

VirtualHdos
HDOS Emulator
February 14, 2023

Table of Contents

Introduction.....	1
Native Files.....	1
Configuration.....	2
Environment Variables.....	3
Running the Emulator.....	3
Console Input.....	4
Batch Execution (SUBMIT).....	4
Setup Example.....	4
Makefile Example.....	5
Tracing.....	5

Introduction

VirtualHdos is a JAVA program that emulates a HDOS OS running on a Z80 with 64K RAM. It is normally used to execute a single HDOS command, but is capable of handling “submit” files as well as native (on the host PC) text files filled with commands. Currently, it is only HDOS v2.0.

Native Files

Native files must have lower-case only names. Mixed-case filenames will cause unpredictable results. All files created by HDOS programs will be in lower-case.

The file’s write permission is used to reflect the HDOS RO attribute. HDOS programs that change a file’s RO attribute will change the native file’s write permission.

The file’s execute permission is used to reflect the HDOS SYS attribute. Note that Windows will always show files as executable, and thus files on a Windows host will always have the SYS attribute set. Also remember that HDOS normally hides files that have the SYS attribute set. There is a server configuration setting that disables the SYS attribute, to avoid these issues on Windows hosts. Note that some non-Windows systems may set file execute bits incorrectly, such as ZIP archive extractions or files copied from a Windows system. On non-windows systems the execute permissions can be removed from HDOS files that should not be SYS.

HDOS is unable to handle years beyond 1999, so the timestamps created for native files will always have the year “1999” (unless they were actually stamped with an earlier date). This behavior may be relaxed with a configuration property.

Files that are not an even multiple of 256 bytes in size will be padded to a 256-byte multiple, using NUL (0x00), when reading. Writing to a file from HDOS always involves a full 256-byte record, so no additional padding is performed.

Configuration

The emulator is configured using a “configuration file”, which is plan text formatted as “property = value” lines. The configuration file to be used is based on the following search order. First, it will look in the current directory for “**vhdos.rc**” and second in the user’s home directory for “**.vhdosrc**”. Comments in the file start with ‘#’ as the first non-space character of the line. Property values may use backslash (‘\’) to extend long values to the following line.

The following properties are recognized:

silent

Suppresses informative messages about configuration, etc.

vhdos_root_dir = path

Specifies the top-level (root) directory to be exported. Subdirectories named “sy0” through “sy3”, “dk0” .. “dk3”, are assumed, but not created automatically. Default will be `~/HostFileHdos`. Note that JAVA Properties does not recognize the ‘~’ syntax, so the absolute path must be used.

vhdos_nosys

Disable the HDOS SYS attribute, so files will not be hidden on Windows.

vhdos_y2k

Disable capping datestamps at 1999. Some programs (that examine directories) may not tolerate this setting.

vhdos_drive_X = path

Where “X” is one of “sy0” through “dk3”. Specify the path to use instead of “root_dir/X” for the HDOS drive.

vhdos_dump = file

Triggers a core dump of RAM when exiting the emulation. The dump file contains the raw (binary) contents of RAM. Note that subsequent dumps to the same file will overwrite any previous dump. Default file name is “vhdos.core”.

vhdos_trace = command

Enables program (instruction) tracing based on *command*. See section “Tracing”.

vhdos_cpu = cpu

Select which CPU to use. Value *cpu* may be “I8080”, “I8085”, “Z80”, or “Z180”. Default is Z80.

vhdos_disas = zilog

For Z80 and Z180 CPUs, this selects Zilog mnemonics in the disassembled instruction when tracing. Default is to use Intel mnemonics with standard macro syntax for Z80/Z180 instructions.

Environment Variables

The following environment variables are recognized:

HDOSDrives = path-list

The *path-list* is a list of (up to) 8 paths separated by commas. Each path is positional dependent, i.e. the first path is for drive SY0:, second for drive SY1:, etc. If a path is empty (“,”), then the default is used. All non-empty paths are translated into **vhdos_drive_X** properties.

HDOSDrive_X = path

Specifies the path to use for drive *X*. This is translated into a **vhdos_drive_X** properties.

VHDOSCoreDump = file

Triggers a core dump of RAM when exiting the emulation. The dump file contains the raw (binary) contents of RAM. Note that subsequent dumps to the same file will overwrite any previous dump. This overrides any **vhdos_dump** property. Default file name is “vhdos.core”.

VHDOSTrace = command

Enables program (instruction) tracing based on *command*. This overrides any **vhdos_trace** property in the config file. See section “Tracing”.

VHDOSCPU = cpu

Select which CPU to use. Value *cpu* may be “I8080”, “I8085”, “Z80”, or “Z180”. This overrides any **vhdos_cpu** property. Default is Z80.

VHDOSShow = drive

This variable causes vhdos to print the path being used for *drive* (“sy0”, ...) and exit without running any program or command. It may be used from Makefiles to directly access the HDOS files without using hard-coded paths.

Running the Emulator

Typical startup command:

```
java -jar Virtualvhdos.jar hdos-command
```

Typically, a wrapper script or batch file will be created, to simplify running commands. On Linux, this might look like:

```
#!/bin/bash
exec java -jar /path/to/VirtualHdos.jar "${@}"
```

If this script were named “vhdos”, and is on the user’s PATH, then one would run HDOS commands using:

```
vhdos hdos-command
```

Note that this emulator does not provide any I/O hardware emulations. Any programs that execute input/output instructions will cause (non-fatal) messages to be printed for each I/O. While these are non-fatal incidents to the emulator, it is likely that the HDOS program will not run correctly.

On Linux/Unix systems (or compatible shells on Windows), a command may be prefixed by environment variables, which is a convenient way to do ad-hoc tracing and coredumps. For example:

```
VHDOSCoreDump=mydumpfile vhdos hdos-command
```

Console Input

Translating HDOS console input to JAVA and modern “stdin” is problematic. This emulator is optimized for whole-line input, so some programs that use “raw” input may not work well. Programs doing full-line input should work fine, although because of echoing there is an extra blank line after entering input.

Batch Execution (SUBMIT)

[NEEDS TESTING] Conventional HDOS 3 submit files may be executed as a command. In addition, a local (native) file may be used even though it does not reside on a HDOS drive. The file must be entered as a path, i.e. must contain a slash (/), even if only a “./” prefix.

Each line is a command, or if prefixed with ‘<’ a line of input to the program being run at the time. Argument substitutions are supported, using the standard “\$1”, “\$2”, ... “\$9” notation. Comment lines must begin with ‘#’ or ‘;’. Empty lines are skipped.

There is no way to disable searching for “command.sub” when only “command” is entered, but entering “command.com” will avoid matching any submit file.

Setup Example

Here is an example set of commands used on Linux to setup vhdos. These assume no vhdos config file is present. In this example, the repository ~/hdos-dist is used as a source for HDOS distribution files. Also, the ‘vhdos’ script has already been copied and edited and is on the user’s PATH.

```
$ mkdir -p ~/HostFileHdos/{sy0,sy1,sy2,sy3,dk0,dk1,dk2,dk3}
$ cp ~/hdos-dist/*.abs ~/HostFileHdos/sy0
$ vhdos dir
>dir
NAME      .EXT    SIZE    DATE      FLAGS
C          .ABS     154    13-Feb-1999  W
SCANF     .H        1     13-Feb-1999
CCONFIG   .ABS     34     13-Feb-1999  W
```

XREF	.ABS	13	13-Feb-1999	W
MATHLIB	.REL	19	13-Feb-1999	
ASM	.ABS	33	13-Feb-1999	W
PRINTF	.H	1	13-Feb-1999	
XREF3	.ABS	12	14-Feb-1999	W
CLIBRARY	.REL	14	13-Feb-1999	
STDLIB	.REL	16	13-Feb-1999	
M80	.ABS	69	14-Feb-1999	W
SYSCMD	.SYS	12	13-Feb-1999	S
BOOT	.ABS	4	13-Feb-1999	W
L80	.ABS	33	14-Feb-1999	W
FPRINTF	.H	1	13-Feb-1999	
FLIBRARY	.REL	29	13-Feb-1999	
DEBUG	.ABS	15	13-Feb-1999	W
AS	.ABS	54	13-Feb-1999	W
FOO	.ABS	0	14-Feb-1999	W
PIP	.ABS	19	13-Feb-1999	W
ASM3	.ABS	32	14-Feb-1999	W
DIRTEST	.ABS	20	13-Feb-1999	W
\$				

The exact list of files, and order, above depends on what files were copied into the SY0: directory.

Note that any startup messages can be suppressed by setting up a vhdos config file that contains the “silent” property. For example, “echo silent >>vhdos.rc” in the current directory will add the ‘silent’ property, possibly creating a new “vhdos.rc” properties file.

Makefile Example

Here is an example Makefile that uses vhdos to assemble using ASM.ABS into an ABS file, using source file “prog1.asm”. This makefile and the source file reside in the same directory. The default VirtualHdos SY0: drive must contain ASM.ABS and SY1: must contain any ACM files needed. Note that some of the vhdos commandline arguments may need to be quoted, because otherwise the host shell may try to interpret them or stumble over special characters.

```
export HDOSDrive_dk0 = $(PWD)

all: prog1.abs

%.abs: %.asm
    vhdos asm dk0:$*,dk0:$*=dk0:$*,sy1:/err
```

Tracing

Trace commands take the form:

range [count] [oneshot]

Where *range* has the syntax:

low:high

A range expression of only ':' matches the entire address range (a missing *low* or *high* equates to the respective end of the full address range). An expression of only *low* (no ':') matches a single address. The *high* address is exclusive, i.e. a match means $low \leq PC < high$.

A special range expression '.' means immediate trigger, for use with specifying only *count*. This implies **oneshot**.

The *count* argument sets the number of instructions to be traced after matching the range. Note that this counter starts after "leaving" the range, i.e. all instructions within the range will always be traced. Default is 0 (no addition lines traced).

The **oneshot** keyword causes the tracing to occur only once. Tracing is essentially disabled after being triggered (after "leaving" the range).

The *range* argument is required and must be the first.