The MikeOS User Handbook

For version 4.5, 21 December 2014 - (C) MikeOS Developers

This documentation file explains how to boot and use the MikeOS operating system on a real PC or an emulator. If you have just downloaded MikeOS and want to run it, this is the guide you need. If you have any questions, see the MikeOS website for contact details.

Click the links on the left to navigate around this guide.

Booting

Disk images

After you have extracted the MikeOS **.zip** file, switch into the **disk_images/** directory and you'll see three files:

- **mikeos.flp** -- Floppy disk image containing MikeOS and programs
- **mikeos.dmg** -- Same as above, but with a Mac-friendly extension
- **mikeos.iso** -- CD ISO image built using the floppy disk image

So, these files are virtual disk images that you can write to real floppy disks or CD-Rs, or run in a PC emulator as described in a moment.

Writing

For running MikeOS on a real PC, you will need to write one of the virtual disk images to physical media. If you have a USB key then this is simple -- we can write the floppy disk image to the USB key, and the PC will boot it like a virtual floppy.

On Linux, insert your USB key and unmount it when it appears (but don't remove it). Then open a command line window and enter **dmesg** to view the kernel messages. You will see an indication at the end of the messages of the device you just plugged in -- eg **/dev/sdb**. Note that you just need the device, eg /dev/sdb, rather than the number (eg /dev/sdb1). Then enter (in the disk_images directory):

```
dd if=mikeos.flp of=/dev/sdb
```

Of course, replace sdb with the device node. The key is now ready for booting.

On Windows, download the open source Flashnul program, plug in your USB key and enter **flashnul -p** to get a list of drives. When you've spotted your USB key, enter **flashnul [number] -L mikeos.flp** (in the disk_images directory), replacing [number] with the number you got before. The key is now ready for booting.

Note: if you plug your USB key back into the machine, the operating system may try to alter the partition structure and stop it from working properly. So treat it as a MikeOS-only key until you want to reformat it for normal use.

For floppy disks, on Windows you can use a program called RawWrite to copy **mikeos.flp** to a floppy disk. On Linux, use the **dd** utility like this:

```
dd if=mikeos.flp of=/dev/fd0
```

If you want to run MikeOS on a machine that doesn't have a floppy drive and doesn't boot from USB keys, you can burn and boot the **mikeos.iso** CD image. Any decent Windows CD burning software will allow you to write an ISO image to a CD-R; if you don't have one, try InfraRecorder.

On Linux, a graphical burning program such as K3b should do the trick, or you can use the command line:

```
cdrecord -dao dev=/dev/cdrom mikeos.iso
```

Real PCs

At a minimum, any 386 PC with 1MB of memory and a keyboard should be able to run MikeOS. In fact, you may be able to get it running on an older machine -- please do let us know if so! Just start your PC with the MikeOS floppy, CD-ROM or USB key inserted, and you should see the initial dialog screen.

On some systems, you may need to change the boot order in your BIOS so that the PC boots from the floppy, CD or USB key rather than the hard drive.

Emulators

A quick way to try MikeOS, and one that doesn't involve writing disk images to physical media, is to use an emulator. This is particularly useful if you're writing MikeOS software or changing the OS as described in the other two Handbooks.

Some of the best emulators:

- **QEMU** -- Small, simple and open source (link)
- **VirtualBox** -- Very powerful with a good GUI (link)
- **VMware** -- Popular proprietary virtualisation app (link)
- **Bochs** -- Takes a bit of work to set up, but good debugging tools (link)

For VirtualBox and VMware, configure the boot device to use the MikeOS floppy disk or CD ISO image. With QEMU on Linux, run **test-linux.sh**, or for QEMU on Windows switch into the directory where you installed the emulator and enter:

```
qemu.exe -L . -m 4 -boot a -fda mikeos.flp -soundhw all -localtime
```

You will need to change the path to **mikeos.flp** accordingly.

Running

Usage

When MikeOS starts up, you'll see a dialog box which gives you the option of a program list of a command line interface. Using the cursor keys and Enter, choose OK for the former and Cancel for the latter.

In the program list you can select a **.BIN** or **.BAS** program with the up/down cursor keys and hit Enter to run it. Also, you can press Esc to return back to the original list/CLI selection screen.

At the command line, enter **DIR** to show a list of programs, and **HELP** to display inbuilt commands. You can run a program by entering the full filename (eg **EDIT.BIN**) or just the name without the extension (eg **EDIT**). There are also file management commands such as COPY, REN, DEL and SIZE.

Programs

MikeOS includes several programs to perform various tasks and demonstrate features of the OS, such as:

- **EDIT.BIN** -- Simple full-screen text editor (Unix-type text files only)
- **EXAMPLE.BAS** -- Demonstration of BASIC features (open it in EDIT.BIN to explore)
- **FILEMAN.BIN** -- Delete, rename and copy files on the floppy disk
- **HANGMAN.BIN** -- Guess the names of cities around the world
- **MEMEDIT.BAS** -- Colourful, powerful memory editor
- **DRAW.BAS** -- ASCII art drawing program
- **CALC.BAS** -- Powerful calculator
- **SUDOKU.BAS** -- Sudoku game
- **CF.BAS** -- Cosmic Flight game
- **MUNCHER.BAS** -- Snake-like game (use WASD keys)
- **ADVNTURE.BAS** -- A text adventure
- **KEYBOARD.BIN** -- Musical keyboard; use the bottom row of keys to play and Q to quit

- **MONITOR.BIN** -- Simple machine code monitor (see below)
- **SERIAL.BIN** -- Minicom-like serial terminal program (see below)
- **VIEWER.BIN** -- Views text files and 320x200x16 PCX images such as **SAMPLE.PCX**

Note that **FILEMAN.BIN** and **EDIT.BIN** try to write to the floppy drive, so if you've booted from a CD-R and try to manipulate files you will see write errors as it's a read-only medium after burning.

Copying files

If you've written MikeOS to a real floppy disk, you can just copy extra files onto that disk in your file manager. But if you want to add files to the floppy disk images, that requires a bit of extra work -- you need to access the disk image as if it was a real floppy. First up is Linux: switch to the MikeOS main directory, then enter the following commands as root:

```
mkdir looptmp
mount -o loop -t vfat disk_images/mikeos.flp looptmp
```

Now the contents of the MikeOS virtual floppy disk image are accessible in the newly-created **looptmp/** directory. (We have loopback-mounted the disk image onto our filesystem.) Copy your programs into that directory, for example:

```
cp MYPROG.BIN looptmp/
```

When you're done, unmount the virtual floppy image and remove the temporary directory:

```
umount looptmp
rm -rf looptmp
```

You can now write **mikeos.flp** to a floppy disk or boot it in an emulator. If you want to recreate the CD ISO image, run **build-linux.sh** as root; this will update **mikeos.iso** with the new floppy contents.

If you're running Windows, you will need a special program to access **mikeos.flp** as if it was

a real floppy. One tool you can use is the ImDisk Virtual Disk Driver; download and run it to install. You can then mount the floppy disk image like this:

```
imdisk -a -f mikeos.flp -s 1440K -m B:
```

Copy your files into the **B:** drive. When you are finished, enter:

```
imdisk -d -m B:
```

Now the files that you copied to **B:** have been written into **mikeos.flp**.

Monitor

Yutaka Saito has contributed a MikeOS program that lets you enter machine code in hexadecimal format and execute it. Run **MONITOR.BIN** from the command line and you'll be presented with a '=' prompt. Now you can enter your instructions, or just 'x' to exit back to the OS.

MikeOS programs are loaded at the 32K (32768) point. The monitor converts hex code and executes it at location 36864 in RAM -- that is, 4K after the where the monitor program is loaded. This is so that your code doesn't overwrite the monitor! Consequently, any code you run should be ORGed to 36864. For example, this is a small MikeOS program which displays the letter 'M' on the screen. After we've assembled it, we can run **ndisasm** on the resulting binary to see the hexadecimal codes:

Source code	**Hexadecimal**
BITS 16	
%INCLUDE "mikedev.inc"	
ORG 36864	

mov si, message	BE0790
call os_print_string	E8FD6F
ret	C3
message db 'M', 0	4D00

(The first three lines are merely assembly directives, so they don't generate any code.) Now that we have the hex codes, we can enter them into the monitor. Note that the code must be terminated with a dollar sign ($) character, and spaces are allowed. So, you can enter at the '=' prompt:

```
BE0790 E8FD6F C3 4D00$
```

When you enter this, the monitor will convert the hex codes to machine code at location 36864 in RAM, and call that location to execute it. (Just like normal MikeOS programs, you should finish with a **ret** instruction.) After execution, you'll be returned to the monitor. You can then enter 'r' to re-run the converted code, or 'x' to exit.

Serial port

You can use MikeOS as a Minicom-like serial terminal emulator with **SERIAL.BIN**. This lets you connect a MikeOS machine to, for instance, a UNIX machine, and operate the UNIX machine from MikeOS. Connect a serial (null-modem) cable between the two machines, then set up your UNIX machine with a terminal session on the serial port.

For instance, if you have a Linux machine, you would add a line like this to /etc/inittab:

```
T0:2345:respawn:/sbin/getty/ -L ttyS0 9600 vt100
```

When you restart your Linux machine, it will wait for a login on the serial port. Connect the null-modem cable to a MikeOS machine, and run SERIAL.BIN in MikeOS. You can now enter

your username and password to log in.

Note that MikeOS configures the serial port to be 9600 baud, no parity, 8 data bits, 1 stop bit. If you wish to change these settings, edit **source/features/serial.asm** and see the port port setup code at the start of the file (then rebuild MikeOS as described in the *System Developer Handbook*). Also note that only a handful of VT100 commands have been implemented at present, so programs which do complicated things with the screen (such as Emacs) may not display properly.

To exit the program, press the F8 key. (You can change this to a different key by editing the source code near the start of **programs/serial.asm**.)

Extra

Help

If you have any questions about MikeOS, or you're developing a similar OS and want to share code and ideas, go to the MikeOS website.

License

MikeOS is open source and released under a BSD-like license (see **doc/LICENSE.TXT** in the MikeOS **.zip** file). Essentially, it means you can do anything you like with the code, including basing your own project on it, providing you retain the license file and give credit to the MikeOS developers for their work.