder als Parameter im CLI angegeben werden:

```
CX_POPUP=YES|NO
CX_PRIORITY=<n>
CX_POPKEY=<hotkey definition>
SETTINGS=<filename>             andere Einstellungsdatei verwenden
QUIT                            VMM verlassen
FORCE                           Dies verhindert VMM Nachfrage beim
```

Überschreiben einer anderweitig
genutzten Partition, z.B. für LINUX

1.6 VMM/doc/VMM.guide/CHANGES

4. ÄNDERUNGEN

Änderungen von V1.0 zu V1.1

Änderungen von V1.1 zu V1.2

Änderungen von V1.2 zu V1.3

Änderungen von V1.3 zu V2.0

Änderungen von V2.0 zu V2.1

Änderungen von V2.1 zu V3.0

Änderungen von V3.0 zu V3.1

Änderungen von V3.1 zu V3.2

Änderungen von V3.2 zu V3.3

Änderungen von V3.3 zu V3.3a

1.7 VMM/doc/VMM.guide/To_1_1

Änderungen von V1.0 zu V1.1:

- VMM lagerte bei Systemen mit mehreren Festplatten immer auf die Platte mit Unit 0 aus.

- Es gibt jetzt eine dynamische Allokationsstrategie für Kacheln. VMM allokiert bei einem Seitenfehler eine neue Kachel falls möglich, um Festplattenzugriffe zu sparen. Wenn dieser Speicher für andere Zwecke benötigt wird, wird dieser von VMM wieder freigegeben.

- Auslagerung auf Datei. Dies ist leider relativ langsam aufgrund des hohen Overheads des Amiga-Dateisystems.

- VMM sollte jetzt auf allen echten 68040-Prozessoren problemlos laufen. Bei Bedarf wird eine eigene MMU-Tabelle installiert.

- Das Statistik-Fenster in nun Font-sensitiv und stellt mehr Informationen über den Auslagerungsprozeß dar. Zusätzlich gibt es ein Programm "VMMStat", sodaß das Statistikfenster nicht die ganze benötigt wird.

- Bis zu 64 MB virtueller Speicher werden nun unterstützt.

- Es gibt zwei verschiedene Programmversionen für Seitengrößen von 4 und 8 KB.

- Die Zugriffszeit auf die Platte wurde reduziert, indem die Kopfbewegungen minimiert werden.

1.8 VMM/doc/VMM.guide/To_1_2

Änderungen von V1.1 zu V1.2:

- Aus Sicherheitsgründen wurde es in V1.1 Tasks verboten, bei gesetztem Forbid() virtuellen Speicher zu allokieren. Dies wurde wegen Problemen mit AdPro rückgängig gemacht.

- Es gibt jetzt eine Programm "ShowPageSize", daß die möglichen Seitengrößen für ein gegebenes System ermittelt.

- Fehler behoben, der zu merkwürdigen Fehlern führte, falls der Name der Auslagerungsdatei oder der Auslagerungspartition länger als 20 Zeichen war. Die maximale Pfadlänge beträgt nun 80 Zeichen.

- Die maximal verfügbare Menge virtuellen Speichers wurde nach Anfrage auf 128 MB angehoben.

- Es wurde eine spezielle Bibliothek zu VMM hinzugefügt, die es nur dafür geschriebenen Programmen ermöglicht, virtuellen Speicher zu nutzen. Es gibt Funktionen wie AllocVMem, FreeVMem und AvailVMem. Siehe vmm_lib.doc.

1.9 VMM/doc/VMM.guide/To_1_3

Änderungen von V1.2 zu V1.3:

- FreeMem markiert freigegebene Seiten jetzt als leer, um das Auslagern unbenutzter Seiten beim Freigeben des Speichers zu vermindern. Leider werden dadurch Tools wie Mungwall behindert, die Speicher beschreiben, den sie nicht allokiert haben.

- Fehler behoben, der ein merkwürdiges Verhalten während Festplattenzugriffen verursachte, deren Units das gleiche Device verwenden. Vermutlich war dies für Probleme mit der Grafikausgabe von Text und Icons verantwortlich.

- Das Statistikfenster ist jetzt "zoombar", sodaß nur die Titelzeile zur Anzeige des freien virtuellen Speichers sichtbar ist. Position und Anfangszustand können konfiguriert werden.

- Die Einstellungen können jetzt verändert werden, während VMM läuft. Alle Parameter außer der Auslagerungspartition/-datei, deren Größe und der Position des Statistikfensters werden jetzt sofort geändert.

- Es gibt nun eine "fortgeschrittene" Einstellmöglichkeit für die Speicherallokation in VMMPrefs. Man kann jetzt für jede Task die minimale Größe der VM-Allokationen für PUBLIC und nicht-PUBLIC-Speicher getrennt einstellen.

- Anzahl der Signale, die eine Task während eines Seitenfehlers benötigt, reduziert. Vorher gab es Probleme mit Tasks, die all ihre Signale selbst benötigten.

- Die Wait-Funktion wurde gepatcht, um Probleme mit Tasks zu umgehen, deren Stack im virtuellen Speicher liegt.

- VMM kann nun verlassen werden, selbst wenn noch virtueller Speicher allokiert ist. In diesem Fall lädt VMM alle noch in Gebrauch befindlichen Seiten in den Hauptspeicher und modifiziert die MMU-Tabelle entsprechend. Anschließend wird das Programm verlassen.

- Modifizierte Seiten werden geschrieben, bevor dies unbedingt nötig ist. Dies reduziert die mittlere Zeit für einen Seitenfehler.

- Resethandler eingefügt, der einen Reset so lange verzögert, bis alle laufenden Ein-/Ausgabeoperationen auf die Festplatte abgeschlossen sind. Ein Validieren der Festplatte nach dem Neustart wird so verhindert.

- Fehler behoben, der verursachte, daß VMM beim Schreiben der ersten Seite auf eine Festplatte mit DMA hing.

- Kleinere Änderungen und Aufräumarbeiten.

1.10 VMM/doc/VMM.guide/To_2_0

Änderungen von V1.3 zu V2.0:

- VMM läuft jetzt auf dem 68030. Daher wurde es von VMM40 zu VMM umbenannt.

- Sogenannte Pseudo-Partition implementiert. Diese sieht aus wie eine Datei, darauf kann jedoch mit der Geschwindigkeit einer Partition zugegriffen werden.

- VMM ist jetzt ein Commodity, dessen Benutzerschnittstelle mittels eines Hotkeys angezeigt wird. Daher hat sich der Aufbau von VMM etwas verändert. VMM40 wurde nach L:VMM-Handler umbenannt und VMM40Prefs heißt jetzt nur noch VMM. Das VMM: Assign und das StartVMM-Programm sind damit überflüssig.

- Hoffentlich alle Probleme mit DMA-Geräten beseitigt (CachePreDMA und CachePostDMA gepatcht).

- Das Benutzerinterface wurde erweitert, um einige neue Einstellmöglichkeiten aufzunehmen.

- VMM patcht jetzt die Titelzeile der Workbench, um dort den freien

virtuellen Speicher mit anzuzeigen. Dies wird durch einen Schalter freigegeben.

- Saubereres Verlassen von VMM, falls noch virtueller Speicher allokiert ist.

- Weitere kleine Fehlerbehebungen und Erweiterungen.

1.11 VMM/doc/VMM.guide/To_2_1

Änderungen von V2.0 zu V2.1:

- AvailMem liefert jetzt den freien physikalischen Speicher zurück, falls eine Task keinen virtuellen Speicher verwenden darf.

- Fehler behoben, der dazu führte, daß VMM abstürzte, wenn die Einstellungsdatei nicht gefunden wurde. Jetzt wird ein Requester geöffnet.

- Fehler bei der Installation behoben, der die Standard-Einstellungsdatei nicht kopierte.

- Schreibpuffer implementiert, der mehrere Seite auf ein Mal auf die Platte schreibt. Zwar müssen dafür die Seiten einmal umkopiert werden, meist resultiert dies aber in einem substantiellen Geschwindigkeitsgewinn.

- Die Option, den größten freien Speicherblock für die Kacheln zu verwenden, wurde gelöscht, da offensichtlich niemand dies benutzt.

- Auslagerung auf Datei wurde drastisch beschleunigt, indem zusätzliche Puffer beim Dateisystem angefordert werden. FFS und OFS sind leider sehr ineffizient bei der Positionierung in langen Dateien. VMM fordert so viele Puffer an, wie für die Speicherung der Dateiverkettungsinformation benötigt werden. Andere Dateisysteme wie das MSDOS-Dateisystem benötigen dies nicht.

- FreeMem markiert freigegebene Seiten nun als unbenutzt und nicht als ungültig. Vorher wurde beim nächsten Zugriff jeweils ein Seitenfehler ausgelöst, der jedoch keinen Plattenzugriff hervorrief.

- Fontberechnung für Statistikfenster korrigiert.

- Verbesserte Fehlerbehandlung

- Schwer zu findenden Fehler behoben, der zufällige Abstürze auf Systemen mit 68030 und 68882 hervorrief. Ich hatte die benötigte Stackgröße bei einem Seitenfehler unterschätzt, wenn die FPU beschäftigt ist.

- Unterstützung für externe Statistikprogramme implementiert. Es können nun eigene (z.B. grafische) Statistiktools für VMM geschrieben werden.

- DOS Fehler umgangen, der Fehler bei Pseudo-Partitionen auf Partitionen

mit dem gleichen Volume- und Devicenamen verursachte.

1.12 VMM/doc/VMM.guide/To_3_0

Änderungen von V2.1 zu V3.0:

- Code-Paging eingebaut, d.h. man kann jetzt den Programmcode in den virtuellen Speicher legen und VMM lagert diesen aus wie normalen Speicher.

- Speicher-Tracking eingebaut. Es ist nun möglich zu verfolgen, welche Task wieviel virtuellen Speicher verwendet.

- Zusätzlicher Modus zur Angabe des Hauptspeichers für den virtuellen Speicher. Der 'eingeschränkt dynamische Modus' funktioniert wie der dynamische Modus mit einer Angabe für Unter- und Obergrenze des belegten Speichers.

- Komplett neue Benutzeroberfläche, die MUI verwendet. Dadurch werden jetzt auch verschiedene Sprachen unterstützt.

- Hoffentlich Probleme mit einigen Prozessorkarten beseitigt, indem die Transparent Translation Register besser genutzt werden.

- Zusätzliches 'spezielles MMU-Setup' implementiert, das alle Karten, die bisher Probleme bereiteten, zur Zusammenarbeit bringen sollte.

- VMM kann über ARexx angesteuert werden.

- VMM unterstützt nun die Kombination 68020 + 68851.

- Fehler behoben, der VMM eine Nachricht ' Nicht genug Speicher' ausgeben ließ, wenn das Statistikfenster während des Programmlaufs geöffnet wurde.

- Das Format der Konfigurationsdatei wurde auf ein Binärformat geändert. Es gibt ein Programm zum Konvertieren der alten Konfigurationsdateien.

1.13 VMM/doc/VMM.guide/To_3_1

Änderungen von V3.0 zu V3.1:

- Fehler behoben , der verhinderte, daß VMM unter OS2.0 lief. Ein mir unbekanntes Auto-OpenLibrary der libnix-Library versuchte die Locale.library zu öffnen. Wenn diese nicht vorhanden war, weigerte sich VMM zu starten.

- Es gibt nun ein neues Schlüsselwort 'FORCE'. Wenn es im Icon oder auf der Kommandozeile angegeben wird, fragt VMM nicht mehr nach, ob eine vorher anderweitig genutzte Partition überschrieben werden soll. Dies ist insbesondere für Leute nützlich, die die Swap-Partition z.B.

für LINUX nutzen möchten.

- Die Obergrenze des verfügbaren virtuellen Speichers wurde nach mehreren Anfragen von 128 MB auf 512 MB erhöht.

- Es gibt nun eine FastROM option für VMM. Auf manchen Rechnern gibt es Probleme mit anderen Tools, die eine FastROM-Option anbieten.

- Pseudo-Partitionen sind jetzt auch auf DC-FFS Partitionen möglich.

- Es wurde ein Algorithmus zum Sammeln unbenutzter Seiten implementiert. Dies reduziert die Anzahl der Plattenzugriffe wegen Seitenfehlern um bis zu 30 %.

- Fehler im Resethandler behoben, der auf manchen Rechner beim Reset einen GURU hervorrief.

- Kleinen Fehler behoben, der VMM manchmal veranlaßte, immer wieder Speicher freigeben zu wollen . Diese passierte meistens, wenn das Chip-Memory knapp wurde.

- Das MMU-Mapping wurde leicht verändert, wenn die VMM_MMU.config-Datei verwendet wird. Vorher konnte VMM auf manchen Rechnern nicht starten, da es versuchte für die kompletten 4 GB Adressraum Seitentabellen anzulegen (4 MB).

- Das Memory-Tracking wurde geändert, so daß das jeweilige Programm als Besitzer seines Codes angezeigt wird und nicht die Task, die den entsprechenden Speicher allokiert. Dies ist hauptsächlich für Libraries von Interesse.

- Fehler gefunden, den ich schon lange gesucht habe. Ein Fehler in ramlib führte zu nicht reproduzierbaren Abstürzen. Ein Patch wird wenn möglich von VMM durchgeführt.

- Kleinen Fehler behoben, wegen dem VMM fälschlicherweise Partitionen mit ein anderen Blockgröße als 512 Byte für Pseudo-Partitionen akzeptierte.

- Workaround für einen Fehler in GCC eingebaut, der VMM immer die Standardeinstellungen für die Compilerpässe benutzen ließ.

1.14 VMM/doc/VMM.guide/To_3_2

Änderungen von V3.1 zu V3.2:

- Enforcer-Hit entfernt, der auftrat, wenn VMM mit dem Statistikfenster als Titelzeile gestartet wurde.

- Mögliche Resetverzögerung entfernt.

- Fehler in ReadMMUConfig behoben. Wenn es mit 8K-Seiten verwendet wurde, wurde eine leere VMM_MMU.config-Datei erzeugt.

- Man kann jetzt auch Verzeichnisse in der Task-Liste eingeben. Man

kann so seine Programme in Verzeichnisse gruppieren, für die die gleichen Einstellungen gelten sollen. Nur Dateien direkt aus dem angegebenen Verzeichnis werden berücksichtigt, nicht die Dateien aus Unterverzeichnissen.

- VMM erkennt jetzt automatisch LINUX-Swap-Partitionen.

1.15 VMM/doc/VMM.guide/To_3_3

Änderungen von V3.2 zu V3.3:

- Die Ausgabe von VMMUsageCLI sieht jetzt etwas schöner aus (empfohlen von Steve Koren).

- Jeder Eintrag in der Taskliste kann jetzt beliebig lang sein. Dies ist insbesondere für komplizierte Suchmuster wichtig.

- VMM läuft jetzt auch auf dem 68060.

- Fehler behoben, der eine Absturz hervorrief, wenn die FastROM-Option auf einem 68030 eingeschaltet wurde.

1.16 VMM/doc/VMM.guide/To_3_3a

Änderungen von V3.3 zu V3.3a

- Deutsche Dokumentation

- Russische Sprachunterstützung

- Update der BGUI-Oberfläche

1.17 VMM/doc/VMM.guide/VMMPREFS

5. DAS EINSTELLUNGS-FENSTER

Um alle nötigen Einstellungen einzugeben, die VMM zum Arbeiten benötigt, existiert eine graphische Benutzeroberfläche, die MUI verwendet. Fast alle Parameter können geändert werden, während VMM läuft. Wo dies nicht möglich ist, ist es entweder unsinnig oder sehr schwierig zu implementieren.

Es gibt drei Hauptbereiche in den Einstellungen von VMM:

Tasks / Programme

Speichereinstellungen

Verschiedenes

Die drei Knöpfe an der unteren Kante des Fensters funktionieren ↩
wie
bei jedem Einstellungsprogramm. Die Auswahl von 'Abbrechen' läßt VMM die Änderungen der Einstellungen vergessen und startet anschließend. Wenn Sie VMM verlassen wollen, wählen Sie 'Beenden' im Menü oder starten Sie VMM ein zweites Mal mit dem Parameter 'QUIT'.

1.18 VMM/doc/VMM.guide/Tasks_Gadget

Liste der Tasks / Programme:

VMM führt eine Liste von Tasks/Programmen, die spezielle Aufmerksamkeit erfordern, wenn sie virtuellen Speicher nutzen sollen. Zusätzlich gibt es eine Standardeinstellung. Virtueller Speicher kann entweder standardmäßig ausgeschaltet werden, um dann die Programme in die Liste einzutragen, die virtuellen Speicher nutzen sollen oder genau entgegengesetzt. Der Name, der in die Liste eingetragen wird, kann ein Taskname, ein Programmname (ohne Pfad), ein Verzeichnis oder ein AmigaDOS-Muster sein, das die Programme spezifiziert, die mit oder ohne virtuellen Speicher laufen sollen. Bei Verwendung eines Verzeichniseintrags sind alle Programme aus diesem Verzeichnis betroffen, nicht jedoch die Programme in dessen Unterverzeichnissen. Dies trifft leider nur auf DOS-Prozesse zu, da es keinen Weg gibt festzustellen, welches Programm eine einfache Exec-Task ausführt. Für jeden Eintrag werden zwei Fälle betrachtet:

1. Code Paging: Dies bestimmt, ob der Programmcode für das angegebene Programm in den virtuellen Speicher geladen wird.
2. Daten Paging: Hier existieren drei Möglichkeiten: Virtueller Speicher wird verwendet, wenn bei der Speicheranfrage das PUBLIC-Bit nicht gesetzt ist, er wird nie verwendet oder es werden Zusatzoptionen ausgewertet.

Die Zusatzoptionen wurden implementiert, da es Programme gibt, die entweder immer ihren Speicher mit dem Public-Flag allokieren oder die das Public-Flag dort weglassen, wo es benötigt wird. Um diese Programme mit VMM lauffähig zu machen, kann man VMM mitteilen, wie groß ein Block mindestens sein soll, damit er in den virtuellen Speicher kommt. Wenn man eine Zahl in eines der beiden Felder einträgt, werden alle Allokationen, die größer sind als diese Zahl, in den virtuellen Speicher gelegt. Diese Einstellung kann für Allokationen mit und ohne Public-Flag getätigt werden. Der spezielle Wert -1 teilt VMM mit, daß keine Allokation mit diesem Flag in den virtuellen Speicher gehen soll. Da dies etwas schwierig zu erklären ist, folgen zwei Beispiele.

Beispiel 1: Ein Programm allokiert all seinen Speicher mit dem Public-Flag und benutzt daher keinen virtuellen Speicher.
Abhilfe: Setzen Sie das Feld "Min. PUBLIC Allok." auf einen Wert, z.B. 200. Den sinnvollsten Wert müssen Sie herausfinden. Nun werden alle kleinen Allokationen, deren Public-Flag gesetzt ist, in normalen Speicher umgeleitet, während die größeren virtuellen Speicher verwenden. Tragen Sie 0 in das Feld "Min. non-PUBLIC Allok." ein, um VMM

mitzuteilen, daß alle Allokationen ohne das Public-Flag in den virtuellen Speicher sollen.

Beispiel 2: Ein Programm allokiert Systemstrukturen ohne das Public-Flag und stürzt daher ab.
Abhilfe: Setzen Sie das Feld "Min. non-PUBLIC Allok." z.B. auf 200 und das "Min. Public Allok."-Feld auf -1. Auf diese Weise werden Systemstrukturen, die meist kleiner sind als 200 Bytes, in den öffentlichen Speicher gelegt.

Der Zustand "VM benutzen" entspricht einer Einstellung von -1/0 für die "Min Public / min non-Public"-Felder. Der Zustand "VM nicht benutzen" entspricht -1/-1.

Falls ein zu überprüfender Name auf zwei Einträge in der Liste paßt, wird der erste der beiden verwendet. Falls er zu keinem Eintrag paßt, werden die Standardeinstellungen verwendet. Auf diese Weise kann z.B. die muimaster.library virtuellen Speicher verwenden, während ein weiter hinten stehendes "#?.library" allen anderen Libraries den virtuellen Speicher verwehrt.

Falls Sie Probleme haben, die nötigen Einstellungen herauszufinden, können Sie das
Memory Tracking
verwenden, um für jeden Prozeß
herauszufinden, wieviel virtuellen Speicher er verwendet.

1.19 VMM/doc/VMM.guide/Memory_Settings

Dieser Abschnitt bestimmt die Einstellungen für das ↩
Auslagerunggerät
und der Menge Speicher, die für das Auslagern verwendet wird.

Auslagerungsspeicher

Speicherart

Schreibpuffer

VM Priorität

Auslagerungsmedium

Dateigröße

1.20 VMM/doc/VMM.guide/MemType_Gadget

Speicherallokation für die Kacheln:

Es gibt drei verschiedene Strategien zur Speicherallokation:

- Feste Größe: VMM allokiert beim Start soviel Speicher wie im Feld darunter angegeben.

- Dynamisch: VMM allokiert Speicher und gibt ihn wieder frei, wie er zur Laufzeit benötigt wird.

- Eingeschränkt dynamisch: Diese Option funktioniert wie die dynamische Einstellung, außer daß der verwendete Speicher durch die eingestellten Minimal- und Maximal-werte begrenzt wird.

1.21 VMM/doc/VMM.guide/MemFlags_Gadget

Speicherart für Kacheln:

Dieses Feld spezifiziert die Art des Speichers, die VMM zum Auslagern verwendet. Entweder Fast, Chip oder Any können gewählt werden. Normalerweise ist beim Chip-Memory das Caching des Prozessors abgeschaltet, daher können dann auch Kacheln in diesem Bereich nicht gecacht werden. Möglicherweise wird aber bei einigen 68040-Karten das Caching für das Chip-Memory nicht abgeschaltet, sodaß dieses für das Auslagern durchaus verwendet werden kann.

1.22 VMM/doc/VMM.guide/WriteBuffer_Gadget

Schreibpuffer:

Ab V2.1 verwendet VMM einen Schreibpuffer, um mehrere Seiten auf einmal schreiben zu können. Dies beschleunigt den Zugriff sehr stark, da Positionierungen des Kopfes und sonstiger Overhead vermieden werden. Mit diesem Feld können Sie die Größe des verwendeten Schreibpuffers einstellen. Ein Schreibpuffer von 100 bis 200 K genügt in den meisten Fällen.

1.23 VMM/doc/VMM.guide/MemPri_Gadget

VM Priorität:

Hier können Sie einstellen, wann virtueller Speicher allokiert wird. Exec durchsucht alle Speicherlisten in einer bestimmten Reihenfolge, in der normalerweise der Fast-Speicher als erstes mit der Priorität 30 steht. Der Chip-Speicher hat die Priorität -10. Wenn Sie also wollen, daß der virtuelle Speicher zuerst verwendet wird, müssen Sie eine Zahl größer 30 einstellen. Wenn Sie normalen Fast-Speicher zuerst verwenden wollen, aber virtuellen, bevor der Chip-Speicher benutzt wird, verwenden Sie eine Zahl zwischen -10 und 30. Wenn Sie die dynamische Speicherallokation für Kacheln eingestellt haben, sollte der virtuelle Speicher die höchste Priorität im System

erhalten. Sonst ist fast der gesamte physikalische Speicher belegt, und VMM muß den virtuellen Speicher in einem sehr kleinen physikalischen Speicherfenster abbilden.

1.24 VMM/doc/VMM.guide/SwapMedium_Gadget

Auslagerungsmedium:

Es gibt drei Möglichkeiten für das Auslagerungsmedium:

- Auslagern auf eine Partition: Beim Anwählen dieser Option müssen Sie die Partition eingeben, die Sie zum Auslagern verwenden wollen. Wenn eine Partition zum ersten Mal von VMM zum Auslagern verwendet wird, erfolgt eine Sicherheitsabfrage, um ein versehentliches Überschreiben der falschen Partition zu verhindern.

- Auslagern auf eine Datei: Hier müssen Sie eine Dateinamen angeben, der zum Auslagern verwendet wird. Normalerweise ist diese Art deutlich langsamer als das Auslagern auf eine Partition.

- Auslagern auf eine sogenannte Pseudo-Partition: Pseudo-Partitionen sind Dateien, die auf der Festplatte am Stück angelegt werden. Dadurch kann auf Sie genauso schnell wie auf eine normale Partition zugegriffen werden, weiterhin kann sie wie eine normale Datei angesehen und gelöscht werden. VMM erzeugt diese Art von Datei selbständig in der gewünschten Größe. Falls eine Datei bereits existiert, führt VMM beim Start eine Konsistenzprüfung durch.
  Pseudo-Partitionen sind zur Zeit nur mit FFS-Dateisystem mit einer Blockgröße von 512 Bytes möglich. Da Pseudo-Partitionen ihren Platz am Stück benötigen und FFS den Root-Block in der Mitte der Partition anlegt, kann sie höchstens halb so groß wie die Partition sein, auf der sie sich befindet.
  Eine kleine Warnung an dieser Stelle: Dies kann ein gefährliche Option sein. Falls sich in diesem Teil von VMM ein Fehler befindet, kann es passieren, daß unschuldige Daten überschrieben werden. Dieser Teil von VMM ist aber inzwischen gut getestet worden und sollte eigentlich keine Probleme bereiten.

Diese Einstellung und die Größe der Auslagerungsdatei können nicht während des Laufs von VMM verändert werden.

1.25 VMM/doc/VMM.guide/FileSize_Gadget

Größe der Auslagerungdatei:

Dieser Slider bestimmt die Größe der Auslagerungsdatei bzw. der Pseudo-Partition.

Diese Einstellung und das Auslagerungsmedium können nicht während des Laufs von VMM verändert werden.

1.26 VMM/doc/VMM.guide/Misc_Settings

Statistik:

VMM kann ein Statistikfenster öffnen, das Sie über den Zustand des virtuellen Speichers informiert. Unter anderem finden Sie hier die Anzahl der Seitenfehler, die Anzahl der Plattenzugriffen, den freien virtuellen Speicher usw. Dieses Fenster kann ein- und ausgeschaltet werden oder in eine flache Titelzeile verwandelt werden.

Zorro II RAM cachen:

Auf einigen Amigas mit 68040-Karte muß das Caching des RAMs im Zorro II-Bereich abgeschaltet werden. Diese Einstellung wird ignoriert, wenn Sie die VMM_MMU.config Datei verwenden.

VM in WB-Titel anzeigen:

Nach vielen Bitten, die Workbench-Titelzeile zu patchen, um den virtuellen Speicher dort anzuzeigen, habe ich dies implementiert. Da es sich hier jedoch um einen üblen Hack handelt, kann es sein, daß dies unter bestimmten Umständen nicht funktioniert.

Speicher protokollieren:

Wenn diese Option eingeschaltet ist, protokolliert VMM jede Allokation von virtuellem Speicher mit. Dabei wird die Größe der Allokation und der Name der entsprechenden Task gespeichert. Mit dem VMMUsage-Programm ist es möglich, sich eine Liste aller Programme ausgeben zu lassen, die virtuellen Speicher verwenden. Diese Option ist hauptsächlich dafür gedacht, um herauszufinden, unter welchem Namen VMM die Tasks führt. Manchmal ist dies nicht der Name, den man erwartet, und daher eignet sich dies sehr gut, um kooperationsunwillige Programme zur Zusammenarbeit zu bewegen.
Wenn Sie diese Einstellung während des Laufs ausschalten, wird der Speicher, der zum Protokollieren verwendet wird, nicht wieder freigegeben.

Fast ROM:

Das Einschalten dieser Option veranlaßt VMM, das ROM ins Fast-RAM zu kopieren und MMU-Tabellen so anzupassen, daß diese Kopie statt des ROMs verwendet wird. Dies beschleunigt den Zugriff auf Systemroutinen oft erheblich. Diese Option wurde implementiert, da einige Tools mit FastROM-Option nicht mit VMM zusammenarbeiten. Diese Option wird ignoriert, wenn Sie eine VMM_MMU.config Datei verwenden.

Minimale VM Allokation:

Um die AllocMem-Routine zu beschleunigen, die vom gesamten System sehr häufig verwendet wird, wird für Allokationen, die kleiner sind als der hier angegebene Wert normales Fast-RAM verwendet. Wenn Sie den Wert auf 0 setzen, wird jede mögliche Allokation in den virtuellen Speicher gelegt. Ein Wert von 100 bis 200 stellt einen vernünftigen Kompromiß zwischen der Speicherausnutzung und der Geschwindigkeit dar.

Hotkeys:

Es gibt zwei zusätzliche Hotkeys, die es Ihnen einfach erlauben, die Allokation von virtuellen Speicher zwischendurch zu verbieten bzw. wieder einzuschalten. Die Einträge müssen den normalen Commodities-Spezifikationen genügen. Der Popup-Hotkey kann nur über die Kommandozeile oder die Tooltypes einstellt werden.

1.27 VMM/doc/VMM.guide/PROC_DIFFS

6. PROZESSORABHÄNGIGE UNTERSCHIEDE:

Es gibt einige Unterschiede in VMM bzgl. der Behandlung des virtuellen Speichers wegen der verschiedenen Prozessortypen, die unterstützt werden. Auf der einen Seite sind zwischen 68020+68851 und dem 68030 kaum Unterschiede, während auf der anderen Seite der 68040 und 68060 sehr ähnlich sind.

Zuerst einmal installiert VMM auf dem 68030 fast immer seine eigene MMU-Tabelle. Dies liegt daran, daß die Standard-Einstellung entweder überhaupt keine Tabelle oder eine mit einer anderen Seitengröße verwendet. Dies bedeutet, daß auf dem 68030 VMM nicht mit Enforcer zusammenarbeitet. Ab V2.1 enthält VMM einen kleinen undokumentierten Hack, der verhindert, daß bei Softkick-Amigas nach einem Absturz das Kickfile neu geladen werden muß.

1.28 VMM/doc/VMM.guide/PROBLEMS

7. SCHWIERIGKEITEN

Commodore definierte das MEMF_PUBLIC-Flag für die AllocMem-Funktion vor einer langen Zeit, als niemand sich vorstellen konnte, wozu dies gut sein könnte. Daher verwenden viele Programme dieses Flag entweder immer oder gar nicht. Im ersten Fall passiert nichts Schlimmes, nur verwendet dieses Programm nie virtuellen Speicher. Im zweiten Fall kann es aber passieren, daß das entsprechende Programm abstürzt. Es gibt eine Menge Programme, die IORequests o.ä. auf dem Stack allokieren. Falls dieser im virtuellen Speicher liegt, kann das einen Absturz hervorrufen. Solchen Programmen kann man verbieten, virtuellen Speicher zu verwenden. Wenn Sie selbst Programme schreiben, die auch mit VMM zusammen noch funktionieren sollten, sollten Sie die Datei "VMProgGuideline" lesen, um bestimmte Dinge in Programmen mit virtuellem Speicher vermeiden zu können.
Festplattencaches wie FastCache oder PowerCache sollten keinen virtuellen Speicher erhalten, da dies wohl wenig Sinn macht. Das gleiche gilt für alle Programme, die den BeginIO-Vektor des Auslagerungsgeräts verändern. Wenn Sie Software-Festplattencaches verwenden, muß möglicherweise auch den Dateisystem-Tasks (DH0, DH1,..) die Verwendung von virtuellem Speicher untersagt werden.
Programme, die den Seitenfehlervektor verändern (z.B. Enforcer), müssen vor VMM gestartet werden, da Enforcer sonst alle Seitenfehler als ungültige Speicherzugriffe anzeigt.

Das Code-Paging sollte nicht für Programme verwendet werden, die Input- oder Interrupt-Handler beinhalten. Als Beispiel mag hier die 'ixemul.library' dienen.

1.29 VMM/doc/VMM.guide/TROUBLESHOOTING

8. HÄUFIG AUFTRETENDE PROBLEME

Frage : Programm "X" verwendet keinen virtuellen Speicher. Warum?
Antwort: Möglicherweise setzt das Programm bei jedere Allokation das MEMF_PUBLIC-Flag. Um dies zu umgehen, verwenden Sie die Zusatzoptionen.

Frage: VMM stürzt mit meiner Konfiguration ab.
Antwort: Es gibt zwei Möglichkeiten, warum dies passiert. Wenn VMM sofort beim Start oder beim ersten Festplattenzugriff abstürzt, handelt es sich meist um ein Problem mit der MMU-Tabelle. In diesem Falle lesen Sie

MMU setup
.

Die andere Möglichkeit ist, daß sich in Ihrer Umgebung ein Programm befindet, daß keinen virtuellen Speicher verträgt. Wenn dies der Fall ist, tun Sie Folgendes:
Setzen Sie Standardeinstellung auf "VM nicht benutzen" und stellen Sie die Einstellung für den für VMM verfügbaren physikalischen Speicher auf das Minimum, damit VMM gezwungen ist, häufig Seiten auszulagern . Dann erlauben Sie nacheinander jeder Task in Ihrem System, virtuellen Speicher zu verwenden. Auf diese Weise könne Sie feststellen, welches Programm keinen virtuellen Speicher verträgt. Anschließend können Sie für dieses Programm den virtuellen Speicher verbieten. Zusätzlich können Sie auch die Option

Speicher protokollieren
verwenden.

Frage: VMM hält das System an, wenn auf eine Partition zugegriffen wird, die sich auf der gleichen physikalischen Platte befindet wie die Auslagerungpartition. Warum?
Antwort: Wahrscheinlich verwendet Ihr Festplattentreiber DMA-Zugriffe, ohne bestimmte Systemroutinen korrekt aufzurufen. Als Abhilfe können Sie mit HDToolbox den Maskenparameter für alle Partitionen auf dieser Platten umsetzen. Wenn Sie nicht wissen, was der Maskenparameter ist, setzen Sie ihn auf 0xffffffe, auf diese Weise werden DMA-Zugriffe auf die unteren 16 MB beschränkt. Dieser Fehler sollte ab V2.0 nicht mehr auftreten, wenn doch, so schicken Sie mir eine Mail.

Frage: Mein Festplattencache funktioniert nicht, obwohl er keinen virtuellen Speicher erhält. Warum?
Antwort: Einige Programme verändern Code, der von anderen Tasks ausgeführt wird. Z.B. verändert DynamiCache den BeginIO-Vektor der gecachten Geräte, sodaß alle Allokationen für die Caches nicht von DynamiCache selbst, sondern von den Dateisystem-Tasks durchgeführt werden. Wenn dies ein Problem ist, müssen

Sie für alle Tasks, die den veränderten Code ausführen, den virtuellen Speicher ausschalten. Wegen eines Problems im Zusammenhang mit dem Copyback-Cache von Powercache siehe auch
Bekannte Fehler
.

Frage: Das System stürzt ab, wenn man in der VMM-Oberfläche "Speichern", "Benutzen" oder "Abbrechen" anklickt.
Antwort: Es kann sein, daß Ihre MMU nicht korrekt funktioniert. Leider gibt es keinen Möglichkeit, dies durch Software zuverlässig herauszufinden. Sie müssen das Gerät öffnen und sich die Bezeichnung Ihres Prozessors ansehen. Falls sich in dieser die Buchstaben "EC" befinden, enthält er keine voll funktionsfähige MMU und VMM läuft nicht. Motorola verkauft die Prozessoren, deren MMUs den Endtest nach der Produktion nicht bestanden haben als EC-Typen. Es kann sich dabei um einen nur kleinen Defekt handeln, sodaß die MMU für die meisten Programm voll funktionsfähig erscheint. Falls die MMU nicht defekt ist, kann es sich um ein Problem mit der MMU-Tabelle handeln (s.
MMU-Tabelle
).

Frage: Was sind die Vor- und Nachteile von 4K und 8K-Seiten?
Antwort: Normalerweise sollten Sie der Empfehlung des Installationsprogrammes bzgl. der Seitengröße folgen. Es kann aber durchaus sein, daß VMM bei Ihnen auch mit einer anderen Seitengröße funktioniert. Bei 4K-Seiten ist die Rate der Seitenfehler bei Standardapplikationen normalerweise geringer als mit 8K-Seiten. Wenn Sie häufig Bildbearbeitung mit AdPro oder ähnlichen Programmen betreiben, können 8K-Seiten deutlich schneller sein. Dies gilt generell, wenn Sie mit Programmen arbeiten, die auf den Speicher in einer linearen Weise zugreifen.

1.30 VMM/doc/VMM.guide/TECHNICAL_DES

9. TECHNISCHE BESCHREIBUNG

VMM besteht aus drei Tasks, beim Einschalten des Statistikfensters wird eine vierte erzeugt. Die erste ist der VM_Manager, der entscheidet, welcher Prozeß virtuellen Speicher erhält, usw. Der PageHandler führt das Aus- und Einlagern von Seiten durch, wenn ein Seitenfehler aufgetreten ist. Der Prepager verankert Seiten im physikalischen Speicher, wenn IO von oder zur gleichen Platte ausgeführt wird, auf der die Auslagerungspartition liegt. Der meiste Aufwand wurde nicht dafür betrieben, das Auslagern lauffähig zu machen, sondern für die korrekte Behandlung von virtuellem Speicher in allen Situationen. Leider hat Commodore sich seinerzeit recht wenig Gedanken zum Thema virtueller Speicher gemacht, sodaß einige Systemfunktionen verändert werden mußten. Ich habe versucht, VMM so systemkonform wie möglich zu halten, aber leider mußte ich einige Annahmen über undokumentiertes Verhalten machen. Die übelste davon

ist die Tatsache, daß ich die Switch-Funktion patchen mußte, die eine Task vom laufenden Zustand in Bereit-Zustand versetzt. Dies bedeutet, daß VMM evtl. mit zukünftigen Versionen des Betriebssystems nicht arbeitet, obwohl wohl in diesem Basiscode nichts mehr geändert werden wird.
Außerdem hat Commodore nie etwas darüber veröffentlicht, ob man auf nicht als Public gekennzeichneten Speicher innerhalb eines Forbid/Permit bzw. Disable/Enable zugreifen darf. Wenn in diesem Bereich ein Seitenfehler auftritt, kann es zu gefährlichen Nebenwirkungen kommen, da durch einen Seitenfehler ein Taskwechsel erfolgt. Speicher, der innerhalb eines solchen Bereiches freigegeben wird, wird erst nach dem Verlassen dieses Abschnitts wirklich freigegeben.
Momentan ist die Anzahl der gleichzeitig in Bearbeitung befindlichen Seitenfehler plus der Anzahl der Tasks, deren Stack im virtuellen Speicher liegt, auf 20 begrenzt.

Der VM_Manager Prozeß

Der VM_Manager startet alle anderen Tasks und initialisiert alles. Er behandelt auch das Beenden von VMM. Jedes Mal, wenn AllocMem aufgerufen wird, muß VMM entscheiden, ob der aufrufende Prozeß berechtigt ist, virtuellen Speicher zu verwenden. Wenn ein Prozeß zum ersten Mal AllocMem aufruft, wird eine Nachricht an den VM_Manager geschickt, der entscheidet, ob diese Task virtuellen Speicher erhält. Weitere Allokationen dieser Task werden über eine Hashtabelle entschieden, um die Allokation schnell zu halten.
Außerdem behandelt der VM_Manager alle Nachrichten, die durch das Drücken eines Hotkeys oder durch das Exchange-Programm erzeugt werden.

Die PageHandler-Task

Alle Aus- und Einlagerungen werden asynchron vom PageHandler bearbeitet. Wenn ein Seitenfehler auftritt, werden all dessen Parameter in eine sogenannte Trapstruktur aufgenommen. Diese Struktur wird von dem im Ausnahmezustand arbeitenden Traphandler an den Pagehandler geschickt. Der PageHandler sucht eine zu ersetzende Seite aus und schreibt diese auf die Platte, falls diese verändert wurde. Zur Auswahl der Seite wird der sogenannte Second Chance-Algorithmus verwendet. Nachdem eine Seite verdrängt wurde, wird die angeforderte Seite eingelesen. Anschließend wird die Task, die den Seitenfehler verursacht hat, benachrichtigt, und kann weiterarbeiten. Während der Plattenzugriffe können andere Tasks laufen und währenddessen auch weitere Seitenfehler verursachen, die sofort behandelt werden.
VMM verwendet einen Schreibpuffer für die Seiten, die auf die Platte geschrieben werden müssen. Hier werden die Seiten gesammelt, bis der Puffer voll ist, um diese dann in einem Rutsch zu schreiben. Dies senkt die zum Auslagern benötigte Zeit deutlich.

Die Prepager-Task

Während ich VMM schrieb, habe ich einige Fälle entdeckt, in denen auf eine andere Partition der gleichen Platte zugegriffen wird, auf der sich die Auslagerungpartition befindet. Da der Gerätetreiber für das

Auslagerungsgerät nie wegen eines Seitenfehlers blockieren darf, muß dies vermieden werden. Sämtliche Ein- und Ausgaben an dieses Gerät werden daraufhin untersucht, ob sie virtuellen Speicher verwenden. In diesem Falle wird der Block vorher umkopiert und die Ein-/Ausgabe mit diesem Puffer ausgeführt.

Die Statistik-Task

Die Statistik-Task wird nur erzeugt, wenn das Statistikfenster eingeschaltet ist. Jede Sekunde werden einige Zeilen über den freien virtuellen Speicher usw. ausgegeben.

Gepatchte Systemfunktionen

Die folgenden System-Funktionen werden von VMM gepatcht: Switch, AddTask, Wait, AllocMem, FreeMem, AvailMem, CachePreDMA, CachePostDMA, LoadSeg und NewLoadSeg.
Auf 68030-Systemen wird zusätzlich ColdReboot gepatcht, um den ursprünglichen Zustand der MMU vor dem Reset wieder herzustellen.
Außerdem wird noch der BeginIO-Vektor des Auslagerungsgeräts gepatcht.
Die Switch-, Wait- und AddTask-Funktionen mußten gepatcht werden, da sich der Stack einer Task möglicherweise im virtuellen Speicher befindet. Um Seitenfehler im Ausnahmezustand (Taskumschaltung) zu verhindern, wird der Stack kurzzeitig durch einen Bereich ersetzt, der groß genug ist, alle Register während einer Kontextumschaltung aufzunehmen. Wenn die Task weiterarbeitet, wird der Originalstack restauriert.
Wenn das Anzeigen des virtuellen Speichers in der Titelzeile des Workbench-Screens eingeschaltet ist, wird zusätzlich die SetWindowTitles-Funktion gepatcht.

MMU-Tabelle

1.31 VMM/doc/VMM.guide/MMU_SETUP

DIE VON VMM VERWENDETE MMU-TABELLE

Falls möglich versucht VMM eine bereits existierende MMU-Tabelle anzupassen. In manchen Fällen geht dies aber leider nicht, wenn z.B. eine andere Seitengröße oder bestimmte MMU-Register verwendet werden (Für Experten: Die sogenannten Transparent Translation Register).
Einige Prozessorkarten wie das GForce ’040 von GVP, das FusionForty Board und einige GVP-Harddisk-Controller verwenden Adreßbereiche, die von Commodore als reserviert definiert wurden. Es gibt leider keinen Weg, so etwas per Software herauszufinden.
Damit man mit diesen Karten VMM nutzen kann, habe ich einen speziellen Mechanismus entwickelt, mit dem man VMM seine MMU-Einstellung mitteilen kann. Es gibt ein Programm namens ’ReadMMUConfig’, das die aktuelle MMU-Einstellung für den gesamten 4 GB-Adreßbereich auslesen kann. Diese Information wird in eine Datei geschrieben, die von VMM

beim Start gelesen wird. Diese Datei enthält für jeden kontinuierlichen Adreßbereich eine Zeile. VMM erzeugt daraufhin eine identische Tabelle mit der richtigen Seitengröße, ohne die Spezial-Register zu verwenden.
Ab V3.1 ist diese Tabelle dynamisch, d.h. der Speicher für die Seitentabellen wird erst dann allokiert, wenn ein Zugriff auf eine Seite dieses Bereichs stattfindet. In älteren Version kam es dazu, daß die MMU-Tabellen für die kompletten 4GB, d.h. 4 MB beim Start angefordert wurden. Unter Umständen kann es trotzdem erforderlich sein, diese Datei zu editieren, daher eine kurze Beschreibung der Einträge:

Jede Zeile enthält die folgenden vier Einträge:

Log. Start-Adresse | Phys. start-Adresse | Blocklänge | Flags

Dies bedeutet, daß ein Block ab 'log. Startadresse' auf einen Block der Länge 'Blocklänge' ab Adresse 'phys. Startadresse' abgebildet wird. Die Flags sind wie folgt definiert:

Bit 0 muß immer gesetzt sein
Bit 1 muß immer gelöscht sein
Bit 2 muß gesetzt sein, wenn dieser Bereich schreibgeschützt sein soll.
Bit 3 und 4 werden ignoriert
Bit 5 und 6 bestimmen den Cache-Modus wie folgt:

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

Wenn Sie VMM nicht zum Laufen bekommen, indem Sie diese Datei ändern, schicken Sie mir eine Mail mit der Ausgabe von 'ReadMMUConfig', der Ausgabe von ShowConfig o.ä. und ich werde versuchen, Sie bei der Erstellung einer passenden MMU-Tabelle zu unterstützen.

1.32 VMM/doc/VMM.guide/VMM_LIBRARY

10. VMM.LIBRARY

Es wurde eine Library für VMM entwickelt, die es erlaubt, Programme speziell für VMM zu entwickeln. Sie enthält die Funktionen AllocVMem, FreeVMem, AvailVMem, AllocVVec and FreeVVec. Wenn die Library geladen wird, wird VMM automatisch gestartet, aber nur Programme, die über die vmm.library Speicher allokieren, bekommen virtuellen Speicher. Wenn Sie sowohl die vmm.library als auch VMM auf dem normalen Wege nutzen wollen, müssen Sie zusätzlich VMM wie gewohnt starten. Momentan wird der PageHandler nicht beendet, wenn die Library aus dem Speicher entfernt wird. Eine (englischsprachige) Dokumentation der Library-Funktionen im Autodoc-Format ist vorhanden.

1.33 VMM/doc/VMM.guide/EXT_PROGS

11. EXTERNE PROGRAMME

Ab V2.1 ist es möglich, eigene Programme zu schreiben, die die Statistikinformationen von VMM auswerten bzw. darstellen. Es gibt eine C-Include-Datei, die die Message-Struktur beschreibt, die von VMM verwendet wird, um den Status mitzuteilen. Weitere Informationen finden Sie dort. Wenn Sie ein Programm schreiben, daß diese Schnittstelle verwendet, können Sie es mir schicken, damit ich es bei der nächsten Veröffentlichung VMM beilegen kann.
Ab V3.0 ist es zusätzlich möglich, auch die Information über die einzelnen VM-verwendenden Tasks von VMM zu erfragen. Falls Sie VMMUsage also nicht mögen oder eine kombinierte Anzeige für Statistik und Speicherverbrauch einzelner Tasks wünschen, können Sie ein entsprechendes Programm schreiben. Die nötige Information entnehmen Sie bitte der o.g. Include-Datei.

1.34 VMM/doc/VMM.guide/KNOWN_BUGS

12. BEKANNTE FEHLER

So weit mir bekannt befinden sich keine größeren Fehler mehr in VMM. Es gibt jedoch nach wie vor einen kleineren Fehler, der noch behoben werden muß. Wenn eine Task via RemTask von einer anderen Task terminiert wird, und deren Stack in virtuellem Speicher liegt, wird die zugehörige Trap-Struktur nicht freigegeben. Commodore empfiehlt sowieso, RemTask nicht auf andere Tasks anzuwenden, sodaß dies keine ernstzunehmende Beschränkung darstellt. Um dieses Problem zu umgehen, kann man der Task den virtuellen Speicher verbieten, die den Stack allokiert.

Es kann passieren, daß VMM wegen Speichermangel nicht starten kann. In diesem Fall versucht es einen Requester öffnen, der diesen Grund angibt. Falls aber selbst hierfür kein Speicher mehr vorhanden ist, gibt es keine Ausgabe. Dies liegt daran, daß man bei der Verwendung von EasyRequestArgs nicht herausfinden kann, ob der Requester wirklich geöffnet werden konnte. Zukünftig sollte hier ein Alert ausgegeben werden.

Es gibt ein merkwürdiges Phänomen im Zusammenhang mit dem console.device. Wenn man ein CLI-Fenster öffnet, bleibt der Cursor darin manchmal grau, obwohl das Fenster aktiv ist. Indem man ein anderes Fenster aktiviert und dann wieder das ursprüngliche wird der Cursor aktiviert. Anscheinend allokiert die CON-Task ihren Stack im virtuellen Speicher und ruft dadurch dieses Problem hervor. Wenn man 'CON' den virtuellen Speicher verbietet, verschwindet dieses Problem.

Es gibt einen Fehler im Zusammenhang mit dem Copyback-Modus von Powercache. Es können dadurch Dateiinhalte zerstört werden. Dies ist eher ein Fehler in Powercache als in VMM. In der aktuellen Powercache-Version (37.115) gibt es zwei Möglichkeiten, dies zu umgehen: Entweder müssen Sie die Copyback-Option von Powercache ausschalten oder Sie müssen in HDToolbox die Maske für alle

Partitionen, die von Powercache gecacht werden, auf 0x0fffffff setzen. Dies beschränkt alle Speichertransfers von und zur Platte auf die unteren 256 MB und vermeidet damit den virtuellen Speicher.

Es gibt einen seltsamen Fehler beim Kopieren großer Dateien über die Workbench. Wenn Sie eine große Datei (mind. 10 MB) nach RAM: kopieren wollen, indem Sie die Workbench dazu verwenden, kann dies aus Speichermangel fehlschlagen, obwohl genügend Speicher vorhanden ist. Wenn man das gleiche über das CLI versucht, funktioniert alles. Das liegt daran, daß die Workbench einen Riesenpuffer zum Kopieren verwendet und anschließend für das Ziel nicht genügend Platz frei hat. Dieser Fehler tritt auch ohne VMM mit genügend Speicher auf.

VMM patcht den LoadSeg-Vektor um Code-Paging zu ermöglichen. Leider kann dieser Patch nicht einfach die ursprüngliche LoadSeg-Funktion aufrufen, nachdem seine Arbeit getan ist. Daher werden vorherige Patches dieser Funktion ignoriert, nachdem VMM gestartet ist. Wenn z.B. Segtracker vor VMM gestartet wird, wird dessen Patch für alle Programme, deren Code in den virtuellen Speicher geladen wird, nicht durchgeführt. Dies liegt daran, daß das Betriebssystem leider oft seine eigenen Funktionen nicht über die öffentlichen Schnittstellen, sondern über interne Sprünge aufruft.

1.35 VMM/doc/VMM.guide/BUG_REPORTING

13. FEHLERMELDUNGEN

Falls Sie einen Fehler melden wollen oder eine Verbesserung vorschlagen möchten, verwenden Sie bitte das Fehlermeldungsformular aus diesem Archiv. Ich kann häufig Fragen nicht beantworten, da ich nicht weiß, welche Version von VMM jemand verwendet, welche Hardware vorhanden ist etc. Daher ist das Fehlermeldungsformular sehr wichtig für mich. Sie können meine Adresse unter Verschiedenes "MISCELLANEOUS"} finden.

1.36 VMM/doc/VMM.guide/ACKNOWLEDGMENTS

14. DANKSAGUNGEN

Ich möchte den folgenden Leuten für Ideen, Verbesserungsvorschläge und Testen der Beta-Versionen danken (in alphabethischer Reihenfolge):

Michael Berg	(Dänische Übersetzung)
Jürgen Barthmann	
Wayne Cole	
Torsten Ebeling	
Markus Eiblmeier	
Sven Fischer	
Frank Grimm	
Magnus Holmgren	(Schwedische Übersetzung)
Oleg Khimich	(Russische Übersetzung)
Robert Kiehne	

Jeff Koons
Steve Koren
Andree Maedl
Manfred Matzinger
Barry McConnell
Marco Musso (Italienische Übersetzung)
Paul Ney
Stefan Odendahl
Hans Otten
Volker Rudolph (für die Unterstützung bei der Portierung auf den 68030)
Stefan Schmidt
Torsten Stolpmann
Erno Tuomainen
Emmanuel Vacher (Französische Übersetzung)
Nikola Vukovljak
Alg Inge Wang
Christian Wasner
Juergen Zimmermann

Ich möchte auch all denen danken, die mir Wünsche und Fehlermeldungen geschickt haben. Ohne sie würde VMM nicht so gut funktionieren.

1.37 VMM/doc/VMM.guide/MISCELLANEOUS

15. VERSCHIEDENES

Ich würde mich freuen von Ihnen zu hören, ob VMM bei Ihnen läuft und welche Programme Probleme mit VMM bereiten. Wenn Sie einen Fehler melden, benutzen Sie BITTE das Fehlermeldungsformular aus diesem Archiv. Da die meisten Fehler sehr hardwareabhängig sind, benötige ich Ihre Konfigurationsdaten, um herauszufinden, was schiefläuft.

email: apel@tecmath.de

Adresse: Martin Apel
Konrad Adenauer Str. 7
67663 Kaiserslautern
Germany

Telefon: 0631 / 24257

Bankverbindung: 145 009 494
Sparkasse Bonn
BLZ 380 500 00