MEMDISK

From Syslinux Wiki

Contents

What is MEMDISK?

MEMDISK is meant to allow booting legacy operating systems. MEMDISK can boot floppy images, hard disk images and some ISO images.

MEMDISK simulates a disk by claiming a chunk of high memory for the disk and a (very small - 2K typical) chunk of low (DOS) memory for the driver itself, then hooking the INT 13h (disk driver) and INT 15h (memory query) BIOS interrupts.

How can I use MEMDISK?

MEMDISK is an auxiliary module used in conjunction with a boot loader that can load linux kernels (EXTLINUX/ISOLINUX /PXELINUX/SYSLINUX, grub, grub4dos grub2, ...). You need a disk image as well as the memdisk file itself. As far as the bootloader is concerned, MEMDISK is the "kernel" and disk image is the initial ramdisk (initrd).

The disk image, passed as initrd, can be a **compressed** zip or gzip file.

In the next examples **hdt.img** is a floppy image (**hdt.iso** is a ISO image) containing Hardware Detection Tool (http://hdt-project.org/) (runs

directly on SYSLINUX) and **dosboot.img** is a floppy image that contains some version of DOS.

EXTLINUX/ISOLINUX/PXELINUX/SYSLINUX

You can use MEMDISK straight off the boot loader command line like the following:

```
memdisk initrd=hdt.img
```

... where, of course, hdt.img is the name of the disk image file. The memdisk file and the disk image need to be present in the appropriate location (for PXELINUX, on your TFTP server, for ISOLINUX, in the /isolinux directory on your CD, etc.)

Normally, however, you would put something like the following in the configuration file:

```
# Boot Hardware Detection Tool from floppy image
LABEL hdt_floppy
 LINUX memdisk
 INITRD hdt.img

# Boot Hardware Detection Tool from iso image (with 'iso' parameter)
LABEL hdt_iso
 LINUX memdisk
 INITRD hdt.iso
 APPEND iso

# Boot DOS from floppy image (with 'raw' parameter)
LABEL dos_floppy_with_raw
 LINUX memdisk
 INITRD dosboot.img
 APPEND raw
```

GRUB and GRUB4DOS

In your **menu.lst** file, use something like:

```
title Boot Hardware Detection Tool from floppy
  kernel /memdisk
  initrd /hdt.img

title Boot Hardware Detection Tool from iso image (with 'iso' parameter)
  kernel /memdisk iso
  initrd /hdt.iso

title Boot DOS from floppy image (with 'raw' parameter)
  kernel /memdisk raw
  initrd /dosboot.img

title Boot CorePlus (TinyCore installation CD) from ISO image
  root (hd0,0)
  kernel /memdisk iso
  initrd /CorePlus.iso
```

On some systems you may receive an `error 28: Selected item cannot fit into memory` despite the physical memory is far larger than the requirements of the OS in the image. In this case, try to add an appropriate `uppermem <memsize>` statement to your grub configuration, e.g. 1G for CorePlus (http://distro.ibiblio.org/tinycorelinux/):

```
title Boot CorePlus (TinyCore install CD) from ISO image with 1G memory assigned
  uppermem 1048576
  root (hd0,0)
  kernel /memdisk iso
  initrd /CorePlus.iso
```

GRUB2

Add the following in your config scripts for grub2:

```
menuentry "Boot Hardware Detection Tool from floppy" {
  linux16 /memdisk
  initrd16 /hdt.img
}

menuentry "Boot Hardware Detection Tool from iso" {
  linux16 /memdisk iso
  initrd16 /hdt.iso
}

menuentry "Boot DOS from floppy image (with 'raw' parameter)" {
```

```
  linux16 /memdisk raw
  initrd16 /dosboot.img
}
```

Supported image types

The image file should contain a disk image, either a floppy disk or hard disk image, or an iso image.

The disk image can be **compressed** with zip or gzip.

Floppy images

If the disk image is less than 4,194,304 bytes (4096K, 4 MB) it is assumed to be a floppy image and MEMDISK will try to guess its geometry based on the size of the file. MEMDISK recognizes all the standard floppy sizes as well as common extended formats:

```
    163,840 bytes  (160K) c=40 h=1 s=8         5.25" SSSD
    184,320 bytes  (180K) c=40 h=1 s=9         5.25" SSSD
    327,680 bytes  (320K) c=40 h=2 s=8         5.25" DSDD
    368,640 bytes  (360K) c=40 h=2 s=9         5.25" DSDD
    655,360 bytes  (640K) c=80 h=2 s=8         3.5"  DSDD
    737,280 bytes  (720K) c=80 h=2 s=9         3.5"  DSDD
  1,222,800 bytes (1200K) c=80 h=2 s=15        5.25" DSHD
  1,474,560 bytes (1440K) c=80 h=2 s=18        3.5"  DSHD
  1,638,400 bytes (1600K) c=80 h=2 s=20        3.5"  DSHD (extended)
  1,720,320 bytes (1680K) c=80 h=2 s=21        3.5"  DSHD (extended)
  1,763,328 bytes (1722K) c=82 h=2 s=21        3.5"  DSHD (extended)
  1,784,832 bytes (1743K) c=83 h=2 s=21        3.5"  DSHD (extended)
  1,802,240 bytes (1760K) c=80 h=2 s=22        3.5"  DSHD (extended)
  1,884,160 bytes (1840K) c=80 h=2 s=23        3.5"  DSHD (extended)
  1,966,080 bytes (1920K) c=80 h=2 s=24        3.5"  DSHD (extended)
  2,949,120 bytes (2880K) c=80 h=2 s=36        3.5"  DSED
  3,194,880 bytes (3120K) c=80 h=2 s=39        3.5"  DSED (extended)
  3,276,800 bytes (3200K) c=80 h=2 s=40        3.5"  DSED (extended)
  3,604,480 bytes (3520K) c=80 h=2 s=44        3.5"  DSED (extended)
  3,932,160 bytes (3840K) c=80 h=2 s=48        3.5"  DSED (extended)
```

A small perl script is included in the MEMDISK directory which can determine the geometry that MEMDISK would select for other sizes; in general MEMDISK will correctly detect most physical extended formats used, with 80 cylinders or slightly more.

```
LABEL floppy_image
 LINUX memdisk
 INITRD floppy.img
```

If your image is larger than 4 MB and a floppy image, you can force MEMDISK to treat it as a floppy image:

```
LABEL floppy_image
 LINUX memdisk
 INITRD floppy.img
 APPEND floppy
```

Hard disk images

If the image is 4 MB or larger, it is assumed to be a hard disk image, and should typically have an MBR and a partition table. It may optionally have a DOSEMU geometry header; in which case the header is used to determine the C/H/S geometry of the disk. Otherwise, the geometry is determined by examining the partition table, so the entire image should be partitioned for proper operation (it may be divided between multiple partitions, however.)

```
LABEL harddisk_image
 LINUX memdisk
 INITRD harddisk.img
```

If your image is smaller than 4 MB and it is a hard disk image, you can force MEMDISK to treat it as a hard disk image:

```
LABEL harddisk_image
 LINUX memdisk
 INITRD harddisk.img
 APPEND harddisk
```

ISO images

For ISO images, the parameter 'iso' must be passed to MEMDISK.

```
LABEL hdt_iso
  LINUX memdisk
  INITRD hdt.iso
  APPEND iso
```

Sometimes

```
  APPEND iso raw
```

is the better "APPEND" option. For example, some Windows ISO's need the 'raw' option on some PCs.

It is possible to map and boot from some CD/DVD images using MEMDISK. No-emulation, floppy emulation and hard disk emulation ISO's are supported.

The "map" process is implemented using INT 13h - any disk emulation will remain accessible from an OS that uses compatible mode disk access, e.g. DOS and Windows 9x. The emulation via INT 13h can't however, be accessed from an OS which uses protected mode drivers (Windows NT/2000/XP/2003/Vista/2008/7, Linux, FreeBSD) once the protected mode kernel drivers take control. If the OS contains drivers for accessing this mapped ISO, or knows how to find the ISO on the disk, there is no booting problem of course (see next section).

ISOHYBRID images

If the image is a so called ISOHYBRID image, you can also treat it as a harddisk image. This is possible because such an image also contains a MBR (Master Boot Record).

To check if your image is an ISOHYBRID, run:

```
  fdisk -l your_image.iso
```

For a normal ISO image, the output will look like this:

```
  Disk your_image.iso: 140 MB, 140509184 bytes
  64 heads, 32 sectors/track, 134 cylinders, total 274432 sectors
  Units = sectors of 1 * 512 = 512 bytes
  Sector size (logical/physical): 512 bytes / 512 bytes
  I/O size (minimum/optimal): 512 bytes / 512 bytes
  Disk identifier: 0xcbacee9f

  Disk your_image.iso doesn't contain a valid partition table
```

For an ISOHYBRID, the output will look like this:

```
  Disk your_image.iso: 140 MB, 140509184 bytes
  64 heads, 32 sectors/track, 134 cylinders, total 274432 sectors
  Units = sectors of 1 * 512 = 512 bytes
  Sector size (logical/physical): 512 bytes / 512 bytes
  I/O size (minimum/optimal): 512 bytes / 512 bytes
  Disk identifier: 0xcbacee9f

           Device Boot      Start         End      Blocks   Id  System
  your_image.iso1   *           0      274431      137216   83  Linux
```

INT 13h access: Not all images will complete the boot process!

Real mode operating systems and boot loaders that use INT 13h BIOS calls

DOS (MS-DOS, FreeDOS, DR-DOS, ...), Windows 95/98/ME and boot loaders (Syslinux, grub, grub4dos, gujin, gag, mbldr, ...) that use INT 13h to read from disks will successfully complete the boot process with MEMDISK (assuming that there are no BIOS bugs, or software bugs).

Windows NT/2000/XP/2003/Vista/2008/7 (NT based)

These Windows versions use INT 13h access only in the start of the booting process (loading only the necessary drivers). Once the protected mode drivers are functional to access the disks, Windows can't see the memory mapped drives created by MEMDISK (CD/DVD, hard disk and floppy disk images) and it will fail to complete the boot process.

Solutions:

- Drivers that detect the MEMDISK mapped floppy/disk/ISO image

- WinVBlock driver

WinVBlock is a Windows driver that overcomes this problem. It detects the MEMDISK mapped drives (CD/DVD, hard disk and floppy disk images) so Windows can read and write to those disks.

For more info about WinVBlock (it can do more than detecting MEMDISK mapped drives) and how to build your Windows images: http://reboot.pro/8168/

Windows images may need the 'raw' parameter on some PCs:

```
LABEL windows_winvblock
  LINUX memdisk
  INITRD windows_harddisk.img
  APPEND raw
```

```
LABEL windows_winvblock
  LINUX memdisk
  INITRD windows.iso
  APPEND iso raw
```

- Firadisk driver

Firadisk is a Windows driver comparable with WinVBlock. It detects the MEMDISK mapped drives (CD/DVD, hard disk and floppy disk images) so Windows can read and write to those disks.

For more info about Firadisk (it can do more than detecting MEMDISK mapped drives) and how to build your Windows images: http://www.boot-land.net/forums/index.php?showtopic=8804

Windows images may need the 'raw' parameter on some PCs:

```
LABEL windows_firadisk
  LINUX memdisk
  INITRD windows_harddisk.img
  APPEND raw
```

```
LABEL windows_firadisk
  LINUX memdisk
  INITRD windows.iso
  APPEND iso raw
```

- Windows PE based

You can also build a RAM disk based Windows PE. RAM disk based discs use ramdisk.sys and setupldr.bin files from windows 2003 server SP1 source, see:

- http://www.911cd.net/forums/index.php?showtopic=10482
- http://www.911cd.net/forums/index.php?showtopic=19333

- Windows Vista/2008/7 with WIM images

Windows Vista/2008/7 can be booted from a WIM image (loaded from disk via INT 13h).

If you get the following message while booting the ISO:

```
A required CD/DVD drive device driver is missing.
 If you have a floppy disk, CD, DVD, or USB flash drive,
 please insert it now.
```

use the ImDisk solution, described at [1] (http://www.boot-land.net/forums/index.php?showtopic=11855).

Linux

The majority of Linux based CD images will also fail to work with MEMDISK ISO emulation. Linux distributions require kernel and initrd files to be specified, as soon as these files are loaded the protected mode kernel driver(s) take control and the virtual CD will no longer be accessible. If any other files are required from the CD/DVD they will be missing, resulting in boot error(s).

Linux distributions that only require kernel and initrd files function fully via ISO emulation, as no other data needs accessing from the virtual CD/DVD drive once they have been loaded. The boot loader has read all necessary files to memory by using INT 13h, before

booting the kernel.

There are ways to get around the problem of not finding the required files:

- Passing ISO parameter to the kernel

Some distributions allow you to pass an extra parameter to the kernel append line, which tells the init scripts to look for an ISO file on a disk. Some distro's require that the drive and partition number where the ISO is stored is explicitly specified, while others will search each partition for the specified filename.

This parameter is distro specific, so look at the docs of your distro. Some popular ones:

- **findiso=**
- **iso-scan/filename=**

- memdiskfind in combination with phram and mtdblock

There is also another solution, which requires the **phram** and **mtdblock** kernel module and **memdiskfind** utility of the Syslinux package (utils/memdiskfind). memdiskfind will detect the MEMDISK mapped image and will print the start and length of the found MEMDISK mapped image in a format phram understands:

```
modprobe phram phram=memdisk,$(memdiskfind)
modprobe mtdblock
```

This will create a /dev/mtdblock0 device, which should be the .ISO image, and should be mountable.

If your image is bigger than 128MiB and you have a 32-bit OS, then you have to increase the maximum memory usage of vmalloc, by adding:

```
vmalloc=<at_least_size_of_your_image_in_MiB>Mi

Example:
vmalloc=256Mi
```

to your kernel parameters.

memdiskfind can be compiled with the **klibc** instead of with the glibc C library to get a much smaller binary for use in the initramfs:

```
cd ./syslinux-4.04/utils/
make spotless
make CC=klcc memdiskfind
```

More info + links to the Arch Linux implementation: http://reboot.pro/11690/

Parameters and options

Specifying geometry and image type manually

You can also specify the geometry manually with the following command line options:

```
  c=#          Specify number of cylinders (max 1024[*])
  h=#          Specify number of heads (max 256[*])
  s=#          Specify number of sectors (max 63)
  floppy[=#]   The image is a floppy image[**]
  harddisk[=#] The image is a hard disk image[**]
  iso          The image is an El Torito ISO9660 image (drive 0xE0)

  # represents a decimal number.

   [*] MS-DOS only allows max 255 heads;
       and on floppy disks it only allows max 255 cylinders.

  [**] Normally MEMDISK emulates the first floppy or hard disk.  This
       can be overridden by specifying an index, e.g. floppy=1 will
       simulate fd1 (B:). This may not work on all operating systems
       or BIOSes.
```

Example:

```
# Boot harddisk image as second hard disk and specify C/H/S geometry
LABEL hdt_floppy
```

```
  LINUX memdisk
  INITRD harddisk.img
  APPEND harddisk=1 c=255 h=64 s=22
```

Set memory access method (raw, bigraw, int , safeint)

MEMDISK normally uses the BIOS "INT 15h mover" API to access high memory. This is well-behaved with extended memory managers which load later. Unfortunately it appears that the "DOS boot disk" from WinME/XP *deliberately* crash the system when this API is invoked. The following command-line options tells MEMDISK to enter protected mode directly, whenever possible:

```
  raw          Use raw access to protected mode memory.

  bigraw       Use raw access to protected mode memory, and leave the
               CPU in "big real" mode afterwards.

  int          Use plain INT 15h access to protected memory.  This assumes
               that anything which hooks INT 15h knows what it is doing.

  safeint      Use INT 15h access to protected memory, but invoke
               INT 15h the way it was *before* MEMDISK was loaded.
               This is the default since version 3.73.
```

It is possible that your image works fine without those parameters on some PCs or virtual machines, but fail to boot on other PCs. In this situation, adding 'raw' will normally solve your problem.

Example:

```
LABEL dos_with_extended_memory_manager
  LINUX memdisk
  INITRD dos_emm.img
  APPEND raw
```

Write protected floppy and hard disk images

The disk is normally writable (although, of course, there is nothing backing it up, so it only lasts until reset.) If you want, you can mimic a write-protected disk by specifying the command line option:

```
 ro           Disk is readonly
```

Example:

```
# Simulate write protected floppy
LABEL hdt_write_protected_floppy
  LINUX memdisk
  INITRD hdt.img
  APPEND ro
```

Hide real floppy or hard drive

Some systems without a floppy drive have been known to have problems with floppy images. To avoid such problems, first of all make sure you don't have a floppy drive configured on the BIOS screen. If there is no option to configure that, or that doesn't work, you can use:

```
  nopass       Hide all real drives of the same type (floppy or hard disk)
  nopassany    Hide all real drives (floppy and hard disk)
```

Example:

```
# Hide real floppy drive (when no real floppy drive is attached)
LABEL hide_floppy
  LINUX memdisk
  INITRD hdt.img
  APPEND nopass
```

Enable/Disable BIOS Enhanced Disk Drive Services

MEMDISK by default supports EDD/EBIOS on hard disks, but not on floppy disks. This can be controlled with the options:

```
 edd          Enable EDD/EBIOS
 noedd        Disable EDD/EBIOS
```

Example:

```
# Enable EDD for the HDT floppy
LABEL EDD_on_floppy
  LINUX memdisk
  INITRD hdt.img
  APPEND edd
```

MEMDISK and PXE booting

Similarly, if you're booting DOS over the network using PXELINUX, you can use the "keeppxe" option and use the generic PXE (UNDI) NDIS network driver, which is part of the PROBOOT.EXE distribution from Intel: [2] (http://www.intel.com/support/network/sb/CS-006120.htm)

```
  keeppxe      Keep PXE capabilities when booted from PXELINUX
```

Pause MEMDISK before booting

To view the messages produced by MEMDISK, use:

```
 pause        Wait for a keypress right before booting
```

Example:

```
# Pause MEMDISK before booting
LABEL pause_memdisk
  LINUX memdisk
  INITRD hdt.img
  APPEND pause
```

Set stack size

The following option can be used to set the real-mode stack size. The default is 512 bytes, but if there is a failure it might be interesting to set it to something larger:

```
 stack=size   Set the stack to "size" bytes
```

Example:

```
# Change stack size to 2kiB (2048 bytes)
LABEL change_stack_size
  LINUX memdisk
  INITRD hdt.img
  APPEND stack=2048
```

MEMDISK and generic El Torito CD-ROM driver for DOS

If you're using MEMDISK to boot DOS from a CD-ROM (using ISOLINUX), you might find the **generic El Torito CD-ROM driver (eltorito.sys)** by Gary Tong and Bart Lagerweij useful. It is now included with the Syslinux distribution, in the dosutil directory. See the file dosutil/eltorito.txt for more information.

Example usage of eltorito.sys in CONFIG.SYS:

```
device=eltorito.sys /X:MSCD0001
```

Corresponding MSCDEX command which can be placed in AUTOEXEC.BAT:

```
MSCDEX /X:MSCD0001 /L:X
```

Where **X** is the drive letter.

Accessing MEMDISK arguments from DOS

See the How to access MEMDISK arguments from DOS page.

Categories: MEMDISK | Menu
