Files and File Systems

Marek Kozłowski

Faculty of Mathematics and Information Sciences
Warsaw University of Technology

Outline of the Lecture

Outline of the lecture

File Information Structure – *i-node*

- *i-node* is a structure that stores all information about a file except its name and data, that is:
  - type,
  - size,
  - owner and group,
  - permissions,
  - time-stamps: last i-node change and file modification (*ctime* and *mtime*) as well as last access time (*atime*),
  - device ID and pointers to disk blocks,
  - link count (number of hard-linked files, normally: 1).

Getting File Information – *stat*

- `stat` displays all information on a file stored in its i-node structure.

Exemplary *stat* output

```
$ stat somefile
  File: ‘somefile’
  Size: 588         Blocks: 8          IO Block: 4096   regular file
Device: 802h/2050d  Inode: 5494628  Links: 1
Access: (0644/-rw-r--r--) Uid: ( 1000/ smithj) Gid: ( 100/ users)
Access: 2011-03-09 23:19:28.000000000 +0100
Modify: 2009-12-30 14:25:28.000000000 +0100
Change: 2010-09-24 09:54:51.000000000 +0200
 Birth: -
```

- File name being displayed is a command argument not an *i-node* field!

File Information Structure – *i-node* (2)

- Uniform structures used for regular files, symbolic links, directories and all other file system objects (see next slides).
- Created and stored in memory when files systems are made available (mounted).
- Independent of file system type and implementation.
- Act as logical interfaces to mass storage.
- Store numerical values only: file ownership, group and permissions are represented as numbers.

File Names and Directories

- I-nodes don't contain file names.
- File system objects are identified by so-called i-node numbers (*i-numbers*).
- *Directories* are file system objects which store lists of pairs: *(name, i-number)*.
- Numerous names can be associated with a single i-number. We say such names are *hard linked*.
- The number of hard links is stored in an i-node. If this value reaches 0 the file is to be deleted.

File Names and Directories – Example

- `/etc` directory is a list that allows finding files by names:

file name	i-node number
group	4204828
nnswitch.conf	4194327
passwd	4204830
pam.d	4194367
shadow	4204831
shells	4194321

- This information may be obtained by executing `ls -i` for a directory.

Displaying Hard Linked Program Names – Example

Displaying hard linked program names:

```
$ ls -li /usr/bin/ | awk '$3>1' | sort
```

- Short explanation:
  - `/usr/bin` directory contains program files,
  - the 3^{rd} field (without `-i` – the 2^{nd} one) for files and symlinks denotes the number of hard linked names,
  - a short inline `awk` script displays the lines for which the 3^{rd} field is greater than 1.

Time Stamps

- *Access*: most administrators disable updating this field for increasing system performance.
- *Modify* refers to file contents changes.
- *Change* refers to i-node changes, for example: changing file group or permissions.
- Typical file systems used for Linux don't store *Birth/Creation* so this field is usually empty.

File Types

- The following file types are commonly used:
  - `-` – regular file,
  - `d` – directory (cannot be directly edited)
  - `l` – symbolic link,
  - `b` – block device (allows buffered access to a device),
  - `c` – character device (allows non-buffered access to a device),
  - `p` – named pipe.
  - `s` – socket.
- The first three ones have already been discussed. The last two ones are out of the scope of this course.
- Numerous commands require a filename as an argument and work the same for files of any type.

Block Devices

- Allow random buffered access to devices.
- Usually represent mass storage devices.
- Linux (Linux! not Unix in general!) uses the following name conventions:
  - `/dev/sda` , `/dev/sdb` , . . . – disks (SATA, PATA, USB, SCSI, hardware RAID volumes),
  - `/dev/sda1` , `/dev/sda2` , `/dev/sda3` , `/dev/sda4` – physical partitions on the first device,
  - `/dev/sda5` , `/dev/sda6` , . . . – logical disks in an extended partition on the first device,
  - `/dev/md0` , `/dev/md1` , . . . – software RAID volumes (see `mdadm`),
  - `/dev/sr0` – optical disc (usually symlinked as `/dev/cdrom`).

Listing Block Devices

Using regular expressions:

```
$ ls -l /dev | grep -P '^b'
```

Using a dedicated command:

```
$ lsblk
```

Character Devices

- Allow sequential unbuffered access to devices.
- Represent physical devices, ttys, as well as some system byte generators:
  - `/dev/null` – accepts and discards all input; produces no output,
  - `/dev/random` and `/dev/urandom` – generate a pseudo-random stream (`random` – better randomness, `urandom` – faster),
  - `/dev/zero` – produces a continuous stream of 0 bytes.

✍ *Writing to a tty is just writing to a file (root privileges are requisite):*

Sending message to a tty

```
# echo whatever > /dev/tty8
```

Outline of the lecture

Hidden Files

- There are no special *hidden* nor *system* attributes.
- File names that begin with '`.`' (dot) are considered hidden and numerous commands omit them by default. Note that those files don't match the '`*`' wildcard – a special wildcard '`.*`' must be used to indicate them.
- Sometimes empty files named `.keep` or similarly are placed in empty directories to protect them from accidental deletion.

✍ *By default `rm` skips hidden files:*

Deleting hidden files in working directory

```
$ rm *    # delete all files except hidden ones
$ rm * .* # delete all files
```

Calculating Disk Usage for Directories – *du*

- `ls -l` or `stat` display the size of a directory as a file instead of the directory contents size.
- The `du` command displays a directory contents size recursively. With the `-s` switch it prints only the summarized value. With `-h` (human readable) it displays size in bytes, kilobytes, megabytes, gigabytes etc. as the need arise in order to reduce the number of digits to four or fewer.

✍ *Note that `du` includes hidden files and directories:*

Displaying disk usage with *du*

```
$ du -sh ./somedir
1.5G  ./somedir
$ du -h ./somedir
1G    ./somedir/somesubdir1
500M  ./somedir/.somehiddensubdir2
1.5G  ./somedir
```

Advanced File Management Utilities – *lsof*

- Lists all files that are currently open. For each file it prints which process (process ID, owner and group) is using it.
- `lsof` *`somefile`* displays information on a selected file.
- Helps solve any *File already in use* lock problems.

Advanced File Management Utilities – *dd*

dd syntax:

```
# dd if=input_file of=output_file [count=how_many] [bs=block_size]
```

- `dd` copies `count` blocks of the `bs` size from the `if` to the `of` file.
- The command is commonly used with block or char device files.
- If the `bs` is not specified some default value is used (usually 512B). Greater values increase the speed.
- If the `count` parameter is skipped the command copies bytes until EOF is reached.

Using *dd* – Examples

Saving MBR to a file:

```
# dd if=/dev/sda of=/root/mbr.bak count=1 bs=512
```

Overwriting a disk with zeros/random bytes:

```
# dd if=/dev/zero of=/dev/sdb
# dd if=/dev/urandom of=/dev/sdb
```

Initializing a swap file:

```
# dd if=/dev/zero of=/swapfile bs=1024 count=524288
```

Creating a bootable usb drive from a liveDVD iso image:

```
# dd if=liveDVD.iso of=/dev/sdb
```

Outline of the lecture

Unix File Systems

- While various extraneous file systems (including FAT and NTFS) are supported by modern Linux and *BSD systems the most common file systems for Unix include:
  - *ext*, *ext2*, *ext3* and *ext4* (*extended file system*) – defaults for most modern Linux distributions,
  - *UFS* (*Unix File System*) or *FFS* (*Berkeley Fast File System*) – default for *BSD and Solaris (although implementation differences do exist); many vendors adapted it to their needs,
  - *AdvFS* (*Tru64 UNIX Advanced File System*) – Tru64,
  - *JFS* (*Journaled File System*) – designed for IBM AIX,
  - *ReiserFS* (less maintained, read tabloids for explanation: why?),
  - *VxFS* (*Veritas File System*) – developed for HP-UX,
  - *XFS* – originally for IRIX,
  - *btrfs* – a new, promising FS for Linux (not recommended for production environments yet),
  - *HAMMER2* – one of the key features of DragonFly BSD.

Notes on Unix File Systems

- File system choice is a question of performance, stability, security, fault tolerance, journaling and extra features.
- File system types are indistinguishable to users and programmers.
- The same i-node structures are used no matter what file system type is being used.

Default Linux File Systems

- Most Linux distributions assume *ext4* as the default general purpose Linux file system.
- Its predecessor: *ext3/ext2* offers substantially worse performance but some *rescue and recovery* utilities may provide better support for it.
- For dedicated use:
  - few huge files,
  - lots of very small files,
  - FS history, audit or special fault tolerance requirements,
  - frequent reads or frequent writes,
  - etc

  choosing other file systems should be considered.

Linux Partitions

- Contrary to FreeBSD or Solaris Linux creates one file system per partition.
- Reminder: Linux uses the following name conventions (for details consult: `man 4 sd`):
  - `/dev/sda` , `/dev/sdb` , . . . – disks (SATA, PATA, USB, SCSI, hardware RAID volumes),
  - `/dev/sda1` , `/dev/sda2` , `/dev/sda3` , `/dev/sda4` – physical partitions on the first device,
  - `/dev/sda5` , `/dev/sda6` , . . . – logical disks in an extended partition on the first device,
  - `/dev/md0` , `/dev/md1` , . . . – software RAID volumes (see `mdadm`).
- `fdisk` is the basic utility for partition management (for partitioning the first disk run as the root: `fdisk /dev/sda`).

Linux *fdisk*

- Requires root privileges.
- Any changes have to be written to a disk. `fdisk` doesn't save any changes on exit if not instructed to do so.
- Adding, deleting or modifying partitions doesn't overwrite their contents.
- Be careful with setting partition IDs. For Linux you should select `83` (hex).

Notes on BSD Partitions

- BSD terminology may be confusing for Windows or Linux users:
  - BSD term for an fdisk partition is a *slice*,
  - slices contain logical disks called *partitions*.
- File systems are created inside partitions.
- Block device names for partitions contain a disk number (sometimes preceded by a controller and bus numbers), a slice number and a partition identifier.

Creating File Systems

Creating a file system

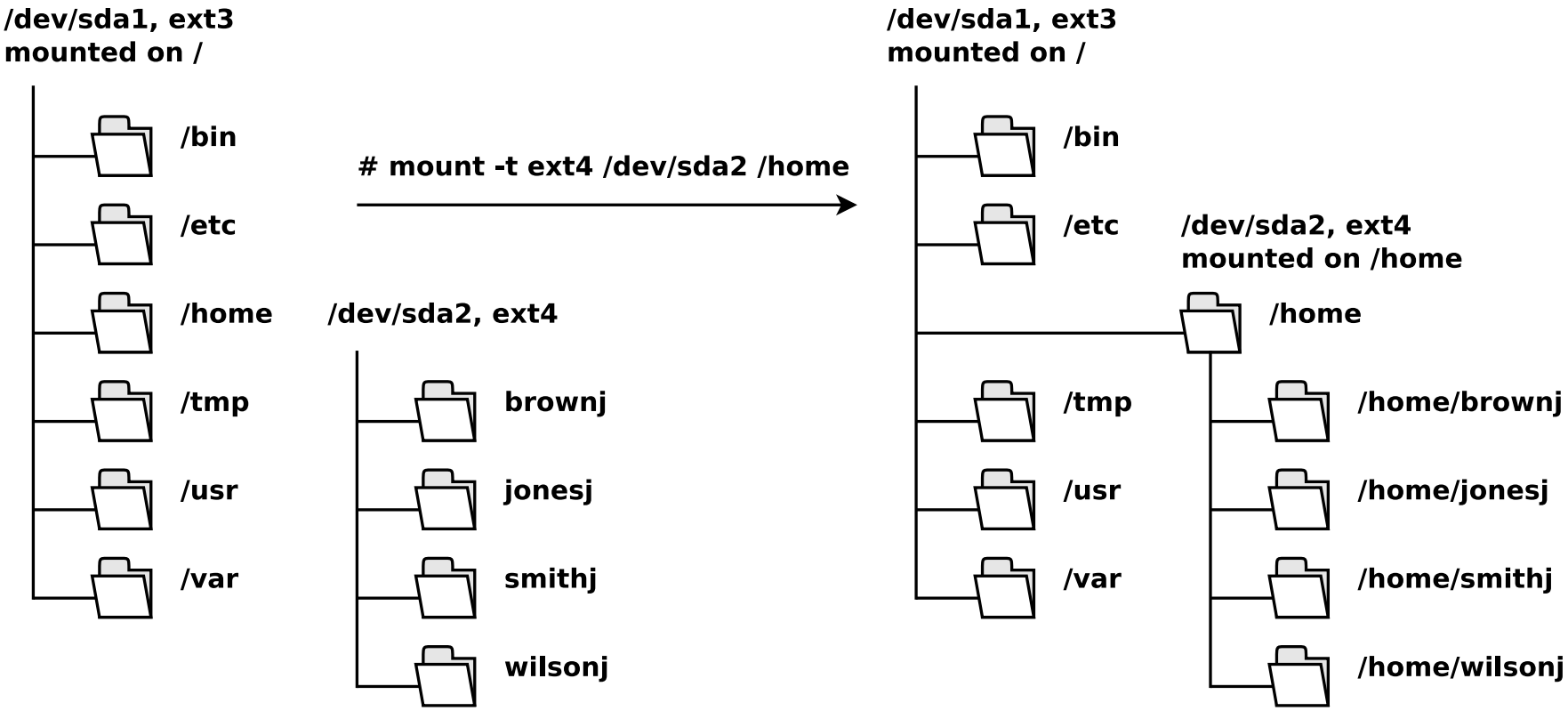
```
# mkfs [-t type] device
```

- *device* is a partition or RAID volume name.
- *type* specifies a file system type.
- In fact `mkfs` acts as a front-end to various file system builders, for example:
  - `mke2fs` – *ext2* (default), *ext3*, *ext4*,
  - `mkreiserfs` – *ReiserFS*.
- Instead of `mkfs -t type` you may usually use an alias: `mkfs.type`, for example:
  - `mkfs.ext2` , `mkfs.ext3` , `mkfs.ext4` ,
  - `mkfs.reiserfs` .

Mounting File System

- In order to access a file system in has to be *mounted*.
- Only one file system can be mounted as a top-level one (/) – so-called *root file system*.
- By mounting other file systems their top directories are identified with some directories that exist on currently mounted file systems. They are seen as branches of some tree-like hierarchical structure.
- If a directory used as a *mount point* in not empty its contents gets inaccessible.
- Parent and child file systems can be of different types.

Mounting – Example



Mounting Pseudo File System

- In addition to normal file systems numerous address spaces or data sets can be mounted and then visible *as if* they were file systems.
- Thanks to it Unix allows accessing almost all resources including kernel and processes (`procfs`), hardware (`devfs`) etc. via standard file system system calls and utilities.
- Usually the directories: `/dev` , `/proc` and `/sys` contain such pseudo file systems (address spaces) mounted.

Mounting File Systems – *mount*

Typical *mount* syntax

```
# mount [-t type] [-o options] device mountpoint
```

- File system type:
  - For most file systems `-t` may be omitted (auto-detection).
  - There are two Linux drivers for NTS: `-t ntfs` (default) and `-t ntfs-3g` (recommended).
  - `-t iso9660` allows mounting ISO images (read only).

Mounting File Systems – *mount* (2)

- Common options include:
  - `-o acl` or `-o acls` – enable ACLs,
  - `-o noatime` – don't update the file access time when reading files (increases performance),
  - `-o nodev` – ignore device files,
  - `-o noexec` – prevent execution of any programs from the file system,
  - `-o nosuid` – do not allow setuid or setgid bits to take effect,
  - `-o ro` – mount read-only.
- Options for currently mounted file system can be changed with `-o remount` .
- A list of options can be specified at a time, for example: `-o acl,noatime,nodev,nosuid` .

Dismounting File Systems - *umount*

umount syntax

```
# umount device|mountpoint
```

- Some file systems (`ext3`, `ext4`, `tmpfs`, etc) can be mounted several times on different mount points at the same time.
- Specifying a mount point instead of a device prevents ambiguity.

Mount Tables – */etc/fstab*

- `/etc/fstab` lists file systems to be mounted by `mount -a` (called at boot time).

/etc/fstab syntax

```
device  mountpoint  type  options  dump/pass
```

- The fields meaning is as follows:
  - *device* – block special device or remote file system to be mounted,
  - *mountpoint* – mount point (`none` for swap),
  - *type* – file system type or `auto`,
  - *options* – coma-separated list of mount options; `noauto` prevents mounting by `mount -a` , `user` allows mounting by regular users,
  - *dump/pass* – flags for `dump` and `fsck` .
- WARNING: Modern desktop-oriented Linux systems automatically mount pluggable devices (CD, pendrive etc.) via *udev*. `/etc/fstab` entries for them may lead to unpredictable results!

Mount Tables – */etc/mtab*

- `/etc/mtab` stores information about currently mounted file systems.
- It is typically a symlink to some file in `/proc` (usually: `/proc/mounts`) or `/sys` pseudo file systems.

  ✍ *The following two commands should display the same information:*

Listing mounted file systems and their mount options

```
$ mount
$ cat /etc/mtab
```

Disk Space Usage – *df*

- `df` displays disk space usage for all mounted file systems. By default it prints data in blocks of 1024 or 512 bytes (depending on the system) so the `-h` (human readable) switch is commonly used. With `-h` it displays size in bytes, kilobytes, megabytes, gigabytes etc. as the need arise in order to reduce the number of digits to four or fewer (`df` and `du` share the same data size format).

File System Repair – *fsck*

- `fsck` (*File System ChecK*) checks and repairs file systems.
- `fsck`, similarly to `mkfs`, acts just as a front end to file system dedicated utilities.
- File systems are usually specified as block devices (for example: `fsck /dev/sda2`).
- Many systems are configured to run `fsck` on system startup (file system mount) if a file system was not checked for some period of time or some number of mounts.
- WARNING: Running `fsck` on live or mounted file systems is highly not recommended and may result in serious data corruption! Always unmount file systems before any repair attempts!

Outline of the lecture

FHS Key Concepts

- Separation of the OS and userland - separate directories for critical system files, system utilities and userland (applications, daemons).
- Directories containing userland files and user data can be placed on different file systems and dismounted shall the need arise.
- Read-write and read-only files in separate directories allow optimization by selecting different file system types for them.
- Simplified privilege management by separate directories for: binaries (programs), configuration files, libraries, data and logs.

Standard Unix Directories

- The following directories do exist in most Unix-like systems:
  - `/bin` – *binaries* (a Unix term for *programs*),
  - `/boot` – *bootstrap* files,
  - `/dev` – *devices*,
  - `/etc` – *editable text configuration*,
  - `/home` – *home* directories (of regular users),
  - `/lib` – (shared) *libraries*,
  - `/mnt` – (temporary) *mount* points,
  - `/opt` – *optional* (add-on) software,
  - `/proc` – *processes* and kernel information,
  - `/root` – *root* home,
  - `/sbin` – *superuser*, *special* or *system binaries*,
  - `/sys` – *system* similar to `/proc`,
  - `/tmp` – *temporary* files,
  - `/usr` – non system critical *unix system resources*,
  - `/var` – *variable* data.

Kernel, Modules, Libraries – */boot* and */lib*

- `/boot` contains the *kernel*, optional *boot manager* configuration files (for Linux: *LILO* or *GRUB*) and other files necessary for system bootstrap.
- `/boot` often resides on a separate partition. It's smart to place it on a disk which doesn't require any non-standard drivers.
- `/boot` access is required only during system boot-up. It is not necessary to keep it mounted for a running system.
- `/lib` (in newer systems two other directories: `/lib32` and `/lib64` can exist) contains kernel modules not critical for system startup as well as shared libraries for the binaries in `/bin` and `/sbin` .

System-wide Configuration – */etc*

- `/etc` stores system-wide configuration files. Some systems, mostly BSD Unices, can use files in `/usr/local/etc/` for less important userland packages.
- For many settings respective user configuration files (preceded with '`.`' and located in home directories) take precedence over system-wide `/etc` configuration files.
- Most configuration files are user-friendly:
  - use ASCII text format,
  - specify each entry/setting in a separate line,
  - use `#` or `;` (rare, PHP) to start a comment,
  - contain numerous commented out examples,
  - have man pages in the 5^{th} section (use `man 5`),
  - provide syntax highlighting files for *vim* (*gvim*).

Binaries – */bin* and */sbin*

- `/bin` and `/sbin` contain critical system utilities (for example: `bash`, `ls`, `mount`, ...).
- `/sbin` in normally not in `$PATH` of regular users and contain utilities for the root.
- Which tools are placed in `/bin` and which in `/sbin` is not strictly defined and can vary. For example: one systems place `mount` in `/bin` while others in `/sbin` .

Home Directories – */home* and */root*

- `/home` is a typical location for user home directories. It usually resides on a separate partition for the following reasons:
  - can be unmounted during system repair/maintenance,
  - a file system optimized for frequent parallel reads/writes or other requirements can be selected,
  - special mount options can be provided (`acl`, `nosuid`, ...),
  - quotas for disk space usage can be implemented.
- Cause accessibility of root home directory is critical it is stored in a separate directory: `/root` on the main partition.

Non Critical Resources – */usr*

- Initially *USeRs*, nowadays the expansion *Unix System Resources* is considered more accurate.
- Contains files that are non-critical for the system itself: user applications, icons, pixmaps, fonts and other desktop-related resources, header and source files etc.
- Read-only access to `/usr` is required except system upgrades.
- `/usr` on servers occupies a separate partition mounted with `ro` option (read-only access).

/usr Subdirectories

- Selected /usr subdirectories:
  - /usr/bin – user applications; for example: `firefox`, `thunderbird`, `pdflatex`, `libreoffice`, `epdfview`, `gimp`, ...,
  - /usr/include – standard and application C header files,
  - /usr/lib – standard and application libraries,
  - /usr/local – secondary applications directory hierarchy,
  - /usr/sbin – *daemons* (services) and non-critical root binaries,
  - /usr/share – architecture-independent and desktop-related data:
    - application icons (/usr/share/pixmaps),
    - desktop menu entries (/usr/share/app or /usr/share/applications),
    - desktop theme icons (/usr/share/icons),
    - GUI fonts (/usr/share/fonts),
    - locale-related files (/usr/share/consolefonts, /usr/share/keymaps, /usr/share/zoneinfo, ...),
    - man pages (/usr/share/man),
  - /usr/src – kernel sources.

Variables: Logs, Databases, Queues – */var*

- `/var` is a directory for variable files – logs, databases, DNS zone files, mail and print queues etc.
- For overfull prevention `/var` should reside on a separate partition.
- Typical subdirectories:
  - `/var/bind` or `/var/named` – DNS (BIND) zone description files,
  - `/var/cache` – non critical application cache (wiping out is generally safe),
  - `/var/lib` – system (package) databases, RDBM's databases,
  - `/var/log` – logs,
  - `/var/run` – runtime system information, process IDs of running services (may be a symlink to `/run`),
  - `/var/spool` – queues,
  - `/var/tmp` – usually same directory/filesystem as `/tmp`,
  - `/var/www` – main WWW document directory,

System Interface – */dev* and */proc* Pseudo-filesystems

- `/dev` is a system managed (*devfs* or *udev*) pseudo-filesystem that contains block and character device files which correspond with supported hardware as well as pseudo-device files.
- `/proc` is a *procfs* pseudo-filesystem which acts as an interface to system and processes. `/proc` files cannot be edited; `cat` and `echo` *somedata* `>` should be used instead (see the examples on the next slide).
- Some systems use `/sys` in addition to or instead of `/proc` .

Using */proc* – Examples

Setting the CPU fan speed on IBM ThinkPad T4x notebook

```
# cat /proc/acpi/ibm/fan
status:  enabled
speed:   3577
level:   auto
commands: level <level> (<level> is 0-7, auto, disengaged, full-speed)
commands: enable, disable
commands: watchdog <timeout> (<timeout> is 0 (off), 1-120 (seconds))
# echo "level 0" > /proc/acpi/ibm/fan # stop the fan
```

Setting LCD brightness on TBM ThinkPad T4x notebook

```
# cat /proc/acpi/ibm/brightness
level:    6
commands: up, down
commands: level <level> (<level> is 0-7)
# echo up > /proc/acpi/ibm/brightness
```

Auxiliary Directories – */mnt*, */opt* and */tmp*

- `/mnt` is a default mount point for temporary manual mounts. In some systems the subdirectories: `/mnt/floppy`, `/mnt/cdrom` etc. for various media can be pre-defined.
- Newer systems use `/media` (or `/run/media`) and created on-the-fly subdirectories instead of `/mnt` for media (CD, DVD, USB) auto-mounting.
- `/opt` is used for external pre-packaged software like *Mathematica*, *Matlab* etc. Software that doesn't strictly follow the `bin`, `etc`, ... scheme should be placed in `/opt` as well.
- `/tmp` contains temporary data. The permissions are typically set to `rwxrwxrwt` . `/tmp` can be safely wiped out during system restart.

Outline of the lecture

Partitioning Goals

- The reason for multi-partition systems:
  - overfull prevention,
  - distinct mount options for better security (`ro`, `nosuid`, `noexec`, ...) and usability (`acl` etc),
  - quota on selected file systems only,
  - simplified troubleshooting (selective and/or parallel file system checks and repairs),
  - performance optimization (various disk types, file system types),
  - different fault tolerance models (RAID) and backup strategy.
- Creating multiple file systems for desktop machines is often being questioned.

Recommended Partitioning Scheme

- A commonly used partitioning follows the scheme:
  - `/boot` – not mounted during system runtime,
  - `/` – all system critical files,
  - `/home` – mounted with `nosuid`, `acl`; an external disk array is often used,
  - `/usr` – read-only (`ro`) except system upgrades,
  - `/var` – file system optimized for frequent reads/writes; some administrator mount it with `noexec`.
- Many administrators allocate a separate partition for `/tmp` and `/var/tmp` .

Outline of the lecture

Linux systemd

- Systemd is a binary sub-system for managing Linux startup and service control.
- While it is an extremely controversial solution (see: https://nosystemd.org/ , https://suckless.org/sucks/systemd/ and others), closer in concept to Windows rather then Unix philosophy, numerous Linux distributions implement it.
- CRUX, Gentoo and Slackware are the main Linux distros which reject adopting systemd.

/*usr*-centric Systemd

- Most of the `systemd` sub-system resides in `/usr/lib` and therefore access to this directory is required at boot time.
- For `systemd` Linux placing `/usr` on a separate partition is highly not recommended!
- Due to merging `/` and `/usr` partitions several directory structure changes were introduced:
  - `/lib` is merged to `/usr/lib` .
  - `/bin`, `/sbin` and `/usr/sbin` are all merged to `/usr/bin` .
  - Symlinks are defined for backward compatibility.

Other File System changes

- `/var/run` has been renamed to `/run`. A symlink is defined for backward compatibility.
- Removable media are auto-mounted under `/run/media/`whoami`/` .
- RAM-based filesystems are auto-mounted under `/tmp` and `/var/tmp` .
- The number of auto-mounted pseudo-filesystems dramatically increased.