

The Midinux2 develop environment kit is a big tarball.

There are 2 method to use the Midinux2 development environment kit. One choice is: install it on a harddisk partition, and boot from that partition. This method is suitable for develop the hardware drivers, but need some modifications to enable it to bootup, and do not support all of computer hardware. Another method is install the develop kit in a directory, and use it by "chroot", start Midinux GUI by "Xnest". This method is suitable for develop applications.

Install on hard disk partition:

1. Installation

Before installation, you must install a linux system on your computer, and boot into this system.

1. Firstly, you must allocate a hard disk partition(i.e. /dev/hda1), format that partition, and mount it on a "***virbuild" directory. This HD partition must larger the 3G.

```
# mke2fs -j /dev/hda1
# mkdir -p /mnt/virbuild
# mount /dev/hda1 /mnt/virbuild
```

2. Un-compress the develop kit, make sure put the files into the mounted partition.

```
# tar xvfj -C /mnt [your path]/develop-Midinux2.tar.bz2
```

Now the DEK files have installed in partition /dev/hda1. But you have to do some modification to make that partition bootable.

3. Change configuration

a) make new initrd for your computer:

```
# cp /etc/modprobe.conf /mnt/virbuild/etc/
# chroot /mnt/virbuild/ mkinitrd -f /boot/initrd-2.6.22.1-0mid.img
2.6.22.1-0mid
```

b) Change the first 2 line of /mnt/virbuild/fstab, according your own harddisk partitons. In my case, these should be:

/dev/hda1	/	ext3	defaults	1 1
/dev/hda2	swap	swap	defaults	0 0

c) Change your grub configuration, add a boot switch as:

```
title Mid develop
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.22.1-0mid ro root=/dev/hda1
    initrd /boot/initrd-2.6.22.1-0mid.img
```

d) Change X configuration. The development kit is pre-configured to use “vesa” driver, use 800x600 resolution.

Intel Poulsbo VV, or Crown Bench, now use “psb” 2D graphic driver. If you want use that driver, please copy /tmp/poulsbo/midinuxdm and /tmp/poulsbo/xorg.conf to /etc/X11

```
# cd /mnt/virbuild
```

```
# cp ./tmp/poulsbo/midinuxdm ./tmp/poulsbo/xorg.conf ./etc/X11/
```

2. Bootup

Now you can reboot the computer.

in grub menu, choose “Mid develop” item. If you are lucky enough, the Midinux system will boot up, and the GUI environment will start.

3. Building drivers

The develop kit contains compiler and kernel source-tree, so building driver is very easy.

For demo propose, there is a uvcvideo driver source located in /opt/. The progress of building it:

```
# cd /opt
```

```
# tar xvfz uvcvideo-r125.tar.gz
```

```
# cd uvcvideo-r125/trunk
```

```
# make
```

The driver uvcvideo.ko will be generated.

Install in a directory

If you do not have a empty harddisk partiton, or the kernel-2.6.22.1-0mid can not support your computer, you have to use the second method. In this case, you need some small tips to build hardware driver.

1. Installation

Before installation, you must install a linux system on your computer, and make sure disable “selinux”. Boot into this system.

Un-compress the develop kit to a directory

```
# tar xvfj -C /mnt develop-Midinux2.tar.bz2
```

2. Start Midinux GUI in Xnest

1. Mount pseudo filesystems:

```
# cd /mnt
# mount --bind /proc virbuild/proc
# mount --bind /sys virbuild/sys
# mount --bind /dev virbuild/dev
# mount --bind /dev/pts virbuild/dev/pts
```

2. Start Xnest

```
# Xnest -geometry 800x480+0+0 -ac :1.0&
```

3. Change root to virbuild system, and start gui

```
# chroot virbuild
# export DISPLAY=localhost:1.0
# /usr/bin/af-sb-init.sh restart
```

Now the GUI will displayed in the Xnest.

The develop kit also provide scripts “startgui” and “stopgui”. You can simply running it to start GUI:

```
# cd /mnt
# ./startgui
```

3. Build drivers

To build applications, you should mount the pseudo filesystem firstly, then change root the the virtual build system. Then you can build your apps.

Since the virtual build system is running over a real linux OS, so you may meet some difficult while build hardware driver. Usually this come from the command “uname -r”. So we have to modify the driver source’s “Makefile”.

For example, we will build the uvcvideo in the virtual build system:

```
# cd /opt
# tar xvfz uvcvideo-r125.tar.gz
# cd uvcvideo-r125/trunk
```

Now modify the Makefile, the 2 lines:

```
KERNEL_VERSION := `uname -r`
KERNEL_DIR      := /lib/modules/$(KERNEL_VERSION)/build
```

Change to:

```
KERNEL_VERSION := 2.6.22.1-0mid
KERNEL_DIR      := /usr/src/kernels/2.6.22.1-0mid-i686
```

Then you can run “make” and get correct driver.