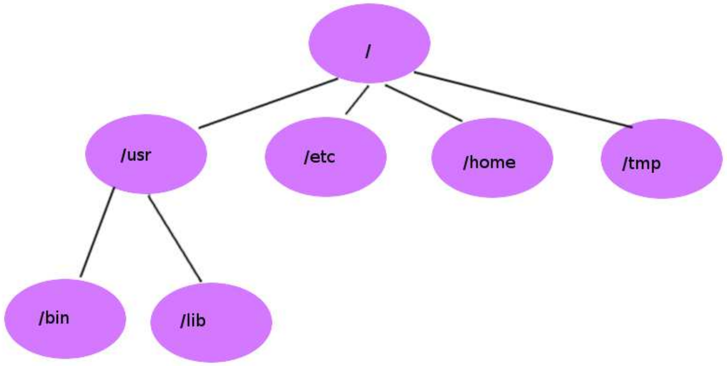
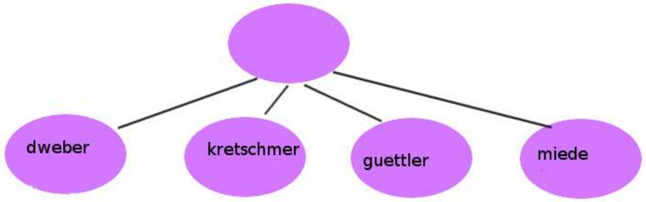
Mounting an FS (1)

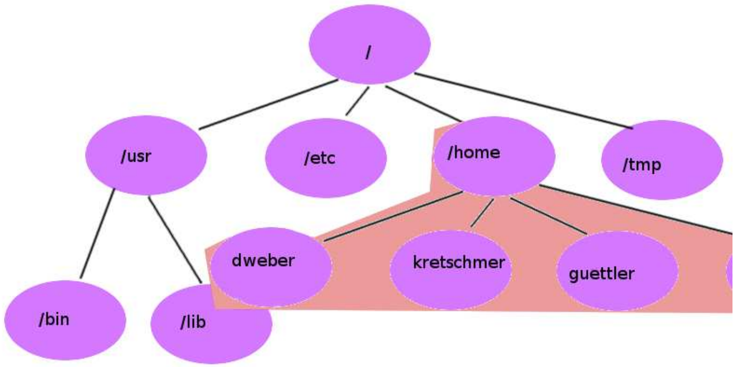
/dev/ada0p2



/dev/ada0p3



Mounting an FS (2)



Mounting an FS (3)

Example:

```
# mount /dev/ada0p6 /tmp
```

Mounts partition /dev/ad0p6 as directory /tmp.

⇝/tmp is called a mount point

⇝mount point = empty directory

Mounting is usually done at boot time.

File /etc/fstab contains device-mount-mapping.

/etc/fstab

```
# Device          M-point   FStype  Options  Dump  Pass#
/dev/ada0p2       /         ufs     rw       1     1
/dev/ada0p3       /usr      ufs     rw       2     2
/dev/ada0p4       /var      ufs     rw       2     2
/dev/ada0p5       /tmp      ufs     rw       2     2
/dev/ada0p10      /TMP      ufs     rw       2     2
134.96.216.92:/home /home nfs     rw       0     0
/dev/acd0         /cdrom  cd9660  ro,noauto 0     0
```

order of entriers important for mount, fsck

dump (# days), pass = order of FS check

Unmounting an FS (1)

Simple:

```
# umount /tmp
```

Or not so easy:

```
# umount /tmp
umount: unmount of /tmp failed: Device busy
```

We should *not* unmount an FS which is currently in use.

But we could:

```
# umount -f /tmp
```

This does *not* work for the *root filesystem*.

Unmounting an FS (2)

Which process uses a disk/file?

```
$ lsof | grep /home
COMMAND  PID   USER  FD TYPE        DEVICE SIZE/OFF    NODE NAME
bash    3627 dweber cwd VDIR 255,117440514     1536 3379712 /home/dweber
lsof    3696 dweber cwd VDIR 255,117440514     1536 3379712 /home/dweber
grep    3697 dweber cwd VDIR 255,117440514     1536 3379712 /home/dweber
```

- alert corresponding users
- kill offending processes
- unmount the FS

Filesystems on a RAMDISK

- create device node for this filesystem
  FreeBSD: `mdconfig`, OpenBSD/NetBSD: `vnconfig`,
  Solaris `ramdiskadm`
  - need info whether to use
    - simply allocated memory (malloc())
    - a file
    - swap space
  - need size
  - should provide a device number
- create filesystem on the device
- mount it

Filesystems on a RAMDISK, Examples

using swap space

```
mdconfig -a -t swap -s 128M -u 10
newfs -U /dev/md10
mount /dev/md10 /tmp
chmod 1777 /tmp
```

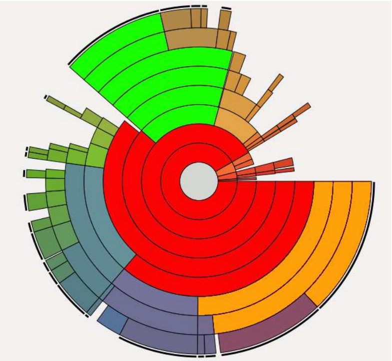
using a file (with bsdlabel)

```
dd if=/dev/zero of=somebackingfile bs=1k count=5k
mdconfig -a -t vnode -f somebackingfile -u 0
gpart create -s gpt md0
gpart add -t freebsd-ufs md0
newfs md0p1
mount /dev/md0p1 /mnt
```

Space Usage on file system: df = disk free

shows mounted file systems with

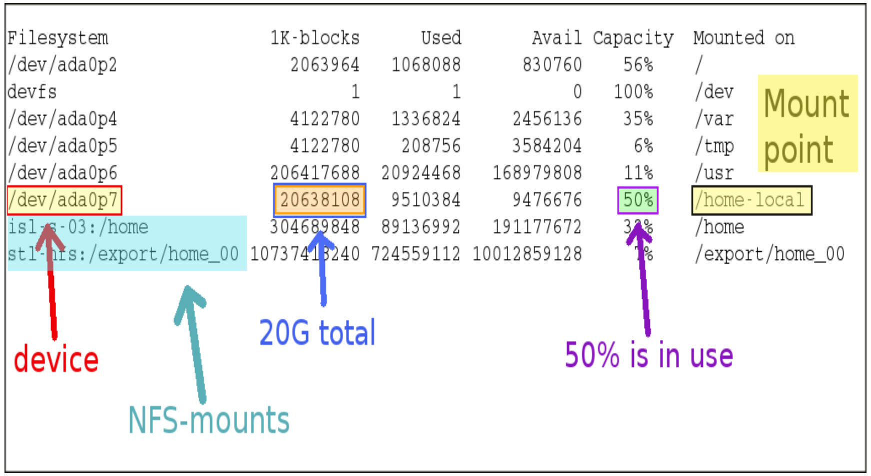
- name
- size in blocks (1K)
- number of used blocks
- number of available blocks
- percentage of use
- mount point



Note:

- must be checked periodically to avoid system failure
- likely overflows in `/home`, `/var`, `/tmp`

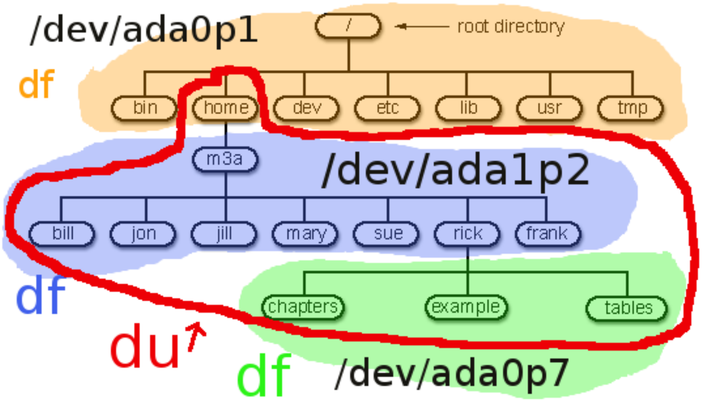
Example: df



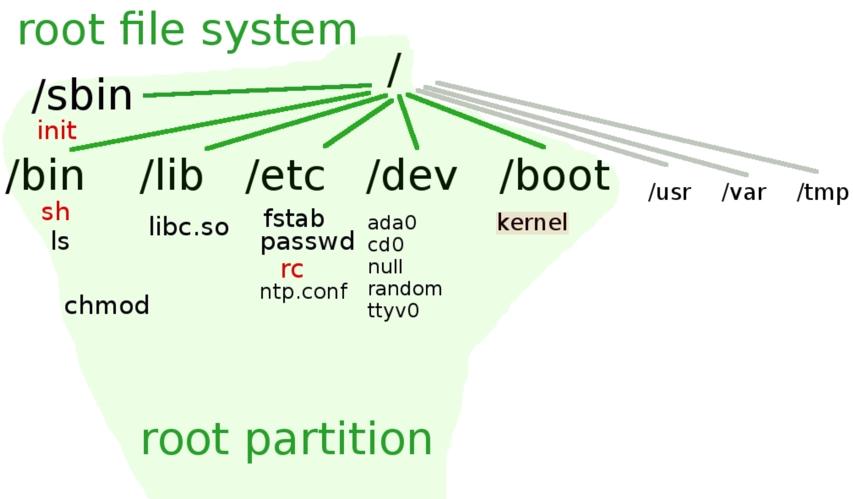
```
Filesystem               1K-blocks      Used        Avail Capacity  Mounted on
/dev/ada0p2                2063964   1068088       830760      56%  /
devfs                            1         1            0     100%  /dev
/dev/ada0p4                4122780   1336824      2456136      35%  /var
/dev/ada0p5                4122780    208756      3584204       6%  /tmp
/dev/ada0p6              206417688  20924468    168979808      11%  /usr
/dev/ada0p7               20638108   9510384      9476676      50%  /home-local
isl-fs-03:/home          304689848  89136992    191177672      32%  /home
stl-nfs:/export/home_00 10737418240 724559112 10012859128       7%  /export/home_00
```

Space Usage on Directory: du = disk usage

```
$ du -s /home
13574540 /home
```



/ – The Root File System (1)



/ – The Root File System (2)

- **system core files**
  - system configuration
  - important shared libs
  - device entries
  - admin commands
- **boot scripts** and corresponding commands
- **mount points** for all other filesystems

/ – The Root File System (3)

Directory	Description	Example
/bin	user commands	/bin/ls
/dev	device entries	/dev/ada0
/etc	configuration	/etc/passwd
/lib	shared libraries	/lib/libc.so
/sbin	system administration commands	/sbin/shutdown
/boot	kernel binary, kernel modules	/boot/kernel/kernel
(/proc)	process information	/proc/curproc/status

The Boot Problem

operating system does

- process management
- memory management
- file system
- I/O

but needs I/O and file system to read the operating system

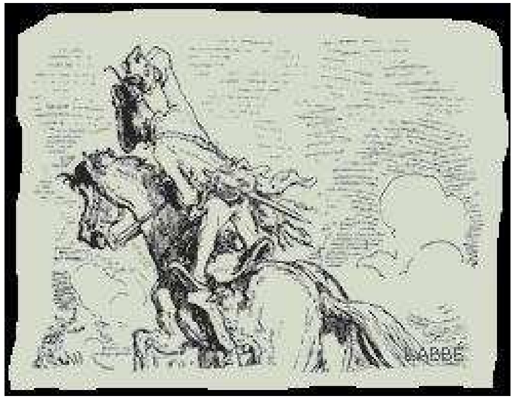
- must determine system disk
- must read boot code from disk
- must read OS kernel from directory tree

⇝**chicken–and–egg problem**

Baron Münchhausen

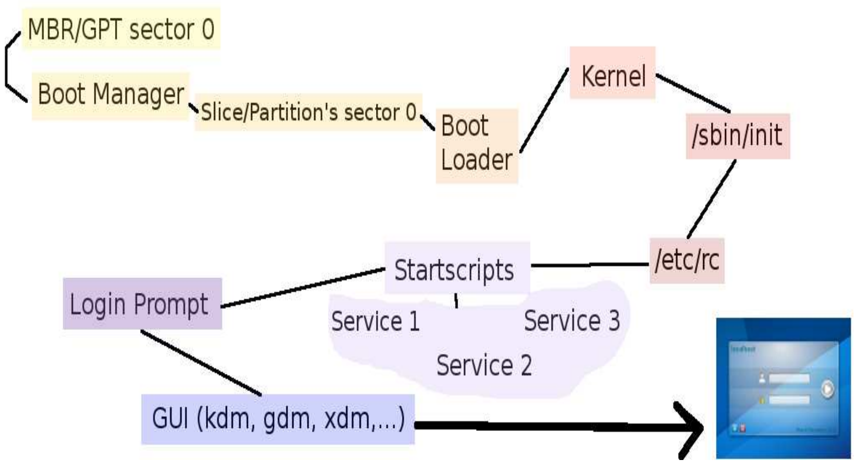
Mein Pferd und ich wären hoffnungslos versunken,
wenn ich es nicht geschafft hätte,
mich an meinem eigenen Haarschopf aus dem Sumpf zu ziehen.

Baron Münchhausen



Es kann eben doch von Vorteil sein,
wenn man einen gut trainierten Körper hat.

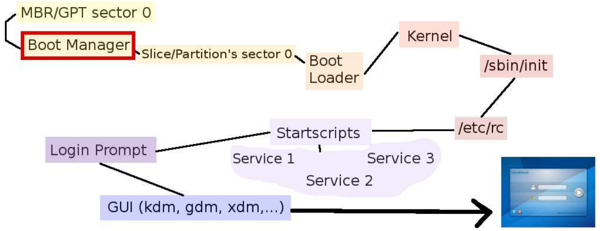
Booting



Starting to Boot (stage 0 boot)

- BIOS = basic input/output system, ROM...EEPROM...Flash
- BIOS locates MBR / GPT
- MBR/GPT code =*boot manager*, 512 bytes, boot menu
  - boot0, standard FreeBSD boot manager
  - GRUB,
  - standard PC MBR (searches active slice)
  - NTLDR, Vista MBR (Windows systems)
- MBR code reads boot loader (BIOS I/O)

Boot Manager / Boot Loader



a *boot manager* is independent from OS (on MBR)

a *boot loader* is OS specific (on slice)

Boot Manager: Select Partition with a Root FS

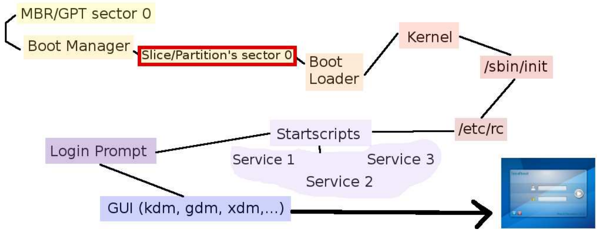
FreeBSD `boot0` start screen (file `/boot/boot0`, 512 bytes)

```
F1 DOS
F2 FreeBSD
F3 Linux
F4 ??
F5 Drive 1

Default: F2
```

source code directory `/usr/src/sys/boot/i386/boot0`

Prepare Loading of Boot Loader



FreeBSD `boot1` (file `/boot/boot1`, 512 bytes)

Located in boot sector of bootable slice ⇝ 512 bytes.

Knows `bsdlabel` data structure.

Finds and loads `boot2` (in the following 15 sectors)

Locate Boot Loader on Partition

FreeBSD `boot2` screenshot (file `/boot/boot2`, 7K bytes)

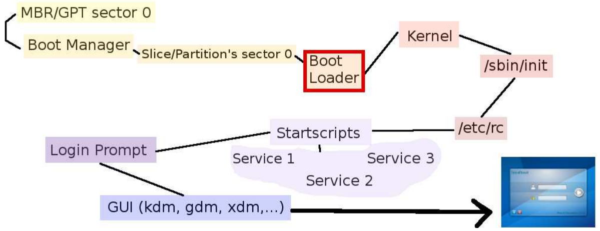
```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Knows how to find files on a UFS filesystem on it

until now, everything coded in machine language directly

Finds and loads `/boot/loader`, (217K)

Boot Loader: Prepare Loading of OS



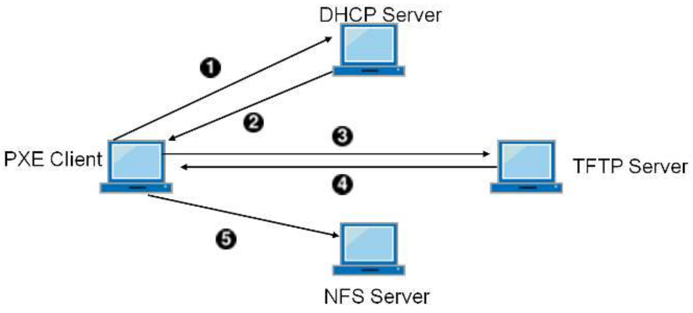
`/boot/loader`

programmed in C, can do:

- probe for a console
- probe for disks,
- figure out what disk it is booting from
- load kernel/modules

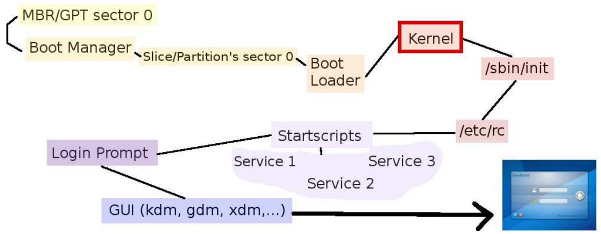
Side note: PXEBOOT

preboot-exec-environment (Intel), on ethernet card



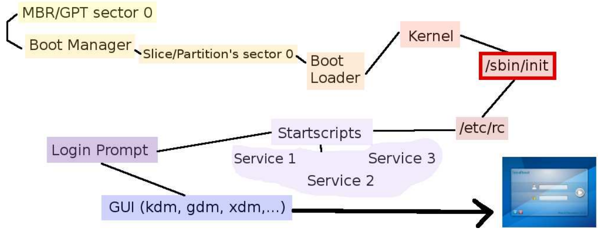
⇝ diskless machines.

Kernel



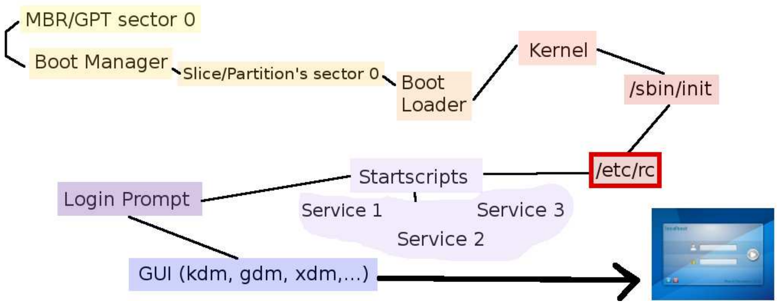
- initialize device drivers
- detect type of root filesystem, if unknown then STOP
- mount root filesystem read–only
- start process `/sbin/init` with PID=1
- init: `/etc/rc`, rc = resource configuration

/sbin/init



- maintains processes for the terminal ports (getty)
- `SIGHUP` ⇝ re–read `/etc/ttys`
- `SIGTERM` ⇝ single-user-mode
- `SIGINT` ⇝ kill all processes, i.e. shutdown
- if `init` terminates ⇝ reboot

/etc/rc



/etc/rc, the original idea

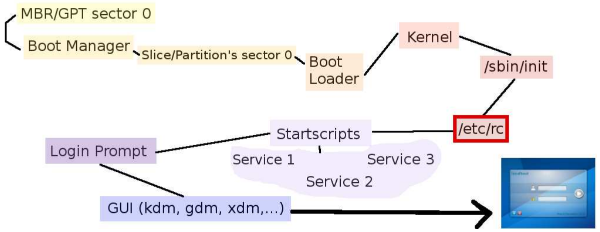
simple shell script, typical start sequences:

```
...
if [ "X${httpd_flags}" != X"NO" ]; then
        # Clean up left-over httpd locks
        rm -f /var/www/logs/{ssl_mutex,httpd.lock,accept.lock}.*
        echo -n ’ httpd’;     /usr/sbin/httpd ${httpd_flags}
fi

if [ "X${ftpd_flags}" != X"NO" ]; then
        echo -n ’ ftpd’;      /usr/libexec/ftpd ${ftpd_flags}
fi

if [ "X${identd_flags}" != X"NO" ]; then
        echo -n ’ identd’;    /usr/libexec/identd ${identd_flags}
fi
...
```

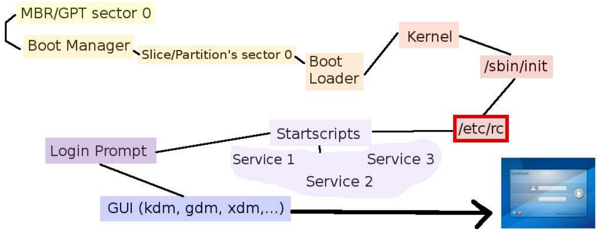
/etc/rc configuration (1)



variables . . .

```
httpd_flags="NO"
ftpd_flags="-t 120"
```

/etc/rc configuration (2)



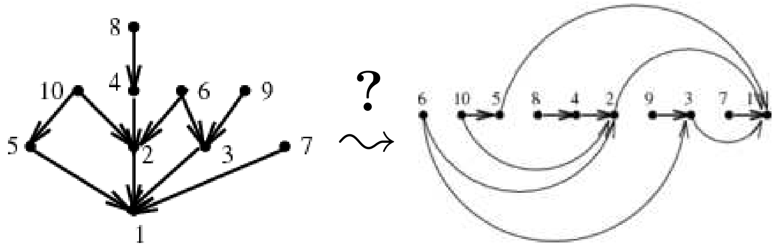
. . . are configured in startup config files

```
/etc/rc.conf
/etc/rc.conf.local
```

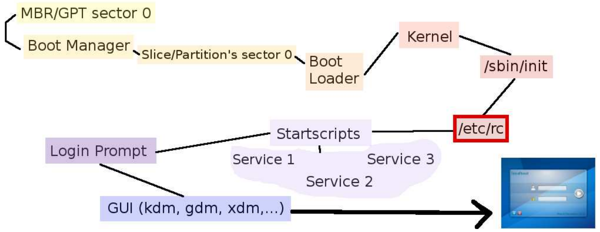
. . . and loaded in rc as follows . . .

```
. /etc/rc.conf
```

Problem: Dependencies between Services



/etc/rc, Enhanced Version



used in FreeBSD/NetBSD

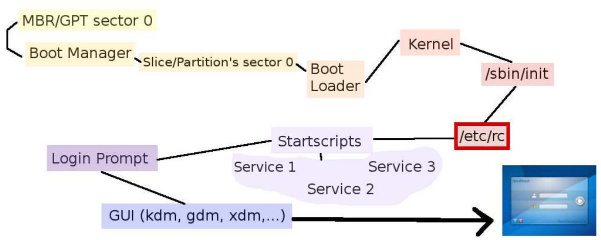
- rc: order all startup scripts by `rcorder`
- /etc/rc.d/*: startup scripts contain dependencies
- rc: starts scripts in `/etc/rc*.d` according to output of **topological sort**

example: RPC service `rpcbind`

```
#!/bin/sh
#

# PROVIDE: rpcbind
# REQUIRE: NETWORKING ntpdate syslogd named
```

/etc/rc, SYSVINIT Version



from UNIX system V, used in Linux, Solaris

⇝`/etc/inittab` exists, configures ,,runlevels''

runlevel: state of a system (which set of services is active)

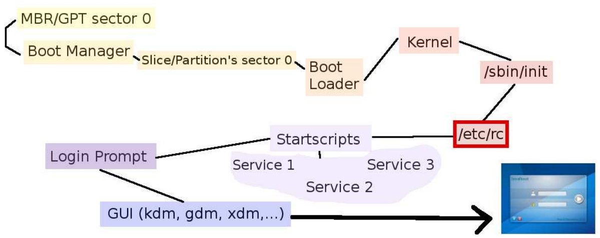
/etc/rc, SYSVINIT Version

- running (runlevels 2, 3, 5)
- shutdown (runlevels 0, 6)
- single user (runlevels 1, S)

normal operation: runlevels 2 or 3 (or 5)

determine set of scripts to be executed

/etc/rc, SYSVINIT Version

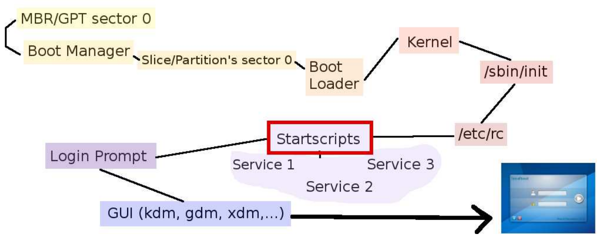


per runlevel there is a directory of softlinks

example `/etc/init.d/rc2.d`

```
...
lrwxrwxrwx 1 root root S05network -> ../network
lrwxrwxrwx 1 root root S06syslog -> ../syslog
lrwxrwxrwx 1 root root S07splash_early -> ../splash_early
lrwxrwxrwx 1 root root S10alsasound -> ../alsasound
lrwxrwxrwx 1 root root S10cups -> ../cups
...
```

Startscripts (1)

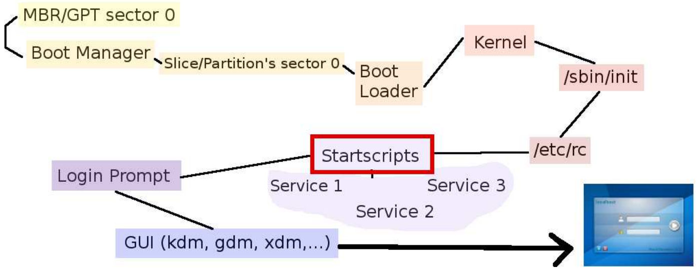


MBR/GPT sector 0
Boot Manager
Slice/Partition's sector 0
Boot Loader
Kernel
/sbin/init
/etc/rc
Startscripts
Login Prompt
Service 1
Service 3
Service 2
GUI (kdm, gdm, xdm,...)

each daemon/service has a start script

- checks configuration files
- determines if service may be started
- starts service (usually in /usr/sbin)

Startscripts (1)



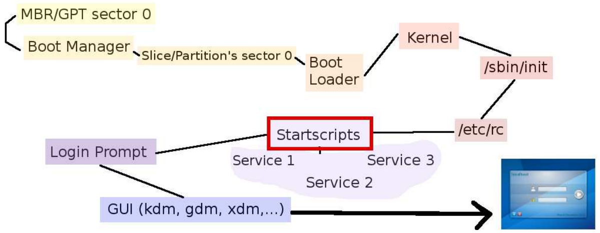
MBR/GPT sector 0
Boot Manager
Slice/Partition's sector 0
Boot Loader
Kernel
/sbin/init
/etc/rc
Startscripts
Login Prompt
Service 1
Service 3
Service 2
GUI (kdm, gdm, xdm,...)

also control shutdown of service

should implement parameters

`start` `stop` `restart` `reload` `status`

Startscripts (2, FreeBSD, NetBSD)



MBR/GPT sector 0
Boot Manager
Slice/Partition's sector 0
Boot Loader
Kernel
/sbin/init
/etc/rc
Startscripts
Login Prompt
Service 1
Service 3
Service 2
GUI (kdm, gdm, xdm,...)

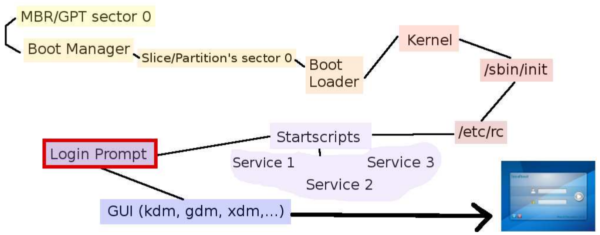
each startscript is located in /etc/rc.d

uses script infrastructure from /etc/rc.subr

points to service that must be started

```
name="sshd"
rcvar=‘set_rcvar‘
command="/usr/sbin/${name}"
start_precmd="sshd_precmd"
pidfile="/var/run/${name}.pid"
extra_commands="keygen reload"
```

Single User Mode, Definition

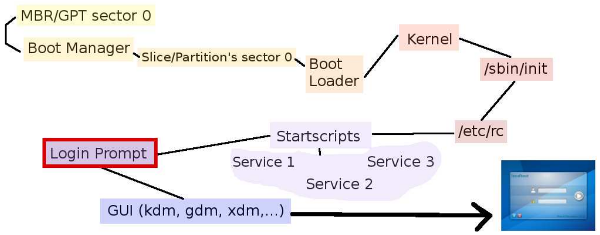


MBR/GPT sector 0
Boot Manager
Slice/Partition's sector 0
Boot Loader
Kernel
/sbin/init
/etc/rc
Startscripts
Service 1
Service 2
Service 3
Login Prompt
GUI (kdm, gdm, xdm,...)

- only root is allowed to log in
- only root filesystem is mounted

use this mode only for special tasks

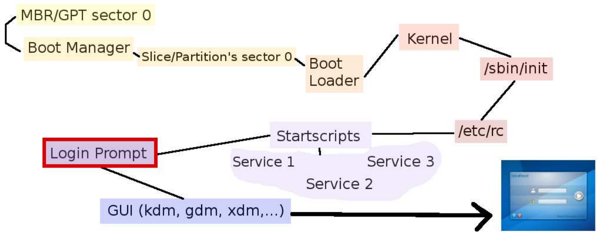
Single User Mode, Examples



MBR/GPT sector 0
Boot Manager
Slice/Partition's sector 0
Boot Loader
Kernel
/sbin/init
/etc/rc
Startscripts
Service 1
Service 2
Service 3
Login Prompt
GUI (kdm, gdm, xdm,...)

- upgrade system (kernel, system lib, tools)
- repair filesystems after system crash
- forensics/clean-up after system break–in
- fix problems in critical system files
  - `/etc/fstab`
  - `/etc/inittab` (if SYSVINIT system)
- restore files from backup

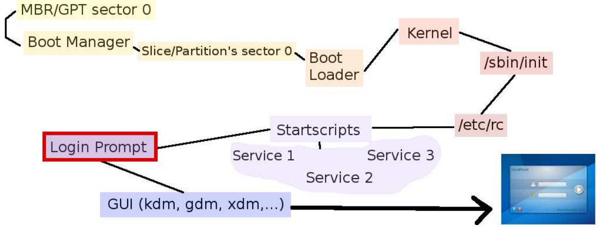
Invoking Single User Mode



MBR/GPT sector 0
Boot Manager
Slice/Partition's sector 0
Boot Loader
Kernel
/sbin/init
/etc/rc
Startscripts
Service 1
Service 2
Service 3
Login Prompt
GUI (kdm, gdm, xdm,...)

- Use `shutdown` without `-h` or `-r`.
- On `loader` prompt use `boot -s`
- On `loader` menu use *single user*

Login Prompt (text–oriented)

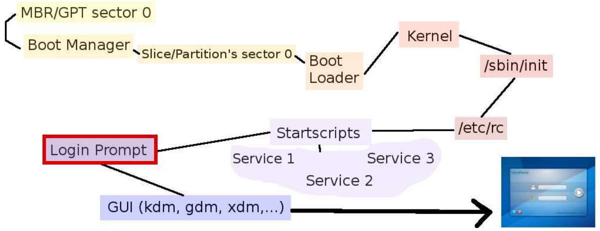


several text–oriented login screens

via `/etc/ttys`

```
# name   getty                       type      status
ttyv0   "/usr/libexec/getty Pc"    cons25l1   on  secure
ttyv1   "/usr/libexec/getty Pc"    cons25l1   on  secure
ttyv2   "/usr/libexec/getty Pc"    cons25l1   on  secure
...
```

Side note: Console

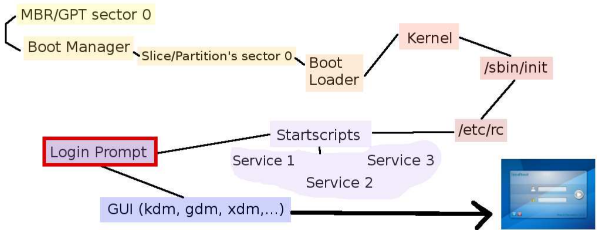


a text-mode terminal (usually 80x25)

- usually used for `root` login
- direct connection to the motherboard
  - PS/2
  - serial
  - ...

- may be used to control root access to the machine (physical presence required)
- change resolution with
  - `vidcontrol` (FreeBSD) (even 1024x768 resolution with MODE_279)
  - kernel boot parameter (Linux)

Side note: Console (2)



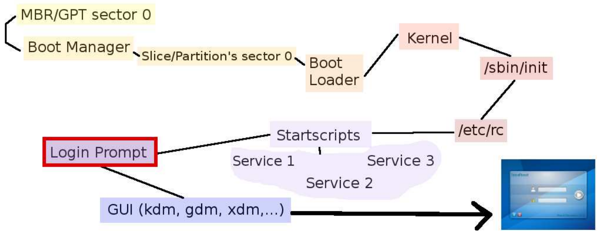
boot and have root ? FreeBSD–Version

see /etc/ttys on a FreeBSD-system

```
# If console is marked "insecure",
# then init will ask for the root password
# when going to single-user mode.

console none      unknown on insecure
```

Side note: Console (3)



boot and have root ? Linux–Version

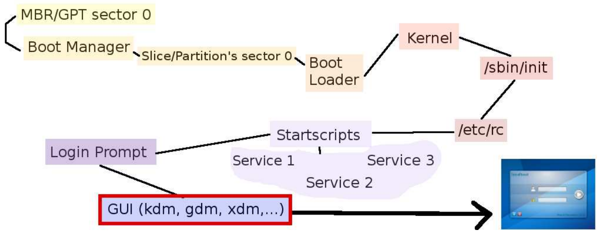
start from GRUB in single user mode

(append `single` on kernel–line and init=/bin/bash)

first process is root shell (no password needed)

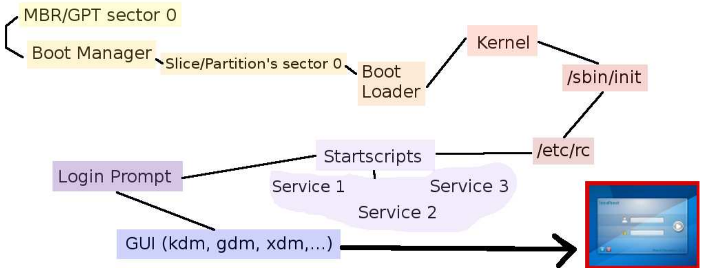
⇝must set password for GRUB/LILO

Login Prompt (for GUI)



- depends on Xorg
  (GUI base system, formerly X11)
- requires root privileges (graphics card)
  - insecure: SETUID `/usr/local/bin/X` from terminal,
  - more secure: display manager
    (xdm, kdm, gdm, slim, . . . as root)

Login Prompt (for GUI)



Login Prompt (Examples)

KDM

GDM

XDM

SLIM

System Up and Running