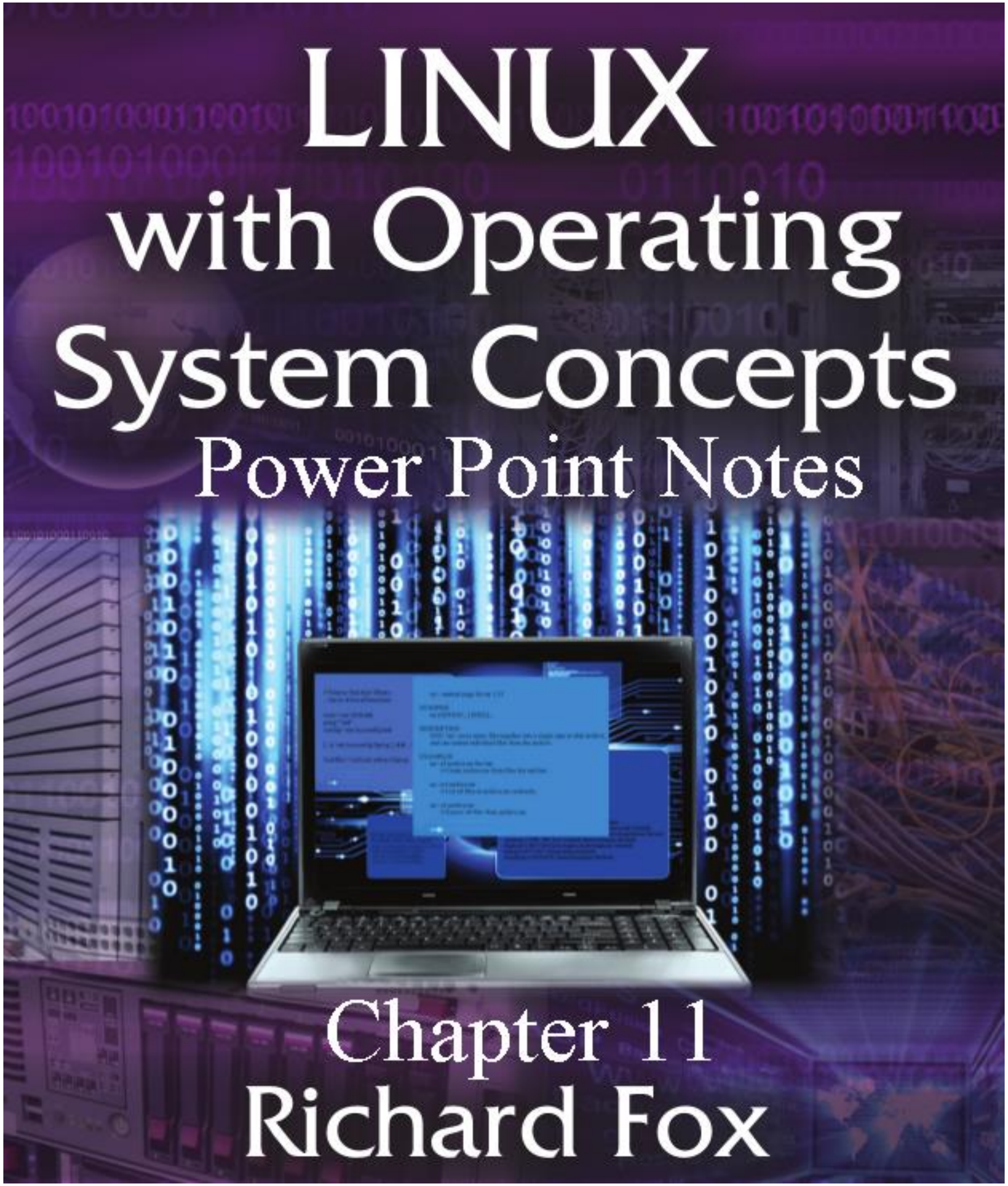


LINUX with Operating System Concepts

Power Point Notes

Chapter 11

Richard Fox

Booting

- Main memory stores the OS
- The OS needs to be in memory and running for us to be able to start and run other processes
- Main memory is volatile – turn off the power and you lose the contents
- When you turn on the computer, main memory is empty
  - How then do you find, load and start the OS when you need the OS to find, load and start a process?
  - We need a one-time startup routine stored somewhere else
  - This is called the boot process

Booting: ROM vs RAM

- The term random access memory is somewhat misleading because DRAM, SRAM and ROM all qualify as random access memories
- We will instead refer to main memory as DRAM (dynamic RAM)
- Cache and register memory as SRAM
- ROM (read only memory) is non-volatile
- We will place part of the boot process here
  - Enough of the boot process so that it can locate and load the rest of the boot process from hard disk to memory
  - We only put part of the boot process in ROM because ROM is expensive memory
    - see the comparison on the next slide

Booting: ROM vs RAM

Type	Volatility	Typical Amount	Relative Expense	Usage
DRAM	Volatile	4–16 GByte	Very cheap	Main memory: stores running program code and data, graphics
SRAM	Volatile	1-2 MByte	Moderately expensive	Stores recently and currently used portions of program code and data
ROM	Non-volatile	4K or less	Very expensive	Stores unchanging information: the boot program, basic I/O device drivers, microcode

Booting: The Process

- Turn on the power
- ROM BIOS (basic IO system) starts
  - Power on self test (POST) – tests various pieces of hardware (CPU registers, main memory, interrupt controller, disk controllers, timer) and identifies all devices connected to the system bus
  - Assembles a list of all bootable devices (hard disk, floppy disk, optical disk, flash drive, network)
  - Unless interrupted, attempts to locate the OS by working through this list in a priority order
  - Run the boot loader program found which loads the OS kernel

Booting: Boot Loaders

- The boot loader is a program responsible for finding the OS kernel on disk and loading it into memory
  - The boot loader is usually partially stored on the first sector of the internal hard disk known as the master boot record
- The two most popular Linux boot loaders are
  - GRUB – can boot between Linux and Windows
  - LILO – boots between different Linux OS's
  - Another is called loadlin which actually runs under DOS or Windows to transfer control from a booted DOS/Windows environment to Linux

Booting: Boot Loaders

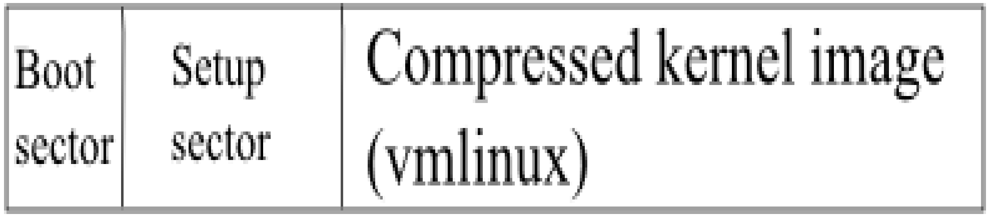
- GRUB - GRand Unified Bootloader
  - Stored in 2 or 3 stages
  - Stage 1: stored in the MBR provides a partition table to indicate where other file systems are located including the rest of the boot loader
  - Stage 1.5 (if any): contains device drivers to communicate with different types of file systems
  - Stage 2: loads the GRUB configuration file from the /boot partition which includes the instruction to launch the Linux kernel

Booting: Boot Loaders

- LILO - LInux Loader
  - Operates in two parts, the first part is responsible for finding the second part
  - LILO is file system independent
  - LILO’s configuration file is stored under /etc/lilo.conf
    - note that GRUB can only access the boot partition but LILO can access /etc

Booting: The Kernel

- The boot loader locates the kernel, now what?
  - – The Linux kernel is partially an executable and partially a compressed file – this file is called vmlinuz
  - – Running the executable uncompresses the remainder of the file providing us the full kernel, vmlinux



Boot sector	Setup sector	Compressed kernel image (vmlinux)

Initialization

- With vmlinux available, it begins executing
- The first step is kernel initialization
  - Power system tests compare various components
  - Ramdisks are loaded
  - Buses are tested and the CPU attempts to communicate with various computer hardware (monitor, keyboard, memory, disk controller, timer, plug and play devices)
  - Interrupt handlers (IRQs) are established
- One specific ramdisk is set up to hold initramfs
  - This is the root of the initial Linux file system
  - This is not the file system we will see but the file system used by the kernel to continue initializing

Initialization: initramfs

- This file system is placed into a ramdisk for quick communication and because we have yet to establish (mount) the full file system
  - This file system to some extent mirrors the regular Linux file system in that there are top-level directories of bin, dev, etc, lib, proc, sbin, sys (as well as others)
  - However, these directories contain only files necessary to initialize and run the kernel

Initialization: pivot_root & init

- After the kernel has initialized, it executes the instruction pivot_root
  - This causes the root file system to change from initramfs to /, the true root of the file system
  - Now the init process (/sbin/init) executes
  - In earlier versions of Linux, init was a synchronous process meaning that each step had to complete before the next step was attempted
    - if a step hangs such as connection to the network, the system hangs without continuing
  - Newer versions of Linux use Upstart
  - Event-based version of init capable of asynchronous action
    - if some step hangs, the rest of the system can still be brought up

Initialization: init

- The init process is always the first started (has a PID of 1) and will remain running until the system is shut down
- With init running, the kernel moves to the background awaiting system calls
  - init’s first step is to invoke /etc/inittab
  - this script’s responsibility is to establish the default runlevel to start in (usually runlevel 5)
  - this file may have other commands as well (see the next slide)

Initialization: init

- Commands are of the form
  - name:#:action:process
    - where name is an identifier, # is a runlevel (optional), action is the operation that inittab will take and process is the invocation of a program (optional)
  - Examples
    - id:5:initdefault: - initialize in runlevel 5
    - rc::bootwait:/etc/rc - execute /etc/rc script during the init process but does not establish a runlevel
    - 2:1:respawn:/etc/getty 9600 tty2 - respawn indicates that the given process should run when a current tty terminates, setting the runlevel to 1
    - ca::ctrlaltdel:/sbin/shutdown –t90 120 "shutting down now" - when the user presses ctrl+alt+del, /sbin/shutdown will run with parameters –t90 120 “shutting down now”
    - si::sysinit:/etc/rc.d/rc.sysinit – execute rc.sysinit after init but before any boot or bootwait entries

Initialization: runlevels

Run Level	Name	Common Usage
0	Halt	Shuts down the system; not used in inittab as it would immediately shut down on initialization.
1	Single-user mode	Useful for administrative tasks including unmounting partitions and reinstalling portions of the OS; when used, only root access is available.
2	Multi-user mode	In multi-user mode, Linux allows users other than root to log in. In this case, network services are not started so that the user is limited to access via the console only.
3	Multi-user mode with Networking	Commonly used mode for servers or systems that do not require graphical interface.
4	Not used	For special/undefined purposes.
5	Multi-user mode with Networking and GUI	Most common mode for a Linux workstation.
6	Reboot	Reboots the system; not used in inittab because it would reboot repeatedly.

Initialization: rcS.conf, rc.sysinit

- Next, the rcS.conf script executes
  - This script looks for the word emergency in the /proc/cmdline file and if found, executes rcS-emergency to handle it
- Next, rc.sysinit executes
  - This script is in charge of initializing hardware, loading kernel modules, mounting special file systems (e.g., /proc, /sys), establishing the SELinux status and executing other scripts

Initialization: rc.conf, rc

- The rc.conf script executes which invokes rc
- rc, based on the runlevel, starts and stops services using code like the following

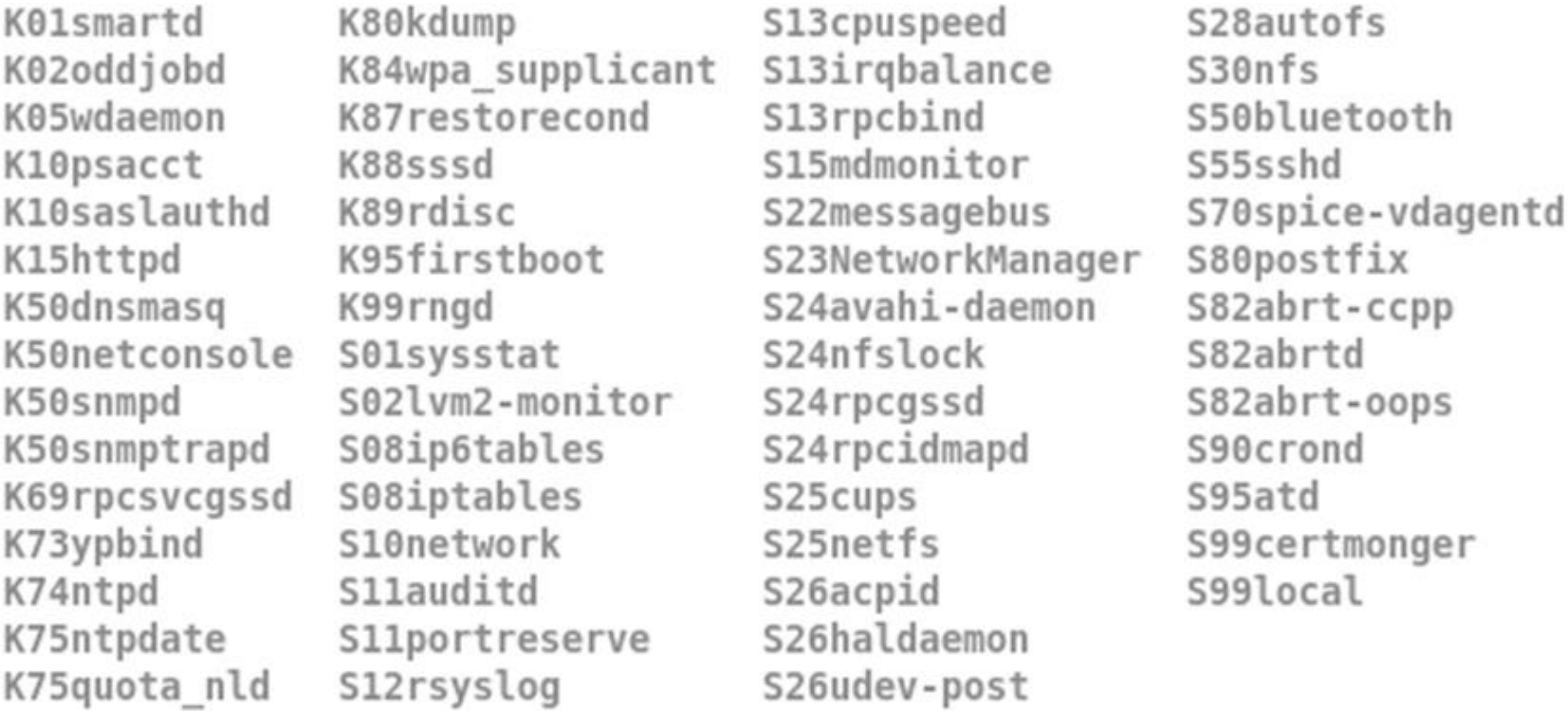
```
for i in /etc/rc$runlevel.d/K* ; do
        $i stop
for i in /etc/rc$runlevel.d/S* ; do
        $i start
```

- There are directories for each runlevel
  - /etc/rc0.d, /etc/rc1.d, …, /etc/rc6.d
  - Entries in these directories are symbolic links whose names are either K##name or S##name
  - K = kill (stop), S = start
  - ## is a 2-digit number to indicate an ordering by which services are stopped and started

Initialization: rc

The following is the listing for /etc/rc5.d

These are symbolic links to the actual scripts in /etc/init.d to start and stop
The various services for runlevel 5



```
K01smartd      K80kdump            S13cpuspeed         S28autofs
K02oddjobd     K84wpa_supplicant   S13irqbalance       S30nfs
K05wdaemon     K87restorecond      S13rpcbind          S50bluetooth
K10psacct      K88sssd             S15mdmonitor        S55sshd
K10saslauthd   K89rdisc            S22messagebus       S70spice-vdagentd
K15httpd       K95firstboot        S23NetworkManager   S80postfix
K50dnsmasq     K99rngd             S24avahi-daemon     S82abrt-ccpp
K50netconsole  S01sysstat          S24nfslock          S82abrtd
K50snmpd       S02lvm2-monitor     S24rpcgssd          S82abrt-oops
K50snmptrapd   S08ip6tables        S24rpcidmapd        S90crond
K69rpcsvcgssd  S08iptables         S25cups             S95atd
K73ypbind      S10network          S25netfs            S99certmonger
K74ntpd        S11auditd           S26acpid            S99local
K75ntpdate     S11portreserve      S26haldaemon
K75quota_nld   S12rsyslog          S26udev-post
```

Initialization: Last Steps

- After rc has completed, the last script to execute is /etc/rc.d/rc.local
- This is an empty (or near empty) script available for the system administrator to add any operations that the system administrator wants to run at system initialization time
  - e.g., running badblocks, rotating log files, starting servers like Apache or Bind, testing network connectivity, mounting additional file systems, etc
  - Once booted, the system is ready for user login
- As system administrator, you can check on the boot and initialization process
  - dmesg displays the kernel ring buffer (the output as the kernel initializes)
  - the /etc/boot.log file will contain information about system initialization

Services

- A piece of OS code used to handle requests
- Services are divided into different categories
- Services have distinct features from other OS components or servers
  - Run in the background
  - Handle requests that could come in from different types of sources (user, application software, system software, network message, hardware)
  - They are configurable
  - Services can be started or stopped as desired

Services: Categories

- boot
- file system
- hardware
- language support
- logging
- network, web/Internet
- power management
- scheduling
- system maintenance

Services: Notable Ones in CentOS

Name	Type	Description
acpi	power management	laptop battery fan monitor
acpid	event handling	handles acpi events
anacron	scheduling	for scheduling startup tasks at initialization time
apmd	power management	laptop power management
arpwatch	web/Internet	logs remote IP addresses with hostnames
atd	scheduling	executes at jobs based on a scheduled time and batch jobs based CPU load
auditd	logging	the Linux auditing system daemon which logs system, software and user-generated events
autofs	file system	automatically mounts file systems at initialization
bluetooth	hardware	bluetooth service
certmonger	web/Internet	maintain up-to-date security certificates
cpufreq, cpufreqd	hardware	configures and scales CPU frequency to reduce possible CPU overheating

Name	Type	Description
crond	scheduling	the daemon for handling cronttab jobs
cups	hardware	service for printing
cvs	system	managing multi-user documents
dhcpd	web/Internet	configure DHCP access
dnsmasq	web/Internet	starts/stops DNS caching
gpm	hardware	mouse driver
haldaemon	hardware	monitors for new or removed hardware
httpd	web/Internet	the Apache web server
iptables, ip6tables	web/Internet	the Linux firewalls
mdadm	file system	manages software for RAID
named	web/Internet	starts/stops the BIND program (DNS)
netfs	file system	allows remote mounting
netplugd	network	monitors network interface
network	network	starts and stops network access
nfs	file system	enables network file system sharing
nscd	network	password and group lookup service

Name	Type	Description
oddjobd	system	fields requests from software that otherwise do not have access to needed Linux operations
postfix	network	mail service
prelude	network	intrusion detection system service
rdisc	network	discovers routers on local subnet
rsync	file system	allows remote mounting of file systems
smartd	hardware	monitors SMART devices, particularly hard drives
snmpd	network	network management protocol for small networks
sshd	network	service to permit ssh access
syslog	logging	system logging
ypbind	network	name server for NIS/YP networks

Services: a Closer Look

- CentOS 6 has over 60 services (Ubuntu 12 has nearly 80)
- Here we look at a few of the most noteworthy
  - atd – the at daemon is a one-time scheduler
    - it runs processes that were scheduled through either the at or batch commands
    - we examine at and batch in chapter 14
  - crond – daemon for handling cron jobs, which unlike at and batch jobs, are scheduled to recur based on some pattern such as hourly or weekly
    - we examine crontab in chapter 14
  - dnsmasq – a mini-DNS server for Linux
    - dnsmasq performs IP alias → IP address caching
  - logrotate – performs operations on log files including rotating logs files, compressing log files and emailing log files

Services: a Closer Look

- auditd – the Linux auditing system daemon
  - Logs entries based on activities that match rules defined in auditd’s rule file (/etc/sysconfig/audit.rules)
  - Rules use options to specify the type of event and specific criteria as shown in the table below

Syntax	Meaning
-D	Delete any previously defined rules
-b #	# is a number, establish # buffers, e.g., -b 1024
-f #	Set failure flag to # (0 is silent, 1 is print failure messages, 2 is panic or halt the system)
-w directory	Log attempts to access the directory
-w filename	Log attempts to access the file
-w filename –p [rwxa]*	Log attempts to read file (r), write to file (w), execute file (x), or change file’s attributes (a). The * indicates that any combination of the options r, w, x, and a can be listed.
-a action,list –S syscall –F field=value	Log system calls; action is either always or never, list is one of task, entry, exit, user or exclude. The –S option allows you to specify a Linux operation such as chmod, mkdir or mount. The –F option allow you to fine-tune the match by testing some system or user parameters such as EUID

Services: Starting and Stopping

- You can establish which runlevels a service is started or stopped for in three ways
  - By altering the symbolic links in the rc#.d directories (e.g., change S11auditd to K88auditd)
- Using the chkconfig command
  - Without arguments, it lists for all services the runlevels that the service starts and stops in
  - Use arguments as in --level levelnumber service start/stop
  - Use the Service Configuration Manager (see next slide)
    - this GUI tool does not actually allow you to configure a service, just start or stop or change the runlevels that it starts and stops

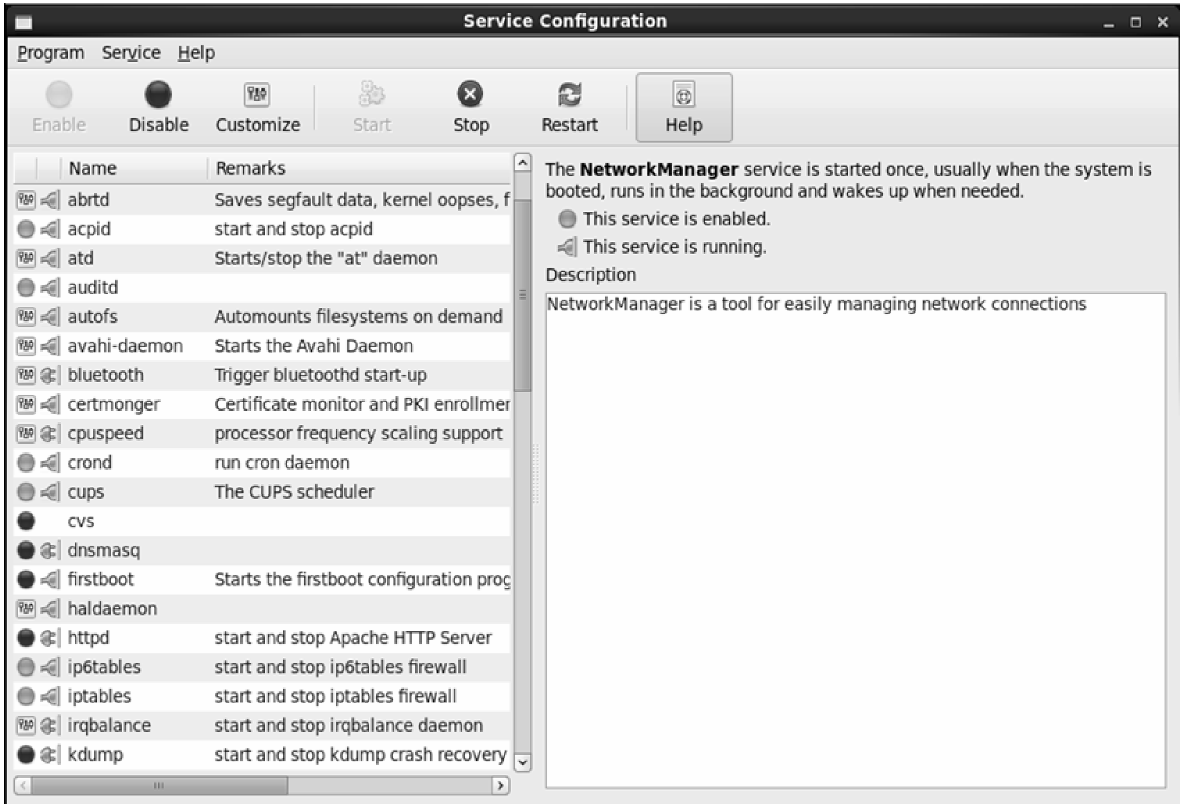
Services: Starting and Stopping

Select a service

Click on Start, Stop, Restart

Click Enable/Disable to indicate that the service should be started or stopped for this runlevel

Select Customize to change start/stop runlevels (only permits runlevels 2-5)



		Name	Remarks
▤	🔈	abrtd	Saves segfault data, kernel oopses, f
○	🔈	acpid	start and stop acpid
▤	🔈	atd	Starts/stop the "at" daemon
○	🔈	auditd	
▤	🔈	autofs	Automounts filesystems on demand
▤	🔈	avahi-daemon	Starts the Avahi Daemon
▤	⚙	bluetooth	Trigger bluetoothd start-up
▤	🔈	certmonger	Certificate monitor and PKI enrollmer
▤	⚙	cpuspeed	processor frequency scaling support
○	🔈	crond	run cron daemon
○	🔈	cups	The CUPS scheduler
●		cvs	
●	⚙	dnsmasq	
●	🔈	firstboot	Starts the firstboot configuration prog
▤	🔈	haldaemon	
●	⚙	httpd	start and stop Apache HTTP Server
○	🔈	ip6tables	start and stop ip6tables firewall
○	🔈	iptables	start and stop iptables firewall
▤	⚙	irqbalance	start and stop irqbalance daemon
●	⚙	kdump	start and stop kdump crash recovery

The **NetworkManager** service is started once, usually when the system is booted, runs in the background and wakes up when needed.

○ This service is enabled.

🔈 This service is running.

Description

NetworkManager is a tool for easily managing network connections

Services: Starting and Stopping

- You can start and stop services from the command line
  - /sbin/service servicename command
    - command is one of start, stop, restart, status
  - Or /etc/init.d/servicename command as in /etc/init.d/auditd start
  - If you are in /etc/init.d, you can also do this as ./auditd start
- The files in /etc/init.d are not the services but are scripts used to start and stop services
  - We explore some portions of the atd script next

Services: the atd Script

```
#!/bin/sh
#
# atd Starts/stop the "at" daemon
#
# chkconfig:   345 95 5
# description: Runs commands scheduled by the "at" command at the time \
#    specified when "at" was run, and runs batch commands when the load \
#    average is low enough.

### BEGIN INIT INFO
# Provides: atd at batch
# Required-Start: $local_fs
# Required-Stop: $local_fs
# Default-Start: 345
# Default-Stop: 95
# Short-Description: Starts/stop the "at" daemon
# Description:      Runs commands scheduled by the "at" command at the time
#    specified when "at" was run, and runs batch commands when the load
#    average is low enough.
### END INIT INFO
```

Services: the atd Script

```
# Source function library.
. /etc/rc.d/init.d/functions

TEXTDOMAIN=initscripts
umask 022
PATH="/sbin:/usr/sbin:/bin:/usr/bin"
export PATH

exec=/usr/sbin/atd
prog="atd"
config=/etc/sysconfig/atd

[ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog
lockfile=/var/lock/subsys/$prog
```

Services: the atd Script

```
start() {
    [ -x $exec ] || exit 5
    [ -f $config ] || exit 6
    echo -n $"Starting $prog: "
    daemon $exec $OPTS
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
}
```

```
stop () {
     echo –n $"Stopping $prog: "
     if [ -n "`pidfileofproc $exec`" ]; then
              killproc $exec
              RETVAL=3
    else
              failure $"Stopping $prog"
    fi
    retval=$?
    echo
    [ $retval –eq 0 ] && rm –f $lockfile
}
```

Services: the atd Script

```
restart() {
        stop
        start
}

reload() {
        restart
}

force_reload() {
        restart
}

rh_status() {
        status $prog
}
```

```
rh_status_q() {
        rh_status >/dev/null 2>&1
}
```

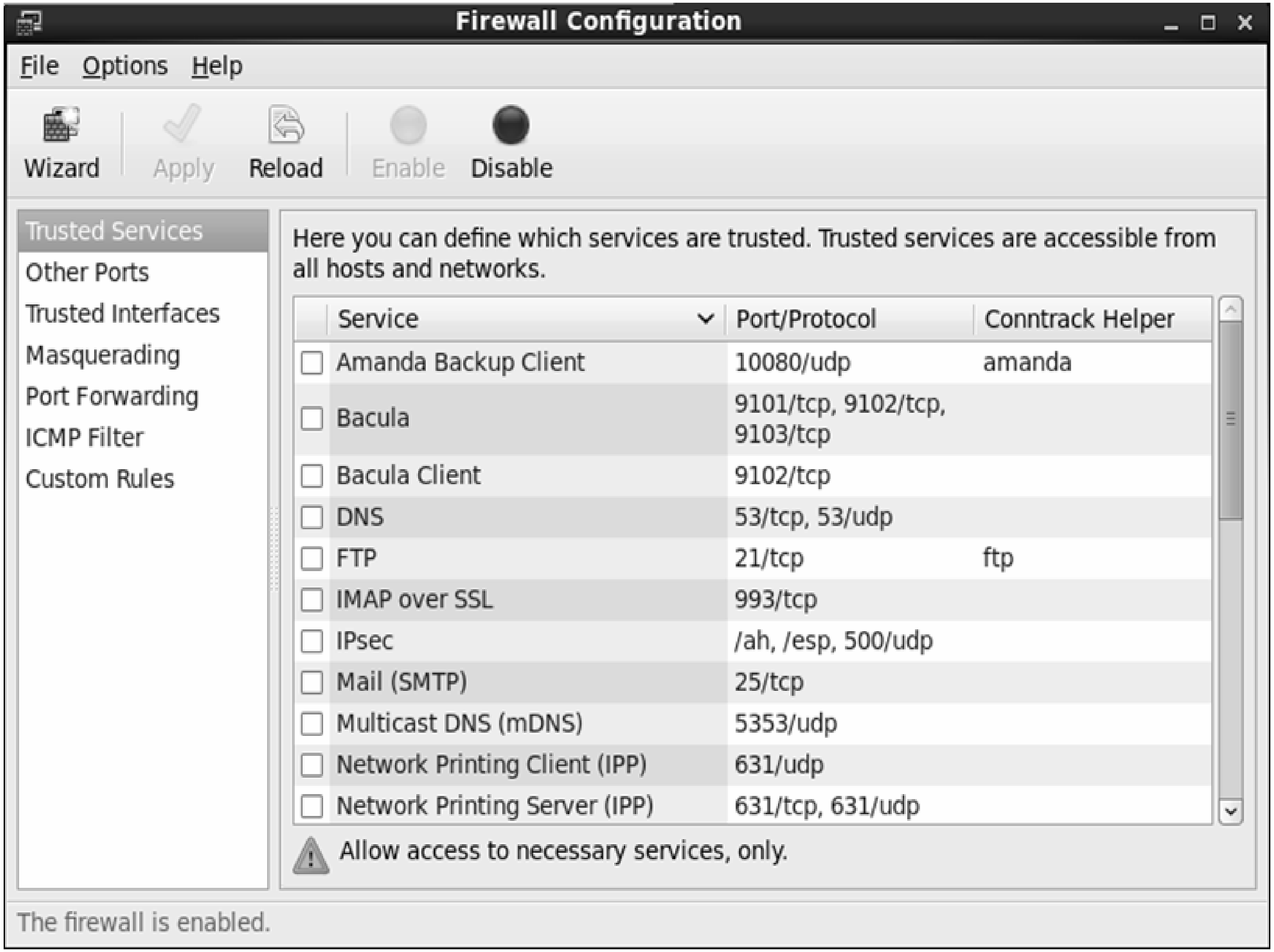
Services: the atd Script

```
case "$1" in
  start)
        rh_status_q || exit 0
        $1
        ;;
  stop)
        rh_status_q || exit 0
        $1
        ;;
  restart)
        $1
        ;;
  reload)
        rh_status_q || exit 7
        $1
        ;;
  force-reload)
        force-reload
        ;;status)
        rh_status
        ;;
  condrestart|try-restart)
        rh_status_q || exit 0
        restart
        ;;
  *)
        echo $"Usage: $0 {start|stop|status|
        restart|condrestart|try-restart|
        reload|force-reload}"
        exit 2
esac
exit $?
```

Services: Configuring Them

- Some services have GUI tools to configure how they operate, we look briefly at the Firewall service (iptables) and kdump
  - Firewall
    - Select wizard to choose between desktop and server configuration (cannot tailor this any more)
    - Or, specify your own trusted services, ports that can be used, trusted interfaces, and custom rules among others
    - Or disable the firewall (not recommended!)
  - Kdump
    - Size of a kernel dump
    - Location to store kernel dump
    - Filtering of what to dump and what actions to perform when the kernel crashes

Services: Configuring Them



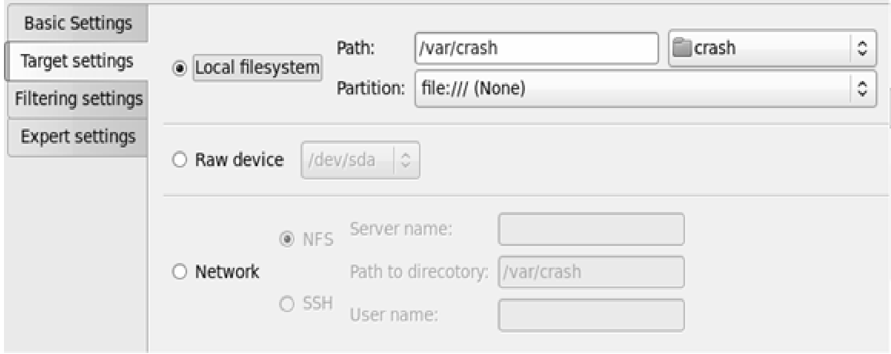
Here you can define which services are trusted. Trusted services are accessible from all hosts and networks.

	Service	Port/Protocol	Conntrack Helper
□	Amanda Backup Client	10080/udp	amanda
□	Bacula	9101/tcp, 9102/tcp, 9103/tcp	
□	Bacula Client	9102/tcp	
□	DNS	53/tcp, 53/udp	
□	FTP	21/tcp	ftp
□	IMAP over SSL	993/tcp	
□	IPsec	/ah, /esp, 500/udp	
□	Mail (SMTP)	25/tcp	
□	Multicast DNS (mDNS)	5353/udp	
□	Network Printing Client (IPP)	631/udp	
□	Network Printing Server (IPP)	631/tcp, 631/udp	

Allow access to necessary services, only.

The firewall is enabled.

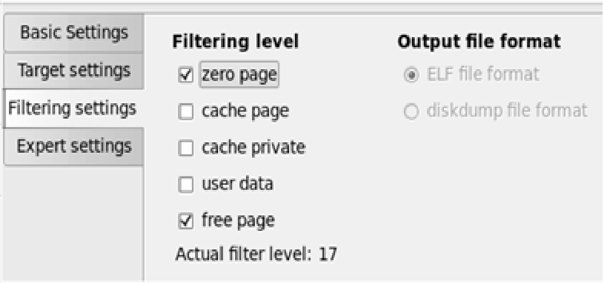
Services: Configuring Them



● Local filesystem — Path: /var/crash — crash — Partition: file:/// (None)

○ Raw device — /dev/sda

○ Network — ● NFS ○ SSH — Server name: — Path to direcotory: /var/crash — User name:



Filtering level

☑ zero page

☐ cache page

☐ cache private

☐ user data

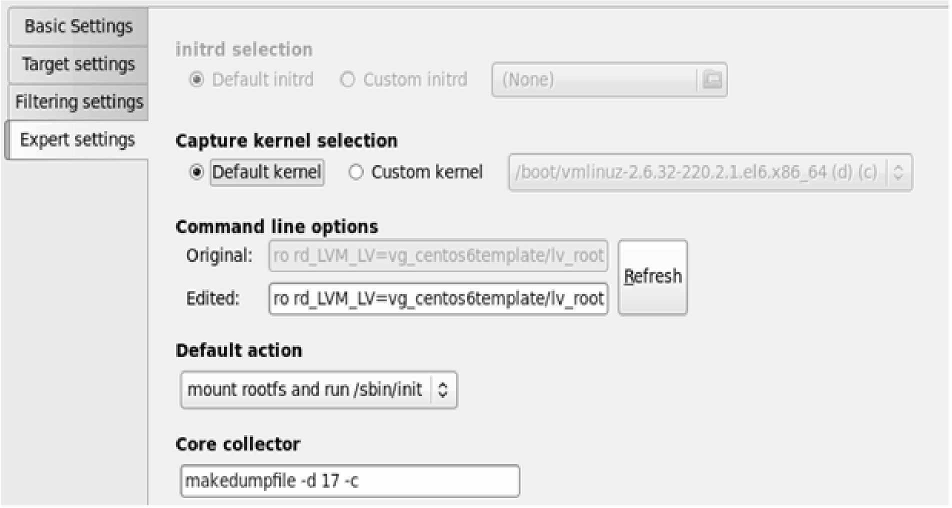
☑ free page

Actual filter level: 17

Output file format

● ELF file format

○ diskdump file format



initrd selection

● Default initrd ○ Custom initrd (None)

Capture kernel selection

● Default kernel ○ Custom kernel /boot/vmlinuz-2.6.32-220.2.1.el6.x86_64 (d) (c)

Command line options

Original: ro rd_LVM_LV=vg_centos6template/lv_root

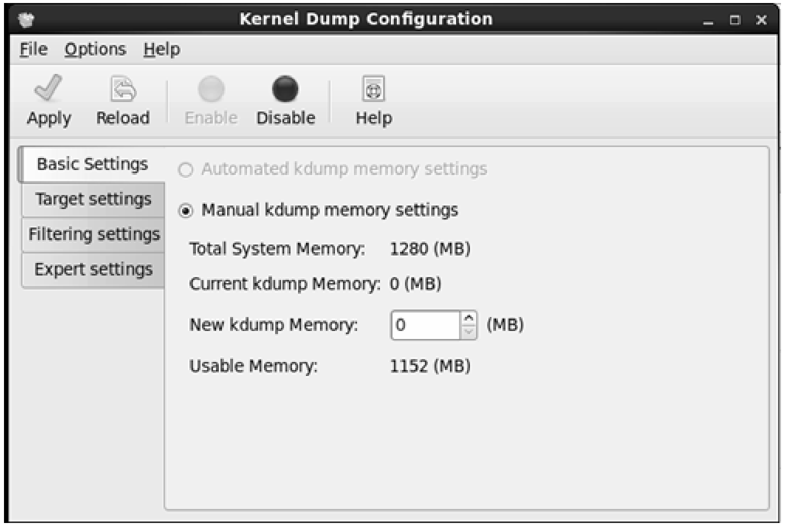
Edited: ro rd_LVM_LV=vg_centos6template/lv_root

Default action

mount rootfs and run /sbin/init

Core collector

makedumpfile -d 17 -c



○ Automated kdump memory settings

● Manual kdump memory settings

Total System Memory: 1280 (MB)

Current kdump Memory: 0 (MB)

New kdump Memory: 0 (MB)

Usable Memory: 1152 (MB)

Services: Configuring Them

- The other, and more common approach to configuring a service is through the service’s configuration file(s)
- Most of these files consist of directives
- Directives might take on several formats such as
  - AUTOCREATE_SERVER_KEYS=YES
  - path /var/crash
  - -A INPUT –s 10.11.12.13 –j ACCEPT
- Once you have altered the configuration file, you must save the file and restart the service for the new configuration to take effect

Services: Configuring Them

- The syslogd daemon logs system and kernel messages to a log file
- Entries in the configuration file, /etc/syslog.conf, denote
  - source.priority   action
  - where source is the type of program whose actions we want to log and priority is the level of action that we want to log
  - action is either the location of the log file or * to indicate that the message should be sent to all active consoles

Services: Configuring Them

- You might find the following entries in your syslog.conf file
  - #kern.*   /dev/console
    - commented out, ignore
  - *.info;mail.none;authpriv.none;cron.none /var/log/messages
    - any informational message, and messages of priority none from these other sources are sent to /var/log/messages
  - authpriv.*   /var/log/secure
    - any other authpriv (authentication) message is sent to /var/log/secure
  - mail.*   -/var/log/maillog
    - the – indicates an asynchronous file so that entries do not have to be written in the order recevied
  - cron.*   /var/log/cron
  - *.emerg   *
    - All emergency messages are sent to console

Services: Configuring Them

Priority level	Meaning
none	No priority.
debug	Log debugging messages; used by programmers and software testers.
info	Log informational messages generated by the program to specify what it is doing.
notice	Log events worth noting such as opening files, writing to disk, mounting attempts.
warning	Log detected potential problems.
err	Log errors that arise that do not cause the program to terminate.
crit	Log errors that arise that will cause the program to terminate.
alert	Log errors that not only cause the program to terminate but may also cause problems with other running programs.
emerg	Log errors that could cause the entire OS to crash.

Priority levels for syslog, you find similar priority levels used in logging for other software like Apache