



Breaking the 2Gb barrier

*Enough physical RAM for all your simultaneous programs,
and enough user memory for one single program,
really are separate things. Solving one will not solve the other.*

Contents

Running out of Memory?.....	3
Monitoring User Memory.....	5
Raising Large Address Awareness.....	7
Enabling Large Address Usage.....	13
Physical and Virtual Memory.....	18

Oct 2010
Updated Aug 2011

Running out of Memory?

When you're running a 32-bit program (note 1), you might experience running out of memory events (note 2). It can happen especially when rendering 3D scenes, or rendering media output for video or music. This can make the program crash, it can make it loose functionality, or it can cripple the results.

You may overcome this problem to some extent (note 3), by adjusting the program itself (note 4).

Notes:

1. when you're not sure that your program is 64-bit, then it's most probably 32-bit
2. see the Monitoring User Memory section in this tutorial
3. by default, each 32-bit program is granted 2Gb User Memory. This can be increased up to 3Gb. That's it. In case you need more, you've got to go for 64-bit programs and thus a 64-bit Operating System as well.
4. see the Raising Large Address Awareness section in this tutorial.
Note that Poser 8, Daz Studio 3 and Carrara 7 and up might have their Large Address Awareness already raised by the supplier and do not require further enhancement. Vue has not raised LAA and does need your attention.

When you're running the program in a 32-bit Windows environment (note 5), you will have to adjust the Windows system settings as well (note 6). This comes at a price. By assigning more memory to user programs, there is less available for system routines. This may slow down some operations, like massive data transfers between disks, or over the network. When you're running a 32-bit program in a 64-bit environment, only the program itself might need an adjustment (note 7).

Notes:

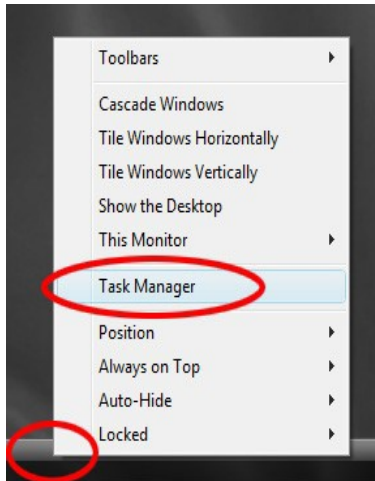
5. when you're not sure that your Windows is 64-bit, then it's most probably 32-bit.
64-bit Windows versions exist for: XP Pro, Vista and Win7 Home Premium, Professional and Ultimate
6. see the Enabling Large Address Usage section in this tutorial.
Note that Mac, Linux, and 64-bit Windows environments do have Large Address Usage enabled by default, so it's a 32-bit Windows thing only.
7. see the Raising Large Address Awareness section in this tutorial

The important thing in this is that's all about program and system settings. Running out of memory has NOTHING to do with the amount of physical RAM in your box, so increasing or decreasing RAM might bring performance effects (note 8), but will NOT affect the issue above.

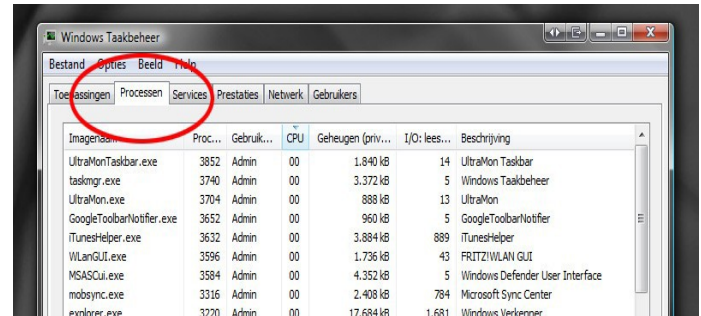
Notes:

8. see the Physical and Virtual Memory section in this tutorial

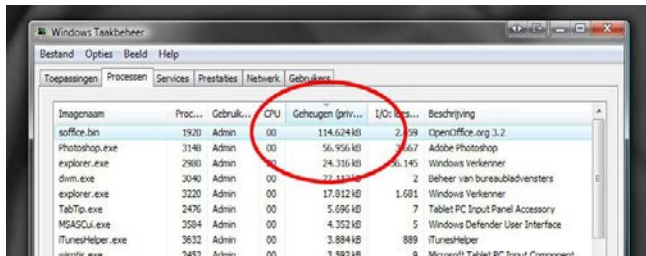
Monitoring User Memory



With a Right-click of your mouse on the Taskbar, you can open Taskmanager.



The Processes tab will show CPU- and memory usage, amongst others.

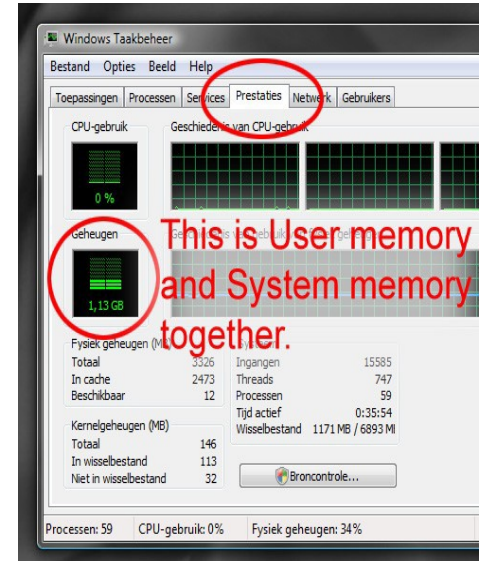


Click twice on the Memory bar on top to get the most hungry program on top. The memory shown is the User Memory, which for 32-bit programs should not exceed 2Gb unless the measures are taken which are described in this tutorial.

Next to User Memory, there is something like System Memory. Those taken together make up the total memory usage, as shown in the Performance tab of Taskmanager.

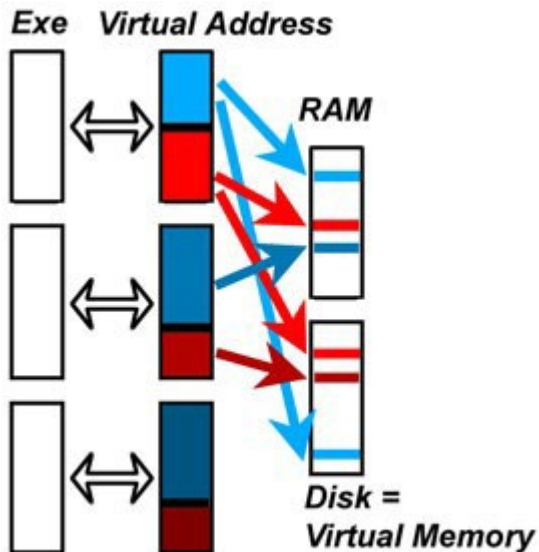
Since the Taskmanager tends to stay on top of all windows, you can see whether the occurrence of some program issues coincide with exceeding the 2Gb boundary. If so, you might profit from the measures described in this tutorial. If not, there is no need to mess around with program and system settings.

Minimizing the Taskmanager will give you some CPU-indicator on the Taskbar. A double click on this icon reopens the Taskmanager. This way, you can work on and have Taskmanager at hand when required.



Raising Large Address Awareness

Each individual program in a Windows environment (and in other environments as well) can access two kinds of memory; System Memory and User Memory.



System Memory (red in schema) contains the program code and various settings and tables handled by Windows. In this memory area, only Windows can read as well as write (to load the program), while the program itself in that area can read only. This is to protect systems against viruses, against self-modifying code and against other potential threatening program behavior.

User Memory (blue in schema) is the area where the program itself is allowed to read and write, to store and retrieve it's intermediate values and results from user actions. When the program is up to something massive, this area is blown to pieces.

While each 32-bit program – without any exception – can deal with a maximum 4Gb of memory in total, most programs are created in such a way that 2Gb is the maximum amount of user

memory they can handle, even when they are assigned more (top in schema, blue and red areas equal in size). Unless the program is made “Large Address Aware” (or: LAA) in which case those kinds of limits are off

(bottom two programs in schema, blue area is larger than red). This can be done at creation time, by the supplier, and happens more and more. It can also be done at production time, by you on your PC, and requires a special piece of software, like “LaaTiDo”.

The zip which included this PDF also includes the LaaTiDo program.

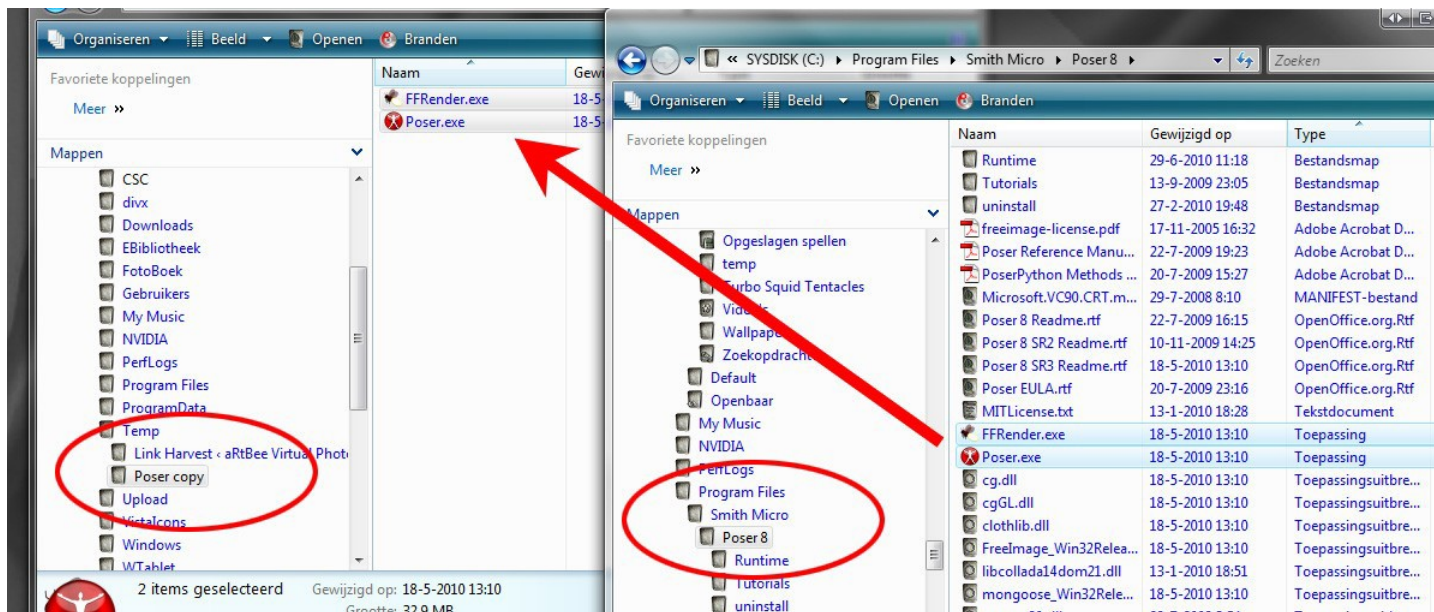
LaaTiDo makes slight changes to your software. As this has some risks involved, you make copies of the original first, of course. An issue is, that each time a new version (update, service pack, ...) of that software is installed (from upgrading or whatever), you have to deal with the new executable all over again.

The following programs: Poser 8, Daz Studio 3, Carrara 7 and 8, all Adobe Elements 9 and CS5 are known to be LAA, and do not require further treatment, while Bryce 7 and earlier, all Vue versions, Photoshop 7, Paintshop Pro X2 and earlier and known NOT to be LAA and do require this enhancement. I've no knowledge yet on Poser 7 and before, Daz Studio 2, Photoshop CS versions (although all recent Adobe programs are LAA already) and the latest PaintShopPro X3. Anyway, for Poser, Daz Studio and Carrara upgrading is a recommended alternative for fiddling with the program's executable.

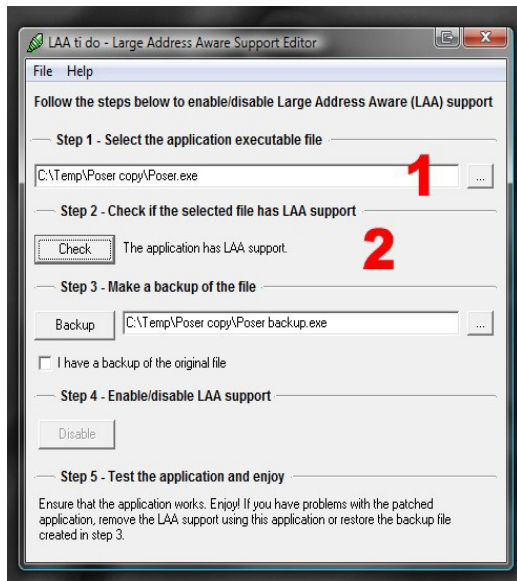
How to proceed?

First of all, you have to make sure you need this kind of solution. Is the problem you're facing caused by the 2Gb User Memory boundary indeed? Read the previous Monitoring User Memory section about it.

Second, you've to locate all the relevant executables in your Program Files folder, and copy (not move!) them to a place to work them, like a new folder in your Temp directory.



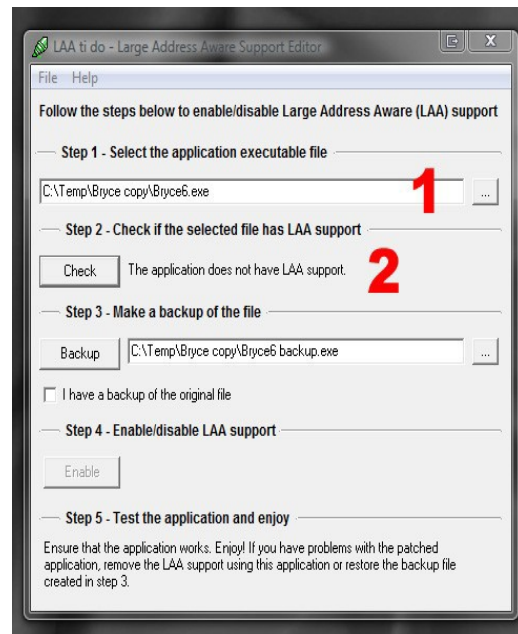
This is because you (and Windows) don't want LaaTiDo or any other application to write in your Program Files directly.

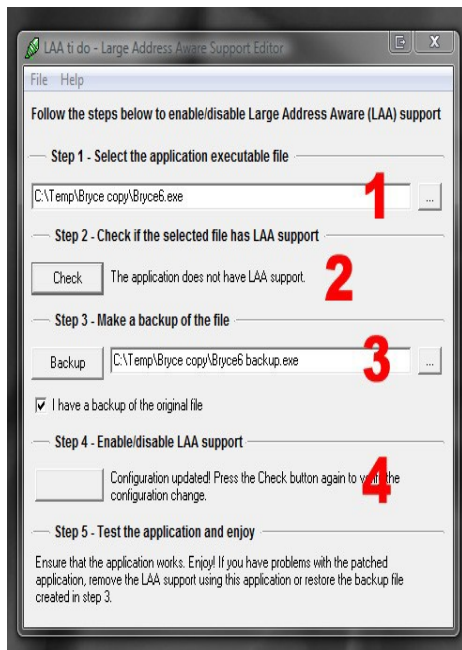


Third, start LAA TiDo and open one of those executables for inspection (1), then click the [Check] button (2). Is it Large Address Aware (LAA)?

Then you're done, and test other executables. You may find that when a supplier made the program LAA, all relevant executables are so already.

When the supplier did not (1, 2), none of those are and you have to make them so.



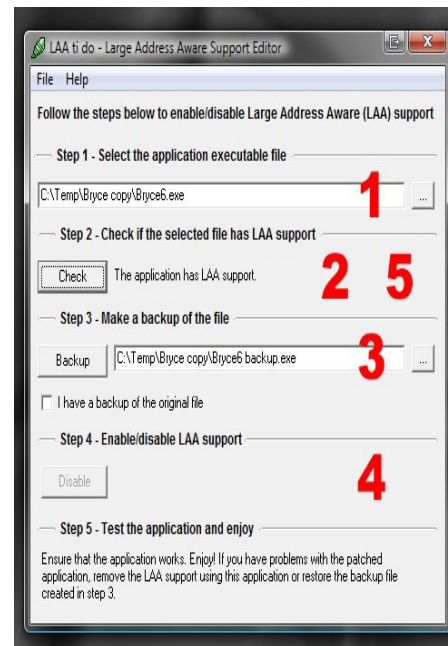
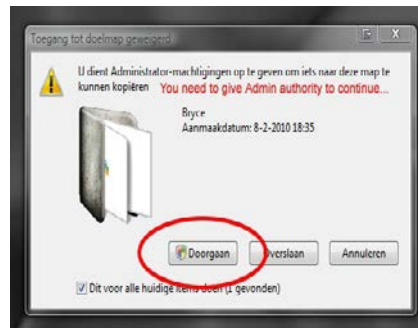


This is manual action, and Windows will ask you for confirmation.

Forth, for non-LAA executables (1), you [Check] (2), then [backup] (3) and then click the [Enable] button (4). That's it, and you're almost done.

Repeating the previous Check (5) will show you that the new executable is LAA aware indeed now.

Finally, you move the adjusted executables AND their respective unchanged copies (!) back to the Program Files environment, overwriting the existing ones.



Now you can test your adjusted program, in simple conditions. Does it seem to work? Then you're fine. Does it fall apart instantly? Then you have to step back, and transfer your backed up unaltered copies onto the altered ones or so, as the LAA progress is not working for them. That means end of the road indeed, stepping towards all 64-bit software is the only way to go now.

When it does seem to work, and you're in a 64-bit Windows environment, or a 32-bit environment which is set ready before, then you're done indeed. If not, then all you've got to do is to adjust your 32-bit Windows as well. The next section tells you how.

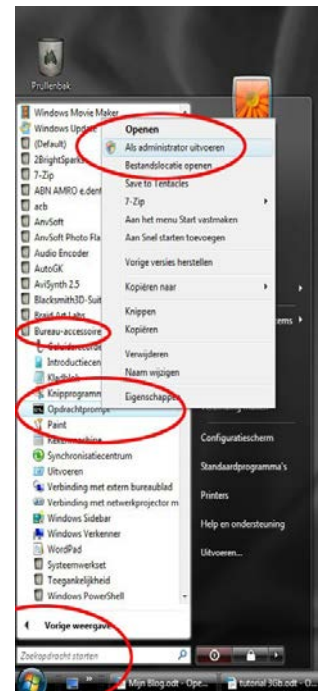
Enabling Large Address Usage

While each 64-bit operating system is enabled to support Large Addresses by definition, and various 32-bit operating systems (like most Unix/Linux variants) are enabled by default too, Windows is not.

Not being enabled implies that the halfway split of 2Gb System Memory and 2Gb User Memory still holds even for LAA (Large Address Aware) programs. Non-LAA programs will fall over when requesting more than the 2Gb maximum, LAA-programs will fall over as the non-enabled 32-bit Windows will not fulfill their request. You won't know the difference. So besides the program being made LAA, you'll need to enable Large Address Usage in Windows.

This is how, in Vista and up (Win7, ...).

Go to Accessoires, and open the CommandPrompt with Admin rights.



```
Administrator: Opdrachtprompt
Microsoft Windows [versie 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. Alle rechten voorbehouden.
C:\Windows\system32>bcdedit
Windows-opstartbeheer
id                {bootmgr}
device            partition=C:
description       Windows Boot Manager
locale            nl-NL
inherit           {globalsettings}
default           {current}
displayorder      {current}
toolsdisplayorder {memdiag}
timeout          0
resume            No

Windows-opstartlaadprogramma
id                {current}
device            partition=C:
path              \Windows\system32\winload.exe
description       Microsoft Windows Vista
locale            nl-NL
inherit           {bootloadersettings}
osdevice          partition=C:
systemroot        \Windows
resumeobject      {f26ched5-dd34-11db-b487-f4ea3d959cc7}
nx                OptIn
C:\Windows\system32>
```

Then you type: bcdedit and {Enter}. This presents just a readout of your Windows settings. Note the blue arrow, there is no additional info below OptIn.

Then you type: bcdedit /set increaseuserva 2900 and {Enter}.

You can check the result by typing bcdedit {Enter}, and you'll find the variable increaseuserva (NB: Increase User Virtual Address)

with the 2900 value. You can do the readout first, to check whether there is something set already.

2900 means 2900MB, or almost 3Gb. The value 3072 (MB, or 3Gb exactly) is the maximum to use, but you can go lower as well. The value chosen is your new User Memory limit, MicroSoft disadvicees against values below 2800.

```
Windows-opstartbeheer
id                {bootmgr}
device            partition=C:
description       Windows Boot Manager
locale            nl-NL
inherit           {globalsettings}
default           {current}
displayorder      {current}
toolsdisplayorder {memdiag}
timeout          0
resume            No

Windows-opstartlaadprogramma
id                {current}
device            partition=C:
path              \Windows\system32\winload.exe
description       Microsoft Windows Vista
locale            nl-NL
inherit           {bootloadersettings}
osdevice          partition=C:
systemroot        \Windows
resumeobject      {f26ched5-dd34-11db-b487-f4ea3d959cc7}
nx                OptIn

C:\Windows\system32>bcdedit /set increaseuserva 2900
De bewerking is voltooid.

C:\Windows\system32>bcdedit
Windows-opstartbeheer
id                {bootmgr}
device            partition=C:
description       Windows Boot Manager
locale            nl-NL
inherit           {globalsettings}
default           {current}
displayorder      {current}
toolsdisplayorder {memdiag}
timeout          0
resume            No

Windows-opstartlaadprogramma
id                {current}
device            partition=C:
path              \Windows\system32\winload.exe
description       Microsoft Windows Vista
locale            nl-NL
inherit           {bootloadersettings}
osdevice          partition=C:
systemroot        \Windows
resumeobject      {f26ched5-dd34-11db-b487-f4ea3d959cc7}
nx                OptIn
increaseuserva    2900
```

This is why I pick the middle route. Higher values leave less memory for Systems use, and might effect your systems performance in bulky operations. Note that non-LAA programs will not be effected by any of those adjustments to your Windows environment.

You can use the same **bcdedit /set increaseuserva ...** command for changing values, or use **bcdedit /deletevalue increaseuserva** to return to the original, nonmodified situation. In which case even LAA-programs will obey the 2Gb limit again.

The Command prompt can be quit with the **exit** {Enter} command, and the new windows setting will be effective after a restart (!) of your PC.

This is how, in Windows NT up till XP

Win7 and Vista offer a startup regime which is an enhancement over the original Windows NT one. Here is the original approach. Enabling LAA is the same for Win NT and above, like Win2000 and Win XP. The images and code examples below were made from my XP system.

Essentially, you have to extent the Windows startup command by adding **/userva=2900 /3gb** switches at the end. The **/userva** switch is optional, when leaving it out, the value defaults to 3072. The **/3gb** switch is mandatory, and it should be at the end.

You may add a second startup command line, which enables you to select whether or not to run in the extended addressing mode. This gives you an escape route as well: when issues occur, you just restart Windows using the first, original command line.

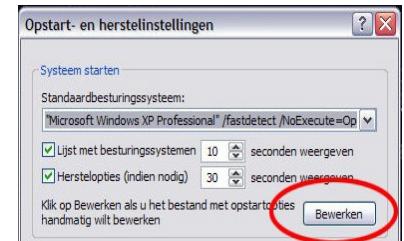
How to:



- Rightclick My Computer on your desktop, and select Properties.

- Choose the Advanced tab, and click the third button, about (Re)Startoptions

- In the options window, click the Edit button



- This opens the text editor on the startup file (boot.ini), which might read something like

```
[boot loader] timeout=10
default=multi(0)disk(0)rdisk(0)partition(2)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="Microsoft Windows XP Professional" /fastdetect /NoExecute=OptIn
```


- Now just add a copy the last line, change the name between the quotes, and add the extra options. So now you have:

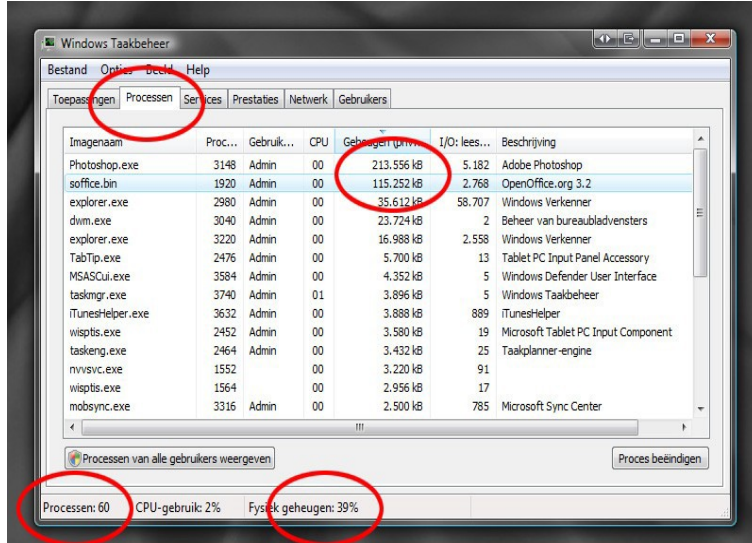
```
[boot loader] timeout=10
default=multi(0)disk(0)rdisk(0)partition(2)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="Microsoft Windows XP Professional" /fastdetect /NoExecute=OptIn
multi(0)disk(0)rdisk(0)partition(2)\WINDOWS="With 3Gb memory limit" /fastdetect /NoExecute=OptIn /userva=2900 /3gb
```

- Save and close the text file
- Then you might choose this new, second option as the startup default, in the options Window. Then click OK. (NB: you might have to click OK first and reopen the options window again, to make it read the edited boot.ini file again and discover the second command line).
- Click OK to close the Properties window too. Note that you have to restart Windows and pick the correct variant to have the 3Gb feature activated.

At startup, you will now see two systems to pick from, for the time as set in the timeout line in the text (10 sec in the example). When issues occur, you might restart Windows and pick the first option. As an alternative, you can delete the first startup option which will make the startup go without offering choices. That's up to you. When issues occur, you have to re-edit the startup file before restarting, deleting the /userva and /3gb switches.

Again, non-LAA programs will not be effected by any of those adjustments to your Windows environment. Changing either the startup commandline or the boot.ini file back to their original state sets you back to the original, nonmodified situation. In which case even LAA-programs will obey the 2Gb limit again.

Physical and Virtual Memory



Windows Taakbeheer

Imagenaam	Proc...	Gebruik...	CPU	Geheugen (priv...)	I/O: lees...	Beschrijving
Photoshop.exe	3148	Admin	00	213.556 kB	5.182	Adobe Photoshop
soffice.bin	1920	Admin	00	115.252 kB	2.768	OpenOffice.org 3.2
explorer.exe	2980	Admin	00	35.612 kB	58.707	Windows Verkenner
dwm.exe	3040	Admin	00	23.724 kB	2	Beheer van bureaubladvensters
explorer.exe	3220	Admin	00	16.988 kB	2.558	Windows Verkenner
TabTip.exe	2476	Admin	00	5.700 kB	13	Tablet PC Input Panel Accessory
MSASGui.exe	3584	Admin	00	4.352 kB	5	Windows Defender User Interface
taskmgr.exe	3740	Admin	01	3.896 kB	5	Windows Taakbeheer
iTunesHelper.exe	3632	Admin	00	3.888 kB	889	iTunesHelper
wispts.exe	2452	Admin	00	3.580 kB	19	Microsoft Tablet PC Input Component
taskeng.exe	2464	Admin	00	3.432 kB	25	Taakplanner-engine
nvsvc.exe	1552	Admin	00	3.220 kB	91	
wispts.exe	1564	Admin	00	2.956 kB	17	
mobsync.exe	3316	Admin	00	2.500 kB	785	Microsoft Sync Center

Processen van alle gebruikers weergeven

Processen: 60 CPU-gebruik: 2% Fysiek geheugen: 39%

In the previous sections, it was discussed how individual programs could use 2, 3 or 4Gb memory.

And when you open the Taskmanager, you can see 50 or more programs running at the same time.

So, how much RAM can one have installed to make all things work?

The answer is that 2Gb is a minimum requirement, 4Gb is an absolute maximum for 32-bit Windows anyway (it's just a non-LAA program itself!) and the 3Gb as in most laptops gives a good price/performance ratio, unless you're doing the bulky things that made you read this tutorial in the first place.

So, how does it work?

I won't go into details, but every time a program needs memory, it just gets a chunk (or: page) of it assigned by the Windows Memory Manager (WMM). This WMM can move filled but hardly used pages to disk, and back. Effectively, it uses disk space to fill in enhanced requirements, and this is why you don't need to have all RAM aboard physically.

The downside of this approach is that when your program needs a lot of RAM actively, the WMM will get very busy by swapping all those memory pages on and off the disk. You will see and hear the rattling of your disks, and you will experience serious performance downgrades. More physical RAM just means: less swapping, and better performances (as long as you have this kind of issue).

For all addressing, programs talk to the WMM too. So whether they request up to 2Gb or even up to 3Gb, it might come from either RAM or disk. Actually, adding more physical RAM will not solve any problems coming from crossing the 2Gb User Memory border. On the other hand, when so much memory is actively used it's likely that you might run into swapping delays as well. Although in my experience those RAM-hungry programs are quite handy in avoiding it.

So, enough physical RAM for all your simultaneous programs, and enough user memory for one single program, really are separate things. Solving one will not solve the other.

