- 78 - Devices, Modules und Hardware

Hardware & Architecture

- General hardware
  - Processor, BIOS, RAM , Address Bus system, Data Bus system
- Address and IRQ conflicts
  - IRQ Table
    - Used by system: 1,2,6,8,14,15
    - Mostly Free: 5,9,10,11,13
    - May be Freeable: 3,4,7,12
- IRQ Sharing: PCI Mostly OK, ISA – Not sharable
- DMA:
  - 8 DMA Channels.
  - DMA-4 is always busy.
  - Normal use: DMA 1,2 or 3.
  - Watch for conflicts DMA!!
- Setting Hardware ADDR,IRQ,DMA
  - Peripherals Integrated in Motherboard: via BIOS
  - Old ISA expansion boards: Jumpers and DIP Switches on boards
  - Newer ISA expansions boards: Jumpers(ADDR)and software (IRQ)
  - ISA Plug-And-Play expansion boards: BIOS or OS PNP feature
  - PCI expansion cards: Via BIOS and OS using PCI bus
  - Memory Base of certain PCI cards for RAM or BIOS direct on the cards

- **Setting and reading the hardware and system time**

```
date                  Show current system time and date
date -s "15:34"       Set the System time

hwclock               Show the hardware clock time setting
hwclock --localtime   Keep hardware clock to local time mode
hwclock --utc         Keep hardware clock to utc time mode

hwclock --set --date="9/22/2002 16:45:05"
hwclock --hctosys     Sets the system time to current hardware clock
```

- Setting the Hardware and System clock in one command:
  `setclock 09/18/2003 21:13:00`
  (Thu Sep 18 21:13:00 EDT 2003)
- Time Variables in `/etc/sysconfig`
  `HWCLOCK="--localtime"` for `localtime` mode `-u` for `utc` time mode
  When SuSE boots-up it set the time from the script `/etc/init.d/boot`
- Files that have some relation to time are:
  `/usr/lib/zoneinfo/localtime` --> `/etc/localtime` (binary)
  `/etc/adjtime` Temporary file used to adjust the time regularly

- `ksysctrl-` Is good at displaying the found system devices a-la-Windows.
- `hwinfo` - Shows a lot of automatically found hardware and their info. (SuSE)
- `lsdev` - Shows a list of recognized devices and their I/O Addr, IRQ and DMA
- `procinfo-` Shows a list of recognized devices and their I/O Addr and IRQ
- `MAKEDEV` - Command to create devices
- `losetup` - Set up and control loop devices

- **KERNEL MODULES** (general)

  - To list all the Kernel modules already loaded:
    `lsmod`
    `cat /proc/modules`

  - To get more info about a module
    `modinfo` *`modulename`*

  - To get the description of all current kernel modules:
    ```
    find /lib/modules -name "*.ko" -exec bash -c \
    "basename {}| cut -d. -f1 ; modinfo -F description \
    $(basename {}| cut -d. -f1)" \;
    ```

  - To load a kernel module use `modprobe` or `insmod`.
    `modprobe` is recommended because it checks the dependencies of the module.

  - To remove a kernel module:
    `modprobe -r` *`modulename`* (without the `.o` or `.ko`) or
    `rmmod` *`modulename`* (without the `.o` or `.ko`)

  - To list all loadable kernel modules who woudn't load properly because of missing symbols: (missing symbols = dependency not respected):
    `depmod` (see `man depmod` for more info on modules dependency)

  - Configuration files for Hardware modules:

`/etc/modules.conf`	Older configuration used by modprobe to change the way a module is loaded or unloaded. Although this file is an older format it provides a lot of functions.
`/etc/modprobe.conf`	Newer configuration file for modprobe command which is used for the same purpose as `/etc/modules.conf` (older).

    Note: It is still unknown to me which one of the two above configuration files would be used if both would be present in a system.

    Listing the modules options of `/etc/modprobe.conf`:
    `modprobe -c`

- **Getting information on hardware**

- **USB**

`lsusb` - Lists all connected USB devices
`/sbin/hotplug` - Script, handles hot-pluggable PCI & USB devices.
`rchotplug {start|stop}` - Starts/Stops USB and PCI configurator.
`usbmodules --device /proc/bus/usb/`*`NNN/nnn`*
- Lists kernel modules corresponding to USB devices currently plugged into the computer. *eg.*
`usbmodules --device /proc/bus/usb/001/009`

- **PCI**

`lspci` - List all PCI devices
`cat /proc/pci` - "" "" ""
`setpci` - Configure PCI devices
`pcitweak` - Read/write/list PCI config space
`scanpci` - Scan/probe PCI buses
`/sbin/hotplug` - SuSE cript to handle hot-pluggable PCI and USB devices
`rchotplug {start|stop}` - Starts/Stops USB and PCI configurator

- **PCMCIA**

`cardinfo` - X-Program to list and control PCMCIA cards
`cardctl` - ASCII program to control the PCMCIA cards
`dump_cis` - ASCII program to list PCMCIA cards and their parameters
`cardmgr` - Daemon who loads and unloads PCMCIA kernel modules for inserted cards.
`/etc/init.d/pcmcia` - Script to load PCMCIA `cardmgr` as daemon

- **PNP**

`lspnp` - To list Plug and Play BIOS device nodes and resources.
`/etc/isapnp.conf` - File used by `isapnp` see also `man setpnp` for info on controlling `pnp` devices resources.
`isapnp /etc/isapnp.conf` - Sets the PNP devices according to `/etc/isapnp.conf`

- **SCSI**

`sg_map` - Displays mapping between `sg` and other SCSI devices.
`cat /proc/scsi/scsi` - Displays information about all SCSI devices that can be : `hdx`, `srx`, `sgx`, `scdx`
`scsiinfo -l` - List of active SCSI device in system. eg. `/dev/sda /dev/scd0` etc.
`sg_reset` - exercises SCSI *device/bus/host* reset capability
`scsi_info` - SCSI device description tool
`sg_test_rwbuf` - Tests the SCSI host adapter by issueing write and read operations on a device's buffer and calculating checksums.
`lsscsi` - list all SCSI devices (or hosts) currently on system
`mover` - utility to control scsi media changers
`sg_scan` - does a SCSI bus scan and prints the results to STDOUT

`sg_senddiag`	- performs a SCSI SEND DIAGNOSTIC command
`sg_logs`	- reads SCSI LOG SENSE pages
`scsidev`	- populate `/dev/scsi` with device names that are persistent against SCSI configuration changes.
`sg_start`	- starts (spins-up) or stops (spins down) SCSI devices
`sg_map`	- displays mapping between sg and other SCSI devices
`scsiinfo`	- query information from a scsi device
`sg_readcap`	- calls a READ CAPACITY command on a SCSI device
`sg_rbuf`	- reads data using SCSI READ BUFFER command
`sg_inq`	- outputs data retrieved from the SCSI INQUIRY command
`sginfo`	- outputs mode sense information for a SCSI generic device the given device
`sg_modes`	- reads SCSI MODE SENSE pages
`xmover`	- X11 frontend for scsi media changers
`scsi_devfs_scan`	- Scan SCSI devices within a devfs tree
`sane-find-scanner`	- find SCSI and USB scanners and their device files
`scsiformat`	- low level format an scsi disk device

- **SERIAL**
  `cat /proc/tty/drivers` - The serial ports being detected.
- **CDROMS**
  `cat /proc/sys/dev/cdrom/info`
  - The CD-ROMS devices names and their capabilities. Note: scsi cdroms can be `scdx`
- **I/O ADDRESSES**
  `cat /proc/ioports` - I/O Addresses used by which device.
- **I/O MEMORY**
  `cat /proc/iomem` - Memory Addresses used by the devices.
- **INTERRUPTS**
  `cat /proc/interrupts` - Interrupts used by the devices
- **DMA**
  `cat /proc/dma` - DMA channels in use.
- **CPU**
  `cat /proc/cpuinfo` - CPU hardware information
- **DEVICES**
  `cat /proc/devices` - Character & Block devices used and their IDs.
  `lsdev` - Display recognized devices IRQ,DMA and IO.
- **KERNEL OPTIONS**
  `cat /proc/cmdline` - Kernel options given at boot time
- **FILESYSTEMS**
  `cat /proc/filesystems` - Filesystem types recognized by linux.
  'nodev' = it doesn't have any physical device.
- **SYSTEM MEMORY**
  `cat /proc/meminfo` - System Memory management information

• The `/proc` file system.

- Connection (schnittstelle) to the kernel. Mostly ReadOnly
- Each process get a directory in `/proc` (named after their PID). Content is:

`cmdline`	What started the process
`cwd`	Symlink to dir where user was when he started the command
`environ`	Environment of process.
`exe`	Symlink to the running program (full path)
`root`	root dir. for the process. (may be changed using command `chroot`)
`fd`	file descriptors (eg. 0,1,2,255. used in `prgm 1>&2` etc.)

- Hardware information/parameters: readable via the program `cat` or `less`:

Hardware Parameters

`interrupts`	IRQ used by peripherals
`ioports`	IO Address used by peripherals
`dma`	DMA used by peripherals
`iomem`	Video RAM/ROM, System RAM/ROM, PCI system memory, VESA Frame buffer, reserved areas.

Other hardware information

`cpuinfo`	Processor type/model, speed, internal cache size, etc.
`partitions`	List of known local PC partitions with major and minor numbers.
`pci`	Scan of peripherals on PCI bus and AGP slot.

Kernel and software information

`cmdline`	Kernel start command and parameters.
`filesystems`	List if file systems know by the Kernel.
`meminfo`	Info about usage of available memory
`modules`	List of loaded modules
`mounts`	List of mounted filesystems. Here are also the mounted filesystems that were mounted with the option `-n` and hidden from `/etc/mtab` and `df` command.
`version`	Present Kernel version.

Extra important directories in `/proc`

`bus`	Info about system busses found in systems
`ide`	Info about IDE controllers and devices
`scsi`	Info about SCSI controllers and devices
`net`	Network info like ARP Info, Routing table etc
`sys`	WRITEABLE system control table.

- ## Plug and Play

  - Description PNP cards have an internal list of Addr,IRQ,DMA to which it can set-itself to, if requested.
    Linux is automatic NOT PNP compatible. It must be done manually.
    Two programs are available for this:

    - `pnpdump` Scans the ISA bus for PNP cards and displays the possible settings of each PNP card found.
    - `isapnp` Reads a PNP configuration file and sets the PNP cards accordingly.

  - Manual Process: (Using: SuSE package:`isapnp` Debian package: `isapnptools`
    1. Collect possible settings from PNP cards. Scans addresses `0x0273` to `0x03f3`
       `pnpdump > /etc/isapnp.conf`
    2. Edit the file and activate the desired settings of each PNP cards (*)
       `vi /etc/isapnp.conf`
    3. Set the PNP cards as per `/etc/isapnp.conf`. Must be done at every boot.
       `isapnp /etc/isapnp.conf`

    (*)Editing the `/etc/isapnp.conf`
    Find:
    IO ADDRESS:
    - First IO base address possible: `Minimum IO base address 0x0`*`240`*
    - Last IO base address possible: `Maximum IO base address 0x0`*`3e0`*
    - Address block size: `Number of IO addresses required:` *`32`*
    - Take a look at the already used IO addresses in system : `cat /proc/ioports`
    - Make a list of possible IO base addresses for this card.
      (First IO base address possible + Address block size) etc
      eg. `240, 260, 280, 2A0, 2c0, ..., ..., 3e0`
    - Chose a free address, write it in the following line and ucomment the line:
      `(IO 0 (BASE 0x`**`340`**`))`

    IRQ:
    - Proceed the same way as above for IRQs and at the end uncomment the line:
      `(INT 0 (IRQ` *`xx`* `(MODE +E)))` (*xx*=chosen IRQ)
    - Finally activate the card by uncommenting the line: `(ACT Y)`

- ## Configure Fundamental BIOS Settings

- **Key files, terms, and utilities:**
  `/proc/ioports`
  `/proc/interrupts`
  `/proc/dma`
  `/proc/pci`

- **Purpose of BIOS:** Middleman program (in ROM) between non-standard hardware architecture (main board) and the operating system. Linux deals directly with some hardware (eg. IDE Controller) for reasons of speed and better control.

- **CMOS Set-up program:**
  Triggered at booting by possible key combinations: DEL (Entf), F2, <Ctrl-Alt-ESC> etc.

- **Hard disk Set-up:**
  Although the hard disk controller is accessed directly by Linux, some HD settings in CMOS are still important.
  - HD cylinders have physically less sectors in the inside of the disk as on the outside.
  - LBA (Large Block Architecture) logically diminishes the number of cylinders and raise the number of heads.
  - LBA is important if Number of physical cylinders is more than 1024 even if Linux doesn't use the BIOS to access the HD.
    Reasons: - `fdisk` reads the BIOS for HD Parameters
    - `lilo` and GRUB are loaded from the BIOS.

- **Errors handling:**
  Normal is: `Halt on all errors` (no booting continues if any type of error occur)
  Linux server without keyboard: `Halt on all errors but keyboard`.

- **Peripherals settings:**
  - Turn off any unused device. eg. COM ports, Mouse, IDE channels if SCSI used, etc

- **IRQ Reserving for older ISA cards**
  These parameters are read from the ISA PNP and PCI cards and deletes them from their possible parameters list. When asked for possible set-ups,(eg. by `pnpdump`) the PNP cards will not have these reserved addresses.

- ## Configure Modem and Sound cards

Modems

- Check the hardware compatibility list from the distribution used.
- Good source of hardware info is the Hardware-HOWTO
- Normal modems are controlled by AT commands (Hayes compatible)
- Watch for WinModems. They are not real hardware modems. This section is not for them. Winmodems are hardware that don't have this AT commands intelligence and rely on drivers to silmulate it. More about it at `www.linmodems.org`

Sound Cards

- LPI concentrates on OSS sound technique. (Open Sound System)
- Each sound board type needs its own kernel module.
- Program for sound card installation: `sndconfig` (RedHat and others)
  - It scans possible sound cards IO ports and is menu driven.

    Standard I/O port for soundcard is: **********

    It handles the PNP and older ISA sound cards as well.

- ## Setup SCSI Devices

- **Key files, terms, and utilities:**
  `SCSI ID`
  `/proc/scsi/`
  `scsi_info`
- **Notes:**
  - SCSI=**S**mall **C**omputer **S**ystem **I**nterface
  - Purpose: Learning to set-up the SCSI devices in respect to BIOS, SCSI-ID, booting
  - Use o SCSI: Still in server industry, offers reliability, endurance, Hot-Plug features.
  - Tools: SCSI-ID, `/proc/scsi, scsi_info`
    (`scsi_info` is from Packages: SuSE: `pcmcia`, Debian: `pcmcia-cs`)

- **Architecture of SCSI:**
  - Number of devices with SCSI, including the SCSI controller itself:
    Standard: 8
    Wide: 16
    Ultra-Wide 32

  - Properties and rules of SCSI
    - Cable joining the devices is 50 wires wide
    - No 'T' branching in the cable
    - Each end of the cable must be terminated 330 Ohms to GND and 220 Ohms to +5
    - Minimum 10cm of cable between SCSI devices
    - Maximum length of 50strands cable: 3 Meters (>4 devices Max:1.5 Meters)
    - End of the cable must have a device attached to it.

  - Types of SCSI:
    - Standard(SCSI-1): 8 Devices 10 Mhz Maximum
    - SCSI-2, FAST-SCSI-2, Wide-SCSI-2(68 strand cable, 16 bit bus): Faster, command set is better
    - SCSI-3 even faster but still in development (no meaning for LPI)

  - SCSI speed table:

Bus width	Cable Width	Standard	Fast	Ultra	Devices
8-Bit	50 Strands	5 MB/sec	10 MB/sec	20 MB/sec	7+Ctrlr
Wide-16-Bit	68 Strands	10 MB/sec	20 MB/sec	40 MB/sec	15+Ctrlr

    Possible names alike Ultra-Wide- or Fast-Wide, etc are possible

  - Addressing SCSI devices:
    - SCSI-ID = 0 to 7 or 0 to 15
    - The SCSI Controller on the highest priority = highest ID: 7 or 15
    - If booting is from SCS then boot HD must be on ID 0
    - Each SCSI-ID can contain LUNs(Logical Unit Number)
    - Each CSCI cable(Bus) receives also a number (0,1,2 etc)
    - Each SCSI device can then be identified as follows:
      BusNumber, SCSI-ID, LUN
      Normally `0,`*`x`*`,0` eg. `/dev/sda` is on `0,0,0`

- SCSI Onboard BIOS
  - Separate and unknown from system BIOS
  - Used to boot SCSI drives and changing controller parametrers
  - Cheap Controllers don't usually have On-Board BIOS. More expensive ones do.
  - Newer Controllers even allows via software to assign SCSI-IDs to devices.
  - Role of the Controller:
    - Assignment of SCSI-IDs to devices
    - Selecting the data transfer rate of devices
    - Section of boot drive

- Booting from SCSI drive.
  - Controller must have an onboard BIOS
  - In SCSI onboard BIOS: Set the boot drive
  - In System BIOS: Set boot drive sequence to 'SCSI'

- SCSI in Linux
  - `/proc/scsi` directory contains all SCSI devices as a sub-directory
  - Each sub-directory contains files named by SCSI-BUS number (0,1,2)
  - Theses files contain the list of devices attached to this bus.
  - The file `/proc/scsi/scsi` contains the list of all found SCSI devices.

- Naming of SCSI devices
  - Hard disks are named from `sda, sdb` ... as per sequence they are found
  - Removable ZIP and USB Chip readers are also in the hard disk class
  - SCSI CD-ROMS are named 2 names at the same time: `sr`*x* & `scd`*x* (*x*=0,1,2,3,..)
  - Each device is also identified by *SCSI-BUS*, *SCSI-ID*, *LUN*
  - Program `scsi_info` shows info on individual device:

    eg. `scsi_info /dev/scd0`

- ## Setup different PC expansion cards

- **Key files, terms, and utilities:**

```
/proc/dma
/proc/interrupts
/proc/ioports
/proc/pci
pnpdump(8)
isapnp(8)
lspci(8)
```

  - Tools used:
    - Info files: `/proc/dma, /proc/interrupts, /proc/ioports, /proc/pci`
    - Programs: `pnpdump(8), isapnp(8), lspci(8)`
  - Important for LPI is: - Hardware parameters (IO Port, IRQ,DMA)
    - `/proc` directory
    - ISA Plug and Play in Linux
    - Setting and reading the time
  - PCI devices are identified by an unique ID just like MAC address in network cards. Linux saves these PCI IDs in the file:
    - `/usr/share/pci.ids` (SuSE)
    - `/usr/share/hwdata/pci.ids` (RedHat & Debian)
    - `/usr/share/mics/pci.ids` (Debian older than above....)

    `update-pciids` command updates the list from internet into: `/usr/share/mics/pci.ids.new` or equivalent as per distribution.
  - Linux support PCI(Bus ID=`00`) devices fully without needing manual settings.
  - AGP Is a separate PCI bus(Bus ID=01) reserved for Graphic Cards, having only one slot, made for undisturbed data transfer between the Graphic chips and the PC.
  - PCI Bus system is addressed the same way as SCSI:

    `BusNr``:`*`SlotNr`*`:`*`FunctionNr`*`(Device Nr.)`
  - `lspci` is used to list the PCI devices in system.
    `lspci` finds the manufacturers etc from the file `/usr/share/pci.ids`.
    `lspci -n` forces the NOT-reading of the `pci.ids` file, giving Mfg info in Numbers.
  - Newer kernels as `2.1.82` has more info about devices on PCI-Bus in the `/proc/pci`.
  - Serial ports known as `COM1`,`COM2` etc are in Linux: `ttyS0`,`ttyS1` etc.
    Parallel Printer ports known as `lpt1 lpt2`, are in llnux: `lp0, lp1` etc .

- ## Configure Communication Devices

- **Key files, terms, and utilities:**
  `/proc/dma`
  `/proc/interrupts`
  `/proc/ioports`
  `setserial(8)`

  - Tools: `/proc/dma`, `/proc/ioports`, `/proc/interrupts`, `setserial(8)` (`setserial` is from package `setserial` for SuSE,RedHat & Debian)

  - `minicom` is one of the modem terminal programs for linux.

  - To check for which serial ports are present in system:
    `setserial -g /dev/ttyS*`

  - `setserial /dev/ttyS`*`X`* Shows the settings of the serial port.
    or `/dev/cua`*`X`* *X*=0,1,2,3...eg. `ttyS0=COM1, ttyS1=COM2`

  - `setserial /dev/ttyS`*`X`* *`parameter`*
    Sets the serial port to the parameters.

    Parameters are:

`port` *`PortNr`*	IO Port number
`irq` *`IRQ`*	IRQ number
`uart` *`UART_Type`*	UART(Universal Assynchroneous Receiver Transmitter) Possible values are: none, 8250, 16450, 16550, 16550, 16550A, 16650V2, 16654, 16750, 16850,16950, 16954. none=Turn device OFF

    Most older application know only up to 38400 Baud. To allow for faster speeds even though the application asks for 38400 Baud, extra parameters to `setserial` set flags in hardware that translates requests from applications of 38.4Kb to higher speeds in UART.

Parameter	Speed requested by Aplication	Real UART speed
`spd_normal`	38.4Kb	38.4Kb
`spd_hi`	38.4Kb	57.6Kb
`spd_vhi` **(Important for LPI)**	38.4Kb	115Kb
`spd_shi`	38.4Kb	230Kb
`spd_warp`	38.4Kb	460Kb

- Modem AT Commands

  Hayes compatible commands that controls most modems.

AT	Sets the baud rate between Modem and PC
ATD Nr.	Dial the Number (Nr.)
ATH0	HangUp
ATH1	Pic-up phone line (Opposite of HangUp)
ATX0	Dial blind, CONNECT when connection OK
ATX1	Dial blind, CONNECT *speed* when connection OK
ATX2	Wait for DIALTONE and CONNECT *speed* when connection OK
ATX3	Dial blind, CONNECT *speed* when connection OK or BUZY
ATX4	Wait for DIALTONE and CONNECT *speed* when connection OK
ATX5	Dial blind, CONNECT *speed* when connection OK, BUSY, VOICE
ATX6	Wait for DIALTONE and CONNECT *speed* when connection OK, BUZY, VOICE
ATZ	Reset the modem.
AT&F	Reset the internal modem configuration to factory settings.

- ## Configure USB devices

  - **Key files, terms, and utilities:**
    ```
    lspci(8)
    usb-uhci.o
    usb-ohci.o
    /etc/usbmgr/
    usbmodules
    /etc/hotplug
    ```
  - Main USB module is usbcore (although often already integrated in kernel)
  - There are 2 types of USB controllers:
    OHCI Open Host Controller Interface (Compaq)
    UHCI Universal Host Controller Interface (Intel)
  - All USB devices are compatible to both OHCI and UHCI.
  - Main boards manufactures using:

OHCI	**UHCI**	**EHCI**(USB 2.0)
Compaq	Intel	Intel
Ali	VIA	VIA
NEC		NEC
Opti Chipset		Phillips

  - `lspci` or `less /proc/pci` To recognize the USB controller type:
    IO address format: `0xHHHH=UHCI`, `0xHH000000=OHCI`
  - The possible modules are: `ohci.o` , `uhci.o` or `ehci-hcd.o`
  - Autoloading at booting: in `/etc/modules.conf`---->entry: `alias usb uhci`
    To also autoload (post-install) other submodules:
    eg.(in `/etc/modules.conf`):
    ```
    alias usb uhci
    post-install uhci modprobe printer
    post-install printer modprobe joydev
    post-install joydev modprobe hid
    ```

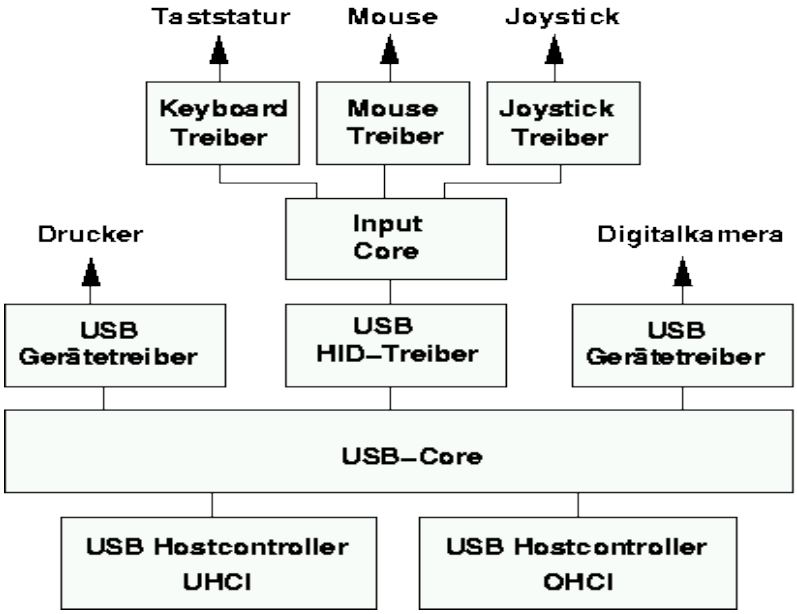
USBDevFS Filesystem:

This dynamic filesystem (like `/proc`) is normally mounted on `/proc/bus/usb`.
Its `/etc/fstab` entry looks like:
```
none  /proc/bus/usb  usbdevfs  defaults  0 0
```

After the mounting, the content(2 files) of `/proc/bus/usb` looks like this:
```
-r--r--r--  1 root root 0 2003-10-18 00:02 devices
-r--r--r--  1 root root 0 2003-10-18 00:02 drivers
```

After loading the driver(`usb-ohci` or `usb-uhci`) then content of this directory grows to include 1 numbered(001,002 etc) directory for each USB device.
The files in these numbered directories are in binary format.

- **Architecture of USB Modules:**



HID = Human Interface Device.(hid.o) and Input Core (input.o) are only for a USB keyboard(`usbkbd.o`), Mouse(`usbmouse.o`) or Joystick otherwies not needed.
Other USB modules:

printers	`printer.o`
storage	`usb-storage.o`

USB devices can be listed via the command: `lsusb`

For printers,when `printer.o` is loaded it creates devices `/dev/usb/lp0 .. lp1.`

List of current usb modules:

Commmand: `find /lib/modules/ -name "usb*" -exec basename {} \;`

```
usb-ohci.o        usbserial.o
   usb-uhci.o          usb-storage.o
   usbcore.o           usb-midi.o
                       usbkbd.o
                       usblcd.o
                       usbmouse.o
                       usbnet.o
                       usbvideo.o
                       usbvnet5.o
                       usbvnet5_2958.o
                       usbvnetr.o
```

Dynamic loading of USB Modules

For dynamically load the proper USB module when a USB device is inserted, 2 dynamic systems are available:

`hotplug`	Oversees the Hotplugged devices: USB, PCMCIA, FireWire (ieee1394)
`usbmgr`	USB manager that oversees only the USB devices.

- **hotplug**
  - At boot time the hotplug daemon is started via the script `/etc/init.d/hotplug`.
  - When a new device is inserted, the kernel senses it , it then passes an agent name as parameter to the daemon listed in the file: `/proc/sys/kernel/hotplug` (normally `/sbin/hotplug`).
  - The kernel then fills in the Environment Variables DEVICES with the info about the device and ACTION about if the device was pluged or unplugged.
  - The hotplug daemon starts the proper agent script.
  - The agent script reads the content of the DEVICES and ACTION variables as well as possibly other variables provided by the kernel. It uses also the program `usbmodules` to find-out all about the device inserted.

  The specific 'agents' scripts are.

The USB system uses	`/etc/hotplug/usb.agent`
The PCMCIA uses	`/etc/hotplug/pci.agent` (via a bridge)
The Firewire(ieee1394) uses	`/etc/hotplug/ieee1394.agent`
The Network system uses	`/etc/hotplug/net.agent`

- **Files involved:**

`/lib/modules/*/modules.*map`	depmod output
`/proc/sys/kernel/hotplug`	specifies hotplug program path
`/sbin/hotplug`	hotplug program (default path name)
`/etc/hotplug/*`	hotplug files
`/etc/hotplug/`*`NAME`*`.agent`	hotplug subsystem-specific agents
`/etc/hotplug/`*`NAME*`	subsystem-specific files, for agents
`/etc/hotplug/`*`NAME/DRIVER`	driver setup scripts, invoked by agents
`/etc/hotplug/usb/`*`DRIVER`*`.usermap`	depmod data for user-mode drivers
`/etc/init.d/hotplug`	hotplug system service script used also to load and configure already plugged hot-plug devices at boot time.

USB Manager (`usbmgr`)

Is a daemon that will load the proper module according to 2 parameters given by the kernel: USB-Vendor-ID and USB-Device-ID

It uses the following configuration files:

`/etc/usbmgr/usbmgr.conf`	List of Vendor-ID/Device-ID and module names
`/etc/usbmgr/preload.conf`	List of modules to load when usbmgr starts.
`/etc/usbmgr/host`	List of module names od the USB controller: either `usb-ohci` or `usb-uhci`.

The `usbmgr` needs the folowing conditions:

- The kernel must be USB capable (`usbcore`)
- The USBDEVFS mus be supported
- The needed modules must be available.

- ACPI (Deutsch)
  Bezieht sich auf: SUSE LINUX 8.1 - 9.0

Anliegen

Sie möchten das Verhalten von ACPI beeinflussen. Seit **SuSE Linux 8.1** wird der neue ACPI-Code verwendet.

Wir haben uns für den neuen ACPI-Code entschieden, weil viele neuere Computer das Interrupt-Handling über ACPI steuern. Ebenso basiert das Powermanagement neuer Computer nicht mehr auf APM, sondern auf ACPI. Ein Nachteil ist, dass manche ältere Computer Probleme damit haben können aufgrund einer unvollständigen oder einer nicht stabilen ACPI-Implementation im BIOS. Weitere Hintergrundinformationen finden Sie hier:

http://www.suse.de/de/private/products/suse_linux/i386/acpi.html

Vorgehen

Es gibt ab **SuSE Linux 8.2** aus den oben genannten Gründen mehrere Kernelparameter, mit denen Sie den ACPI-Code entscheidend beeinflussen können.

- `acpi=off` - Dieser Parameter schaltet das komplette ACPI-System ab. Dies ist zum Beipiel sinnvoll, wenn Ihr Computer über gar keine ACPI-Unterstützung verfügt oder Sie den konkreten Verdacht haben, dass die ACPI-Implementierung Probleme bereitet.
- `acpi=oldboot` - Schaltet das ACPI-System fast komplett aus, und nur die Teile, die für das Booten nötig sind, werden verwendet.
- `acpi=force` - Schaltet ACPI ein, auch wenn Ihr Rechner ein BIOS von vor 2000 hat. Dieser Parameter überschreibt `acpi=off`. Dieser Parameter kann auch bei neueren Rechnern wirksam sein wenn Sie trotz `apm=off` keine ACPI unterstützung bekommen.
- `pci=noacpi` - Dieser Parameter schaltet das PCI IRQ-Routing vom neuen ACPI-System aus.

In **SuSE Linux 8.1** gibt es folgende Parameter:

- `acpi=off` - Dieser Parameter schaltet das komplette ACPI-System ab. Dies ist zum Beipiel sinnvoll, wenn Ihr Computer über gar keine ACPI-Unterstützung verfügt oder Sie den konkreten Verdacht haben, dass die ACPI-Implementierung Probleme bereitet.
- `acpi=oldboot` - Schaltet das ACPI-System fast komplett aus, und nur die Teile, die für das Booten nötig sind, werden verwendet.
- `pci=acpi` - Dieser Parameter schaltet das PCI IRQ-Routing vom neuem ACPI-System ein. Normalerweise wird aus Kompatibilitätsgründen das alte IRQ Routing benutzt.

`pci=noacpi` und `acpi=force` gibt es in **SuSE Linux 8.1** nicht. Der Update-Kernel, der für **SuSE Linux 8.1**, verfügbar ist verhällt sich genauso wie der Kernel von **SuSE Linux 8.2**.

Bei Mulitprozessor-Rechnern ist bei Verwendung von ACPI häufig auch APIC einzuschalten. Dies erreichen Sie mit dem Kernelparameter:

`apic`

Diese Kernelparameter können Sie schon vor der Installation aktivieren, indem Sie sie am Bootprompt dem Kernel übergeben. Dabei ist eine Englische Tastaturbelegung aktiv. Das = finden Sie zwischen der Backspace und der Fragezeichen Taste. Wenn Sie diese Parameter schon vor der Installation eintragen, werden diese Parameter automatisch in die Konfiguration des Bootloaders übernommen. Sie können die Parameter aber auch nachträglich in der Konfiguration des Bootloaders in YaST2 eintragen. Nähere Informationen zur Konfiguration des Bootloaders mit YaST2 finden Sie im Handbuch. Weiterführende Informationen zu ACPI finden Sie auf der Projekt-Homepage der ACPI-Entwickler:

http://acpi.sf.net

Siehe auch:

- Bootoptionen dauerhaft eintragen