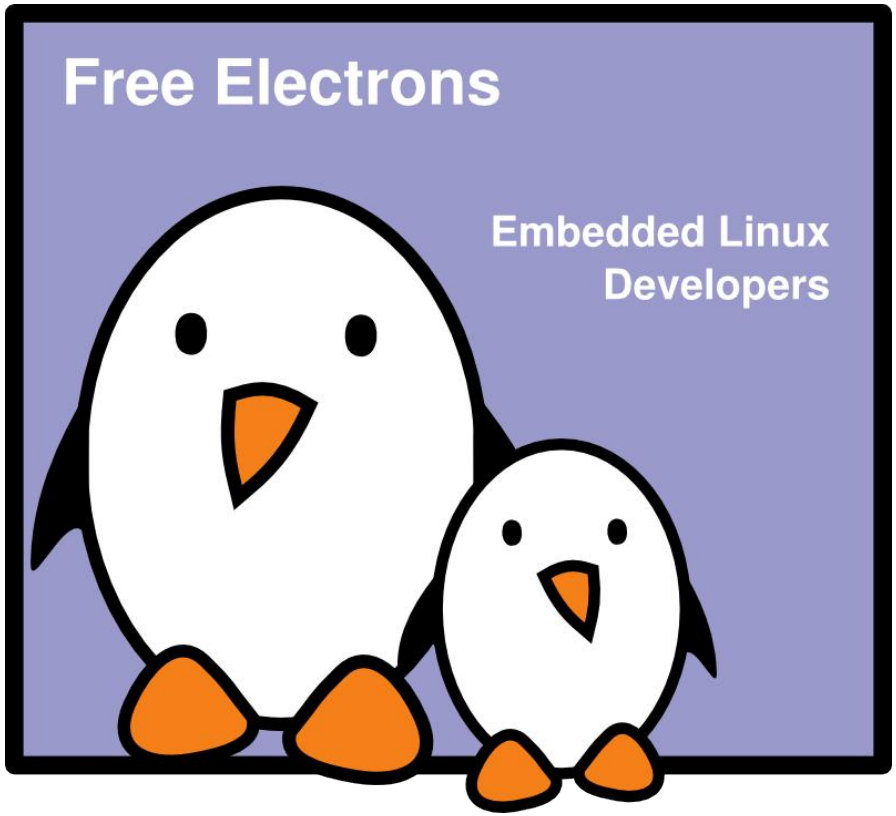
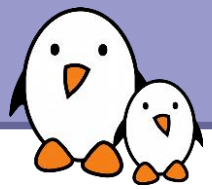
System administration basics

Michael Opdenacker
Thomas Petazzoni
Free Electrons



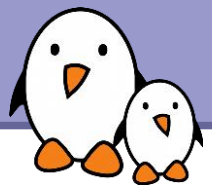
Latest update: Dec 20, 2010,
Document sources, updates and translations:
http://free-electrons.com/docs/command-line
Corrections, suggestions, contributions and translations are welcome!



Networking

Network setup (1)

- `ifconfig -a`
  Prints details about all the network interfaces available on your system.
- `ifconfig eth0`
  Lists details about the `eth0` interface
- `ifconfig eth0 192.168.0.100`
  Assigns the `192.168.0.100` IP address to `eth0` (1 IP address per interface).
- `ifconfig eth0 down`
  Shuts down the `eth0` interface (frees its IP address).

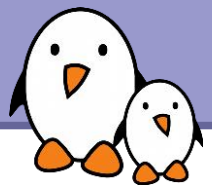


Network setup (2)

- `route add default gw 192.168.0.1`
  Sets the default route for packets outside the local network. The gateway (here `192.168.0.1`) is responsible for sending them to the next gateway, etc., until the final destination.
- `route -n`
  Lists the existing routes

`-n` option: immediately displays ip addresses instead of trying to find their domain names

- `route del default`
  or `route del <IP>`
  Deletes the given route
  Useful to redefine a new route.

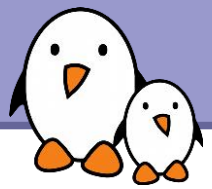


Network setup (3)

- Your programs need to know what IP address corresponds to a given host name (such as `kernel.org`)
- Domain Name Servers (DNS) take care of this.
- You just have to specify the IP address of 1 or more DNS servers in your `/etc/resolv.conf` file:

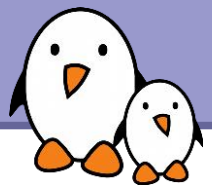
```
nameserver 217.19.192.132
nameserver 212.27.32.177
```

- The changes take effect immediately!

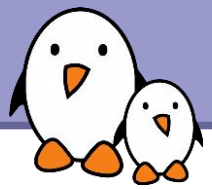


Network testing

- First, try to ping the IP address of your gateway.
  This will confirm that your network adapter works fine.
- Then, make sure you can ping the name server IP address, which will confirm that your gateway is configured properly.
- Finally, make sure you can ping any host using its name, which will confirm that the nameserver configuration is correct.



Filesystems and devices



Creating filesystems

Examples

- `mkfs.ext2 /dev/sda1`
  Formats your USB key (`/dev/sda1`: 1st partition raw data) in ext2 format.

`-F`: force. Execute even if not a real device file.

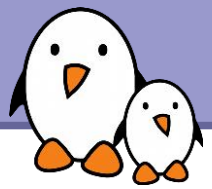
- `mkfs.ext2 -F disk.img`
  Formats a disk image file in ext2 format

- `mkfs.vfat -v -F 32 /dev/sda1` (`-v`: verbose)
  Formats your USB key back to FAT32 format.

- `mkfs.vfat -v -F 32 disk.img`
  Formats a disk image file in FAT32 format.

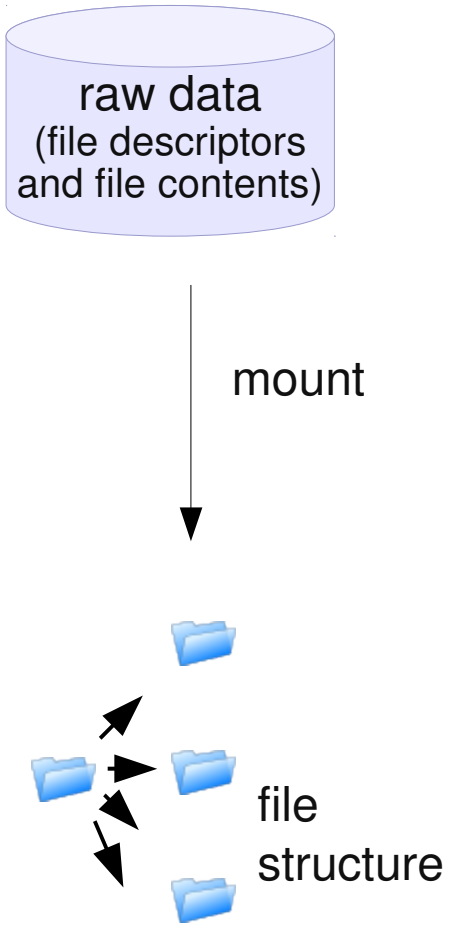
Blank disk images can be created as in the below example (64 MB file):

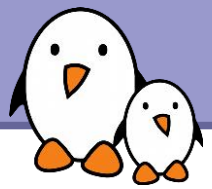
```
dd if=/dev/zero of=disk.img bs=1M count=64
```



Mounting devices (1)

- To make filesystems on any device (internal or external storage) visible on your system, you have to *mount* them.
- The first time, create a mount point in your system: `mkdir /mnt/usbdisk` (example)
- Now, mount it:
  `mount -t vfat /dev/sda1 /mnt/usbdisk`
  `/dev/sda1`: physical device
  `-t`: specifies the filesystem (format) type (`ext2`, `ext3`, `vfat`, `reiserfs`, `iso9660`...)



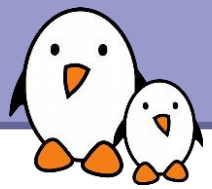


Mounting devices (2)

You can also mount a filesystem image stored in a regular file (*loop devices*)

- Useful to develop filesystems for another machine
- Useful to access the contents of an ISO cdrom image without having to burn it.
- Useful to have a Linux filesystem inside a file in a Windows partition.

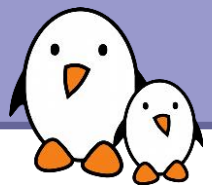
```
cp /dev/sda1 usbkey.img
mount -o loop -t vfat usbkey.img /mnt/usbdisk
```



Listing mounted filesystems

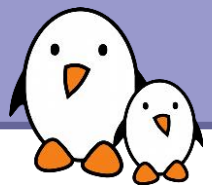
Just use the `mount` command with no argument:

```
/dev/hda6 on / type ext3 (rw,noatime)
none on /proc type proc (rw,noatime)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/hda4 on /data type ext3 (rw,noatime)
none on /dev/shm type tmpfs (rw)
/dev/hda1 on /win type vfat (rw,uid=501,gid=501)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

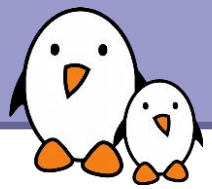


Unmounting devices

- `umount /mnt/usbdisk`
  Commits all pending writes and unmounts the given device, which can then be removed in a safe way.
- To be able to unmount a device, you have to close all the open files in it:
  - Close applications opening data in the mounted partition
  - Make sure that none of your shells have a working directory in this mount point.
  - You can run the `lsof <mount point>` command (list open files) to view which processes still have open files in the mounted partition.

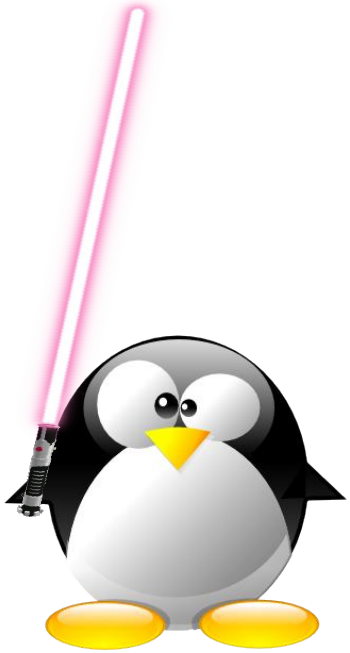


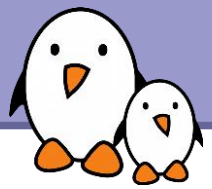
Package management



Beware of the dark side of root

- `root` user privileges are only needed for very specific tasks with security risks: mounting, creating device files, loading drivers, starting networking, changing file ownership, package upgrades...
- Even if you have the `root` password, your regular account should be sufficient for 99.9 % of your tasks (unless you are a system administrator).
- In a training session, it is acceptable to use `root`. In real life, you may not even have access to this account, or put your systems and data at risk if you do.

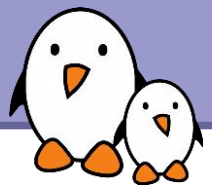




Using the root account

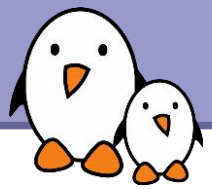
In case you really want to use `root`...

- If you have the `root` password:
  `su -` (**s**witch **u**ser)
- In modern distributions, the `sudo` command gives you access to some `root` privileges with your own user password.
  Example: `sudo mount /dev/hda4 /home`



Software packages

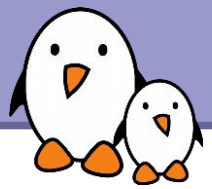
- The distribution mechanism for software in GNU/Linux is different from the one in Windows
- Linux distributions provides a central and coherent way of installing, updating and removing applications and libraries : **packages**
- Packages contains the application or library files, and associated meta-information, such as the version and the dependencies
  - .deb on Debian and Ubuntu, .rpm on Mandriva, Fedora, OpenSUSE
- Packages are stored in **repositories**, usually on HTTP or FTP servers
- One should only use packages from official repositories of its distribution, unless strictly required.



Managing software packages (1)

Instructions for Debian based GNU/Linux systems (Debian, Ubuntu...)

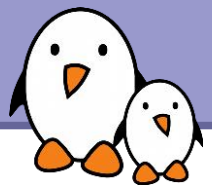
- Package repositories are specified in `/etc/apt/sources.list`
- To update package repository lists:
  `sudo apt-get update`
- To find the name of a package to install, the best is to use the search engine on http://packages.debian.org or on http://packages.ubuntu.com. You may also use:
  `apt-cache search <keyword>`



Managing software packages (2)

- To install a given package:
  `sudo apt-get install <package>`
- To remove a given package:
  `sudo apt-get remove <package>`
- To install all available package updates:
  `sudo apt-get dist-upgrade`
- Get information about a package:
  `sudo apt-cache show <package>`
- Graphical interfaces
  - Synaptic for GNOME
  - Adept for KDE

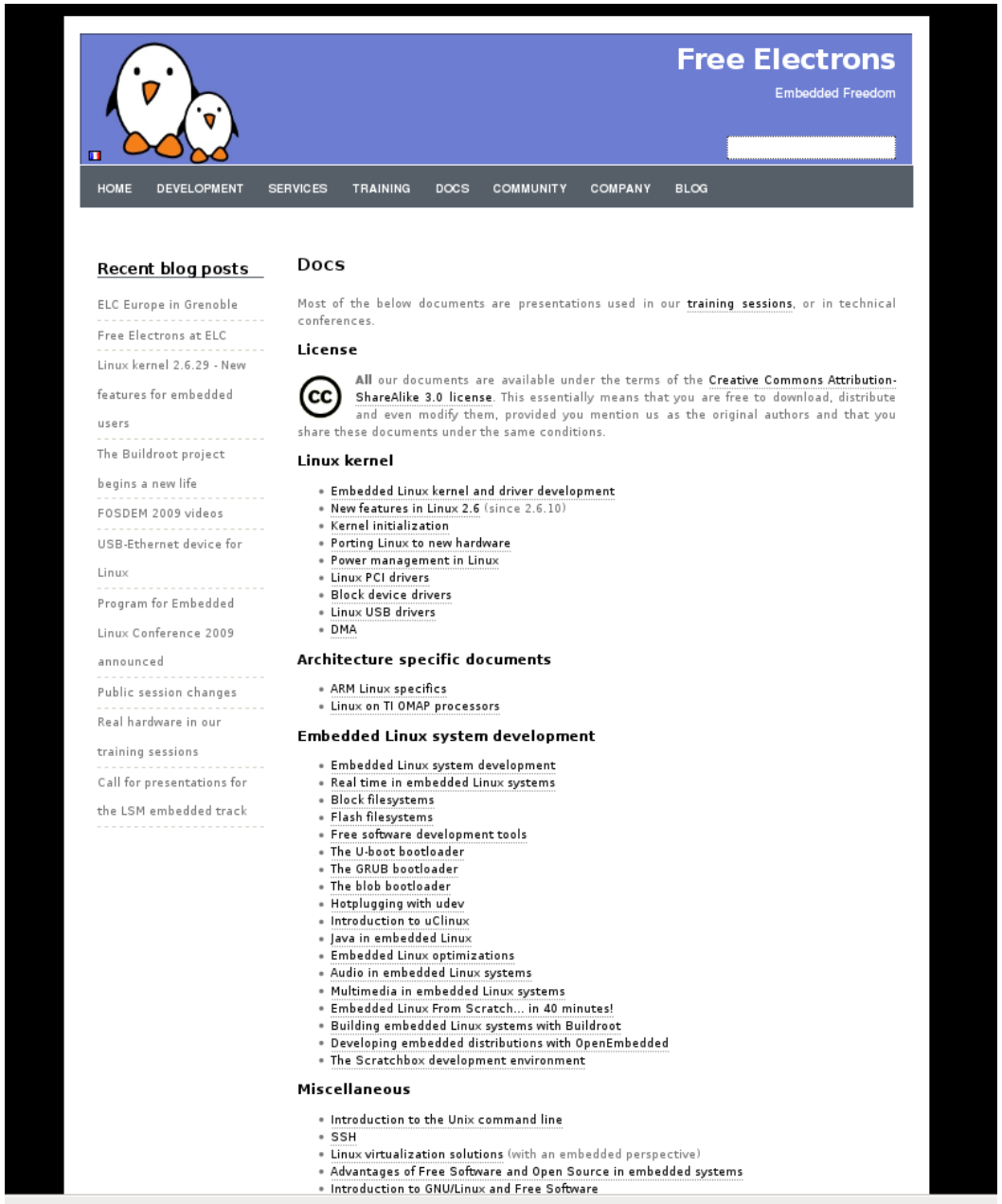
Further details on package management:
http://www.debian.org/doc/manuals/apt-howto/



Shutting down

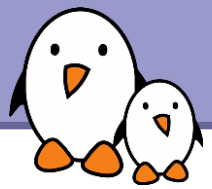
- `halt`
  Immediately halts the system.
- `reboot`
  Immediately reboots the system.
- `[Ctrl][Alt][Del]`
  Also works on GNU/Linux to reboot.
  Embedded systems: you must use an implementation of `init` and can specify any key combination in `/etc/inittab`.

Related documents



All our technical presentations on http://free-electrons.com/docs

- Linux kernel
- Device drivers
- Architecture specifics
- Embedded Linux system development



How to help

You can help us to improve and maintain this document...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order development, consulting and training services performed by the authors of these documents (see http://free-electrons.com/).
- By sharing this document with your friends, colleagues and with the local Free Software community.
- By adding links on your website to our on-line materials, to increase their visibility in search engine results.

Linux kernel

Linux device drivers
Board support code
Mainstreaming kernel code
Kernel debugging

Embedded Linux Training

All materials released with a free license!

Unix and GNU/Linux basics
Linux kernel and drivers development
Real-time Linux, uClinux
Development and profiling tools
Lightweight tools for embedded systems
Root filesystem creation
Audio and multimedia
System optimization

Free Electrons

Our services

Custom Development

System integration
Embedded Linux demos and prototypes
System optimization
Application and interface development

Consulting and technical support

Help in decision making
System architecture
System design and performance review
Development tool and application support
Investigating issues and fixing tool bugs