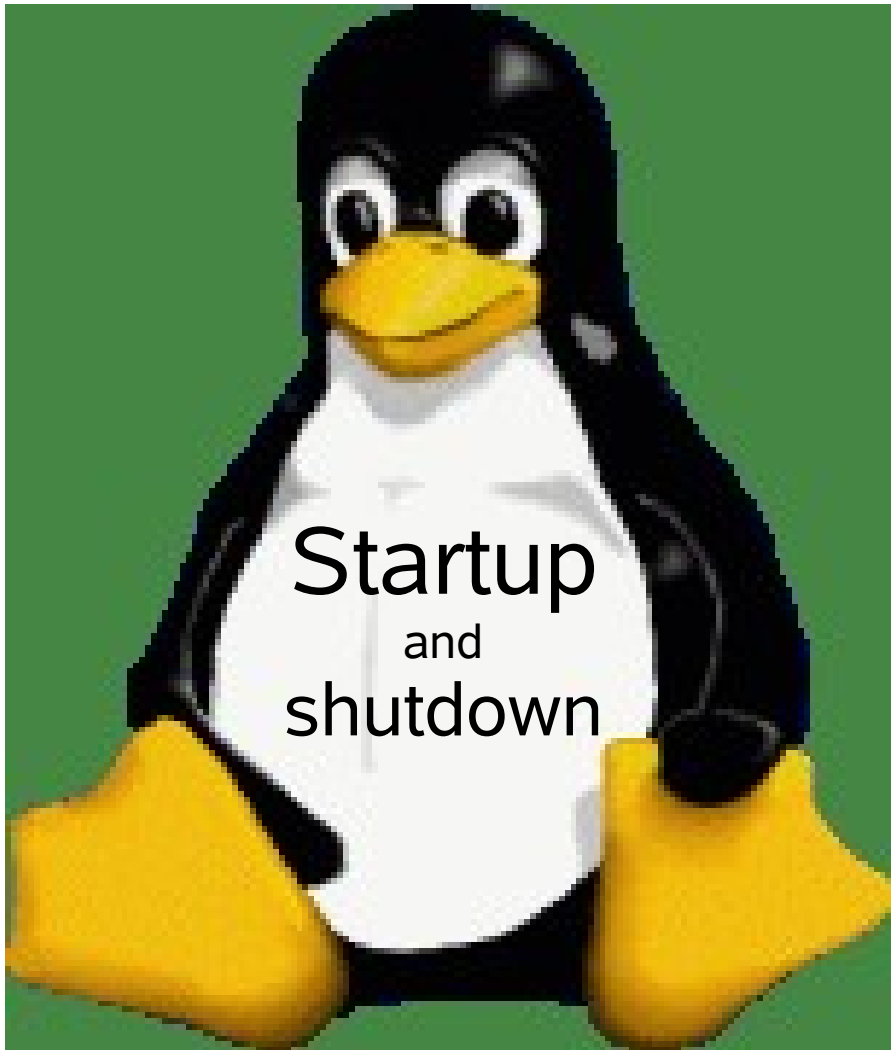
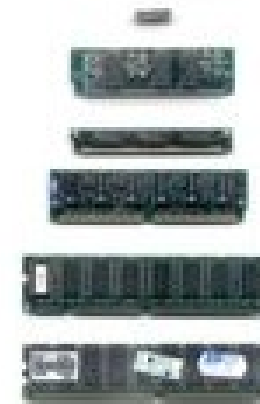
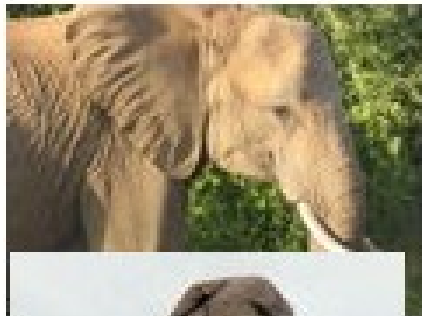


# What goes up must come down.



- Shutdown your LINUX.
- New file systems.
- umount
- Application cleanup.
- disconnect from remote hosts.

# The really real world.



# Philosophy

- You can't press the undo button if an elephant knocks down your house, or eats your crop!
- You can't create a new elephant by calling the copy constructor!



# Philosophy

- Human beings do not create objects; they represent them or mold them.
- Computers are used to represent objects.
- In elephant management software, there are no elephants in the computer.



# Philosophy

- In designing, running, or debugging software, you must make a distinction between an object and the **representation of an object!**
- A program may represent an object.



# Application caching



- Applications “cache” to postpone updating their representations of reality.
- They write “postit” notes to themselves reminding themselves to do it later!
- A system crash causes these notes to be lost!

# Startup RunLevels



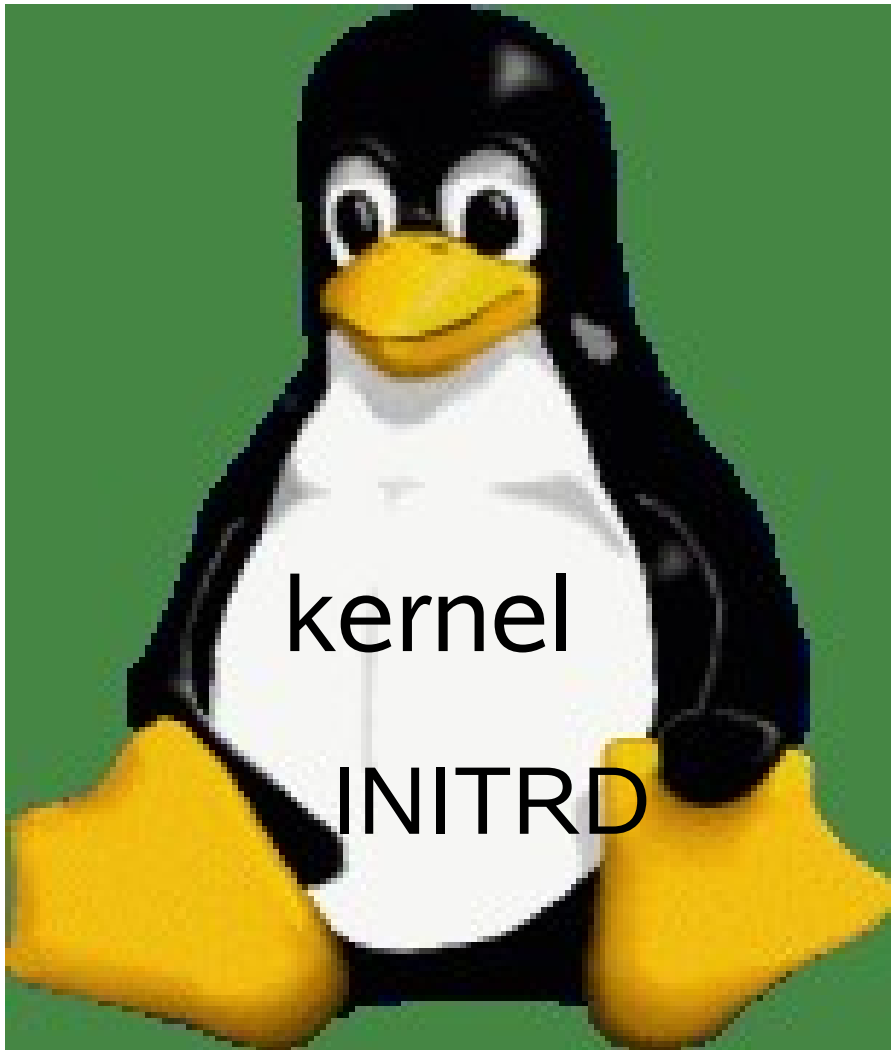
- 0 system halt
- 1 Single user mode
- 2 multiuser, no net, no gui
- 3 multiuser with net, no gui
- 4 not used
- 5 multiuser with gui
- 6 reboot

# Run levels on startup

Add number to command line on startup to boot with reduced functionality, to avoid crashes other, problems!



# What is INITRD?



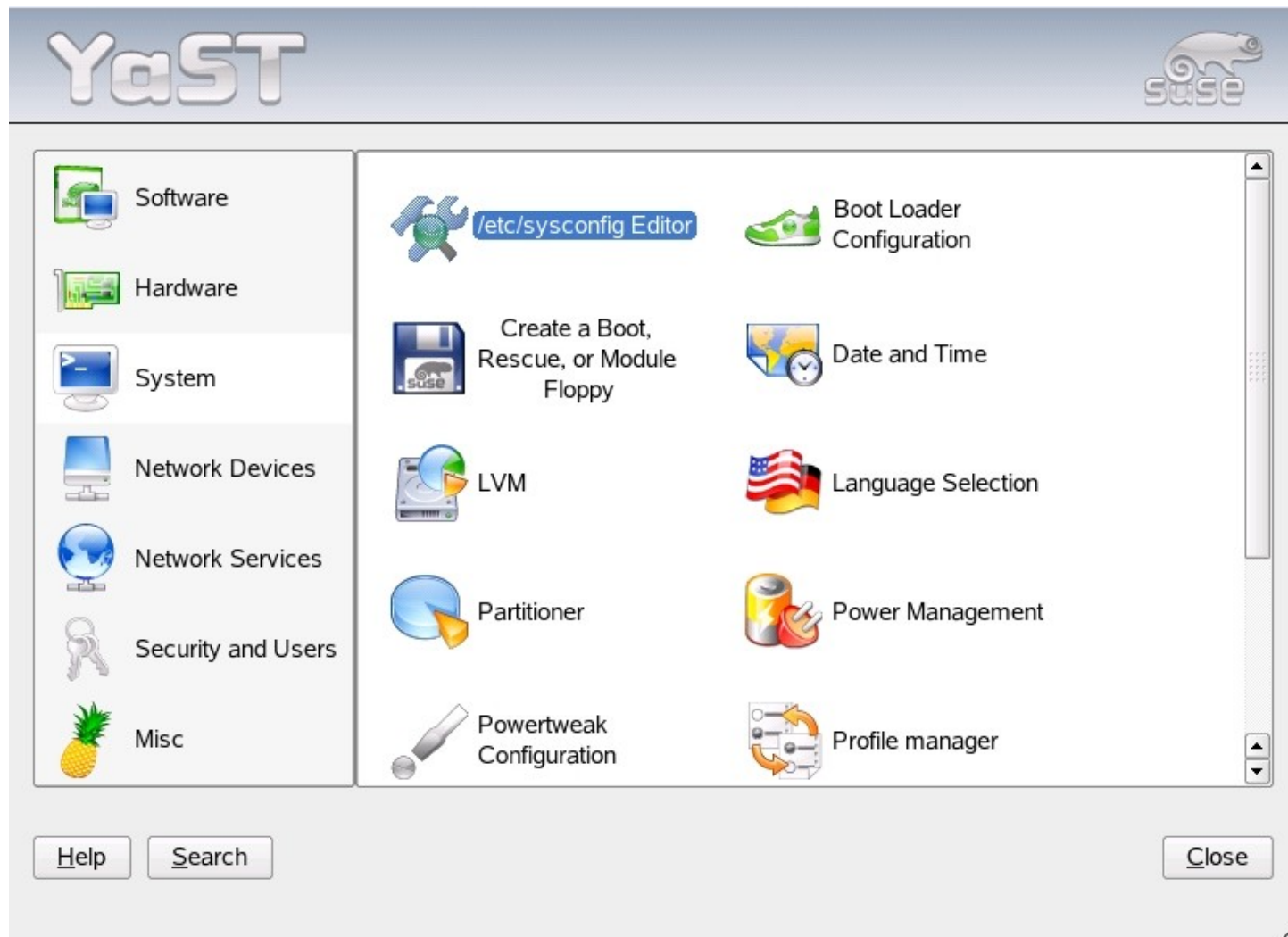
- Modular Kernels – no built in drivers!
- LINUX bootloaders
- GRUB, LILO
- use BIOS to boot
- How do I bring up LINUX, if I do not know how to read the hard disk?

# Initrd

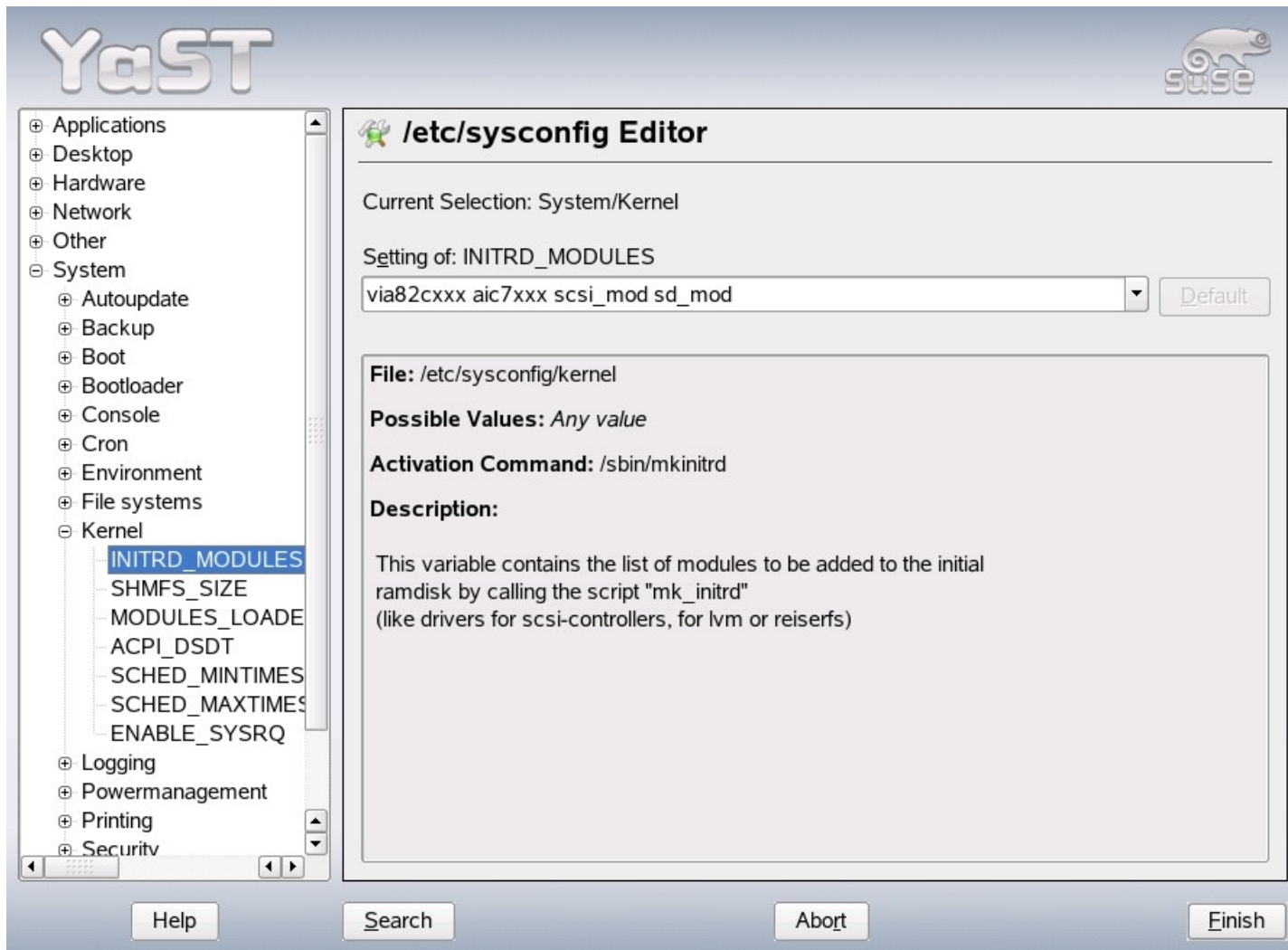
Initrd allows Linux to have a temporary virtual “disk” during booting. On this disk are the drivers necessary to start LINUX (and do other things).

Initrd's are basically the same, but the software to create them is distro dependent. Usually called mkinitrd.


# Yast example



# sysconfig parameter



The image shows the YaST2 configuration utility interface. On the left is a tree view of system categories, with 'Kernel' expanded to show 'INITRD\_MODULES' selected. The main window is titled '/etc/sysconfig Editor' and shows the current selection as 'System/Kernel'. The parameter being edited is 'INITRD\_MODULES', with a value of 'via82cxxx aic7xxx scsi\_mod sd\_mod' entered in a text field. A 'Default' button is next to the field. Below the field, the file path is '/etc/sysconfig/kernel', and the possible values are 'Any value'. The activation command is '/sbin/mkinitrd'. The description states: 'This variable contains the list of modules to be added to the initial ramdisk by calling the script "mk\_initrd" (like drivers for scsi-controllers, for lvm or reiserfs)'. At the bottom of the window are buttons for 'Help', 'Search', 'Abort', and 'Finish'.

**YaST** 

**/etc/sysconfig Editor**

Current Selection: System/Kernel

Setting of: INITRD\_MODULES

via82cxxx aic7xxx scsi\_mod sd\_mod

**File:** /etc/sysconfig/kernel

**Possible Values:** Any value

**Activation Command:** /sbin/mkinitrd

**Description:**

This variable contains the list of modules to be added to the initial ramdisk by calling the script "mk\_initrd" (like drivers for scsi-controllers, for lvm or reiserfs)

# Other distros are different!

Red Hat has a different way of specifying which modules are to be included in INITRD. **READ YOUR DOCUMENTATION.** Luckily, most users don't have to mess with this stuff. It is all figured out by your point and clicky installation installer thingy!

# /boot partition



- Bootloader uses BIOS to read in kernel, INITRD!
- Therefore, kernel, INITRD must be in a partition readable **by the BIOS!**
- So put it in /boot
- overcome bios limits!

# Large Disk Howto disk limits

<http://www.tldp.org/HOWTO/Large-Disk-HOWTO-4.html>

- ATA Specification (for IDE disks) - the 137 GB limit
- BIOS Int 13 - the 8.5 GB limit
- 528 MB limit
- 2.1 GB limit
- 3.2 GB limit
- 7.9 GB limit
- 8.4 GB limit
- 4.2 GB limit
- 7.9 GB limit
- 8.4 GB limit
- 33.8 GB limit
- 137 GB limit
- 2 TiB limit

# Init Scripts



- located in `/etc/init.d/`
- control what happens at startup, shutdown time.
- LINUX STANDARD BASE defines
- Really controlled by links in `/etc/init.d/rc?.d` (don't touch)



# Install / Uninstall Init scripts

- Use `insserv` to install.
- `/usr/lib/lsb/install_initd`
- Use `insserv -r` to remove
- `/usr/lib/lsb/remove_initd`

These programs really manipulate the links in `/etc/init.d/rc.d`. Every time they run they recompute these links. Take a run level parameter so you can control what happens at various run levels.

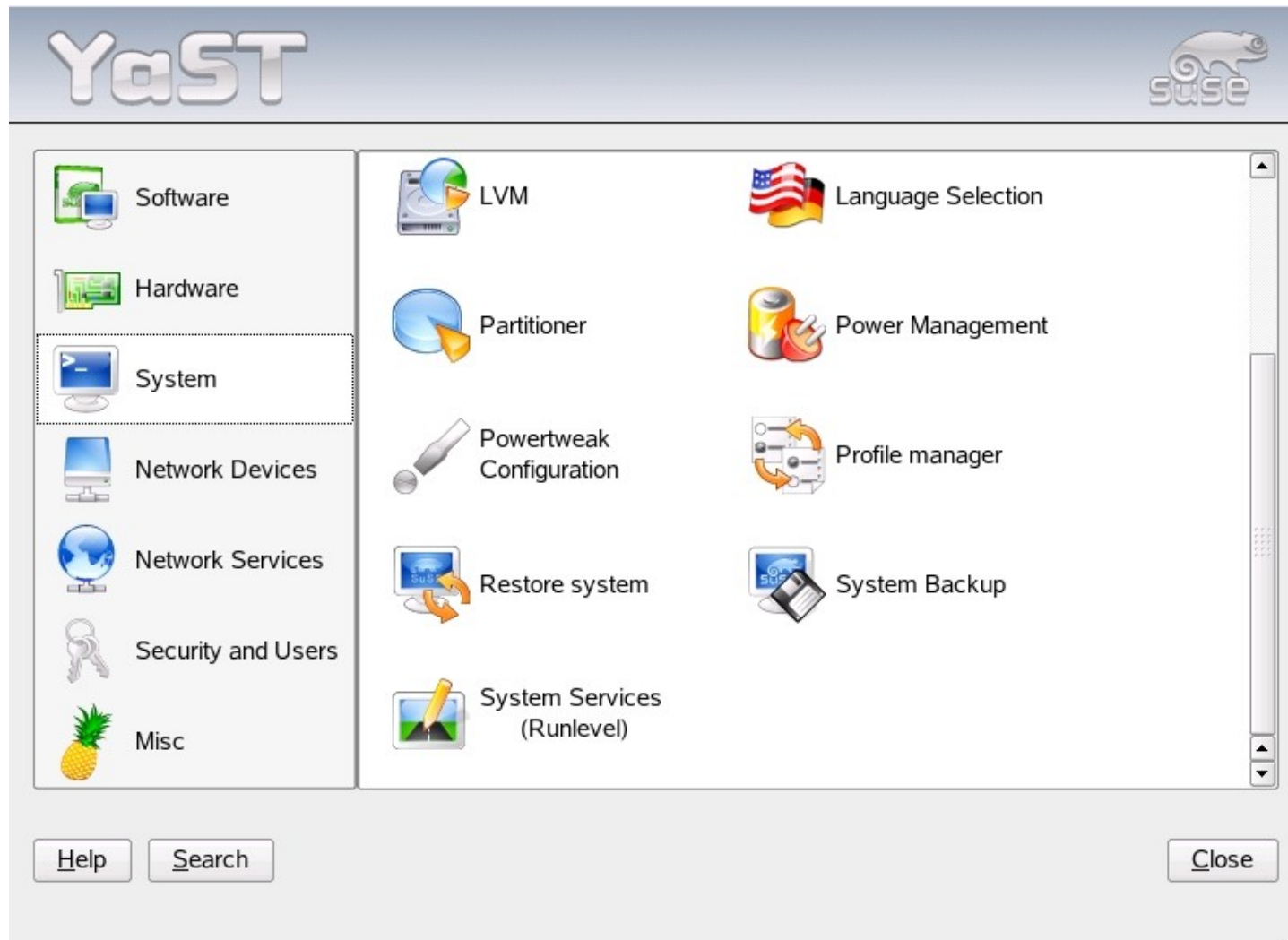
These scripts can be run manually:

```
#cd /etc/init.d
```

```
#!/sendmail restart
```

```
#restarts sendmail!
```

# System Services (runlevel)



# Control scripts for runlevels

Assign system services to runlevels by selecting the list entry of the respective service then checking or unchecking the **check boxes B-S** for the runlevel.

**Start/Stop/Refresh:** Use this to start or stop services individually.

**Set and Reset:** Select runlevels in which to run the currently selected service.

- ◆ **Enable the service:** Activates the service in the standard runlevels.
- ◆ **Disable the service:** Deactivates service.
- ◆ **Enable all services:** Activates all services in their standard runlevels.

Changes to the **default**

### System Services (Runlevel): Details

Simple Mode     Expert Mode

Set default runlevel after booting to:

5: Full multiuser with network and display manager

Service	Running	B	0	1	2	3	5	6	S	Description
SuSEfirewall2_init	Yes	B								SuSEfirewall2 phase 1
<b>SuSEfirewall2_setup</b>	<b>Yes</b>					<b>3</b>	<b>5</b>			<b>SuSEfirewall2 phase 2</b>
acct	No									Process accounting
acpid	No									Listen and dispatch ACPI
alsasound	Yes				2	3	5			Loading ALSA drivers and
apache2	Yes						3	5		Apache2 httpd
atd	Yes				2	3	5			Start AT batch job daemo
autoyast	Yes									A start script to execute a
bluetooth	No									Bluetooth protocol stack s

SuSEfirewall2\_setup does some basic setup and is the phase 2 of 2 of the SuSEfirewall initialization.

Service will be started in following runlevels:

B     0     1     2     3     5     6     S

Start/Stop/Refresh    Set/Reset

Back    Abort    Finish

**Other Distributions are different**

# Write your own init scripts

- In Bash
- Start with `/etc/init.d/skeleton`
- Read all the documentation first.
- define start, stop, maybe restart.
- Control script execution order with
- `# Provides:`
- `# Required-Start:`
- `# Required-Stop:`

# Start/Kill Daemons

- startproc
- start\_daemon
- to start daemons.
- killproc
- to kill daemons.