System Admin

Class 4 - System Administration
by Ian Robert Blair, M.Sc.

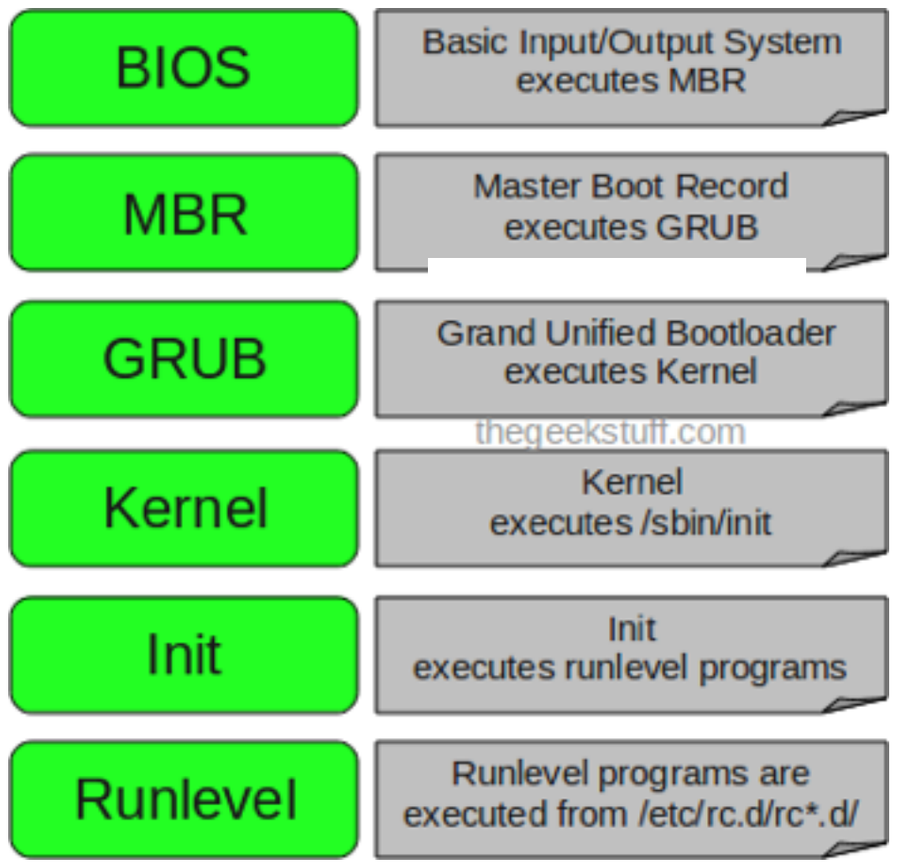
Agenda

- SysVinit and Systemd
- Services
- SELinux
- Software Installation
- Cron/AT Scheduler
- Logs

Booting and Service Management

Boot Process

- After testing the hardware, the **BIOS** transfers control to the MBR, which usually passes control to the boot loader on the partition boot record
- The boot loader loads the kernel into memory and starts it running
- The boot loader frequently resides on the starting sectors of a hard disk called the **MBR (master boot record)**
- The **grub-install** utility installs the MBR and the files that GRUB needs to boot the system
  - grub-install /dev/sda
- The configuration file for grub is /etc/grub.conf



The init Daemon

- The **init** daemon is the system and service manager for Linux
- It is the first process Linux starts when it boots, has a **PID 1,** and is the parent of all processes
- The first Linux init daemon was based on the UNIX System V init daemon and is referred to as **SysVinit**
- SysVinit does not deal well with modern hardware, including (i.e. hotplug devices) Fedora/RHEL replaced it with the Systemd (Ubuntu, Debian) init daemon

Run Levels

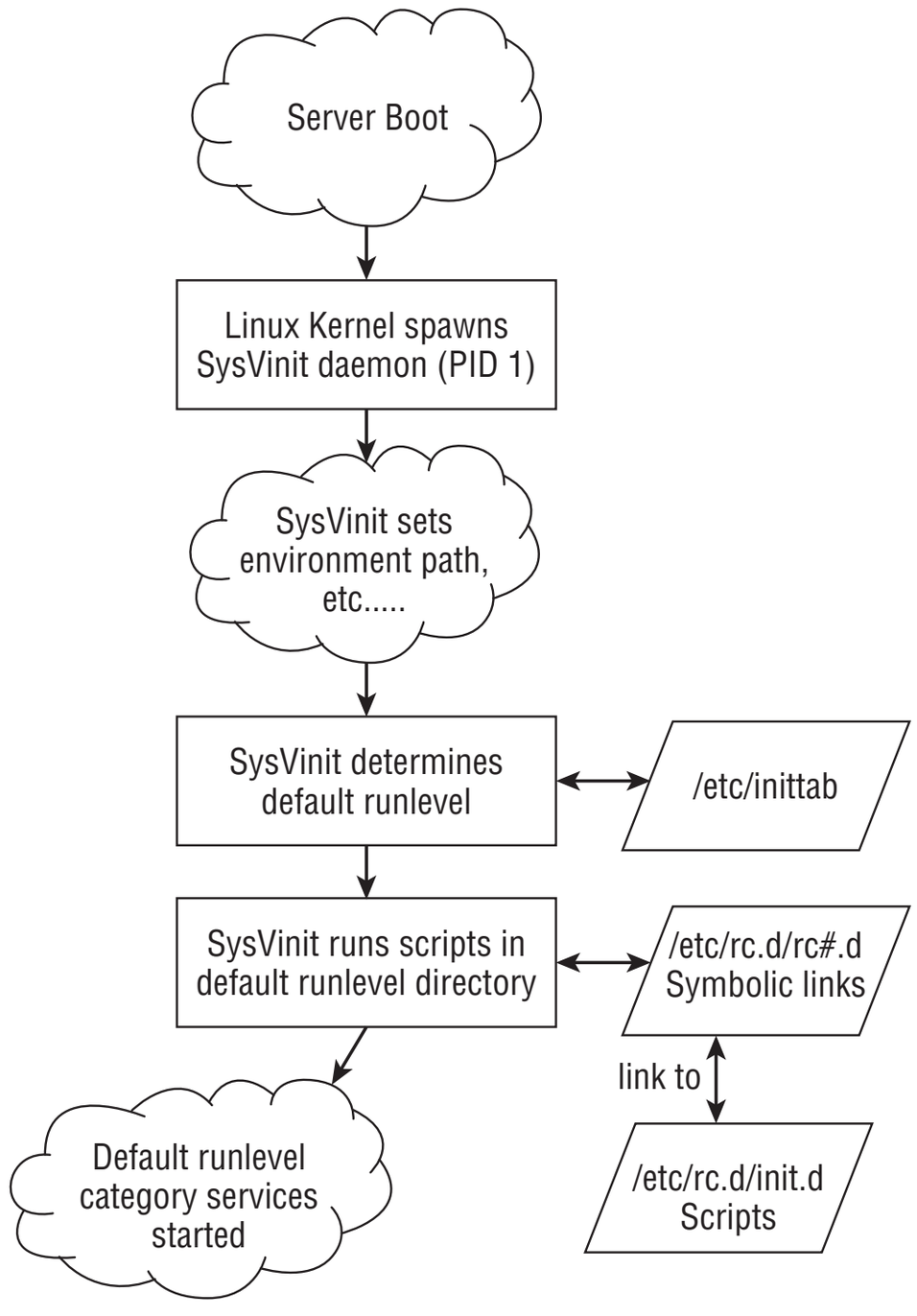
SysVinit runlevel	systemd runlevel (target unit)	Name/function
0	runlevel0.target, poweroff.target	Shuts down the system
1	runlevel1.target, rescue.target	Brings the system to single-user/ rescue mode
2, 3, 4	runlevel2.target, runlevel3.target, runlevel4.target, multi-user.target	Brings the system to multiuser textual mode
5	runlevel5.target, graphical.target	Brings the system to multiuser graphical mode
6	runlevel6.target, reboot.target	Reboots the system
	emergency.target	Emergency shell

Service Control

- Two Systems…

SysVinit 1

- The **SysVinit** daemon is driven by init (initialization scripts) called rc (run command) scripts, are shell scripts located in the **/etc/rc.d/init.d** directory
- The scripts are run via symbolic links in the **/etc/rc.d/rc*n*.d** directories, where *n* is the runlevel the system is entering
- The /etc/rc.d/rc*n*.d directories contain scripts whose names begin with K and scripts whose names begin with S

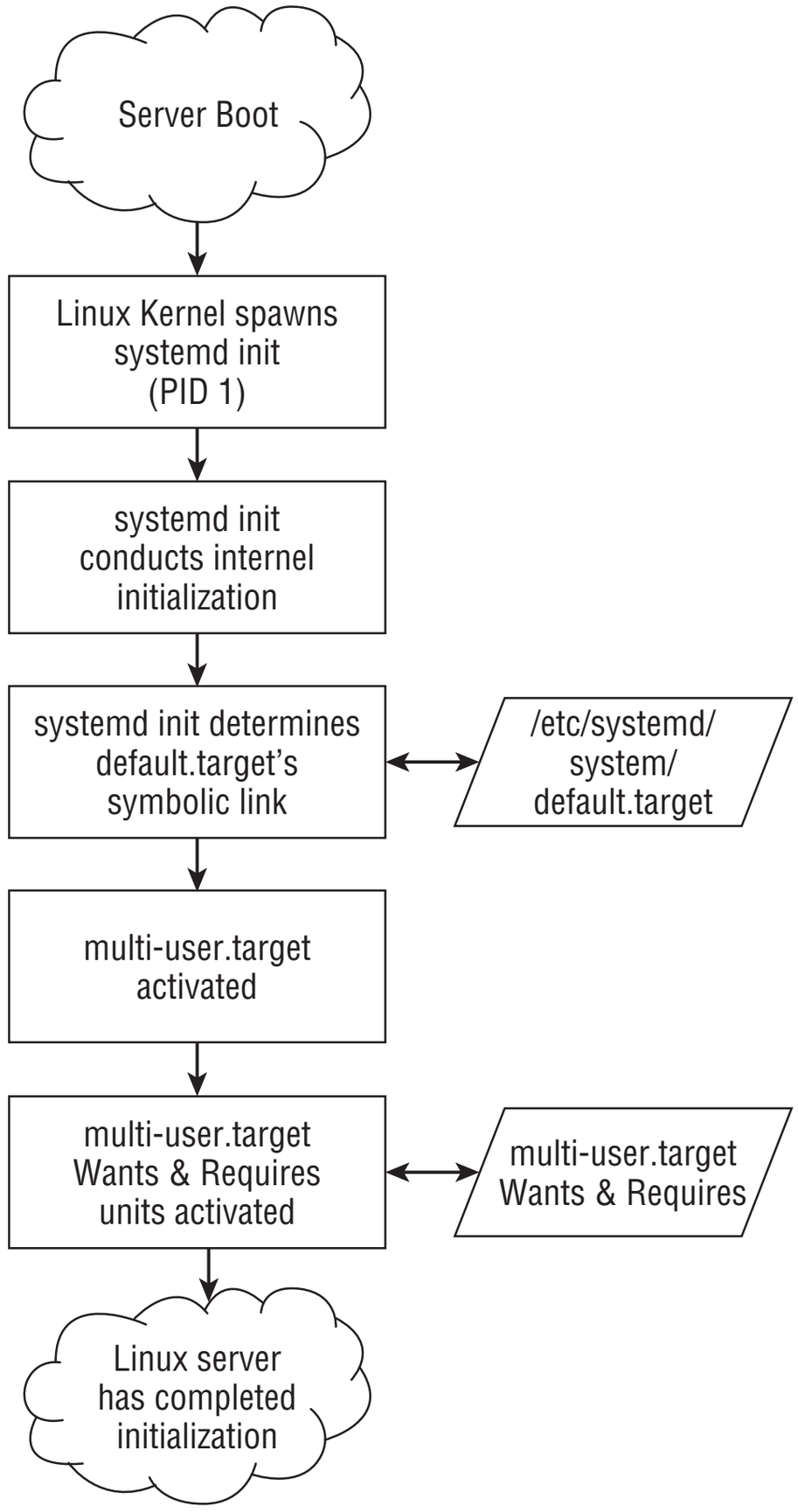


SysVinit 2

- When entering a new **runlevel**, each K (kill) script is executed with an argument of stop, and then each S (start) script is executed with an argument of start
- Each of the K and S file is run in numerical order
- Put commands that customize the system in **/etc/rc.d/rc.local**, it is executed after all other rc scripts
- The **service** utility can report on or change the status of any of the jobs in /etc/init and any of the system services in /etc/rc.d/init.d
  - service nfs stop
- The command service —status-all displays the status of all system services controlled by SysVinit

Systemd Init

- Init has been replaced by a link to link to **systemd**
- The **systemd** init daemon is based on the concept of **units**, each of which has a name and type
- Typically information about a unit is stored in a file that has the same name as the unit (e.g., dbus.service)
- A **service unit** refers to a **daemon (service)** that systemd controls, including those controlled natively by systemd and those controlled by systemd via SysVinit scripts
- A **target unit** groups other units (ie. graphical.target, multi-user.target)



Systemd Init 2

- The services wanted by a runlevel target appear in directories named *.wants under the **/etc/systemd/system** directory
- All service unit files are kept in the **/lib/systemd/system** directory (linked by /etc/system/system)
- If you want to **customize** service files put them in the **/etc/systemd/system** hierarchy, because they take precedence and won't be overwritten by future updates
- **Systemd** is backward compatible with SysVinit and Upstart

Setting Run Level Fedora

- The **/etc/systemd/system/default.target** file is a link to the file that specifies the target the system will boot to by default
- To set run level:
  - systemctl set-default multi-user.target (or graphical.target)
  - systemctl get-default

Service Control

- Persistent service control for systemd and sysvinit (backward compatible)
  - systemctl disable(or enable) ntpd.service
- Current service control for systemd and sysvint
  - systemctl start (or status/retstart/stop) ntpd.service
- The **system-config-services** is a GUI utility that displays the Service Configuration window (RHEL)

Run Levels 2

- The **runlevel** utility displays the previous and current runlevels
- The **telinit** utility allows a user working with root privileges to bring the system down, reboot the system, or change between runlevels
  - telinit 1

Rescue Mode: Update

- In **single-user/rescue mode** init displays the root prompt
- You can perform system maintenance, filesystems can be unmounted
- No one except you will be using it, so no user programs will interfere with disk maintenance
- Give the command to bring the system down to single-user mode
  - systemctl isolate rescue.target
  - telinit 1

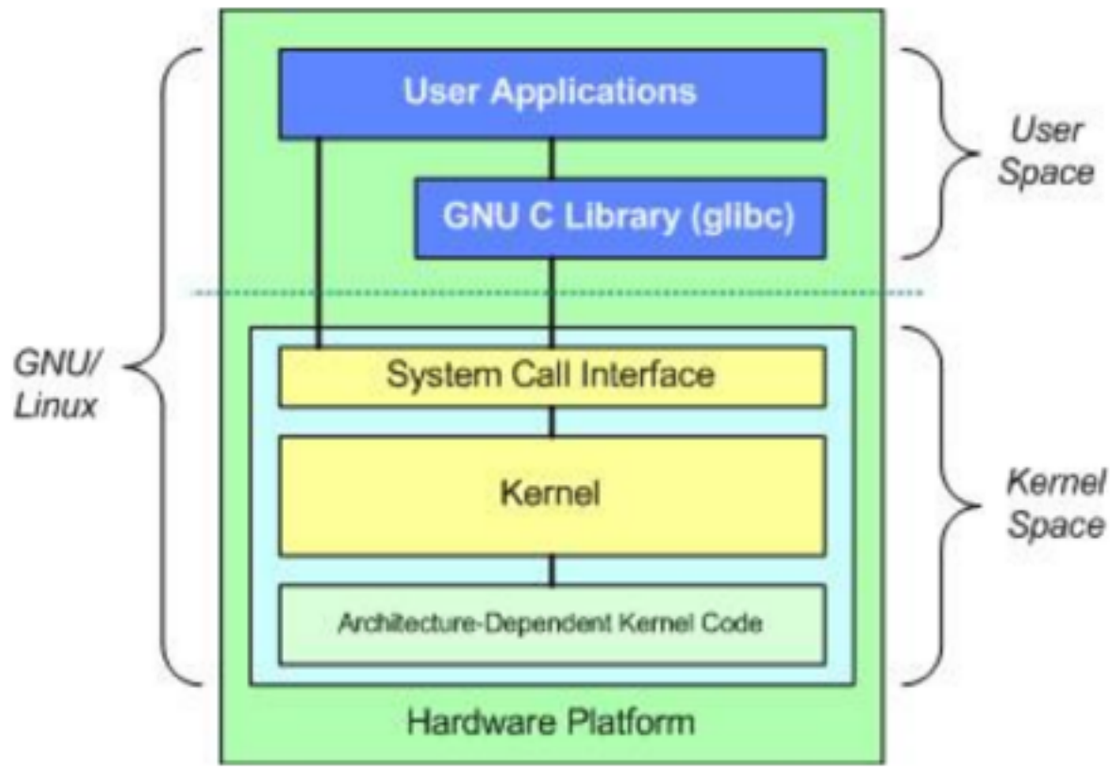
Recover Root Password

- Reboot, press “e” to edit
- At the end of the “linux” line, add rd.break (or init=/bin/sh if not on a VM)
- Press “control+x”, to continue boot
- Remount the file system as writable, run “mount -o remount,rw /sysroot”
- chroot /sysroot
- echo “*PaSsw0rd!~*” | passwd —stdin root
- Fix SELinux context, “touch /.autorelabel”
- exec /sbin/reboot
- Enter “control+d”, then “reboot”

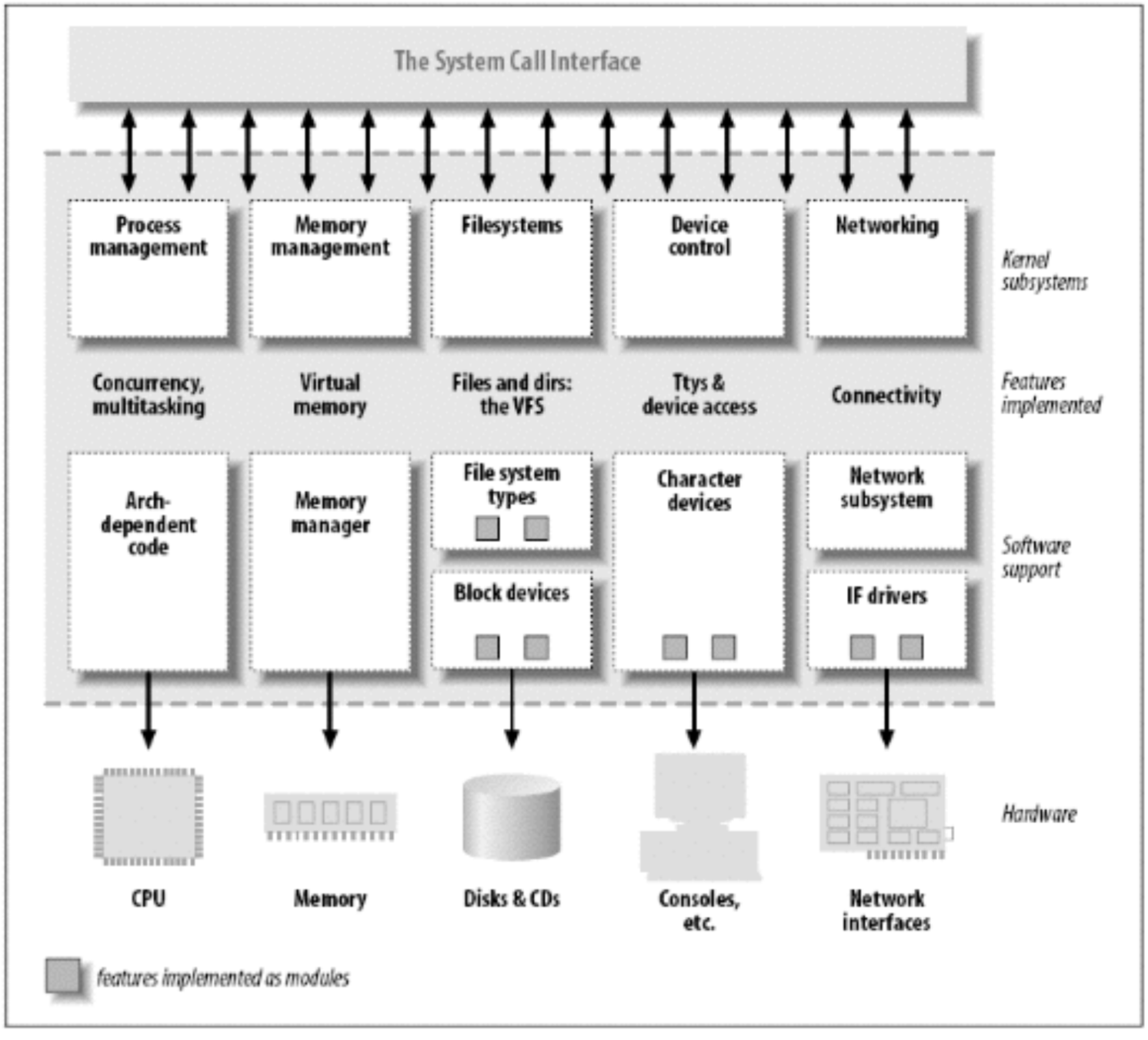
Kernel Building

What is the Kernel?

- The **kernel** is a computer program that manages hardware
- It is usually loaded into a protected area of memory (kernel space)
- Processes (user space) request resources by making system calls to the kernel
- The kernel:
  - Controls access to the system CPU via scheduling algorithms
  - Allocates memory for storing instructions and data
  - Controls input/output to and from devices such as keyboards, mice, disk drives, printers, network adapters, and display devices



Linux Kernel Architecture



Kernel Building

- Smaller and faster
- Customized (building the feature into the kernel or by specifying the feature as a loadable kernel module)
- The goal is to make the kernel as small as possible while minimizing how often modules have to be loaded
- However, compiling and maintaining a custom kernel takes a lot of time and work

Sysctl

- The **sysctl** utility can modify kernel parameters while the system is running
- This utility takes advantage of the facilities of **/proc/sys**, which defines the parameters that sysctl can modify
- The command **sysctl –a** displays a complete list of sysctl parameters
  - sysctl kernel.domainname
  - su -c 'sysctl -w kernel.domainname="example.com"'

Kernel Build 1

- Install the ELRepo
- Update current kernel version
  - yum --enablerepo=elrepo-kernel install kernel-ml
- Install dependencies
  - yum install gcc ncurses ncurses-devel (openssl-devel)
- Download kernel (www.kernel.org)
- Extract package
  - tar -xf linux-3.18.4.tar.xz -C /usr/src/
- Configure
  - make menuconfig (make oldconfig, for prev. build)

Kernel Build 2

- Build
  - make
- Install
  - make modules_install install
- Check
  - uname -r

Kernel Source Tree

Directory	Description
arch	Architecture-specific source
crypto	Crypto API
Documentation	Kernel source documentation
drivers	Device drivers
fs	The VFS and the individual file systems
include	Kernel headers
init	Kernel boot and initialization
ipc	Interprocess communication code
kernel	Core subsystems, such as the scheduler
lib	Helper routines
mm	Memory management subsystem and the VM
net	Networking subsystem
scripts	Scripts used to build the kernel
security	Linux Security Module
sound	Sound subsystem
usr	Early user-space code (called initramfs)

Kernel Modules

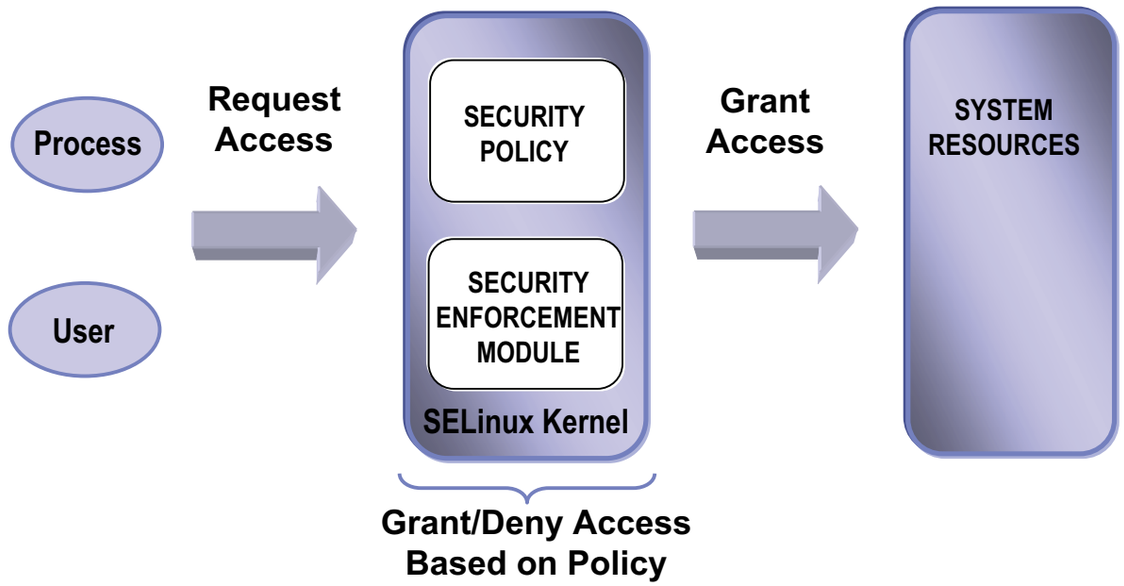
- A loadable **kernel module** is an object file—part of the kernel—that is linked into the kernel at runtime

- This ability gives the kernel the flexibility to be as small as possible at any given time

Tool	Function
depmod	Works with dependencies for modules
insmod	Loads modules in a running kernel
lsmod	Lists information about all loaded modules
modinfo	Lists information about a module
modprobe	Loads, unloads, and reports on modules (including dependencies)
rmmod	Unloads modules from a running kernel

SELinux 1

- Traditional Linux security, called **DAC (Discretionary Access Control)** owner controls access
- **SELinux (Security-Enhanced Linux)** implements **MAC (Mandatory Access Control)**
- It defines a security policy that controls some or all objects, such as files, devices, sockets, and ports, and some or all subjects, such as processes



The kernel checks MAC rules after it checks DAC rules; either can deny access

SELinux 2

- SELinux can be in one of three states (modes):
    - Enforcing/Active, Permissive/Warn, or Disabled
- SELinux implements one of the following policies:
    - Targeted, MLS, or Strict

SELinux 3

- The **/etc/selinux/config** file, which has a link at /etc/sysconfig/selinux, controls the state of SELinux on the local system or **system-config-selinux** (GUI)
    - getenforce
    - setenforce permissive
    - sestatus

SELinux 4

- For a list of Booleans, an explanation of what each one is, and whether they are on or off, run the ***semanage boolean -l** command as the Linux root user (install policycoreutils-python)
- The **getsebool -a** command lists Booleans, whether they are on or off, but does not give a description of each one
- The **setsebool boolean-name x** command turns Booleans on or off, where boolean-name is a Boolean name, and x is either on or off
- To make changes persistent across reboots, run the **setsebool -P boolean-name on** command as the Linux root user

PAM

- **PAM (Linux-PAM, or Linux Pluggable Authentication Modules)** allows a system administrator to determine how applications use authentication to verify the identity of a user
- Instead of building the authentication code into each application, PAM provides shared libraries that keep the authentication code separate from the application code
- PAM warns you about errors it encounters, logging them to **/var/log/messages** or **/var/log/secure**

Software Maintenance

Software Installation I

- A **software package**, or simply **package**, is the collection of scripts, programs, files, and directories required to install and run an application, utility, server, or system software
- A package also includes a list of other packages the package depends on (dependencies)
- Using software packages makes it easier to install, update, and uninstall software
- Fedora/RHEL and SUSE use **RPM** (Red Hat Package Management System) and Ubuntu and Debian use **dpkg** (Debian Package Management System)

Software Installation: YUM 1

- The **yum** utility downloads package headers and packages from servers, called repositories, that can reside on the Internet, a CD, a DVD, or the local network
  - yum install tcsh
  - yum remove tcsh
  - yum info tcsh
  - yum whatprovides "*X11/Xlib.h"
  - yum search vim (searches description, summary, and name fields)

Software Installation: YUM 2

- yum list '*emacs*' (searches for a package name field)
- yum update [package]
- yum grouplist
- yumdownloader samba

Software Installation: YUM 3

- The primary configuration file, **/etc/yum.conf**, holds global settings
- Files in the **/etc/yum.repos.d** directory define repositories

Software Installation: RPM I

- **RPM** works only with software packages built for processing by RPM
- Used install, uninstall, upgrade, query, and verify RPM packages (.rpm)
- RPM uses the local RPM installation database to track the locations where software packages are installed, the versions of the installed packages, and the dependencies between the packages
- RPM uses the Berkeley Database (Berkeley DB or BDB) and stores the database files in **/var/lib/rpm**
- RPM forms the basis for yum: yum finds packages, downloads them from repositories, and then calls RPM to install/upgrade/remove the packages

Software Installation: RPM 2

- **rpm –q** option queries the package database, **–a** option specifies all packages

- **–ql** options to list the files in a package.

- **–qi** options displays information about a package:

- **–qf** options to determine which package a file belongs to; it works with installed packages only (yum whatprovides searches repositories)

Software Installation: RPM 3

- The **–U** option upgrades existing packages and installs new packages

- Add the **–v** (verbose) option to display more information about what is happening and the **–h** (or —hash) option to display hash marks as the package is unpacked and installed

    - rpm -Uvh package_name.rpm

- **-e** Uninstalls a package

    - rpm -e package_name

Software Installation: Source

- The GNU Configure and Build System makes it easy to build a program that is distributed as source code (see www.gnu.org/software/autoconf)

- This process requires a shell, **make** (make package), and **gcc** (the GNU C compiler; gcc package)
  - yum groupinstall “Development Tools”

- First unpack and decompress the file and cd to the new directory

- After reading the README and INSTALL files, run the **configure** script, which gathers information about the local system and generates the Makefile file

- Next, run **make**

- If you want to install it, run **“make install”** while running with root privileges

Source Code Security

- To check whether software has been tampered with check it's hash using MD5 or SHA

  sha256sum CentOS-6.5-x86_64-minimal.iso

  f9d84907d77df62017944cb23cab66305e94ee6ae6c1126415b81cc5e999bdd0 CentOS-6.5-x86_64-minimal.iso

- It should match the sum on the developer's website

Patch

- The **patch** program is used to apply changes to text files (usually source code)
- To create a patch file:
  - diff -Naur old_file new_file > diff_file
- To apply a patch to a file (within the same directory as the new file):
  - patch < diff_file
  - patch -p1 < diff_file

Scheduling Tasks

CRON 1

- The **crond** daemon executes scheduled commands periodically (for systems the are always running, ie. servers)
- System crontab files are kept in the **/etc/cron.d** directory and in **/etc/crontab**
- Format: minute, hour, day-of-month, month, day-of-week, user, command

```
20 1 * * *     root    /usr/local/bin/checkit
25 9 17 * *    root    /usr/local/bin/monthly.check
40 23 * * 7    root    /usr/local/bin/sunday.check
10/* * * * *   root    /usr/local/bin/tenmin.check
```

ANACRON

- The **anacron** utility executes scheduled commands when it is called (for systems the aren't always running, ie. laptops)
- The **anacron** utility keeps track of the last time it ran each of its jobs so when it is called, it can tell which jobs need to be run
- When anacron is run, it reads the commands it is to execute from the **/etc/anacrontab file.**
- Format: period delay jobID command

```
#period in days   delay in minutes   job-identifier   command
1       5       cron.daily              nice run-parts /etc/cron.daily
7       25      cron.weekly             nice run-parts /etc/cron.weekly
@monthly 45     cron.monthly           nice run-parts /etc/cron.monthly
```

User CRON

- Users can use the crontab utility to set up personal crontab files in **/var/spool/cron**
- A user **crontab** file (same format) runs as the user who is running it
- Users can work with their own crontab files by giving the command **crontab** followed by **–l** to list the file, **–r** to remove the file, or **–e** to edit the file

AT

- Like the crond daemon, at runs a job sometime in the future
- Unlike crond, at runs a job only once
- For instance, you can schedule an at job that will reboot the system at 3:00 AM
- at 3am
  at> reboot
  at> <EOT>
  job 1 at Wed Jan 26 03:00:00 2011

Performance

Performance Tools 1

- The **top** utility is a useful supplement to ps. At its simplest, top displays system information followed by the most CPU-intensive processes
  - The top utility updates itself periodically; type q to quit
  - h (help), k (kill), s (seconds), u (user)
- **free** [-m]— memory analysis
- **vmstat** - virtual memory analysis

Performance Tools 2

- **sar** - system activity report for system monitoring and logging
  - Creates cron jobs that monitor systems daily
  - sar -r 1 3 (memory, 1 sec., x3)
  - sar -n DEV 1 3 (Network interfaces)
  - sar -P 0 1 3 (CPU core 0)
- **time** - used to time scripts and commands

nice/renice

- Run a program with modified scheduling priority
- **renice** allows you to change the priority of a command executed with nice
- ADJUST is 0 by default. Range goes from -20 (highest priority) to 19 (lowest)
- Examples:
  - nice -n 5 perl test.pl
  - renice -n -19 -p 3534

Limits

- Allows you to control CPU usage, maximum file size, job priority, number of logins for a user or group

- /etc/security/limits.conf

Logs

Logs 1

- The **logrotate** utility manages system log files automatically by rotating, compressing, mailing, and removing each file as you specify

- The **logrotate** utility is controlled by the **/etc/logrotate.conf**

- Typically **logrotate.conf** has an include statement that points to utility-specific specification files in **/etc/logrotate.d**

Logs 2

- **Rsyslogd** daemon listens for log messages and stores them in the **/var/log** hierarchy

- In addition to providing logging facilities, rsyslogd allows a single machine to serve as a log repository for a network and allows arbitrary programs to process specific log messages

- **The /etc/rsyslog.conf** file stores configuration information for rsyslogd

Logs 3

- A selector is split into two parts, a **facility** and a **priority,** which are separated by a period

- Facilities: auth, authpriv, cron, daemon, kern, lpr, mail, news, user, uucp

- Priorities: debug, info, notice, warning, err, crit, alert, emerg

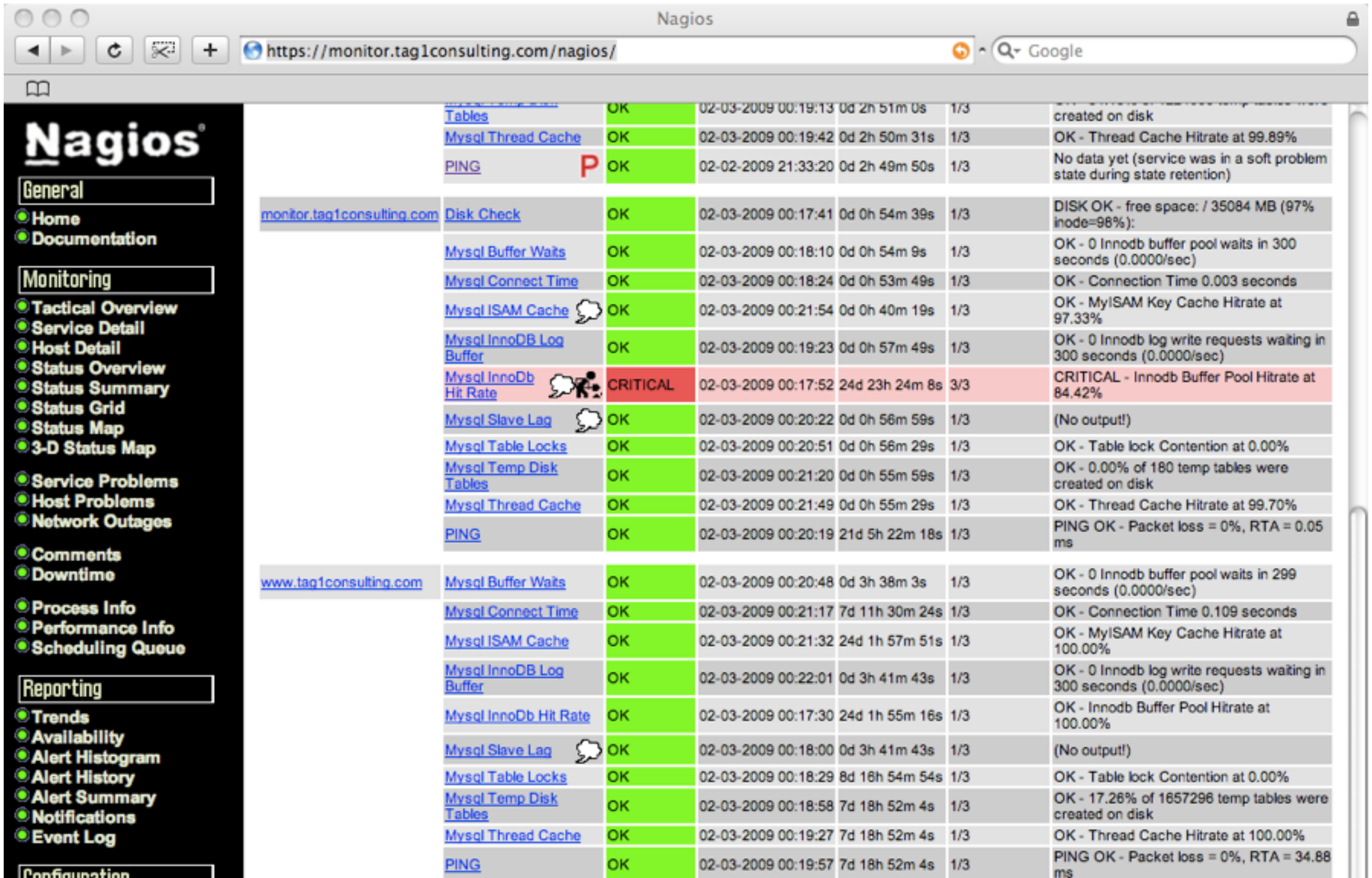
- A selector consisting of a single facility and priority, such as kern.info, causes the corresponding action to be applied to every message from that facility with that priority or **higher (more urgent)**

Logs 4

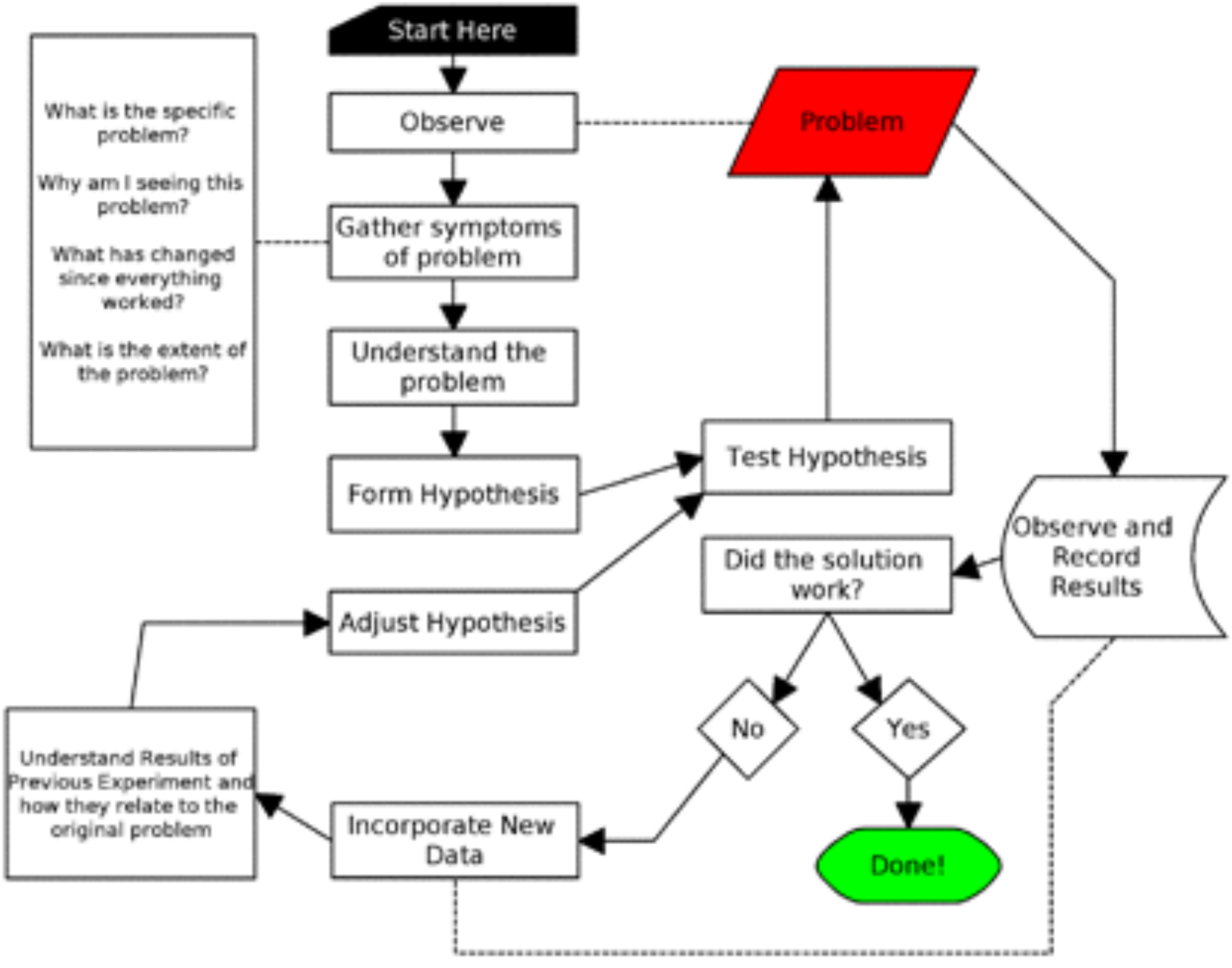
- To forward messages to rsyslogd on a remote system, specify the name or IP address of the system preceded by @ (sends messages to UDP port 514) or @@ (sends messages to TCP port 514)
    - ex. kern.crit @@plum
    - On the remote system edit **/etc/rsyslog.conf**

Troubleshooting

Monitoring: Nagios



Troubleshooting



Troubleshooting Tips, pt. 1

- Check the log files in **/var/log**
- The **/var/log/messages** file accumulates system errors, messages from daemon processes, and other important information
- Determine whether only that one user or others
- Check the user's entry in **/etc/passwd**, home directory, and startup files (.profile, .login, .bashrc, and so on)

Troubleshooting Tips, pt. 2

- Check cables, network connectivity, and DNS is working properly
- Use **df** to check for full filesystems or if file systems are mounted
- Restart the service (i.e. systemctl restart httpd.service)
- Always log hardware changes, system software updates, hardware malfunctions, and user complaints help you find and fix system problems

dmesg

- The **dmesg** utility displays the kernel-ring buffer, where the kernel stores messages
- Messages in the kernel-ring buffer are often useful for diagnosing system problems
  - dmesg | grep DMA

Lab Assignments

- Change the system to boot to run level 3 as default
- Disable SELinux, Re-enable it
- Use YUM to update system and install software
- Install software using Source code (lynx browser)

Homework

- Read 10, 15, 17, 18
- Page 498, Exercises 1, 4, 6
- Page 557, Exercises 2, 4
- Page 643, Exercises 2, 4

Contact

Ian Robert Blair, MSc.

ian.robertblair@icloud.com

QQ: 2302412574