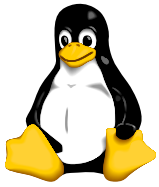

LPIC 102

Compiling a kernel

1 Aim

We aim to be able to compile a Linux kernel.

2 Background

In this laboratory the old DEC network cards use a driver that is included in the kernel source code, but it is not built in the default setup that we use in this laboratory. We need this built as a *module* that can be loaded into the kernel so that we can use the network card.

There are a number of steps in the process:

- We verify that the kernel source code is installed.
- I prefer to build the kernel not as root, but as a normal user. We will change the ownership of the kernel source code to our own account.
- We edit the Makefile and change the `EXTRAVERSION` name near the top so that the kernel will have a different name from the kernel provided by the distribution.
- We copy a suitable configuration file from those provided by the distribution, or we can go through and choose the details of configuration ourselves, though this is quite time consuming, as there are many configuration options to choose. The name of the configuration file is `.config`
- We start a configuration program that provides a graphical user interface that edits the configuration file with `make xconfig`
- We calculate the dependencies (which things need to be compiled) with `make dep`.
- We build the kernel with `make bzImage`
- We build the modules with `make modules`
- We install the modules with `sudo make modules_install`
- We copy the kernel to the `/boot` directory
- We also copy the `System.map` file and the `.config` files to the `/boot` directory.
- Generate a RAM disk file with mkinitrd.
- We edit the GRUB configuration so we can boot our new kernel.
- We boot into our new kernel and watch with a happy smile as everything works.

3 Procedure

3.1 Change the Ownership of the Source Code

1. Notice the dot '.' at the end of the `chown` command. Use your own username instead of `jimmy`.

```
$ cd /usr/src ↩
$ ls ↩
debug  linux-2.4  linux-2.4.21-15.EL  redhat
$ cd linux-2.4.21-15.EL ↩
$ chown -R jimmy.jimmy . ↩
```

   We will call the current location the *top level* of the source directory.

2. Let us clean up any other residue from previous compiles with

```
$ make clean ↩
```

3.2 Change the Name of the Kernel: Edit the `Makefile`

1. With the editor of your choice (`emacs`, `gedit`, `vi`, `nano`... , edit the file `Makefile` in the top level directory, and

2. change the line

```
EXTRAVERSION = -15.EL
```

   to

```
EXTRAVERSION = -15.ELde4x5
```

3.3 Get a Configuration File

1. Red Hat provide some configuration files in the `config` directory. Get an appropriate configuration for the computer and...

```
$ cd configs ↩
$ ls ↩
kernel-2.4.21-athlon.config         kernel-2.4.21-ia32e.config
kernel-2.4.21-athlon-smp.config     kernel-2.4.21-ia64.config
kernel-2.4.21-i386-BOOT.config      kernel-2.4.21-ppc64.config
kernel-2.4.21-i386.config           kernel-2.4.21-ppc64iseries.config
kernel-2.4.21-i586.config           kernel-2.4.21-ppc64pseries.config
kernel-2.4.21-i586-smp.config       kernel-2.4.21-s390.config
kernel-2.4.21-i686.config           kernel-2.4.21-s390x.config
kernel-2.4.21-i686-hugemem.config   kernel-2.4.21-x86_64.config
kernel-2.4.21-i686-smp.config       kernel-2.4.21-x86_64-smp.config
$ cp kernel-2.4.21-i686.config ../.config ↩
```

   ...copy it to the name `.config` in the top level.

2. Have a look at the configuration file with `less` or your editor.

3.4 Edit the Configuration with `make xconfig`

1. In the top level directory, type:

   ```
   $ make xconfig ↩
   ```

   This will start a Tk program that allows you to select configuration options.

2. Choose the menu item Network Interfaces

3. Find the item that corresponds the the `de4x5` module, and select **`m`** for *module* You can click on the help buttons to find what each configuration option is for.

4. Save the configuration and exit the program.

3.5 Make the Dependencies

```
$ make dep ↩
```

3.6 Build the Kernel

```
$ make bzImage ↩
```

3.7 Build the Modules

```
$ make modules ↩
```

3.8 Install the Modules

```
$ sudo make modules_install ↩
```

3.9 Copy the kernel to /boot

```
$ sudo cp -a arch/i386/boot/bzImage /boot/vmlinuz-2.4.21-15.ELde4x5 ↩
$ sudo cp -a System.Map /boot/System.map-2.4.21-15.ELde4x5 ↩
$ sudo cp -a .config /boot/config-2.4.21-15.ELde4x5 ↩
```

3.10 Make an RAM disk file with mkinitrd

```
$ cd /boot ↩
$ /sbin/mkinitrd -v initrd-2.4.21-15.ELde4x5.img 2.4.21-15.ELde4x5 ↩
```

3.11 Edit GRUB Configuration to Boot the New Kernel

1. Edit `/boot/grub/grub.conf` (or `/boot/grub/menu.lst`) and copy the section for booting the old kernel

2. Edit it, changing the name of the kernel and initrd file to match what you have made

3. Boot the computer into your new kernel and rejoice!