Linux Kernel

Morteza Noorelahi Alamdari(Ipo)

- Linux kernel history
- Structure
- What we need
- Configuring / Building / Installing / Upgrading
- Customizing

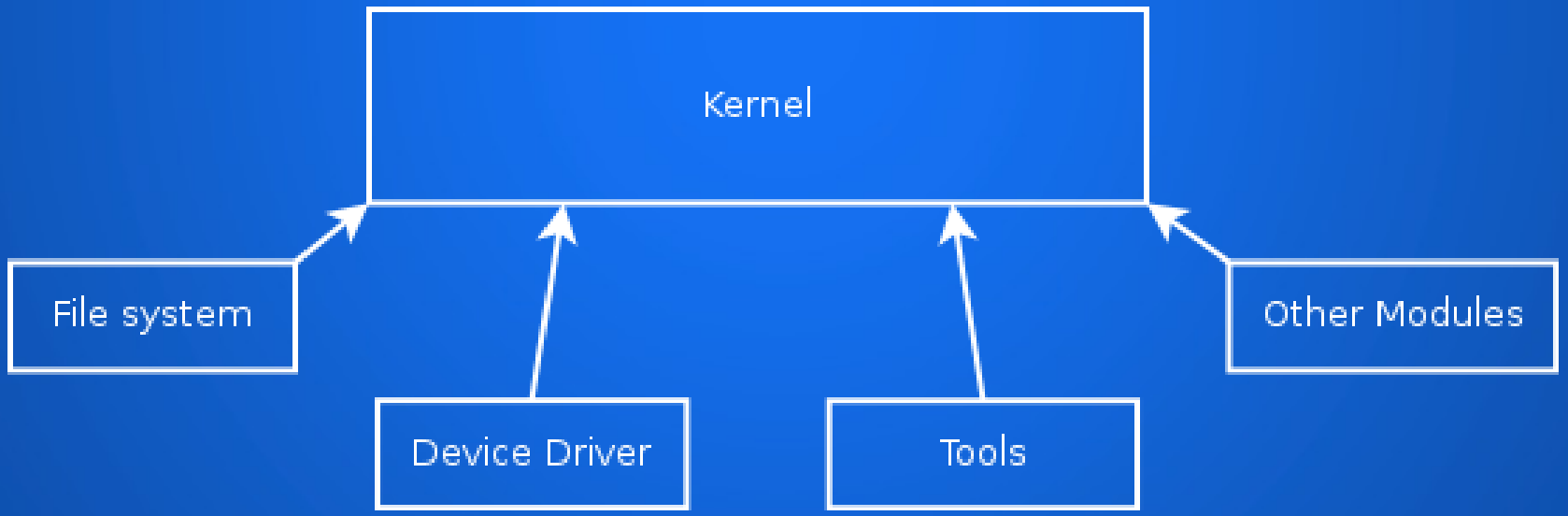
History:

Linux is a clone of the operating system Unix, written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net. It aims towards POSIX and Single UNIX Specification compliance.

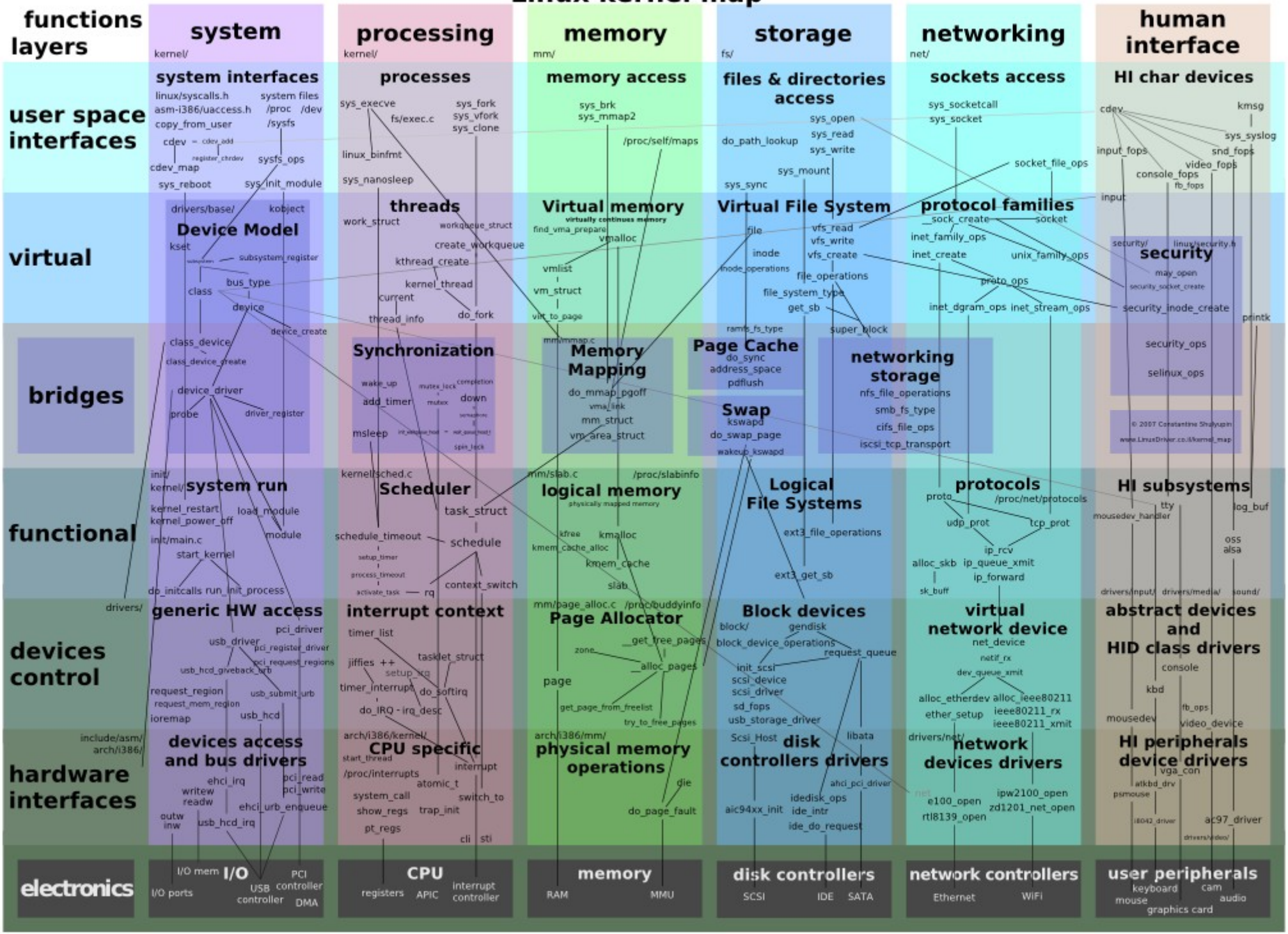
License : GNU General Public License

Although originally developed first for 32-bit x86-based PCs (386 or higher), today Linux also runs on (at least) the Compaq Alpha AXP, Sun SPARC and UltraSPARC, Motorola 68000, PowerPC, PowerPC64, ARM, Hitachi SuperH, Cell, IBM S/390, MIPS, HP PA-RISC, Intel IA-64, DEC VAX, AMD x86-64, AXIS CRIS, Xtensa, Tilera TILE, AVR32 and Renesas M32R architectures.

Structure:



Linux kernel map



What we need:

GCC is a C compiler
Make is a tools that walks the kernel source tree to determine which files need to be compiled.
Binutils is linking and assembling of source files.
Git is version controller [optional]

Getting Kernel source code :

$ wget http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.6.6.tar.bz2

Or :

$ curl http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.6.6.tar.bz2 -o linux-3.6.6.tar.bz2

Then :

$ tar -xjzf linux-3.6.6.tar.bz2

$ cd linux-3.6.6

What is Git?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Redhat:

```
# yum install git
```

Debian:

```
# apt-get install git
```

OpenSuse:

```
# zypper install git
```

Arch:

```
# pacman -S git
```

Simple example about Git:

You can get clone projects via this command:

$ git clone https://github.com/torvalds/linux.git

Redirect to linux directory:

$ cd linux

For showing list of logs:

$ git log

For showing list of versions:

$ git tags

For getting last update from the git server:

$ git pull

Simple example about Git:

For pushing changed parts to the git server:

```
$ git push
```

For showing status of project:

```
$ git status
```

For configing git :

```
$ git config [--global] user.name “your name”
$ git config [--global] user.email “your email”
```

Configuring:

For showing help:

$ make help

We have several ways for configuring kernel:

1- For configuring step by step:

$ make config

2- For configuring with default config:

$ make defconfig

3- For configuring in graphical mode:

$ make menuconfig/gconfig/xconfig

4- see `make help` ...

For removing all generated files + config + various backup files:

$ make mrproper

Building:

For building kernel:

```
$ make
```

Advanced building options:

```
$ make -j4
$ make -j8
$ make -j
```

For getting iso file:

```
$ make isoimage
```

For making deb package:

```
$ make deb-pkg
```

For making rpm package:

```
$ make rpm-pkg
```

For making targz file:

```
$ make targz-pkg
```

Installing:

For install any modules that we want:

```
# make module_install
```

Then for installing kernel in your system:

```
# make install
```

Updating:

For updating this version we need to get patch files from the official site:

```
$ wget http://kernel.org/patches/patch-3.6.6.7.10.bz
```

Redirect to the directory of kernel source:

```
$ cd linux-3.6.6
```

And for adding patch in the kernel:

```
$ bzip2 -dv patch-3.6.6.7.10.bz | patch -p1
```

Or:

```
$ bzip2 -dv patch-3.6.6.7.10.bz
$ cd linux-3.6.6
$ patch -p1 < ../patch-3.6.6.7.10
```

Config file:

Sometimes you want to use other distributions config file, Like ubuntu.
You can find config file of ubuntu via live cd!

```
$ cp /proc/config.gz .
$ gzip -dv config.gz
$ cp config linux-3.6.6/.config
```

And then you can run `make` command.

Customizing:

Determining the driver:

Example network driver:

ls /sys/class/net

$ basename $(readlink /sys/class/net/eth0/device/driver/module)

$ r8169 << output

$ cd linux-3.6.6

$ fine . -type f -a -name Makefile | grep -i r8169

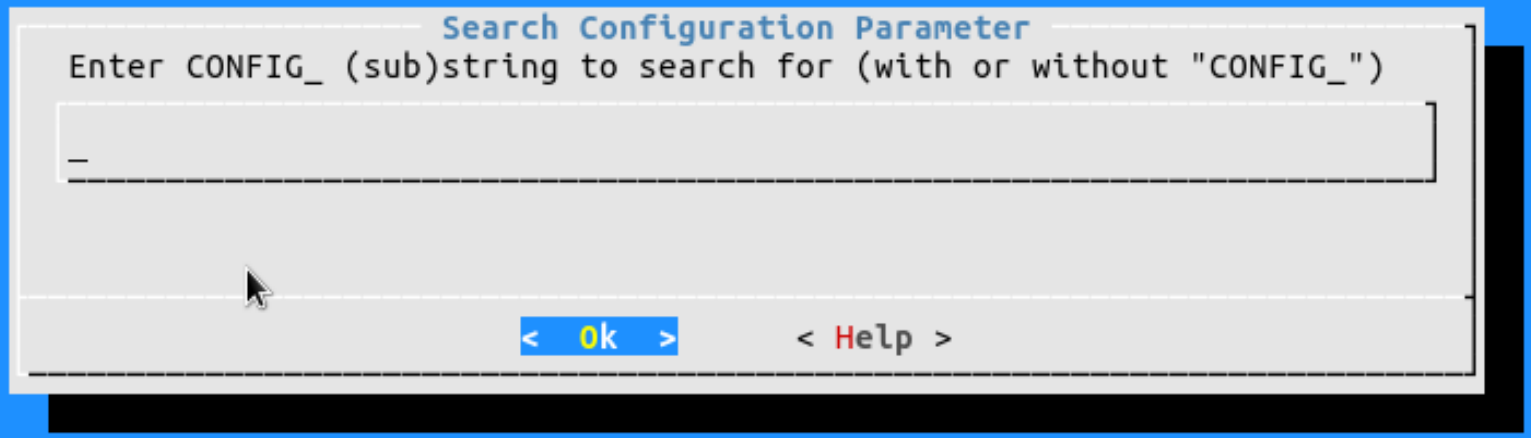
$ ./drivers/net/ethernet/realtek/Makefile:obj-$(CONFIG_R8169) += r8169.o

$ make menuconfig

=> press /

=> type module name: r8169

Now you can enable this module in the kernel.



Search Results

```
Symbol: R8169 [=n]
Type  : tristate
Prompt: Realtek 8169 gigabit ethernet support
  Defined at drivers/net/ethernet/realtek/Kconfig:105
  Depends on: NETDEVICES [=y] && ETHERNET [=y] && NET_VENDOR_REALTEK [=y] && PCI [=y]
  Location:
    -> Device Drivers
      -> Network device support (NETDEVICES [=y])
        -> Ethernet driver support (ETHERNET [=y])
          -> Realtek devices (NET_VENDOR_REALTEK [=y])
  Selects: FW_LOADER [=y] && CRC32 [=y] && NET_CORE [=y] && MII [=y]
```

Determining the correct module from scratch:

PCI devices:

```
# lspci | grep -i ethernet
# 13:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168B
PCI Express Gigabit Ethernet controller (rev 06)
$ cd /sys/bus/pci/devices/
$ ls
$
0000:00:00.0  0000:00:1a.0  0000:00:1c.1  0000:00:1d.0  0000:00:1f.2  0000:12:00.0  0000:ff:00.1
0000:ff:02.2 0000:00:02.0  0000:00:1b.0  0000:00:1c.2  0000:00:1e.0  0000:00:1f.3  0000:13:00.0
0000:ff:02.0  0000:ff:02.3 0000:00:16.0  0000:00:1c.0  0000:00:1c.4  0000:00:1f.0  0000:00:1f.6
0000:ff:00.0  0000:ff:02.1
$ cd cd 0000\:13\:00.0/
$ cat vendor
$ 0x10ec   << output
$ cat device
$ 0x8168   << output
$ cd linux-3.6.6
$ grep -i 0x10ec include/linux/pci_ids.h
$ #define PCI_VENDOR_ID_REALTEK             0x10ec
```

$ grep -R1 PCI_VENDOR_ID_REALTEK * | less
/0x8168 << search device
Result:

```
--
drivers/net/ethernet/realtek/r8169.c-static DEFINE_PCI_DEVICE_TABLE(rtl8169_pci_tbl) = {
drivers/net/ethernet/realtek/r8169.c:   { PCI_DEVICE(PCI_VENDOR_ID_REALTEK,     0x8129), 0, 0, RTL_CFG_0 },
drivers/net/ethernet/realtek/r8169.c:   { PCI_DEVICE(PCI_VENDOR_ID_REALTEK,     0x8136), 0, 0, RTL_CFG_2 },
drivers/net/ethernet/realtek/r8169.c:   { PCI_DEVICE(PCI_VENDOR_ID_REALTEK,     0x8167), 0, 0, RTL_CFG_0 },
drivers/net/ethernet/realtek/r8169.c:   { PCI_DEVICE(PCI_VENDOR_ID_REALTEK,     0x8168), 0, 0, RTL_CFG_1 },
drivers/net/ethernet/realtek/r8169.c:   { PCI_DEVICE(PCI_VENDOR_ID_REALTEK,     0x8169), 0, 0, RTL_CFG_0 },
drivers/net/ethernet/realtek/r8169.c-   { PCI_DEVICE(PCI_VENDOR_ID_DLINK,       0x4300), 0, 0, RTL_CFG_0 },
--
```

All PCI drivers contain a list of different devices that ther support.
Vendor id matches with device.

Summery:

Here are steps needed in order to find which pci driver can control a specific PCI device:

1. Find the PCI bus ID of the device for which you want to find the driver , using lspci.
2. Go into the /sys/bus/pci/devices/0000:bus_id directory , where bus_id is the PCI bus ID found in the previous step.
3. Read the values of the vendor and device file in the PCI device directory.
4. Move back the kernel source tree and look in include/linux/pci_ids.h for the PCI vendor and device IDs found in the previous step.
5. Search the kernel source tree for references to those values in drivers . Both the vendor and device ID should be in a struct pci_device_id definition.
6. Search the kernel Makefile for the CONFIG_ rule that builds this driver by using find and grep: $ find . -type f -a -name Makefile | xargs grep DRIVER_NAME
7. Search in the kernel configuration system for that configuration value and go to the location in the menu that it specifies to enable that driver to be built.

References:

Oreilly : Linux Kernel in the nutshell
http://www.wikipedia.org/
https://github.com/torvalds/linux
http://kernel.org/

Thanks for your attention