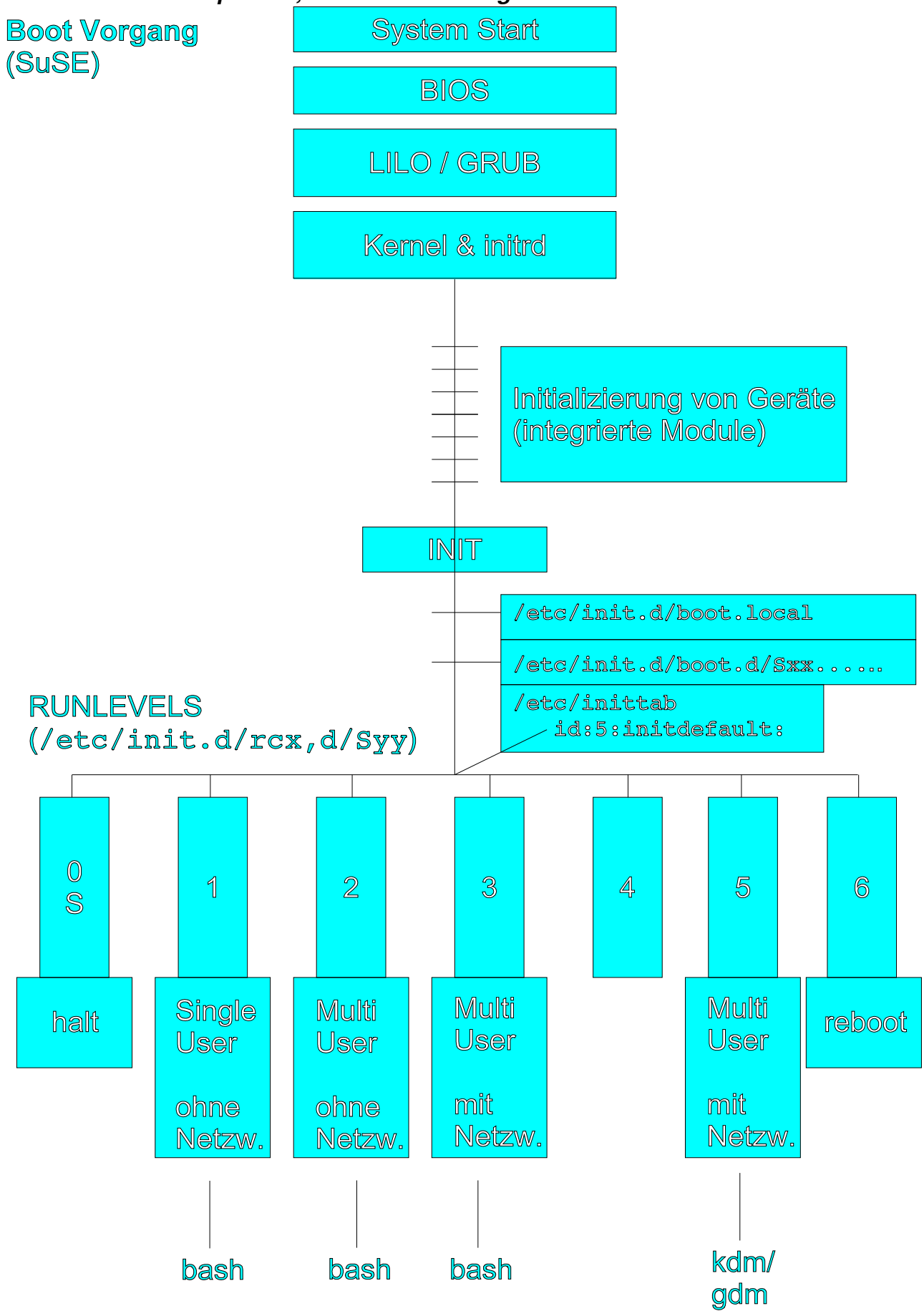
51 - *Linux Boot sequence , Runlevels and Login*



- **From Booting .....to Bash prompt.**

- **Bios** load LILO from MBR (first hard disk activated for booting)

- **LILO**
  1. Initialize the screen (ASCII mode) and keyboard
  2. Loads Kernel from hard disk (`/boot` or `/` directories)
  3. Load the file `initrd`
  4. Gives control to the Kernel

- **Kernel**
  1. Detects and initialize some system devices (mouse,cpu,clock,pci bus, etc)
  2. Detects and initializes Block devices (IDE,SCSI,floppy, cdrom, etc.)
  3. Mount the `initrd` file system
     1. Start `linuxrc` from `initrd`
        1. Loads kernel modules to prepare to mount root `(/)` file system.
        2. Manages RAID and LVM Setups(optional)
        3. Mounts network(NFS) root `(/)` file system(optional)
  4. Unmounts `initrd` and mounts root `(/)` file system.
  5. Starts `/sbin/init`

- `init`
  1. Reads `/etc/inittab` and runs all `boot` and `bootwait` actions
  2. Loads the `blogd` daemon:
     It logs all the boot scripts output to `/var/log/boot.msg`
  3. Runs `/etc/init.d/boot` script
     - Mounts(creates) the `/proc` device
     - Activates the swap-devices in `/etc/fstab`
     - Starts the file system checking with `fsck`
     - Mounts the local file system (according to `/etc/fstab`)
     - Setting up local parameters:
       - `/etc/ld.so.cache`, Time zone,Loopback device,Hostname,CMOS Clock etc.
     - Executes scripts (if present) in `/etc/init.d/boot.d/`
     - Executes the script (if present) `/etc/init.d/boot.local`
  4. Reads `/etc/inittab (id:`*x*`:initdefault:)` and starts `/etc/init.d/rc` script:
     1. S*yy*... scripts(S-links) from `/etc/init.d/rc`*x*`.d` will be run in sequence order of the *yy*.
        x =the runlevel *yy*=execution sequence
     2. As these scripts are run, their results are shown on screen (`OK` or `done`)

        Notes: - Each time a child terminates, `init` records the fact and the reason it died in `/var/run/utmp` and `/var/log/wtmp`, if they exist.
        - Each time the runlevel is changed from 1 or S to another, `rc` runs also `/etc/init.d/boot.setup`: (keyboard and virtual consoles setup)
  5. Closes Log file `/var/log/boot.msg` and terminates `blogd`

6. Starts `mingetty` (`tty` controller)

- `mingetty`
  1. Displays Banner: the content of `/etc/issue` or `/etc/issue.net`
  2. Prompts for a `login` name and waits for the *username*
  3. Starts the `/bin/login` command with the *username* as argument.

- `login`
  1. Reads `/etc/login.defs` and `/etc/pam.d/login`
  2. Authenticates the user using PAM,
     PAM uses `/etc/passwd` and `/etc/shadow` or another method.
  3. Sets `HOME, PATH, SHELL, TERM, MAIL, & LOGNAME` env. var .
  4. Read `/etc/passwd` to know which shell to start.(eg. `/bin/bash`)

- `bash`
  1. Displays the content of `/etc/motd`
  2. Reads `/etc/profile`
     - Reads `/etc/profile.local` (SuSE only , if it exists)
     - Reads `/etc/bash.bashrc`(SuSE only , if it exists)
  3. Reads `~/.bash_profile` if it exists.
     if `~/.bash_profile` doesn't exist then
     Reads `~/.bash_login`
     if `~/.bash_login` doesn't exist then
     Reads `~/.profile`
     which reads `~/.alias` if it exists(SuSE)
  4. Presents **prompt** and waits for user commands.

- **Linux Run Levels**

  - **Shutting down the system**
    - `halt` - Uses `shutdown -h` to shut down
    - `halt -f` - Forces a halt immediately without calling `shutdown`
    - `reboot` - Uses `shutdown -r` to reboot the system
    - `reboot -f` - Forces a reboot immediately without calling `shutdown`
    - `poweroff` - Uses `halt -p` (powers off the PC)
    - `shutdown -h` - Uses `init 0` to shut down
    - `shutdown -r` - Uses `init 6` to reboot
    - `shutdown -f` - Reboot fast.
      Creates the file `/fastboot` to bypass `fsck` on reboot
    - `shutdown -F` - Force full `fsck` filesystem check on reboot.
      Creates the file `/forcefsck` to force `fsck` on reboot
    - `shutdown -c` - Cancel an already launched shutdown command
    - `shutdown -a` - Checks if one of the users in `/etc/shutdown.allow` is logged in in one of the 6 virtual consoles and proceed if so. `shutdown.allow` format is one name per line. Maximum 32 users.

- `shutdown -t 10 5m "The system is shutting down in 5 Min."`
  The system send the message "The system is ....." to logged in users
  - wait 5 minutes(`5m`)
  - then send the `-TERM` signal to all processes
  - waits 10 seconds(`-t 10`)
  - then send the `-KILL` signal to processes
  - then call `init 1`(this is the default if `-r` or `-h` is not used)

- **Run levels description and purpose**
  Autostart annd changing of processes, services or setups (eg. Daemons)

- **What happens when changing run levels**
  `init` starts `/etc/init.d/rc` *x* (*x* = new run level)
  If runlevel is changed from 1(or S) to another, the script
  `/etc/init.d/boot.setup` is also run before changing run level.

- **Yast run level editor**
  SuSE graphic interface for managing the run levels' properties.

- **Files involved**
  - `/sbin/init` Father of all processes
  - `/etc/inittab` Configuration file of `init`
  - `/etc/init.d/rc` Main runlevel control script
  - `/etc/init.d/yyyyyyy` Run levels Scripts
  - `/etc/init.d/rcx.d/Sxxyyyyyyy` Start-Links to Run levels Scripts
  - `/etc/init.d/rcx.d/Kxxyyyyyyy` Stop-Links to Run levels Scripts
  - `/usr/sbin/mingetty` `tty` controller that starts `login`
    - `/etc/issue` and `/etc/issue.net` banner shown before `login` prompt. `issue.net` is for telnet sessions
  - `/dev/console` Default system console
  - `/var/run/utmp` (binary log file of `init` children process born and died)
  - `/var/log/wtmp` (binary log file of `init` children process born and died and why as well as all logins, logouts and shutdowns)
  - `/dev/initctl` (`init` control fifo)

- **The purpose of `/etc/init.d/rcx.d` directories for default run level**
  - Scripts running with start and stop parameters

- **To provoke `init` to read again its configuration file `/etc/inittab`:**
  `init q` or `telinit Q` or `telinit q`

- **To create/delete run level services symbolic links automatically:**
  (SuSE 8.0 and higher only)
  `chkconfig -s servicename on|off`
  or
  `insserv servicename` (installs links for boot start/stop of service)
  `insserv -r servicename` (deletes links for boot start/stop of service)
  (*servicename*=script filename in `/etc/init.d/` directory)

• The `/etc/inittab` file format:

Each entry in `/etc/inittab` is made of 4 fields:

1) ***id***

`id` is a unique sequence of 1-4 characters which identifies an entry in inittab

(for versions of `sysvinit` compiled with libraries < 5.2.18 or `a.out` libraries the limit is 2 characters).

Note: For `getty`s or other login processes, the `id` field should be the `tty` suffix of the corresponding `tty`, e.g. `1` for `tty1`. Otherwise, the login accounting might not work correctly.

2) ***runlevels***

lists the runlevels for which the specified action should be taken.
The runlevels field may contain multiple characters for different runlevels.
For example:
`123` specifies that the process should be started in runlevels 1, 2, and 3.

The runlevels for `ondemand` entries may contain an `A, B,` or `C.`

The runlevels field of sysinit, boot, and bootwait entries are ignored.

When the system runlevel is changed, any running processes that are not specified for the new runlevel are killed, first with SIGTERM, then with SIGKILL.

3) ***action***

action describes which action should be taken.(see list of actions below)

4) ***process***

Specifies the process to be executed. If the process field starts with a `+' character, init will not do `utmp` and `wtmp` accounting for that process.

This is needed for gettys that insist on doing their own `utmp/wtmp` house keeping. This is also a historic bug.

Actions (field 3 in `/etc/inittab`)

`respawn` The process will be restarted whenever it terminates (e.g. getty).

`wait` The process will be started once when the specified runlevel is entered and init will wait for its termination.

`once` The process will be executed once when the specified runlevel is entered.

`boot` The process will be executed during system boot. The runlevels field is ignored.

`bootwait` The process will be executed during system boot, while init waits for its termination (e.g. /etc/rc). The runlevels field is ignored.

`off` This does nothing.

`ondemand` A process marked with an ondemand runlevel will be executed whenever the specified ondemand runlevel is called. However, no runlevel change will occur (ondemand runlevels are `a', `b', and `c').

`initdefault` An initdefault entry specifies the runlevel which should be entered after system boot. If none exists, init will ask for a runlevel on the console. The process field is ignored.

`sysinit` The process will be executed during system boot. It will be executed before any boot or bootwait entries. The runlevels field is ignored.

`powerwait` The process will be executed when the power goes down. Init is usually informed about this by a process talking to a UPS connected to the computer. Init will wait for the process to finish before continuing.

`powerfail` As for powerwait, except that init does not wait for the process's completion.

`powerokwait` This process will be executed as soon as init is informormed that the power has been restored.

`powerfailnow` This process will be executed when init is told that the battery of the external UPS is almost empty and the power is failing (provided that the external UPS and the monitoring process are able to detect this condition).

`resume` This process will be executed when init is told by the kernel that Software Suspend has resumed the machine. This way you may specify userland programs what can restore hardware states the kernel cannot (for example svgatextmode and hdparm).

`ctrlaltdel` The process will be executed when init receives the SIGINT signal. This means that someone on the system console has pressed the CTRL-ALT-DEL key combination. Typically one wants to execute some sort of shutdown either to get into single-user level or to reboot the machine.

`kbrequest` The process will be executed when init receives a signal from the keyboard handler that a special key combination was pressed on the console keyboard. The documentation for this function is not complete yet; more documentation can be found in the kbd-x.xx packages (most recent was kbd-0.94 at the time of this writing). Basically you want to map some keyboard combination to the "KeyboardSignal" action. For example, to map Alt-Uparrow for this purpose use the following in your keymaps file: `alt keycode 103 = KeyboardSignal`

Ausschnitt aus der Datei `/etc/inittab`:

```
id:3:initdefault:
si:I:bootwait:/sbin/init.d/boot
# runlevel 0 is halt
# runlevel S is single-user
# runlevel 1 is single-user
# runlevel 2 is multi-user without network
# runlevel 3 is multi-user with network
# runlevel 5 is multi-user with network and xdm
# runlevel 6 is reboot
l0:0:wait:/sbin/init.d/rc 0
l1:1:wait:/sbin/init.d/rc 1
l2:2:wait:/sbin/init.d/rc 2

ca::ctrlaltdel:/sbin/shutdown -r -t 4 now

1:123:respawn:/sbin/mingetty --noclear tty1
2:123:respawn:/sbin/mingetty tty2
```

Modus	***Bedeutung***
`initdefault`	Legt fest welche Arbeitsstufe beim Systemstart angesteuert wird
`bootwait`	Wird nur beim Systemstart ausgeführt. Es wird bis zum Ende der Ausführung gewartet (synchron)
`wait`	Es wird bis zum Ende der Ausführung gewartet (synchron)
`ctraltdel`	Bestimmt was mit dieser Tastenkombination passieren soll
`respawn`	Wenn nicht schon existent, wird ein Kindprozeß erzeugt (asynchron). Sobald er beendet ist, wird ein neuer gestartet (spawned)
`boot`	Erste process gestarted wann `/etc/inittab` ist gelesen
`powerfail`	Was passierte wann Strum abgebrochen ist
`sysinit`	The process will be executed during system boot. It will be executed before any boot or bootwait entries. The runlevels field is ignored.
`ondemand`	A process marked with an `ondemand` runlevel will be executed whenever the specified `ondemand` runlevel is called. However, no runlevel change will occur (`ondemand` runlevels are `a', `b', and `c').
`off`	This does nothing.
`once`	The process will be executed once when the specified runlevel is entered.
`powerwait`	The process will be executed when the power goes down. `init` is usually informed about this by a process talking to a UPS connected to the computer. `init` will wait for the process to finish before continuing.