Getting started

Antivirus program blocks vDosSetup

Before a new version of vDos is released, vDosSetup and vDos are submitted to www.virustotal.com. There tested by over 70 antivirus programs. Eventual false positives are then reported to those companies.

If your antivirus program complains about vDosSetup: First make sure its virus database is **up-to-date**. Eventually go to www.virustotal.com, and check vDosSetup.exe yourself.

Install vDos

The vDosSetup installation program will offer to create a vDos folder on your C: drive. You can select another location, but C:\vDos is a good starting point for a first-time impression and testing.

vDosSetup doesn't modify anything to your Windows system internals. It only creates that vDos folder and a desktop shortcut. There's even no uninstall option. If vDos doesn't work for you, you'll have to delete those two items by hand!

The vDos folder will contain these files:

- **autoexec.txt** The equivalent of DOS autoexec.bat.
- **config.txt** DOS had config.sys, Windows config.nt, vDos has config.txt. Like autoexec.txt, the .txt extension is for easy editing, without Windows trying to 'execute' it.
- **FAQs.pdf** This document.
- **Icons.icl** Some alternative icons for vDos shortcuts.
- **Printing.pdf** Documentation of the printing capabilities.
- **Readme.pdf** An introduction to vDos.
- **vDos.exe** The Windows program emulating a DOS PC in a window to run your application(s).

And two subfolders:

- **DPTEST** The DOS DataPerfect TestDrive demo program, started by the initial autoexec.txt file.
- **EXTRA** Some rarely used DOS specific code page files, missing in Windows.

Start vDos by the desktop shortcut "vDos - Initial test". Its window will show, awaiting a keystroke to start the before mentioned demo program.

The demo ran fine, how to run/test my DOS application

Your application once ran in DOS, then it migrated to Windows 32-bit. Now you're facing vDos, a Windows program that enables you to run it in Windows 32 or 64-bit.

For now, we're in a testing phase, and assume vDos is installed to the default Windows C:\vDos folder.

- Open autoexec.txt and just delete all its contents. The demo program doesn't need to start anymore. Save the empty file and start vDos again.
- That will bring the famous `C:\>` DOS prompt. Notice that this vDos C: isn't Windows C:! Instead it defaults to the Windows C:\vDos folder. Eventually do a `DIR` command to display the files and folders in this vDos C: drive. Leave the vDos window open.
- You will still have your application running on some system. Copy its folder(s) to the vDos folder. So you'll have something like C:\vDos\DosApp, C:\vDos\DosData…
- Have a close look at how the application was started before. That would be by some batch file, we'll assume it is "start.bat". If it isn't already in the copied folder(s), copy that also to the C:\vDos folder.
- Do another `DIR` command, the copied folder(s) will show. Start the application by `start.bat` (it's in vDos C:), or `folder\start.bat` (substituting folder by that start.bat is located in). If the application starts, you end it and get back to the DOS prompt; Close the vDos window by issuing `EXIT`.
- If the application doesn't start, that will be because it expects (relies on) a drive letter or directory structure that doesn't match what we got so far in vDos. If it for instance expects a F: drive with the program and data files (DosApp, DosData…), you'll have to assign vDos F: to the appropriate Windows folder. That would be C:\vDos in this example. Issue `USE F: C:\vDos`, then select that drive by `F:`, so you'll get a `F:\>` prompt. Now try to start the application again. If it still doesn't start, open the batch file in for instance Notepad. Look for the line that starts your application. If some other program is started before that, temporary disable it by adding REM. For instance REM program, save the batch file and try again.
- If you still have trouble running the application, post your problem at the vDos forum. Specify what vDos version you use (should be the most recent one), what directory structure you have (C:\vDos\DosApp…), what application you try to start, and how it is supposed to start (the start.bat contents).

Start the application by starting vDos

- Open autoexec.txt, if you didn't already delete its contents, do it now.
- Add the command lines that you entered by hand in the previous section, for instance:

```
USE F: C:\vDos
F:
CALL DosApp\start.bat
```

- Save the file, and start vDos again.
- If your application doesn't start, verify the command lines in autoexec.txt with those you entered by hand.
- Add a final command to autoexec.txt: `EXIT`, so the vDos window will close the moment you end the application.

Roundup, 'going live'

We assumed you copied the application folder(s) temporary to Windows C:\vDos. And the application was started by autoexec.txt with:

```
USE F: C:\vDos\
F:
CALL DosApp\start.bat
EXIT
```

To start the application with the 'live' data, change the folder reference in the first line to where DosApp is actually located. For instance:

```
USE F: D:\DOS\
```

or

```
USE F: \\server\share_name\DOS\
```

That's it.

If you have more than one DOS application:

- Copy config.txt and autoexec.txt to a separate folder, preferably nearby the application
- Modify autoexec.txt to start that application. Eventually temporary rem-out the last `EXIT` and test.
- Create a new (desktop) shortcut to vDos.exe, or create a copy of the installed one.
- Give it a sensible name, like "Accounting". That will also be displayed in the caption bar of the vDos window.
- Go to the properties of the shortcut (right click). Set "Start in:" to the separate folder.
- Eventually assign another icon image to the shortcut.

Miscellaneous

FILES= and other directives once in config.sys

BUFFERS=
A kind of basic local disk caching, without meaning anymore. vDos isn't the operating system (DOS once was) controlling the drive, any caching could be a disaster for your data on the drive, waiting to happen.

DOS=
DOS in vDos is simulated outside the virtual PC. It is no longer some 16-bit code executed by the (emulated) CPU. So this directive is also meaningless in vDos, there is essentially no DOS code in the virtual PC's memory.

DEVICE=
vDos shouldn't need device drivers. For instance ansi.sys is built-in.

FILES=
The number of global DOS file handles in vDos is 255, the maximum.

STACKS=
Simulated DOS in vDos has no use for this directive.

SHARE - Record locking (RL) by multi-user applications

In the DOS days, RL was provided by the SHARE program. If more than one need to access the same data, the (mostly a database oriented) program occasionally requires exclusive access to some parts of files (RL data and indexes). “I’m going to modify those, can’t have someone else interfering while doing that”.

The operating system (OS) in control of the disk maintains a table of file regions granted a RL request by a program. DOS is since Windows NT no longer that OS, SHARE useless: Its functionality is provided by Windows, or the OS managing the network drive. vDos secures RL is functioning for a drive, then mostly just forwards RL requests to Windows. Effective RL by database programs is more complex, but this will do.

Some DOSBox mods claim RL support, but turn out to be a disaster. Besides granted exclusive access, the internal program data has to match what’s actually on disk that moment. DOSBox (mods) however caches disk operations, possibly outdated data lurking all around (the network). One DOSBox mod ‘solution’ first updates the caches with RL. That doesn’t do it either: Local caches at other PC’s can still destroy the database. Especially indexes are vulnerable due to the compact and structured organization.

My application (occasionally) runs slower than before

To overcome Windows 64-bit not able to execute 16-bit programs, vDos emulates a 80386 CPU in software. A single emulated CPU instruction will so result in several instructions to the actual CPU. The operating state and registers of the emulated CPU are maintained in relatively slow memory, and more technical stuff.
In raw processing power the emulated CPU can be up to 40 times slower than the real thing! The emulated CPU got over 100% faster the past years, but no further significant improvements should be expected.

DOS applications however don’t run that much slower. They frequently call BIOS and DOS functions, mostly executing faster than before. Reading data from and writing to disk is generally the real bottleneck. That is also at least as fast. On average DOS programs will operate only a few times slower than on a real DOS PC or Windows 32-bit/NTVDM. Shouldn’t however be that an issue: With user interaction, a program is mostly waiting for user input. Whether she/he then has to wait 0.01 or 0.05 seconds before the program responds, isn’t noticeable.

But w/o user interaction, few BIOS/DOS calls, little disk access, lengthy CPU intensive tasks. Like sorting database records in memory, the slowness of the emulated CPU will become more profound. You just have to life with that.

No Long File Names (LFN) support

DOS was originally restricted to 8.3 filenames: 1-8 characters for the name, 1-3 for an optional extension. Windows brought LFN, supporting up to 255 characters. Microsoft also implemented a scheme so DOS programs could access those LFN files: 8.3 “Short File Names” (SFN).

vDos doesn't support LFN directly: Those are optionally used by only a few (mainly utility) DOS programs. A Zip/Unzip program version, ditto file manager and some other obscure examples, just silly to use in vDos. You have genuine Windows program for that! A list of LFN aware DOS programs.

vDos/DOS drive letters are not those of Windows

DOS systems were dedicated to run DOS programs. DOS got replaced by Windows. For simplicity we forget about Windows 3.x - 98 (mainly DOS programs) and skip to Windows NT/XP (not DOS based anymore).

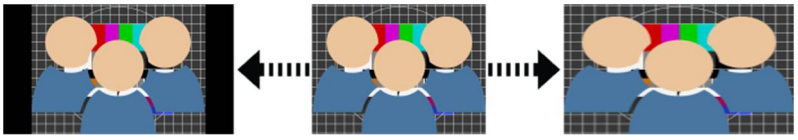
Microsoft had to make sure DOS programs would still run. And so came with NTVDM (New Technology Virtual DOS Machine), integrated in Windows 32-bit. DOS programs could run, largely as before. But that integration came at a cost: The Windows PC was no longer dedicated to run DOS programs. The Internet surfed, email checked, Office used, and a lot more Windows specific stuff. Most folders and files have no meaning to DOS programs.

DOS programs got access to all Windows drives (letters) and folders, while there is no need for that. Certainly not the root of Windows boot drive C:. Windows even doesn’t like that. DOS programs require drive letters, these then had to be created for non-local drives in Windows, while Windows (programs) doesn’t need those.

Although you might be accustomed to DOS drive letters being those of Windows, reconsider that. The vDos `USE` command allows to only make available those regions of Windows filesystem your DOS program needs to access. Even if you don’t want to forgo for instance a Windows F: being mapped to a network share: Don’t do `USE F: F:\`. That is a possibly confusing, even troublesome roundabout. Instead do the direct `USE F: \\server\share\`.

vDos full screen mode isn’t real full screen

PC monitors evolved since DOS: CRT to LCD/(O)LED - 4:3 to 16:9 widescreen. So did TV’s, occasionally a popular old serial (DOS) is rerun, 4:3 content however doesn’t fit a 16:9 sized modern TV (or PC monitor). You get bars alongside, or a stretched image:



It’s not vDos or Windows to blame you don't get what you perhaps hoped for.

Some programs of all I tried out, didn’t run

vDos is meant to run productive DOS programs still in use. Not to play with.

Printing

DOS/vDos printing

Printing by DOS applications differs completely from Windows printing. vDos should by default mostly determine correctly how to handle that. So just start with that will bring.

One exception could be printing to an actual old-style DOS printer. vDos supports that also, but you might consider switching to Windows printing. So you can use any printer supported by Windows (for instance also virtual PDF).

Keep in mind, DOS LPT/COM ports/devices in vDos aren’t those of Windows. With that you were only able to print to actual DOS printers. Even if you convinced Windows to print to an USB or network printer instead.

Troubleshooting

The printing capabilities and options are covered by the Printing.pdf document in the vDos folder. If printing doesn't work as expected: **Read** that document. Many questions in the past just came from not reading (anything).

vDos creates two files before the actual printing process starts: #LPTx/#COMx.asc (x being the port number) and a .txt version of that. Those however have no value for vDos itself. The .txt version is mostly a means to export (large amounts of) data, to be imported by a Windows program (the DOS ASCII text is translated to Windows Unicode). The .asc version contains the data stream as send by your program to the printer. It is created to facilitate an eventual third-party DOS-to-Windows print processor and for debugging purposes.

If you have an issue with printing: Submit a copy of the .asc file, so the problem can be reproduced and investigated (that’s the debugging part).

The COMx port isn’t related to a printer

By default, writing to LPTx/COMx ports (DOS, BIOS or hardware) is considered to be a printing task.

If COMx is instead to access some serial device; use the COMx = "COMy": directive in config.txt. Mind the trailing colon, it is mandatory. You also have to initialize the Windows COMy device before starting vDos. By the MODE COMy command, or the device manager.

Support of serial devices is basic. To add further support, I would to have such a port, a connected hardware device, and DOS software using those. I don’t and gave up communicating with those (initially) willing to do some testing. Though that basic support should mostly work, as reported so far.

Digital printing with stationary (PDF)

Documents are more and more stored and sent digitally, instead of printed out on paper. You want those documents to resemble closely the former printouts on your stationary. vDos internal print processor doesn’t support bitmap graphics, but that actually is of no concern to produce high quality digital documents.

You'll need:

- A digital copy of your stationary, ask your print supplier for the PDF file, **vector based**! Eventually modify that, or even create your own.
- A virtual PDF printer creating a PDF file, instead of printing to paper. A freeware PDF printer, capable of merging two PDF files, could do. But for instance novaPDF is more versatile with profiles, options and different handling of documents: What stationary to use, how to name the PDF, where to store…

I created two files by hand:

- InvoiceBack.pdf: The stationary used for invoices.
- Invoice.txt: Output of an imaginary DOS invoice program, just plain ASCII (DOS) text.

The setup in config.txt: LPT1 = sel:"novaPDF 11".

Since there's no actual invoice program to print, this was mimicked by `COPY INVOICE.TXT LPT1` at the command prompt. The Select Profile dialog of novaPDF pops up (I have several), I chose the temporary test profile "Invoice". That is setup to merge the printers output with InvoiceBack.pdf to a PDF file with an incrementing number as its name (matching the invoice number). Save it to the designated folder for invoices and open it in the default PDF viewer. The result isn't that bad for a simple test.

Generating PDF's this way, keep in mind:

- Make sure the PDF's can actually be printed as expected. PDF version 1.4 gives consistent good results for me.
- Consider what actions to allow with the PDF's. You won't want invoices to be modified. Just set them to print-only.
- Invest some initial time on fine-tuning the PDF's and optimizing the automated document handling.

Advanced

Windows environment variables in vDos

To use a Windows environment variable like %systemroot% in your DOS program, you first have to explicitly declare it. In autoexec.txt or a batch file. The Windows variable must be surrounded by two percent signs. For example, before you can use Windows %systemroot% in your DOS program, you first declare it with:

```
set systemroot=%%systemroot%%
```

At startup one Windows variable is already set in DOS: WIN_VDOS: The VDOS variable if set in Windows (%VDOS%) or the command line parameters of vDos.exe. You can also use %0-%9 to access those in autoexec.txt.

Start a Windows program or command processor

The vDos CMD command is linked to Windows command processor CMD. It can be used at the vDos command line, or from within a program by:

```
CMD ["program"] [WAIT][HIDE] command line
```

"program": If supplied, the quotes are mandatory.
WAIT: Wait for the Windows program or command processor to complete.
HIDE: Start it minimized.
command line: Passed on to the program or command processor.

Eventual paths in command line (and "program") are those of Windows.

The optional WAIT and HIDE have to be in capitals. Mind, WAIT in combination with the Windows command line has a major limitation: If that executes a program, it will exit immediately. While the external program could still be running. The returned exit code then merely indicates the program was started.

If you want to redirect the input or output, enclose the entire command line in quotes.

Some examples:

```
CMD notepad mydoc.txt
CMD HIDE /c "mytest.bat>mytest.txt"
CMD /c start /max notepad
```

You can also call a Windows program directly inside a DOS program:

```
program [WAIT][HIDE] command line
```

Here program is a DOS file name reference. vDos will determine what it is supposed to be: A DOS or Windows program. And will start that inside the virtual PC (DOS program), or in a new window (Windows program). Program can for instance even be a document, to be opened by the associated Windows program.

Note: Some DOS programs don't start external programs directly, but the DOS command processor to do that.

A portable DOS application

In autoexec.txt you could have something like:

```
USE F: \\server\DOS\
F:
CD DOSapp
mydosapp.exe
```

Great, you already used the direct UNC path, instead of mapping a Windows drive letter to it and refer to that. Can we do without a (absolute) Windows path?

Sure, when vDos is started, the Windows current work directory for that vDos instance is set to its folder. If vDos.exe would be located in \\server\DOS, that will become the Windows work directory. A relative path works equally well for the `USE` command. So we can also do:

```
USE F: .\
```

Or if we have a separate vDos folder (\\server\DOS\vDos):

```
USE F: ..\
```

What will a portable DOS application eventually bring?
First, if you can get to (so start) vDos.exe, your application will run. No matter from where it is started, or its location (local, network, RDP, VPN, the cloud…).
Second, you can relocate the application without any modification needed. Or create a copy of it at another location, for testing, backup…

Build your own copy/version of vDos

Here is a beginner's guide to building a copy of vDos from the sources. There is only one reason to do this: If you want to make your own copy of vDos that includes changes in the source code. This procedure was tested under Windows 8.1, but should work with any recent Windows version.

This assumes a basic understanding of Windows software (for example, you should know how to open .7z archives). You will be able to do very little with the source code unless you know something about C/C++, but you may be able to make some useful changes with very little knowledge.

If you have any questions about these instructions, you should be able to figure out the answers for yourself by experimenting or by searching the Internet.

- Request the vDos source code; start by visiting this page.
- Extract the vDos folder to any convenient place on your system (such as your desktop).
- Download the free Visual Studio Community Edition from the Download link on this page.
- Run the downloaded installation program, and proceed as follows.
- Check the box where you agree to the license terms; click Next.
- Remove the checkmarks from everything on the list of "Optional features to install"; click Install and proceed through the installation. At the end, do not click "Launch" but simply close the installation program.
- In the vDos folder that you extracted, open the visualc_net folder and double-click vDos.sln.
- When Visual Studio opens, you can sign in with a Microsoft account if you want, or ignore the sign-in option.
- Visual Studio will offer to upgrade the VC++ Compiler and Libraries. Click OK.
- In the Solution Explorer on the left you will see vDos in bold. Right-click on it, and choose "Build" from the pop-up menu, and wait while the project builds.
- When the Output window shows "Build: 1 succeeded…", go back to the vDos folder and go to the vDos\bin folder. You will find the copy of vDos.exe that you built.

You may now experiment with the vDos source code and generated vDos.exe.