

This document is the instruction of how to use USB 2.0 Card Reader on Linux.

Note. The kernel might use 2.4.0 or above.

Step 1: Getting the source code:

You can obtain the source via anonymous ftp from ftp.kernel.org in /pub/linux/kernel/vx.y, where x.y is the version (eg 2.2), and as mentioned before, the ones that end with an odd number are development releases and may be unstable.

It is typically labelled linux-x.y.z.tar.gz, where x.y.z is the version number.

The sites also typically carry ones with a suffix of .bz2, which have been compressed with bzip2 (these files will be smaller and take less time to transfer).

It's best to use ftp.xx.kernel.org where xx is your country code; examples being ftp.at.kernel.org for Austria, and ftp.us.kernel.org for the United States.

Step 2: Unpacking the source:

Log in as or su to 'root', and cd to /usr/src.

If you installed kernel source when you first installed linux (as most do), there will already be a directory called 'linux' there, which contains the entire old source tree. If you have the disk space and you want to play it safe, preserve that directory. A good idea is to figure out what version your system runs now and rename the directory accordingly.

The command 'uname -r' prints the current kernel version. Therefore, if 'uname -r' said '1.0.9',

you would rename (with 'mv') 'linux' to 'linux-1.0.9'. If you feel mildly reckless,

just wipe out the entire directory. In any case, make certain there is no 'linux' directory in /usr/src before unpacking the full source code.

Now, in /usr/src, unpack the source with 'tar xzpvf linux-x.y.z.tar.gz' (if you've just got a .tar file with no .gz at the end, 'tar xpvf linux-x.y.z.tar' works.).

The contents of the source will fly by. When finished, there will be a new 'linux' directory in /usr/src.

cd to linux and look over the README file. There will be a section with the label 'INSTALLING the kernel'.

Carry out the instructions when appropriate -- symbolic links that should be in place, removal of stale .o files, etc.

If you have a .bz2 file and the bzip2 program (read about it at <http://www.muraroa.demon.co.uk/>), do this:

```
bz2cat linux-x.y.z.tar.bz2 | tar xvf -
```

Step 3: Backup old kernel:

The below step is kernel compile and build so it's more important to backup old kernel.

If the compile and build fail it can recover the system.

The backup files have:

```
/boot/vmlinuz  
/boot/System.map
```

Step 4: Make mrproper:

It is important to begin a kernel build with the source tree in a known condition.

Therefore, it is recommended that you begin with the command `make mrproper`.

This will remove any configuration files along with the remains of any previous builds that may be scattered around the source tree.

Step 5: Configuring the kernel:

Note: Some of this is reiteration/clarification of a similar section in Linux' README file.

The command `make config` while in `/usr/src/linux` starts a configure script which asks you many questions.

It requires bash, so verify that bash is `/bin/bash`, `/bin/sh`, or `$BASH`.

However, there are some much more pleasant alternatives to `make config` and you may very well find them easier and more comfortable to use.

`make menuconfig` is probably the most widely-used. Whatever you choose, it's best to get familiar with the interface because you may find yourself back at it sooner than you think.

For those `running X`, you can try `make xconfig` if you have Tk installed (`click-o-rama` - Nat).

`make menuconfig` is for those who have (n)curses and would prefer a text-based menu.

These interfaces have a rather clear advantage: If you goof up and make a wrong choice during configuration,

it is simple to go back and fix it.

The configuration options will appear in hierarchies with `make menuconfig` and `make xconfig`.

You are ready to answer the questions, usually with `y` (yes) or `n` (no).

Device drivers typically have an `m` option. This means `module`, meaning that the system will compile it,

but not directly into the kernel, but as a loadable module. A more comical way to describe it is as `maybe`.

Some of the more obvious and non-critical options are not described here; see the section `Other configuration options` for short descriptions of a few others.

With `make menuconfig`, the space bar toggles the selection.

The USB reader need following items:

SCSI emulation support = y

SCSI support = y

SCSI disk support = y

SCSI generic support = y

Support for USB = m

UHCI Alternate Drive(JE) or OHCI (Compaq, iMacs, OPTi, SiS, ALi, ...) support = m

USB Mass storage support = m

Preliminary USB device filesystem = y

Dos FAT fs support = y

MSDOS fs support = y

UMSDOS: Unix-like file system on top of standard MSDOS fs

VFAT (Windows-95) fs support

The option items:

USB verbose debug message = y

USB Mass storage verbose debug =y

Step 6: Make dep:

This step is start compile and build kernel.

## LINUX\_HOWTO.txt

### Step 7: Make clean:

This step is to clean old kernel and driver

### Step 8: Make bzImage:

If compile successd please do the following step:

- (1) `cd /usr/src/linux/arch/i386`
- (2) `cp bzImage /boot/vmlinuz-x.x.x` (x.x.x is kernel version)
- (3) `cd /boot`
- (4) `ln -s -i vmlinuz-x.x.x vmlinuz`

### Step 9: Make modules and Make modules\_install:

This step is compile the modules.

Because we configure have choose m so we need make modules and make modules\_install.

### Step 10: Refresh the System.map

The setp is flloing:

- (1) `cp System.map /boot/System.map-x.x.x` (x.x.x is kernel version)
- (2) `cd /boot`
- (3) `ln -s -i System.map-x.x.x System.map`

### Step 11: Edit the lilo.conf or grub.conf:

If you selected GRUB as your boot loader, you need to modify the file /boot/grub/grub.conf like as following:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You do not have a /boot partition. This means that
#           all kernel and initrd paths are relative to /, eg.
#           root (hd0, 0)
#           kernel /boot/vmlinuz-version ro root=/dev/hda1
#           initrd /boot/initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
#####Our addition information#####
title Red Hat Linux (x.x.x)
    root (hd0,0)
    kernel /boot/vmlinuz-x.x.x ro root=/dev/hda1
    initrd /boot/initrd-x.x.x.img
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
##### x.x.x is compiler kernel version #####
#####
title Red Hat Linux (2.4.18-3)
    root (hd0,0)
    kernel /boot/vmlinuz-2.4.18-3 ro root=/dev/hda1
    initrd /boot/initrd-2.4.18-3.img
title DOS
    rootnoverify (hd0,4)
    chainloader +1
```

If you selected LIL0 the same of the GRUB

### Step 12: Mount the usb device:

The late thing is mount the device.

- (1) first create a folder of /mnt  
example: `mkdir /mnt/cf`
- (2) mount the device  
example: `mount /dev/sda1 /mnt/cf`

**LINUX\_HOWTO.txt**