3. Host Management

IT5406 - Systems and Network Administration

Level III - Semester 5

Overview

In this topic we discuss about the essential topics related to host management such as user management, process management, access control, software installation management, logging, storage, and file system management.

Intended Learning Outcomes

At the end of this lesson, you will be able to;

- Install software into a Linux distribution using its software package management system.
- Implement and demonstrate user management in a Linux distribution.
- Explain different storage technologies.
- Implement and demonstrate user access control.
- Implement and demonstrate file system access control.
- Implement and demonstrate process management in a Linux distribution.

List of sub topics

3.1 Access Control and Rootly Powers

3.2 Process Control

3.3 File System

3.4 Software Installation and Management

3.5 User Management

3.6 Logging

3.7 Storage

3.1 Access Control and Rootly Powers

3.1.1. Standard UNIX Access Control

- Basic rules for access control schema:
  - Access control decisions depend on which user is attempting to perform an operation, or in some cases, on that user’s membership in a UNIX group.
  - Objects (e.g., files and processes) have owners. Owners have broad (but not necessarily unrestricted) control over their objects.
  - You own the objects you create.
  - The special user account called “root” can act as the owner of any object.
  - Only root can perform certain sensitive administrative operations.

3.1 Access Control and Rootly Powers

3.1.1. Standard UNIX Access Control

- File system access control:
  - The kernel and the filesystem are intimately intertwined.
  - For example, you control and communicate with most devices through files that represent them in **/dev**
  - Since device files are filesystem objects, they are subject to filesystem access control semantics.
  - The kernel uses that fact as its primary form of access control for devices.
  - In the standard model, every file has both an owner and a group where the owner can set permission of the file.
  - Both the kernel and the filesystem track owners and groups as numbers rather than as text names.
    - UID – User Identification Number
    - GID – Group Identification Number

3.1 Access Control and Rootly Powers

3.1.1. Standard UNIX Access Control

- Process ownership:
  - The owner of a process can send the process signals and can also reduce (degrade) the process's scheduling priority.
  - Processes actually have multiple identities associated with them: a real, effective, and saved UID; a real, effective, and saved GID.

3.1 Access Control and Rootly Powers

3.1.1. Standard UNIX Access Control

- The root account:
  - The root account is UNIX’s omnipotent administrative user
  - Some examples of restricted operations that can be carried out by a root user are:

    Creating device files
    Setting the system clock
    Raising resource usage limits and process priorities
    Setting the system’s hostname
    Configuring network interfaces
    Opening privileged network ports (those numbered below 1,024)
    Shutting down the system

3.1 Access Control and Rootly Powers

3.1.1. Standard UNIX Access Control

- SetUID and SetGID execution:
  - Identity substitution system that's implemented by the kernel and the filesystem in collaboration
  - This scheme allows specially marked executable files to run with elevated permission
  - When the kernel runs an executable file that has its "**setuid**" or "**setgid**" permission bits set, it changes the effective UID or GID of the resulting process to the UID or GID of the file containing the program image rather than the UID and GID of the user that ran the command.
    - Example: a user being able to change password
  - You can disable **setuid** and **setgid** execution on individual filesystems by specifying the **nosuid** option to mount.

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- root account login:
  - root login must not be allowed to directly log in.
  - root access is required for system administration, and it’s also a pivot point for system security.

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- su: substitute user identity:
  - A marginally better way to access root account
  - How to use **su** command?
  - How to exit **su** environment?
  - The exact implications of login mode vary by shell, but login mode normally changes the number or identity of the files that the shell reads when it starts up. For example, bash reads **~/.bash_profile** in login mode and **~/.bashrc** in nonlogin mode.

3.1 Access Control and Rootly Powers

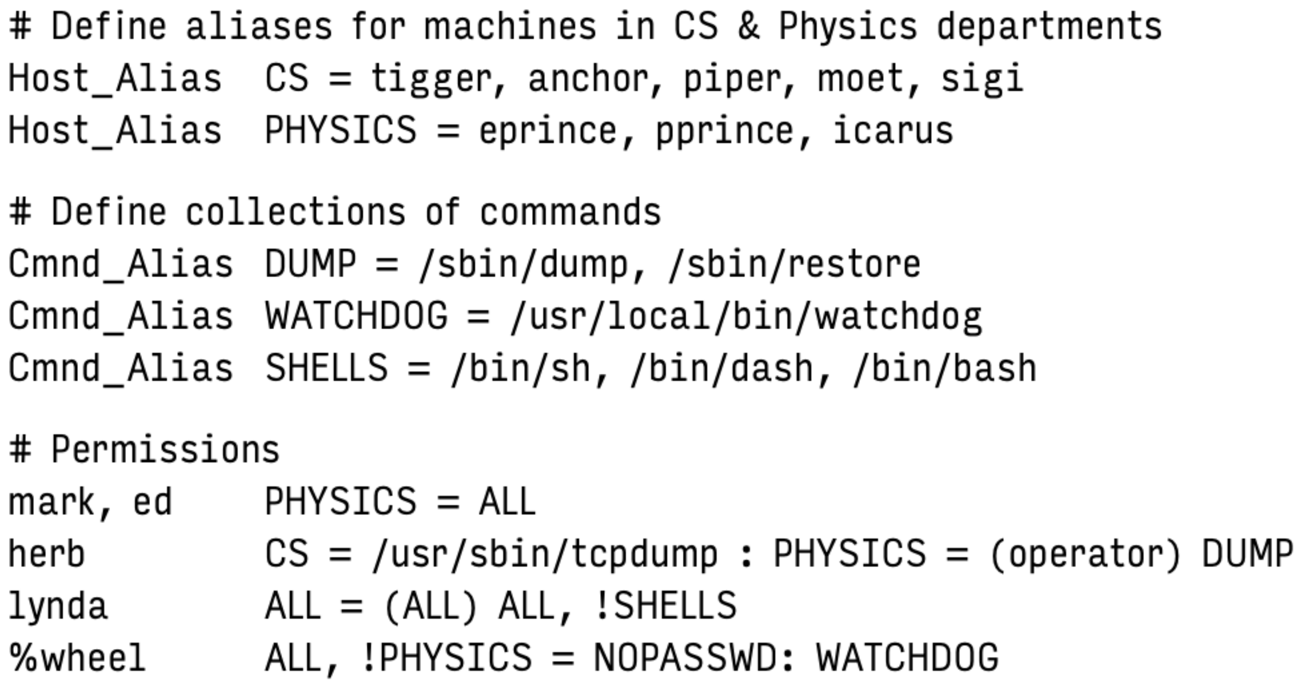
3.1.2. Management of Root Account

- sudo: limited su:
  - If the root account is used by several administrators, you really have only a vague idea of who's using it or what they've done
  - **sudo** provide a solution to this problem.
  - **sudo** consults the file **/etc/sudoers**, which lists the people who are authorized to use **sudo** and the commands they are allowed to run on each host.
  - **sudo** commands can be executed without having to type a password until a five-minute period (configurable)
  - sudo keeps a log of the command lines that were executed, the hosts on which they were run, the people who ran them, the directories from which they were run, and the times at which they were invoked.

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- <u>sudo: example configuration:</u>



```
# Define aliases for machines in CS & Physics departments
Host_Alias  CS = tigger, anchor, piper, moet, sigi
Host_Alias  PHYSICS = eprince, pprince, icarus

# Define collections of commands
Cmnd_Alias  DUMP = /sbin/dump, /sbin/restore
Cmnd_Alias  WATCHDOG = /usr/local/bin/watchdog
Cmnd_Alias  SHELLS = /bin/sh, /bin/dash, /bin/bash

# Permissions
mark, ed    PHYSICS = ALL
herb        CS = /usr/sbin/tcpdump : PHYSICS = (operator) DUMP
lynda       ALL = (ALL) ALL, !SHELLS
%wheel      ALL, !PHYSICS = NOPASSWD: WATCHDOG
```

Reference: 01

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- sudo: command capabilities defined in sudores file:
  - Each permission specification line includes information about
    - The users to whom the line applies
    - The hosts on which the line should be heeded
    - The commands that the specified users can run
    - The users as whom the commands can be executed

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- sudo: advantages:
  - The use of **sudo** has the following advantages:
    - Accountability is much improved because of command logging.
    - Users can do specific chores without having unlimited root privileges.
    - The real root password can be known to only one or two people.
    - Using **sudo** is faster than using **su** or logging in as root.
    - Privileges can be revoked without the need to change the root password.
    - A canonical list of all users with root privileges is maintained.
    - The chance of a root shell being left unattended is lessened.
    - A single file can control access for an entire network.

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- sudo: disadvantages:
  - The use of **sudo** has the following disadvantages:
    - The worst of these is that any breach in the security of a sudoer's personal account can be equivalent to breaching the root account itself.
    - **sudo's** command logging can easily be subverted by tricks such as shell escapes from within an allowed program, or by **sudo sh** and **sudo su**.

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- sudo: sudo vs. advanced access control
  - Benefits of using sudo:
    - You decide exactly how privileges will be subdivided. Your division can be coarser or finer than the privileges defined for you by an off-the-shelf system.
    - Simple configurations—the most common—are simple to set up, maintain, and understand.
    - **sudo** runs on all UNIX and Linux systems. You do need not worry about managing different solutions on different platforms.
    - You can share a single configuration file throughout your site.
    - You get consistent, high-quality logging for free.

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- sudo: details
  - Typical setup
  - Environment management
  - **sudo** without passwords
  - Precedence
  - **sudo** without a control terminal
  - Site-wide **sudo** configuration

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- Disabling the root account:
  - The main effect of locking the root account is that root cannot log in, even on the console.
  - The main advantage of disabling the root account is that you needn't record and manage root's password.
  - Example: Ubuntu ships with the root account locked, and all administrative access is funneled through **sudo** or a GUI equivalent.

3.1 Access Control and Rootly Powers

3.1.2. Management of Root Account

- System accounts other than root:
- UIDs between 10 and 100 are pseudo-users associated with specific pieces of software
- Their shells should be set to **/bin/false** or **/bin/nologin** as well, to protect against remote login exploits that use password alternatives such as SSH key files.

3.1 Access Control and Rootly Powers

3.1.3. Extensions to the Standard Access Control Model

- Drawbacks of the standard model.
    - the root account represents a potential single point of failure
    - The only way to subdivide the privileges of the root account is to write setuid programs. Every setuid program is a potential target.
    - The standard model has little to say about security on a network.
    - In the standard model, group definition is a privileged operation
    - Many access control rules are embedded in the code of individual commands and daemons
    - The standard model also has little or no support for auditing or logging

3.1 Access Control and Rootly Powers

3.1.3. Extensions to the Standard Access Control Model

- PAM: Pluggable Authentication Modules
- Kerberos: network cryptographic authentication
- Filesystem access control lists
- Linux capabilities
- Linux namespaces

3.1 Access Control and Rootly Powers

3.1.4. Modern Access Control

- Separate ecosystems
- Mandatory access control
- Role-based access control

3.1 Access Control and Rootly Powers

3.1.4. Modern Access Control

- Mandatory access control
- Examples:
  - SELinux: Security-Enhanced Linux
  - AppArmor

3.2 Process Control

- A process represents a running program. It’s the abstraction through which memory, processor time, and I/O resources can be managed and monitored.

3.2 Process Control

3.2.1. Components of a process

- A process consists of an address space and a set of data structures within the kernel.
- The kernel's internal data structures record various pieces of information about each process such as:
  - The process's address space map
  - The current status of the process (sleeping, stopped, runnable, etc.)
  - The execution priority of the process
  - Information about the resources the process has used (CPU, memory, etc.)
  - Information about the files and network ports the process has opened
  - The process's signal mask (a record of which signals are blocked)
  - The owner of the process

3.2 Process Control

3.2.1. Components of a process

- PID: process ID number
- PPID: parent PID
- UID and EUID: real and effective user ID
- GID and EGID: real and effective group ID
- Niceness
- Control terminal

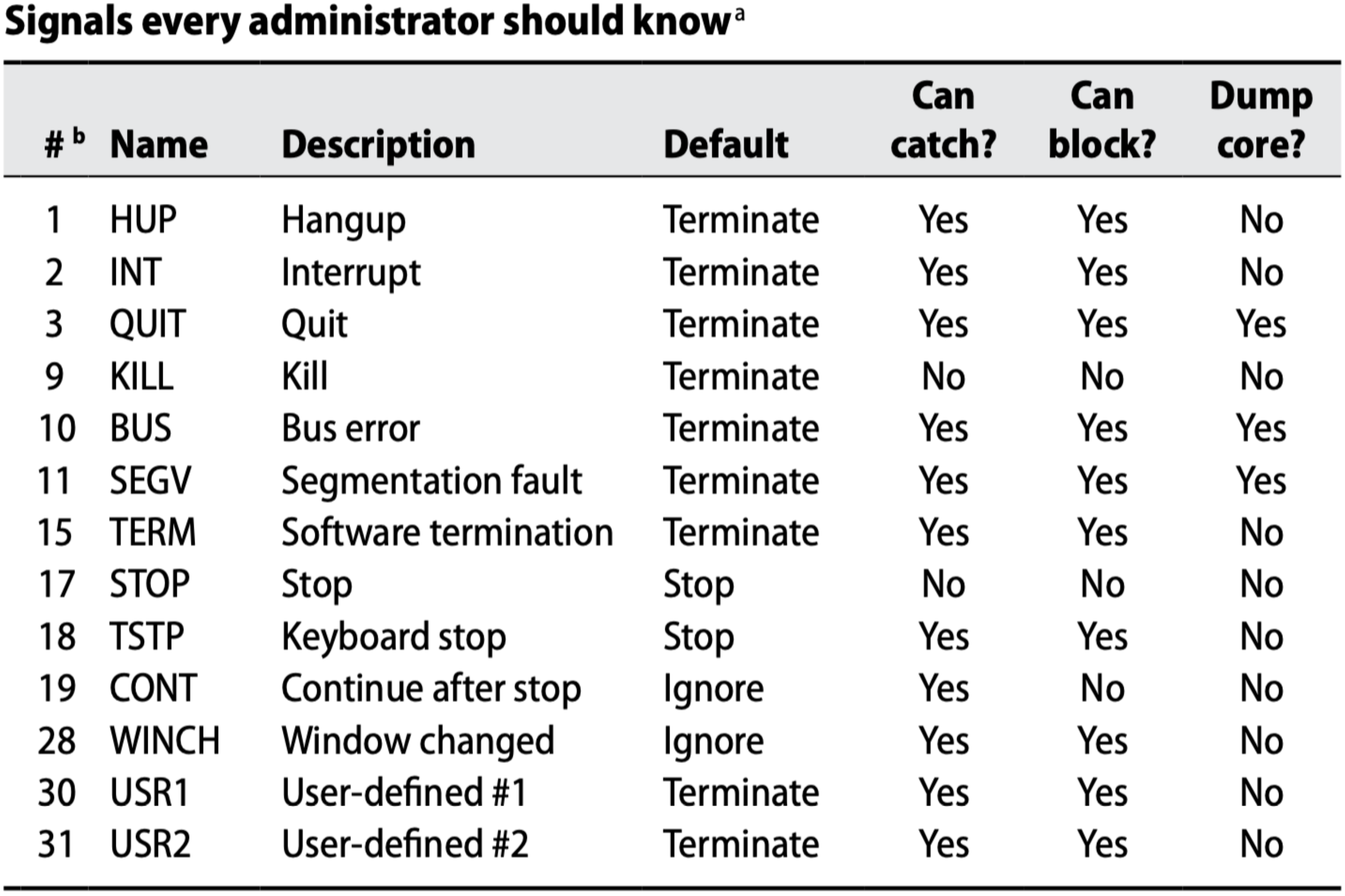
3.2 Process Control

3.2.2. The Life Cycle of a Process

- To create a new process, a process copies itself with the fork system call
- After a fork, the child process often uses one of the exec family of routines to begin the execution of a new program
- **init** or **systemd** depending on the system is very notable process.
- Before a dead process can be allowed to disappear completely, the kernel requires that its death be acknowledged by the process's parent, which the parent does with a call to **wait**

3.2 Process Control

3.2.2. The Life Cycle of a Process



Signals every administrator should know[a]

#[b]	Name	Description	Default	Can catch?	Can block?	Dump core?
1	HUP	Hangup	Terminate	Yes	Yes	No
2	INT	Interrupt	Terminate	Yes	Yes	No
3	QUIT	Quit	Terminate	Yes	Yes	Yes
9	KILL	Kill	Terminate	No	No	No
10	BUS	Bus error	Terminate	Yes	Yes	Yes
11	SEGV	Segmentation fault	Terminate	Yes	Yes	Yes
15	TERM	Software termination	Terminate	Yes	Yes	No
17	STOP	Stop	Stop	No	No	No
18	TSTP	Keyboard stop	Stop	Yes	Yes	No
19	CONT	Continue after stop	Ignore	Yes	No	No
28	WINCH	Window changed	Ignore	Yes	Yes	No
30	USR1	User-defined #1	Terminate	Yes	Yes	No
31	USR2	User-defined #2	Terminate	Yes	Yes	No

3.2 Process Control

3.2.2. The Life Cycle of a Process

- Process and thread states
  - A process can be suspended with a STOP signal and returned to active duty with a CONT signal
  - The state of being suspended or runnable applies to the process as a whole and is inherited by all the process's threads
  - Even when nominally runnable, threads must often wait for the kernel to complete some background work for them before they can continue execution.
  - A process is generally reported as "sleeping" when all its threads are asleep

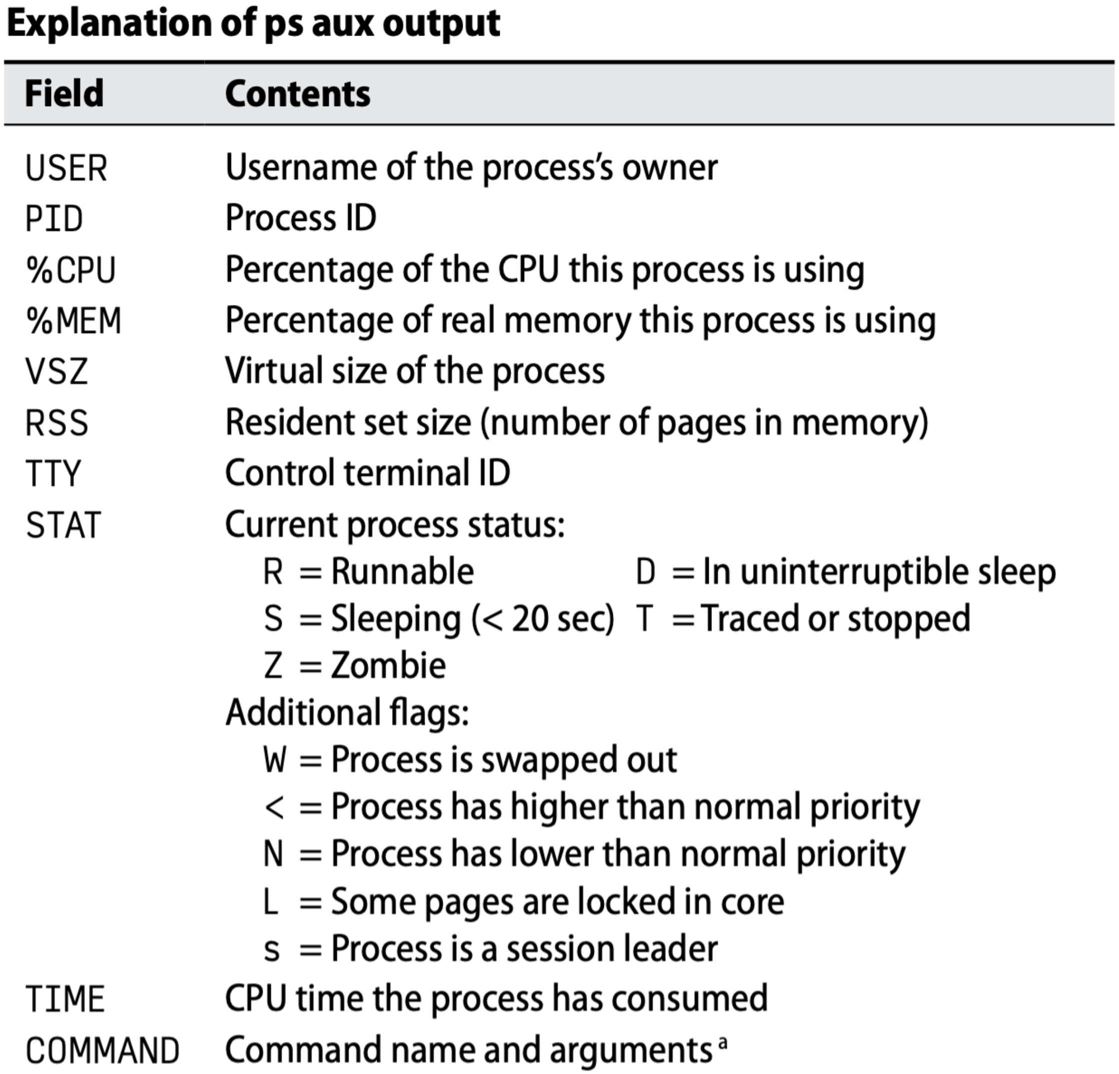
3.2 Process Control

3.2.3. Monitoring Processes

- The **ps** command is the system administrator’s main tool for monitoring processes.

3.2 Process Control

3.2.3. Monitoring Processes



Explanation of ps aux output

Field	Contents
USER	Username of the process's owner
PID	Process ID
%CPU	Percentage of the CPU this process is using
%MEM	Percentage of real memory this process is using
VSZ	Virtual size of the process
RSS	Resident set size (number of pages in memory)
TTY	Control terminal ID
STAT	Current process status: R = Runnable D = In uninterruptible sleep S = Sleeping (< 20 sec) T = Traced or stopped Z = Zombie Additional flags: W = Process is swapped out < = Process has higher than normal priority N = Process has lower than normal priority L = Some pages are locked in core s = Process is a session leader
TIME	CPU time the process has consumed
COMMAND	Command name and arguments[a]

3.2 Process Control

3.2.3. Monitoring Processes

- **ps** : monitor processes
- Interactive monitoring with **top**
- **nice** and **renice**: Influence scheduling priority
- The **/proc** file system
- **strace** and **truss**: Trance signals and system calls

3.2 Process Control

3.2.4. Runaway Processes

- “Runaway” processes are those that soak up significantly more of the system’s CPU, disk, or network resources than their usual role or behavior would lead you to expect.
- You can identify processes that are using excessive CPU time by looking at the output of **ps** or **top**.
- Also check the system load averages as reported by the **uptime** command.
- For CPU bound systems, the load averages should be less than the total number of CPU cores available on your system. If they are not, the system is overloaded.
- If you suspect misfeasance, obtain a system call trace with **strace** or **truss** to get a sense of what the process is doing (e.g., cracking passwords) and where its data is stored.

3.2 Process Control

3.2.5. Periodic Processes

- Common uses for scheduled tasks
  - Sending mail
  - Cleaning up a filesystem
  - Rotating a log file
  - Running batch jobs
  - Backing up and mirroring

3.2 Process Control

3.2.5. Periodic Processes

- cron: schedule commands
    - The format of crontab files
    - crontab management
    - cron access control

3.2 Process Control

3.2.5. Periodic Processes

- systemd timers
  - Structure of systemd timers
  - systemd timer example
  - systemd time expressions
  - Transient timers

3.3 File System

3.3.1. Introduction to File Systems

- The basic purpose of a filesystem is to represent and organize the system's storage resources.
- The filesystem can be thought of as comprising four main components:
    - A namespace – a way to name things and organize them in a hierarchy
    - An API – a set of system calls for navigating and manipulating objects
    - Security models – schemes for protecting, hiding, and sharing things
    - An implementation – software to tie the logical model to the hardware

3.3 File System

3.3.1. Introduction to File Systems

- Pathnames
- Filesystem mounting and unmounting
- Organization of the file tree

3.3 File System

3.3.2. File Types

- Regular files
- Directories
- Character and Block device files
- Local domain sockets
- Named pipes (FIFOs)
- Hard links and Symbolic links

3.3 File System

3.3.3. File Attributes

- The permission bits – owner, group, other
- The **setuid** and **setgid** bits
- The sticky bit

3.3 File System

3.3.3. File Attributes

- GNU/Linux commands on file attributes
  - **ls**: list and inspect files
  - **chmod**: change permissions
  - **chown** and **chgrp**: change ownership and group
  - **umask**: assign default permissions

3.3 File System

3.3.4. Access Control Lists (ACLs)

- Access control lists, aka ACLs, are a more powerful but also more complicated way of regulating access to files.
- An access control entry identifies the user or group to which it applies and specifies a set of permissions to be applied to those entities.
- ACLs have no set length and can include permission specifications for multiple users or groups.
- There are pros and cons of ACLs.

3.3 File System

3.3.4. Access Control Lists (ACLs)

- GNU/Linux ACL support
  - Linux has standardized on POSIX-style ACLs
  - NFSv4 ACLs are not supported at the filesystem level, though of course Linux systems can mount and share NFSv4 filesystems over the network.
  - The standard **getfacl** and **setfacl** commands can be used everywhere, without regard to the underlying filesystem type.

3.3 File System

3.3.4. Access Control Lists (ACLs)

- POSIX ACLs
  - Interaction between traditional modes and ACLs
  - POSIX access determination
  - POSIX ACL inheritance

3.3 File System

3.3.4. Access Control Lists (ACLs)

- NFSv4 ACLs
  - NFSv4 file permissions
  - NFSv4 entities for which permissions can be specified
  - NFSv4 access determination
  - ACL inheritance in NFSv4
  - NFSv4 ACL viewing
  - Interactions between ACLs and modes
  - NFSv4 ACL setup

3.4 Software Installation and Management

3.4.1 Managing Packages and Linux Package Management Systems

- UNIX and Linux software assets (source code, build files, documentation, and configuration templates) were traditionally distributed as compressed archives, usually gzipped tarballs (.tar.gz or .tgz files).
- These source code has to be compiled and build for each and every system, which is a difficult task.
- Packaging systems emerged to simplify and facilitate the job of software management.
- Packages include all the files needed to run a piece of software, including precompiled binaries, dependency information, and configuration file templates that can be customized by administrators.

Ref 1: Pg. (162-163)

3.4 Software Installation and Management

3.4.1 Managing Packages and Linux Package Management Systems

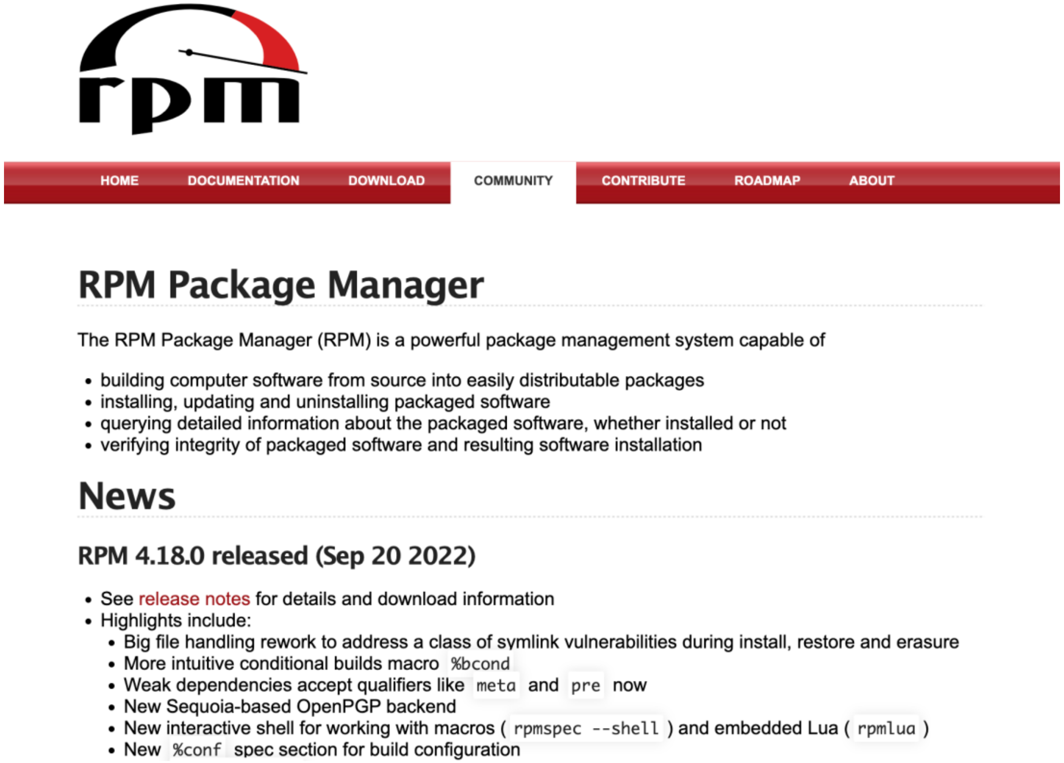
- There are two common package formats in Linux systems
    - RPM Package Manager (rpm)
        - Used in operating systems like Red Hat, CentOS, SUSE, Amazon Linux.
    - deb
        - Used in operating systems based on Debian.

Ref 1: Pg. (164)

3.4 Software Installation and Management

3.4.1 Managing Packages and Linux Package Management Systems

- rpm command installs, verifies, and queries the status of packages



RPM Package Manager

The RPM Package Manager (RPM) is a powerful package management system capable of

- building computer software from source into easily distributable packages
- installing, updating and uninstalling packaged software
- querying detailed information about the packaged software, whether installed or not
- verifying integrity of packaged software and resulting software installation

News

RPM 4.18.0 released (Sep 20 2022)

- See release notes for details and download information
- Highlights include:
  - Big file handling rework to address a class of symlink vulnerabilities during install, restore and erasure
  - More intuitive conditional builds macro `%bcond`
  - Weak dependencies accept qualifiers like `meta` and `pre` now
  - New Sequoia-based OpenPGP backend
  - New interactive shell for working with macros (`rpmspec --shell`) and embedded Lua (`rpmlua`)
  - New `%conf` spec section for build configuration

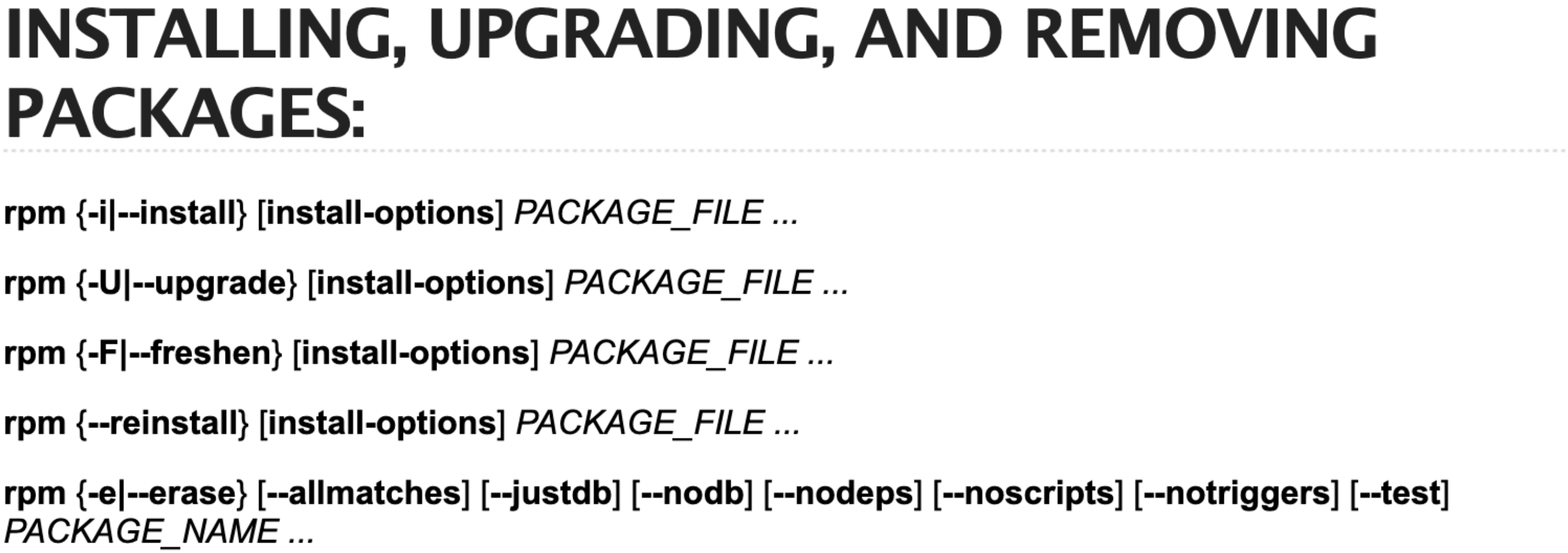
https://rpm.org/

Ref 1: Pg. (164)

3.4 Software Installation and Management

3.4.1 Managing Packages and Linux Package Management Systems

- rpm commands



INSTALLING, UPGRADING, AND REMOVING PACKAGES:

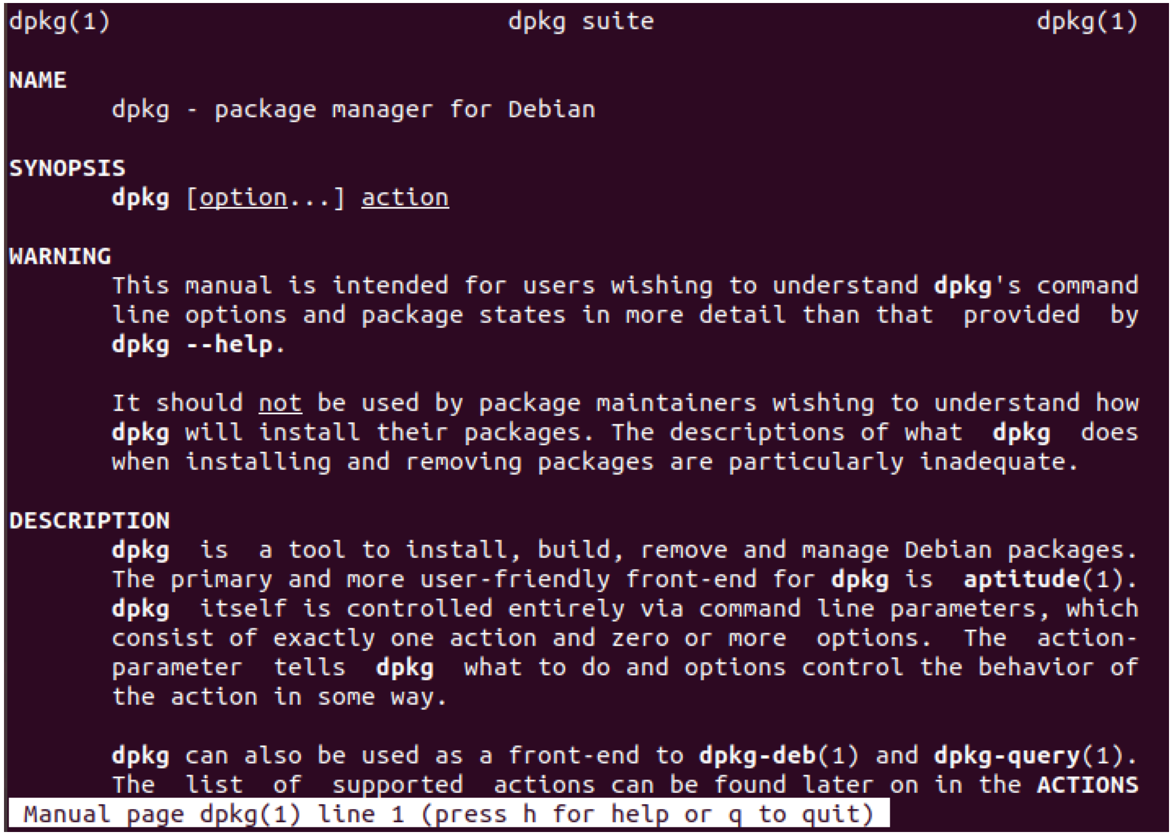
```
rpm {-i|--install} [install-options] PACKAGE_FILE ...

rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...

rpm {-F|--freshen} [install-options] PACKAGE_FILE ...

rpm {--reinstall} [install-options] PACKAGE_FILE ...

rpm {-e|--erase} [--allmatches] [--justdb] [--nodb] [--nodeps] [--noscripts] [--notriggers] [--test]
PACKAGE_NAME ...
```

https://rpm-software-management.github.io/rpm/man/rpm.8.html

3.4 Software Installation and Management

3.4.1 Managing Packages and Linux Package Management Systems

- dpkg: similer to rpm. It has set of commands and options to install, remove packages and many more.



```
dpkg(1)                          dpkg suite                          dpkg(1)

NAME
       dpkg - package manager for Debian

SYNOPSIS
       dpkg [option...] action

WARNING
       This manual is intended for users wishing to understand dpkg's command
       line options and package states in more detail than that  provided  by
       dpkg --help.

       It should not be used by package maintainers wishing to understand how
       dpkg will install their packages. The descriptions of what  dpkg  does
       when installing and removing packages are particularly inadequate.

DESCRIPTION
       dpkg  is  a tool to install, build, remove and manage Debian packages.
       The primary and more user-friendly front-end for dpkg is  aptitude(1).
       dpkg  itself is controlled entirely via command line parameters, which
       consist of exactly one action and zero or more  options.  The  action-
       parameter  tells  dpkg  what to do and options control the behavior of
       the action in some way.

       dpkg can also be used as a front-end to dpkg-deb(1) and dpkg-query(1).
       The  list  of  supported  actions can be found later on in the ACTIONS
 Manual page dpkg(1) line 1 (press h for help or q to quit)
```

Ref 1: Pg. (166)

3.4 Software Installation and Management

3.4.2. High-Level Linux Packages Management Systems

- Metapackage management systems such as APT and yum share several goals:
  - To simplify the task of locating and downloading packages
  - To automate the process of updating or upgrading systems
  - To facilitate the management of interpackage dependencies

Ref 1: Pg. (164)

3.4 Software Installation and Management

3.4.2. High-Level Linux Packages Management Systems

- APT: the Advanced Package Tool
  - It is one of the most matured package management system.
  - There are some basic commands used in apt;
    - apt-get
    - apt-cache
    - apt-get install
    - apt update

Ref 1: Pg. (169)

3.4 Software Installation and Management

3.4.2. High-Level Linux Packages Management Systems

- yum: release management for RPM
  - yum, the Yellowdog Updater, Modified, is a metapackage manager based on RPM.
  - yum command to fetch and install packages
  - yum figures out dependency constraints and does whatever additional work is needed to complete the installation of the requested packages.
  - yum has commands like;
    - yum install
    - yum update
    - yum check

Ref 1: Pg. (174)

3.4 Software Installation and Management

3.4.3. Software Localization and Configuration

- The local (or cloud) environment has to be properly designed. Following are some points to consider,
    - Non-administrators should not have root privileges.
    - Systems should facilitate work and not get in users’ way.
    - Misbehaving users are learning opportunities.
    - Be customer-centered.
    - Keep your local documentation up to date and easily accessible.

Ref 1: Pg. (178-179)

3.4 Software Installation and Management

3.4.3. Software Localization and Configuration

- Organizing your localization - There may be large number of devices with different configurations. It will be a difficult task to manage it. So, split the setup into manageable bits.
- Structuring updates - software installation is not enough. Need to update them. It is the one of the most important security tasks. Do not roll out new software releases en masse. Instead, stage rollouts according to a gradual plan that accommodates other groups' needs and allows time for problems to be discovered while their potential to cause damage is still limited.
- Limiting the field of play - It is a good idea to specify a maximum number of "releases" you are willing to have in play at any given time.
- Testing - test your own configurations and as well as the software that your vendor releases.

Ref 1: Pg. (179-180)

3.5 User Management

3.5.1. The /etc/passwd File

- User accounts
  - A user is just a number.
  - It is known as User ID or UID.
  - Almost everything related to the user management is conducted using this number.
  - /etc/passwd contains the list of users in the system.

Ref 1: Pg. (244)

3.5 User Management

3.5.1. The /etc/passwd File

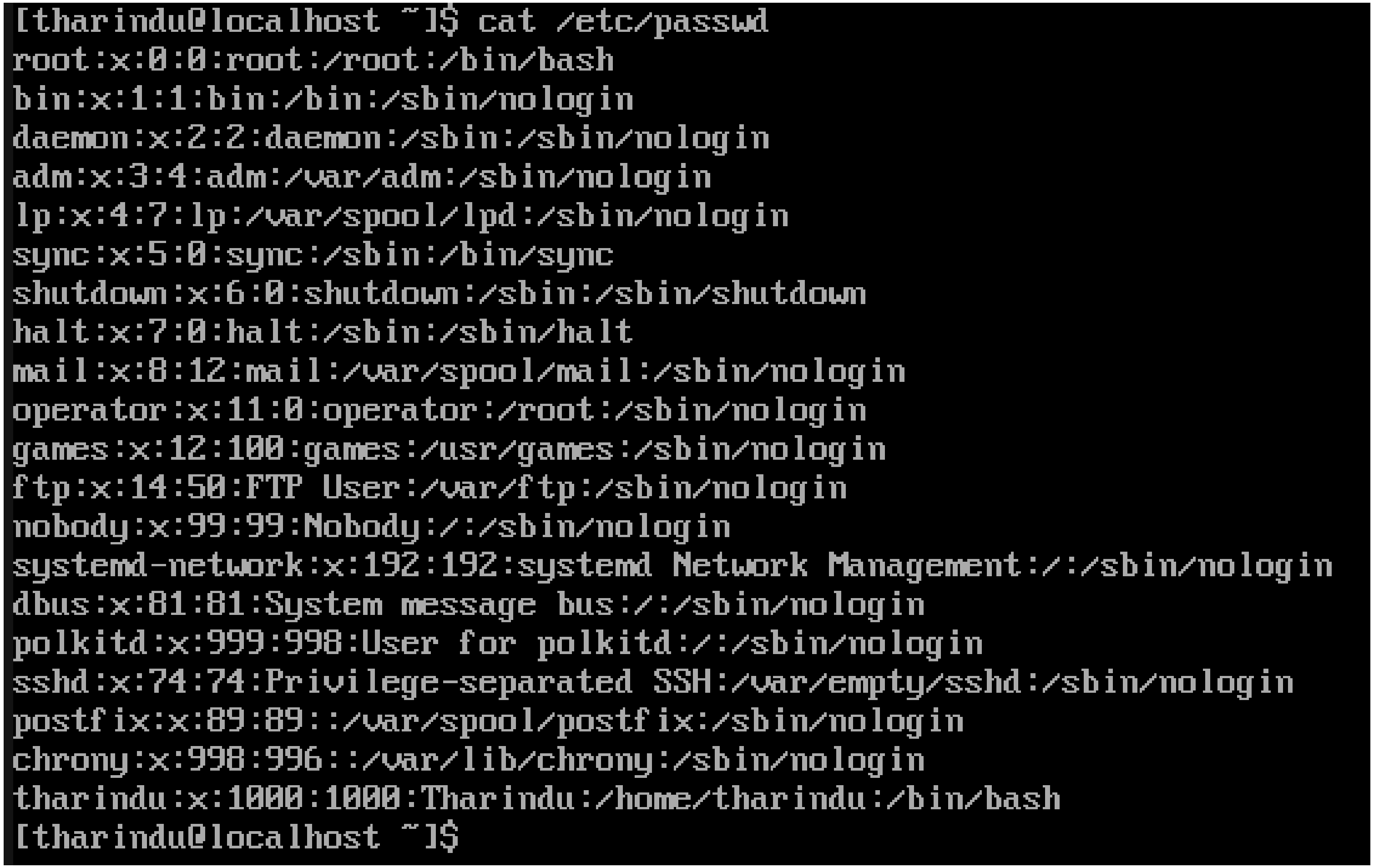
- The system consults /etc/passwd at login time to determine a user's UID and home directory, among other things.
- Each line in the file represents one user and contains seven fields separated by colons:
  - Login name
  - Encrypted password placeholder
  - UID (user ID) number
  - Default GID (group ID) number
  - Optional "GECOS" information : fullname, office, extension, homephone
  - Home directory
  - Loginshell

Ref 1: Pg. (245-250)

3.5 User Management

3.5.1. The /etc/passwd File

- cat /etc/passwd



```
[tharindu@localhost ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:996::/var/lib/chrony:/sbin/nologin
tharindu:x:1000:1000:Tharindu:/home/tharindu:/bin/bash
[tharindu@localhost ~]$
```

3.5 User Management

3.5.2. The /etc/shadow File

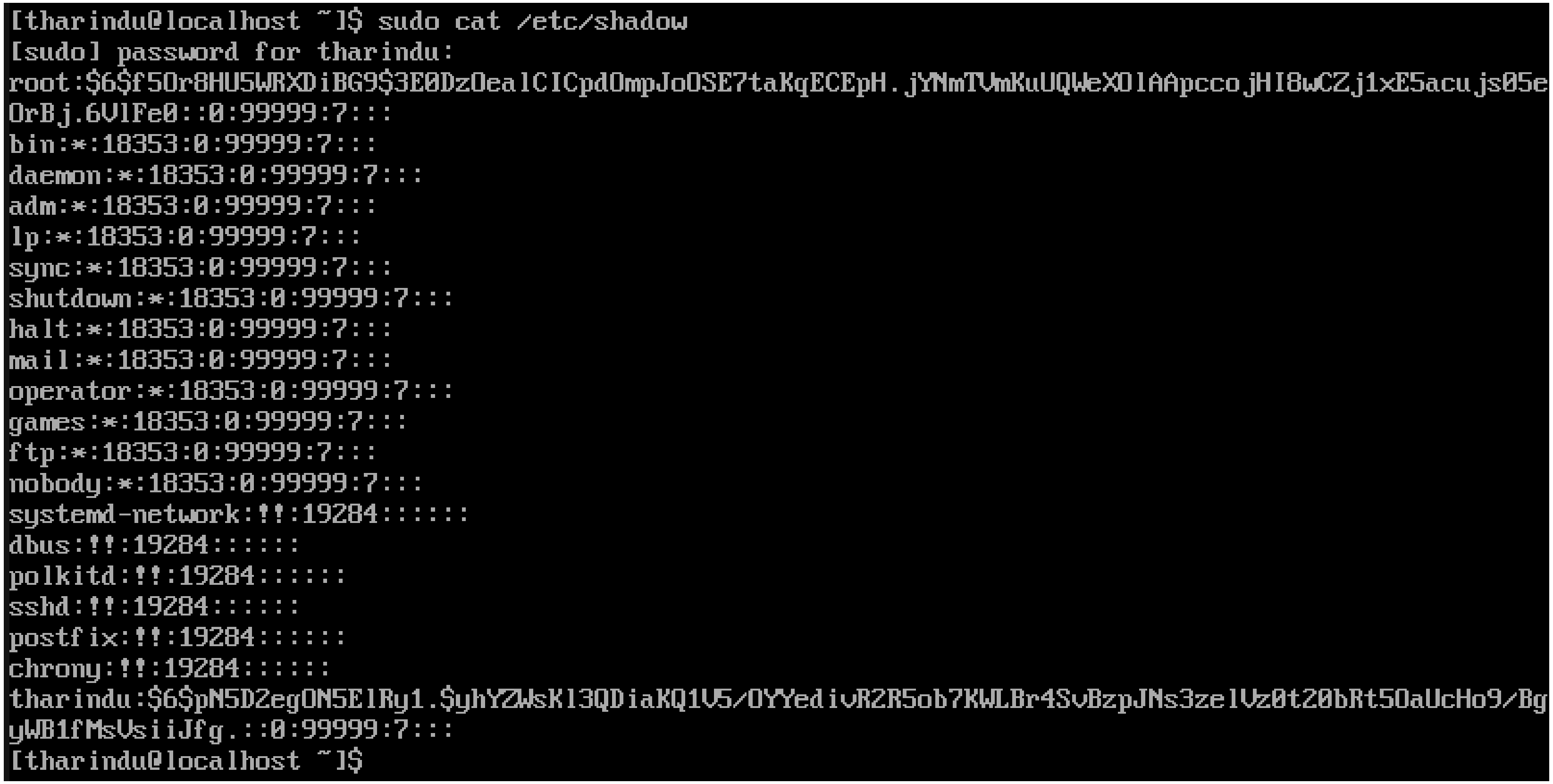
- On Linux, the shadow password file is readable only by the superuser.
- It keeps the encrypted passwords.
- It also includes some additional account information that wasn’t provided for in the original /etc/passwd format, such as;
  - Login name
  - Encrypted password
  - Date of last password change
  - Minimum number of days between password changes
  - Maximum number of days between password changes
  - Number of days in advance to warn users about password expiration

Ref 1: Pg. (250-252)

3.5 User Management

3.5.2. The /etc/shadow File

- cat /etc/shadow



```
[tharindu@localhost ~]$ sudo cat /etc/shadow
[sudo] password for tharindu:
root:$6$f5Or8HU5WRXDiBG9$3E0DzOealCICpdOmpJoOSE7taKqECEpH.jYNmTVmKuUQWeXDlAApccojHI8wCZj1xE5acujs05e
OrBj.6VlFe0::0:99999:7:::
bin:*:18353:0:99999:7:::
daemon:*:18353:0:99999:7:::
adm:*:18353:0:99999:7:::
lp:*:18353:0:99999:7:::
sync:*:18353:0:99999:7:::
shutdown:*:18353:0:99999:7:::
halt:*:18353:0:99999:7:::
mail:*:18353:0:99999:7:::
operator:*:18353:0:99999:7:::
games:*:18353:0:99999:7:::
ftp:*:18353:0:99999:7:::
nobody:*:18353:0:99999:7:::
systemd-network:!!:19284::::::
dbus:!!:19284::::::
polkitd:!!:19284::::::
sshd:!!:19284::::::
postfix:!!:19284::::::
chrony:!!:19284::::::
tharindu:$6$pN5D2egON5ElRy1.$yhYZWsKl3QDiaKQ1V5/OYYedivR2R5ob7KWLBr4SvBzpJNs3zelVz0t20bRt5OaUcHo9/Bg
yWB1fMsVsiiJfg.::0:99999:7:::
[tharindu@localhost ~]$
```

© 2022 e-Learning Centre, UCSC

3.5 User Management

3.5.3. The /etc/group File

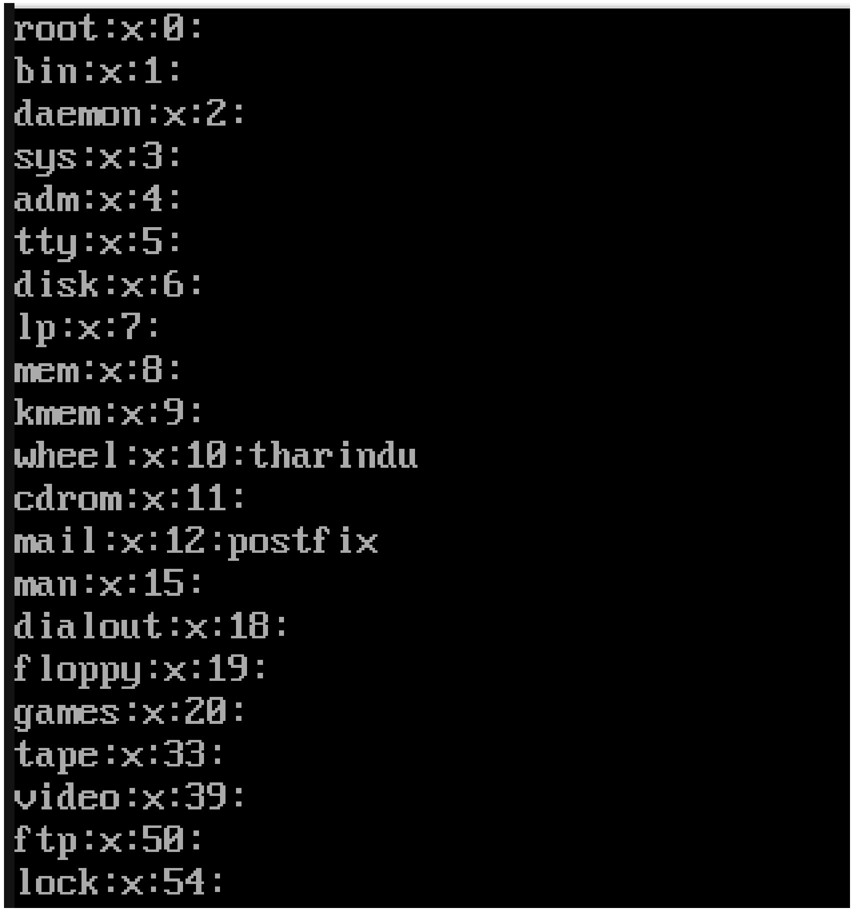
- It contains the names of UNIX groups and a list of each group’s members.
- Each line represents;
  - Group name
  - Encrypted password or a placeholder
  - GID number
  - List of members, separated by commas

Ref 1: Pg. (254-255)

3.5 User Management

3.5.3. The /etc/group File

- cat /etc/group



```
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mem:x:8:
kmem:x:9:
wheel:x:10:tharindu
cdrom:x:11:
mail:x:12:postfix
man:x:15:
dialout:x:18:
floppy:x:19:
games:x:20:
tape:x:33:
video:x:39:
ftp:x:50:
lock:x:54:
```

3.5 User Management

3.5.4. Adding Users – Manual and via Scripts

- Steps to create new users
  - Edit the passwd and shadow files.
  - Add the user to the /etc/group file.
  - Set an initial password.
  - Create, chown, and chmod the user’s home directory.
  - Configure roles and permissions.
  - Copy default startup files to the user’s home directory.

- Use text editors such as vi, vim, nano.. etc to edit the passwd and shadow files. Manually editing this files will create errors. Therefore, it is recommended to use **useradd**, **adduser**, **usermod**, **pw**, and **chsh** tools.

Ref 1: Pg. (256-260)

3.5 User Management

3.5.4. Adding Users – Manual and via Scripts

- Adding new users manually;
  - Use text editor to edit /etc/passwd file.
  - Add the following line to the end of passwd file and save it.

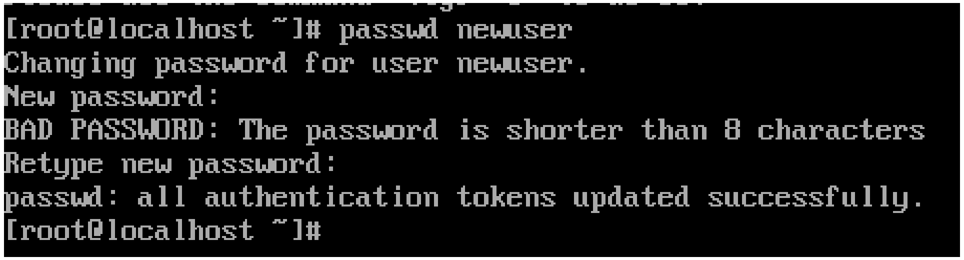
    ```
    newuser:x:1001:1001:NewUser:/home/newuser:/bin/bash
    ```

  - Then edit the /etc/group file and add the following line to the end. Save the file.

    ```
    newuser:x:1001:newuser
    ```

  - Set the password for the user account using the following command.

    ```
    sudo passwd newuser
    ```



```
[root@localhost ~]# passwd newuser
Changing password for user newuser.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]#
```

Ref 1: Pg. (256-260)

3.5 User Management

3.5.4. Adding Users – Manual and via Scripts

- Adding new users manually; (continue…)
  - Create home directory for the *newuser* using the following command.

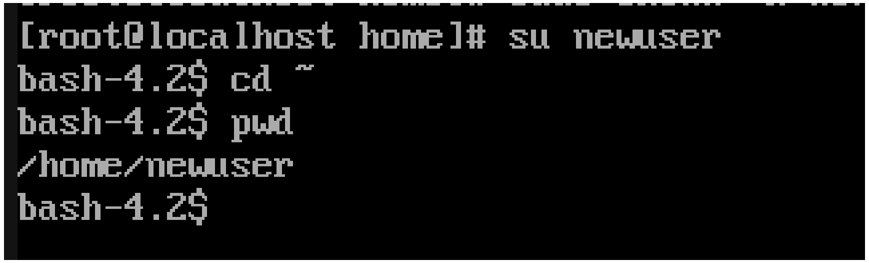
    ```
    mkdir /home/newuser
    ```

  - Set the permissions for the home directory.

    ```
    sudo chown -R newuser:newuser ~newuser
    ```

  - Finally log in as *newuser*.

    ```
    su newuser
    ```



```
[root@localhost home]# su newuser
bash-4.2$ cd ~
bash-4.2$ pwd
/home/newuser
bash-4.2$
```

Ref 1: Pg. (256-260)

3.5 User Management

3.5.4. Adding Users – Manual and via Scripts

- Adding new users using tools

```
sudo useradd newuser2
```

- This command can be customized with more options.

```
sudo useradd -c “Newuser2” -m -d /home/newuser2 newuser2 -s /bin/bash
```

- Then use passwd command to set the password for the account.
- There is another tool called *adduser* to create new users. This tool will guide you step by step to create the new user.

Ref 1: Pg. (260-263)

3.5 User Management

3.5.5. Removing Users

- We can use userdel command to remove the user account.
- After removing the user check the following;
  - Remove the user from any local user databases or phone lists.
  - Remove the user from the mail aliases database, or add a forwarding address.
  - Remove the user's crontab file and any pending at jobs or print jobs.
  - Kill any of the user's processes that are still running.
  - Remove the user from the passwd, shadow, group, and gshadow files.
  - Remove the user's home directory.
  - Remove the user's mail spool (if mail is stored locally).
  - Clean up entries on shared calendars, room reservation systems, etc.
  - Delete or transfer ownership of any mailing lists run by the deleted user.

Ref 1: Pg. (264)

3.5 User Management

3.5.6. User Login Lockout

- There may occasions where you need to lock a user account.
- You can use the following command to lock a user account (as an example lock the "newuser" account).

```
sudo usermod -L newuser
```

- To unlock the user account, you can use;

```
sudo usermod -U newuser
```

Ref 1: Pg. (265)

3.5 User Management

3.5.7. Risk Reduction with PAM

- Pluggable Authentication Modules (PAM)
- PAM centralizes the management of the system's authentication facilities through standard library routines.
- Programs like login, sudo, passwd, and su need not supply their own tricky authentication code.
- An organization can easily expand its authentication methods beyond passwords to options such as Kerberos, one-time passwords, ID dongles, or fingerprint readers.

Ref 1: Pg. (266)

3.5 User Management

3.5.8. Centralized Account Management

- It is essential for medium to large scale organizations.
- Users need a single login name, UID, and password across the site.
- Administrators need a centralized system that allows changes to be instantly propagated everywhere.
- LDAP (Lightweight Directory Access Protocol) and Microsoft's Active Directory system can be used to achieve them.

Ref 1: Pg. (266)

3.5 User Management

3.5.8. Centralized Account Management

- LDAP and Active Directory
  - LDAP is a database-like repository that can store user management data as well as other types of data.
  - Microsoft's Active Directory uses LDAP and Kerberos and can manage many kinds of data, including user information.
- Application-level single sign-on systems
  - User can sign on once and be authenticated at that time.
  - The user then obtains authentication credentials which can be used to access other applications.
  - The user only has to remember one login and password sequence instead of many.
- Identity management systems
  - Identifying the users of the systems, authenticating their identities, and granting privileges according to those authenticated identities.

Ref 1: Pg. (267)

3.6 Logging

3.6.1. Log Locations

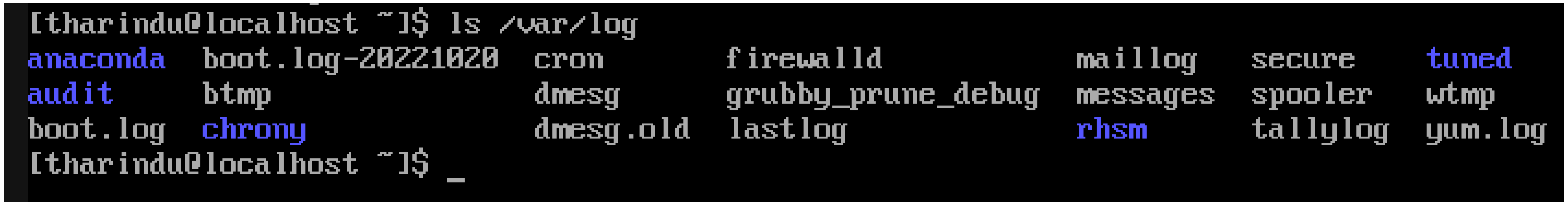
- A log message is usually a line of text with a few properties attached, including a time stamp, the type and severity of the event, and a process name and ID (PID).
- Log management
  - Collecting logs from a variety of sources
  - Providing a structured interface for querying, analyzing, filtering, and monitoring messages
  - Managing the retention and expiration of messages so that information is kept as long as it is potentially useful or legally required, but not indefinitely.

Ref 1: Pg. (294)

3.6 Logging

3.6.1. Log Locations

- Most of the log files can be found in /var/log directory



```
[tharindu@localhost ~]$ ls /var/log
anaconda  boot.log-20221020  cron       firewalld           maillog   secure    tuned
audit     btmp               dmesg      grubby_prune_debug  messages  spooler   wtmp
boot.log  chrony             dmesg.old  lastlog             rhsm      tallylog  yum.log
[tharindu@localhost ~]$ _
```

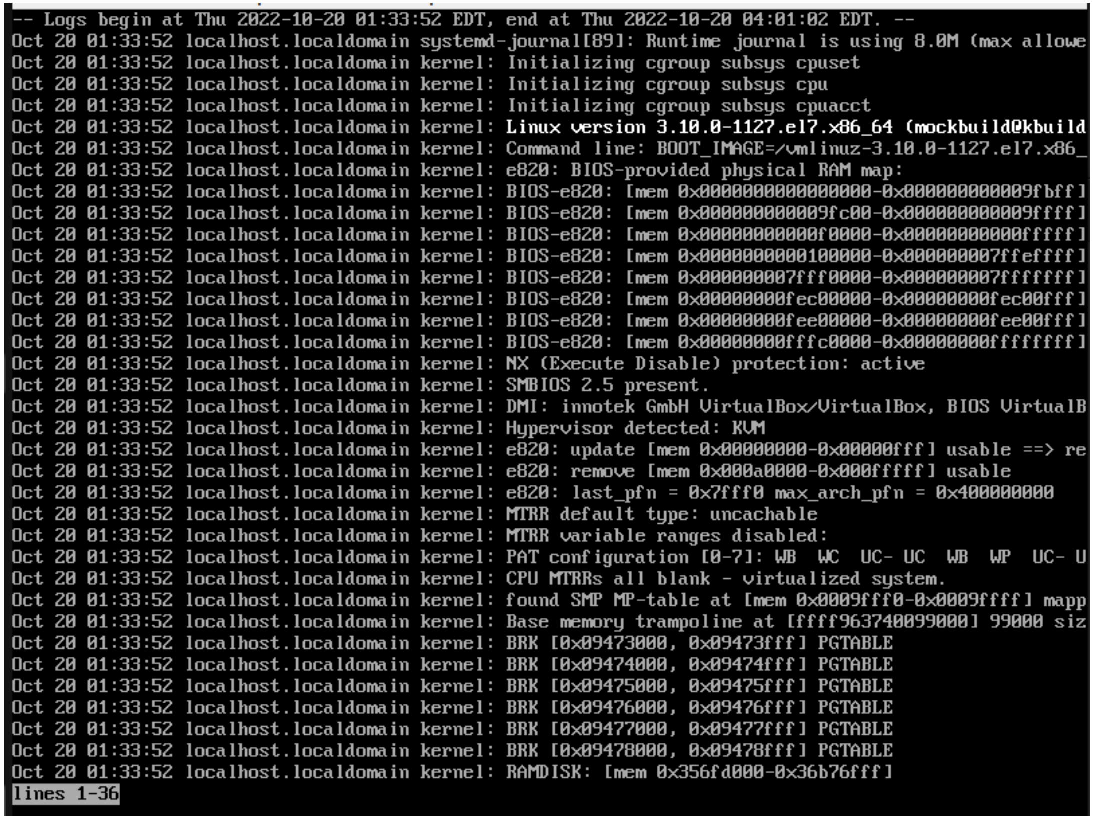
- But some applications write their log files elsewhere on the filesystem.

Ref 1: Pg. (296-297)

3.6 Logging

3.6.1. Log Locations

- To view logs in the systemd journal, type *journalctl* command. You can use some filters to filter the specific log you need to check.



```
-- Logs begin at Thu 2022-10-20 01:33:52 EDT, end at Thu 2022-10-20 04:01:02 EDT. --
Oct 20 01:33:52 localhost.localdomain systemd-journal[89]: Runtime journal is using 8.0M (max allowe
Oct 20 01:33:52 localhost.localdomain kernel: Initializing cgroup subsys cpuset
Oct 20 01:33:52 localhost.localdomain kernel: Initializing cgroup subsys cpu
Oct 20 01:33:52 localhost.localdomain kernel: Initializing cgroup subsys cpuacct
Oct 20 01:33:52 localhost.localdomain kernel: Linux version 3.10.0-1127.el7.x86_64 (mockbuild@kbuild
Oct 20 01:33:52 localhost.localdomain kernel: Command line: BOOT_IMAGE=/vmlinuz-3.10.0-1127.el7.x86_
Oct 20 01:33:52 localhost.localdomain kernel: e820: BIOS-provided physical RAM map:
Oct 20 01:33:52 localhost.localdomain kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff]
Oct 20 01:33:52 localhost.localdomain kernel: BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff]
Oct 20 01:33:52 localhost.localdomain kernel: BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff]
Oct 20 01:33:52 localhost.localdomain kernel: BIOS-e820: [mem 0x0000000000100000-0x000000007ffeffff]
Oct 20 01:33:52 localhost.localdomain kernel: BIOS-e820: [mem 0x000000007fff0000-0x000000007fffffff]
Oct 20 01:33:52 localhost.localdomain kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff]
Oct 20 01:33:52 localhost.localdomain kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff]
Oct 20 01:33:52 localhost.localdomain kernel: BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff]
Oct 20 01:33:52 localhost.localdomain kernel: NX (Execute Disable) protection: active
Oct 20 01:33:52 localhost.localdomain kernel: SMBIOS 2.5 present.
Oct 20 01:33:52 localhost.localdomain kernel: DMI: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualB
Oct 20 01:33:52 localhost.localdomain kernel: Hypervisor detected: KVM
Oct 20 01:33:52 localhost.localdomain kernel: e820: update [mem 0x00000000-0x00000fff] usable ==> re
Oct 20 01:33:52 localhost.localdomain kernel: e820: remove [mem 0x000a0000-0x000fffff] usable
Oct 20 01:33:52 localhost.localdomain kernel: e820: last_pfn = 0x7fff0 max_arch_pfn = 0x400000000
Oct 20 01:33:52 localhost.localdomain kernel: MTRR default type: uncachable
Oct 20 01:33:52 localhost.localdomain kernel: MTRR variable ranges disabled:
Oct 20 01:33:52 localhost.localdomain kernel: PAT configuration [0-7]: WB  WC  UC- UC  WB  WP  UC- U
Oct 20 01:33:52 localhost.localdomain kernel: CPU MTRRs all blank - virtualized system.
Oct 20 01:33:52 localhost.localdomain kernel: found SMP MP-table at [mem 0x0009fff0-0x0009ffff] mapp
Oct 20 01:33:52 localhost.localdomain kernel: Base memory trampoline at [ffff963740099000] 99000 siz
Oct 20 01:33:52 localhost.localdomain kernel: BRK [0x09473000, 0x09473fff] PGTABLE
Oct 20 01:33:52 localhost.localdomain kernel: BRK [0x09474000, 0x09474fff] PGTABLE
Oct 20 01:33:52 localhost.localdomain kernel: BRK [0x09475000, 0x09475fff] PGTABLE
Oct 20 01:33:52 localhost.localdomain kernel: BRK [0x09476000, 0x09476fff] PGTABLE
Oct 20 01:33:52 localhost.localdomain kernel: BRK [0x09477000, 0x09477fff] PGTABLE
Oct 20 01:33:52 localhost.localdomain kernel: BRK [0x09478000, 0x09478fff] PGTABLE
Oct 20 01:33:52 localhost.localdomain kernel: RAMDISK: [mem 0x356fd000-0x36b76fff]
lines 1-36
```

Ref 1: Pg. (298)

3.6 Logging

3.6.2. The system Journal

- The logging daemon of *systemd* is called as *systemd-journald.*
- It stores messages in a binary format.
- All message attributes are indexed automatically, which makes the log easier and faster to search.
- You can use ***journalctl*** command to review messages stored in the journal.

Ref 1: Pg. (299)

3.6 Logging

3.6.2. The system Journal

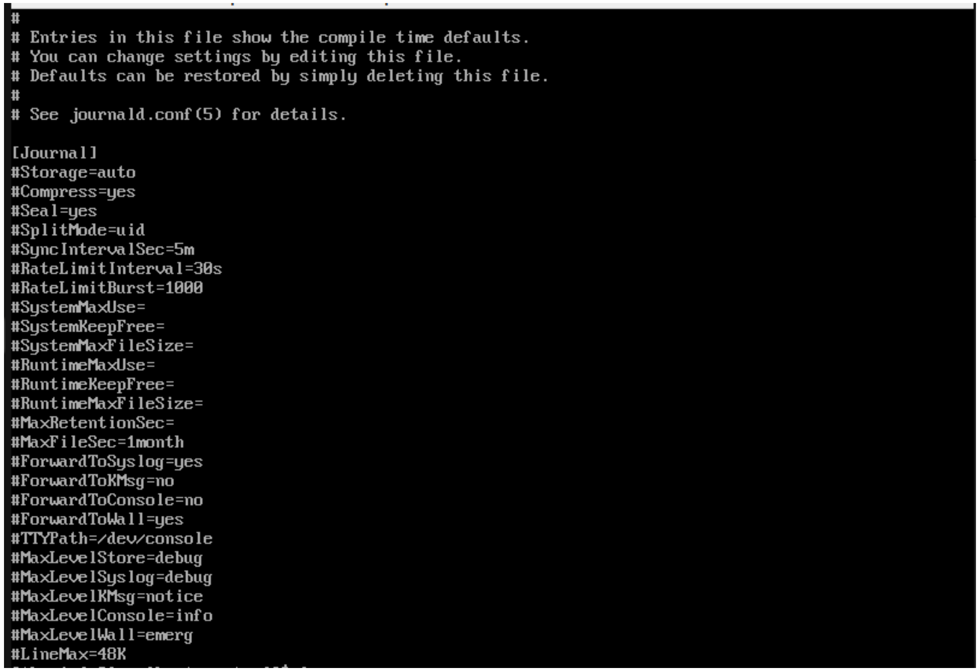
- The journal collects and indexes messages from several sources:
  - The */dev/log* UNIX socket, to harvest messages from software that submits messages according to syslog conventions
  - The device file */dev/kmsg*, to collect messages from the Linux kernel. The *systemd* journal daemon replaces the traditional *klogd* process that previously listened on this channel and formerly forwarded the kernel messages to syslog.
  - The UNIX socket */run/systemd/journal/stdout*, to service software that writes log messages to standard output
  - The UNIX socket */run/systemd/journal/socket*, to service software that submits messages through the *systemd* journal API.
  - Audit messages from the kernel's *auditd* daemon

Ref 1: Pg. (299)

3.6 Logging

3.6.2. The system Journal

- The default journal configuration file is ***/etc/systemd/journald.conf***, however, this file is not intended to be edited directly.
- Instead, add your customized configurations to the ***/etc/systemd/journald.conf.d*** directory.



```
#
# Entries in this file show the compile time defaults.
# You can change settings by editing this file.
# Defaults can be restored by simply deleting this file.
#
# See journald.conf(5) for details.

[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
#SplitMode=uid
#SyncIntervalSec=5m
#RateLimitInterval=30s
#RateLimitBurst=1000
#SystemMaxUse=
#SystemKeepFree=
#SystemMaxFileSize=
#RuntimeMaxUse=
#RuntimeKeepFree=
#RuntimeMaxFileSize=
#MaxRetentionSec=
#MaxFileSec=1month
#ForwardToSyslog=yes
#ForwardToKMsg=no
#ForwardToConsole=no
#ForwardToWall=yes
#TTYPath=/dev/console
#MaxLevelStore=debug
#MaxLevelSyslog=debug
#MaxLevelKMsg=notice
#MaxLevelConsole=info
#MaxLevelWall=emerg
#LineMax=48K
```

Ref 1: Pg. (300-302)

3.6 Logging

3.6.3. Syslog

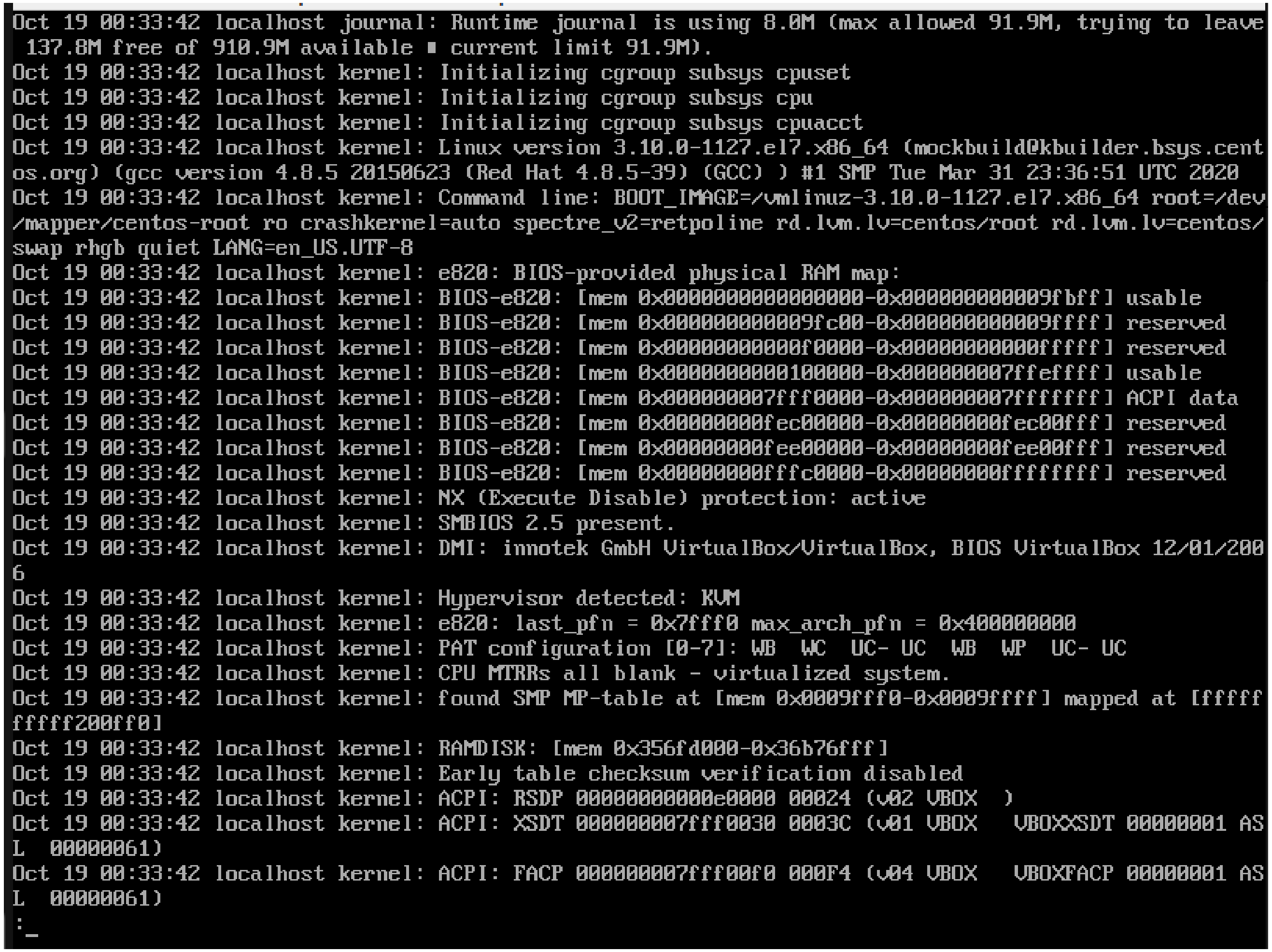
- It is a comprehensive logging system and IETF-standard logging protocol.
- It has two important functions:
  - to liberate programmers from the tedious mechanics of writing log files
  - to give administrators control of logging
- Content of syslog
  - Time stamp
  - System's hostname
  - Name of the process and its PID in square brackets
  - Message payload

Ref 1: Pg. (302-303)

3.6 Logging

3.6.3. Syslog

- cat /var/log/messages



```
Oct 19 00:33:42 localhost journal: Runtime journal is using 8.0M (max allowed 91.9M, trying to leave
 137.8M free of 910.9M available ■ current limit 91.9M).
Oct 19 00:33:42 localhost kernel: Initializing cgroup subsys cpuset
Oct 19 00:33:42 localhost kernel: Initializing cgroup subsys cpu
Oct 19 00:33:42 localhost kernel: Initializing cgroup subsys cpuacct
Oct 19 00:33:42 localhost kernel: Linux version 3.10.0-1127.el7.x86_64 (mockbuild@kbuilder.bsys.cent
os.org) (gcc version 4.8.5 20150623 (Red Hat 4.8.5-39) (GCC) ) #1 SMP Tue Mar 31 23:36:51 UTC 2020
Oct 19 00:33:42 localhost kernel: Command line: BOOT_IMAGE=/vmlinuz-3.10.0-1127.el7.x86_64 root=/dev
/mapper/centos-root ro crashkernel=auto spectre_v2=retpoline rd.lvm.lv=centos/root rd.lvm.lv=centos/
swap rhgb quiet LANG=en_US.UTF-8
Oct 19 00:33:42 localhost kernel: e820: BIOS-provided physical RAM map:
Oct 19 00:33:42 localhost kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
Oct 19 00:33:42 localhost kernel: BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
Oct 19 00:33:42 localhost kernel: BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
Oct 19 00:33:42 localhost kernel: BIOS-e820: [mem 0x0000000000100000-0x000000007ffeffff] usable
Oct 19 00:33:42 localhost kernel: BIOS-e820: [mem 0x000000007fff0000-0x000000007fffffff] ACPI data
Oct 19 00:33:42 localhost kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved
Oct 19 00:33:42 localhost kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved
Oct 19 00:33:42 localhost kernel: BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
Oct 19 00:33:42 localhost kernel: NX (Execute Disable) protection: active
Oct 19 00:33:42 localhost kernel: SMBIOS 2.5 present.
Oct 19 00:33:42 localhost kernel: DMI: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/200
6
Oct 19 00:33:42 localhost kernel: Hypervisor detected: KVM
Oct 19 00:33:42 localhost kernel: e820: last_pfn = 0x7fff0 max_arch_pfn = 0x400000000
Oct 19 00:33:42 localhost kernel: PAT configuration [0-7]: WB  WC  UC- UC  WB  WP  UC- UC
Oct 19 00:33:42 localhost kernel: CPU MTRRs all blank - virtualized system.
Oct 19 00:33:42 localhost kernel: found SMP MP-table at [mem 0x0009fff0-0x0009ffff] mapped at [fffff
fffff200ff0]
Oct 19 00:33:42 localhost kernel: RAMDISK: [mem 0x356fd000-0x36b76fff]
Oct 19 00:33:42 localhost kernel: Early table checksum verification disabled
Oct 19 00:33:42 localhost kernel: ACPI: RSDP 00000000000e0000 00024 (v02 VBOX  )
Oct 19 00:33:42 localhost kernel: ACPI: XSDT 000000007fff0030 0003C (v01 VBOX   VBOXXSDT 00000001 AS
L  00000061)
Oct 19 00:33:42 localhost kernel: ACPI: FACP 000000007fff00f0 000F4 (v04 VBOX   VBOXFACP 00000001 AS
L  00000061)
:_
```

3.6 Logging

3.6.4. Kernel and Boot-time Logging

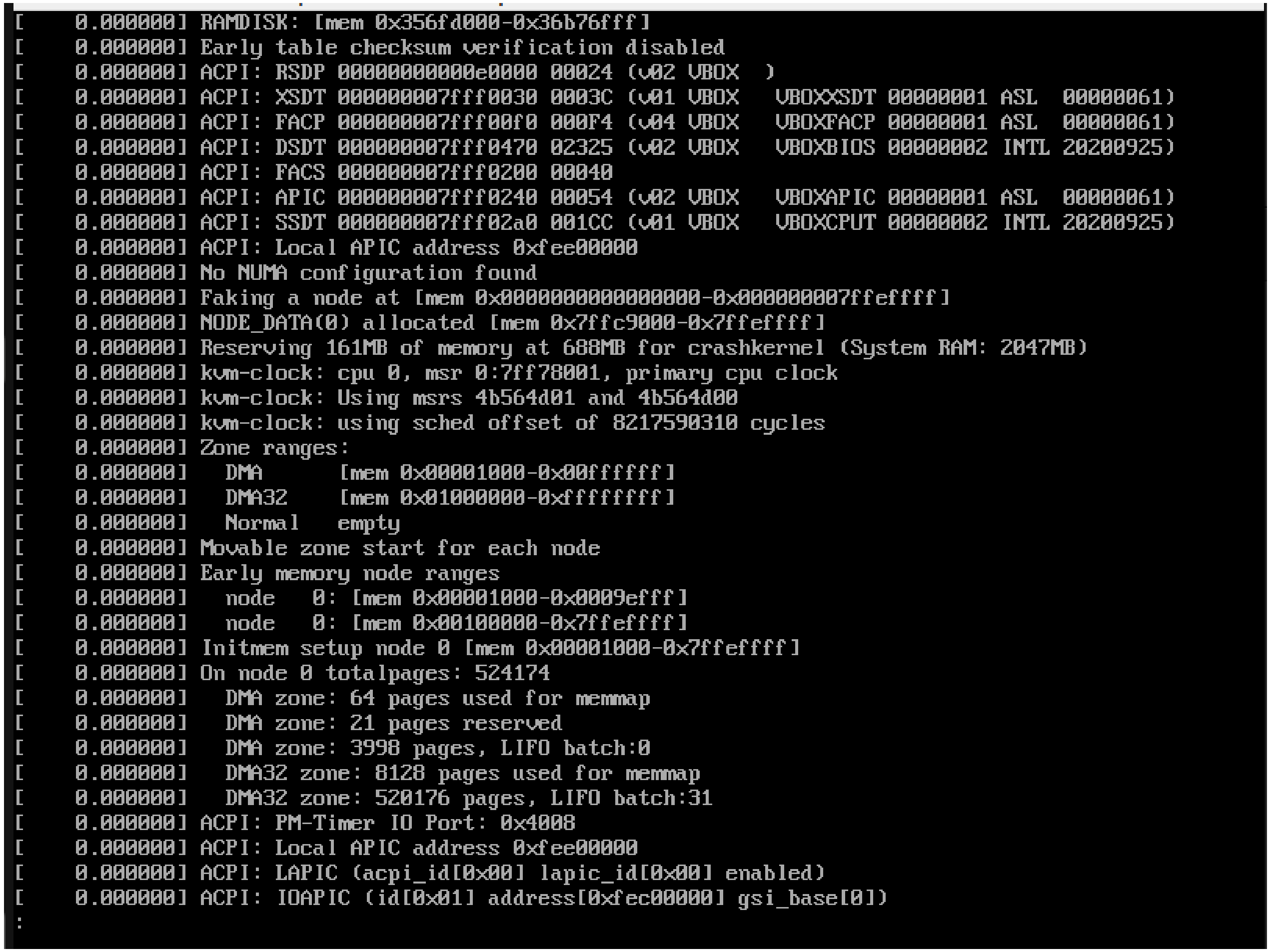
- For kernel logging at boot time, kernel log entries are stored in an internal buffer of limited size.
- The buffer is large enough to accommodate messages about all the kernel’s boot-time activities.
- When the system is up and running, a user process accesses the kernel’s log buffer and disposes of its contents.

Ref 1: Pg. (318)

3.6 Logging

3.6.4. Kernel and Boot-time Logging

- dmesg command to view kernel logs



```
[    0.000000] RAMDISK: [mem 0x356fd000-0x36b76fff]
[    0.000000] Early table checksum verification disabled
[    0.000000] ACPI: RSDP 00000000000e0000 00024 (v02 VBOX  )
[    0.000000] ACPI: XSDT 000000007fff0030 0003C (v01 VBOX   VBOXXSDT 00000001 ASL  00000061)
[    0.000000] ACPI: FACP 000000007fff00f0 000F4 (v04 VBOX   VBOXFACP 00000001 ASL  00000061)
[    0.000000] ACPI: DSDT 000000007fff0470 02325 (v02 VBOX   VBOXBIOS 00000002 INTL 20200925)
[    0.000000] ACPI: FACS 000000007fff0200 00040
[    0.000000] ACPI: APIC 000000007fff0240 00054 (v02 VBOX   VBOXAPIC 00000001 ASL  00000061)
[    0.000000] ACPI: SSDT 000000007fff02a0 001CC (v01 VBOX   VBOXCPUT 00000002 INTL 20200925)
[    0.000000] ACPI: Local APIC address 0xfee00000
[    0.000000] No NUMA configuration found
[    0.000000] Faking a node at [mem 0x0000000000000000-0x000000007ffeffff]
[    0.000000] NODE_DATA(0) allocated [mem 0x7ffc9000-0x7ffeffff]
[    0.000000] Reserving 161MB of memory at 688MB for crashkernel (System RAM: 2047MB)
[    0.000000] kvm-clock: cpu 0, msr 0:7ff78001, primary cpu clock
[    0.000000] kvm-clock: Using msrs 4b564d01 and 4b564d00
[    0.000000] kvm-clock: using sched offset of 8217590310 cycles
[    0.000000] Zone ranges:
[    0.000000]   DMA      [mem 0x00001000-0x00ffffff]
[    0.000000]   DMA32    [mem 0x01000000-0xffffffff]
[    0.000000]   Normal   empty
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x00001000-0x0009efff]
[    0.000000]   node   0: [mem 0x00100000-0x7ffeffff]
[    0.000000] Initmem setup node 0 [mem 0x00001000-0x7ffeffff]
[    0.000000] On node 0 totalpages: 524174
[    0.000000]   DMA zone: 64 pages used for memmap
[    0.000000]   DMA zone: 21 pages reserved
[    0.000000]   DMA zone: 3998 pages, LIFO batch:0
[    0.000000]   DMA32 zone: 8128 pages used for memmap
[    0.000000]   DMA32 zone: 520176 pages, LIFO batch:31
[    0.000000] ACPI: PM-Timer IO Port: 0x4008
[    0.000000] ACPI: Local APIC address 0xfee00000
[    0.000000] ACPI: LAPIC (acpi_id[0x00] lapic_id[0x00] enabled)
[    0.000000] ACPI: IOAPIC (id[0x01] address[0xfec00000] gsi_base[0])
:
```

Ref 1: Pg. (318)

3.6 Logging

3.6.5. Management and Rotation of Log Files

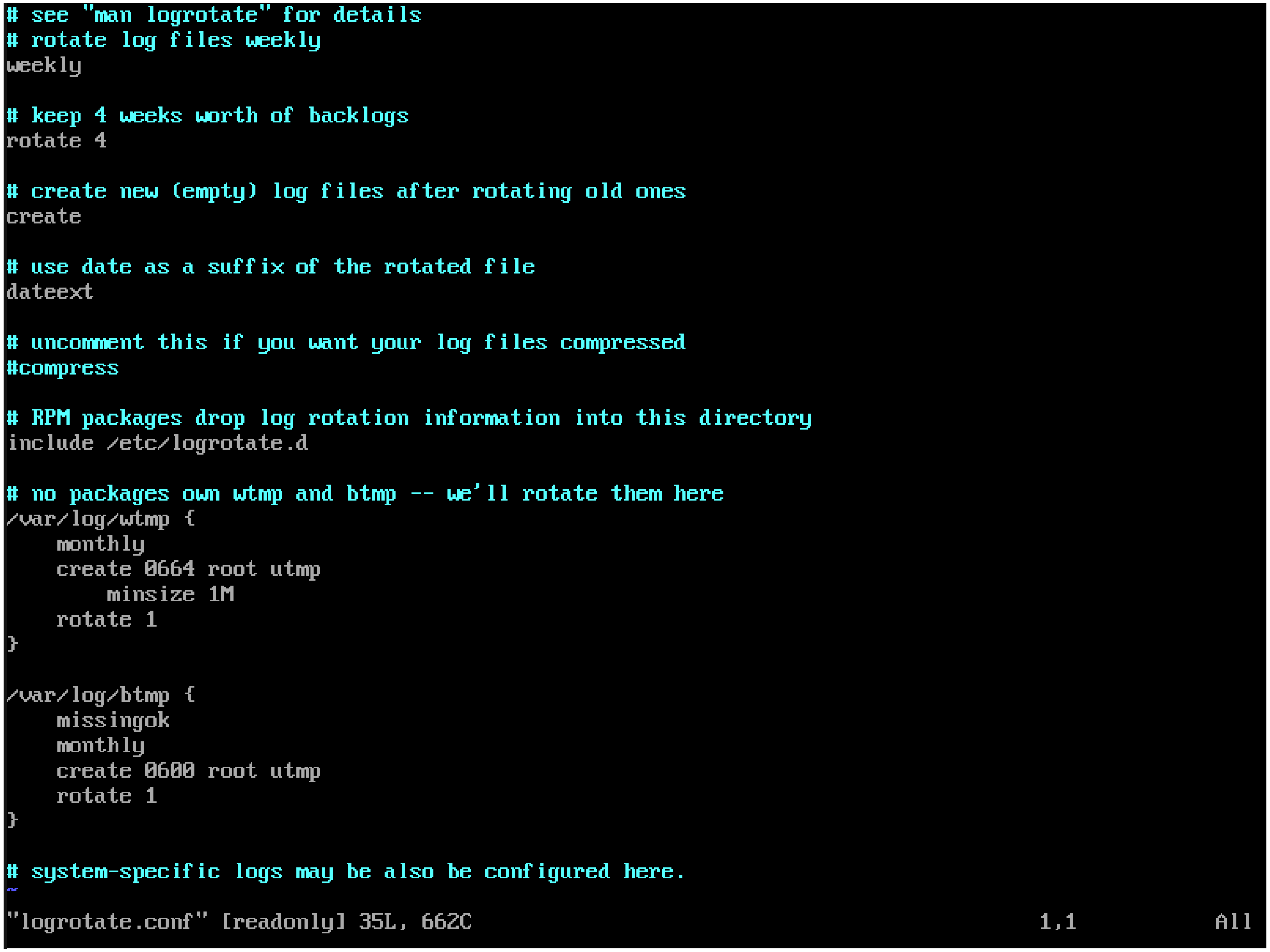
- *logrotate*: cross-platform log management
  - A *logrotate* configuration consists of a series of specifications for groups of log files to be managed.
  - logrotate is normally run out of cron once a day.
  - Its standard configuration file is /etc/logrotate.conf, but multiple configuration files can appear on logrotate's command line.

Ref 1: Pg. (319-321)

3.6 Logging

3.6.5. Management and Rotation of Log Files

- cat /etc/logrotate.conf



```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
dateext

# uncomment this if you want your log files compressed
#compress

# RPM packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp and btmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
        minsize 1M
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0600 root utmp
    rotate 1
}

# system-specific logs may be also be configured here.
~
"logrotate.conf" [readonly] 35L, 662C                                     1,1           All
```

3.6 Logging

3.6.5. Management and Rotation of Log Files

- Management of logs at large scale
  - Handling logging data from hundreds or thousands of servers is challenging. Following are some tools can be used to handle log files in large scale
    - The ELK stack
    - Graylog
    - Splunk

Ref 1: Pg. (321-323)

3.6 Logging

3.6.6. Logging Policies

- Points need to be considered when preparing the logging strategies.
  - How many systems and applications will be included?
  - What type of storage infrastructure is available?
  - How long must logs be retained?
  - What types of events are important?
- The answers to these questions depend on business requirements and on any applicable standards or regulations.

Ref 1: Pg. (323)

3.6 Logging

3.6.6. Logging Policies

- For application logs, at least try to consider capture the following information:
  - Username or user ID
  - Event success or failure
  - Source address for network events
  - Date and time (from an authoritative source, such as NTP)
  - Sensitive data added, altered, or removed
  - Event details
- A log server should have a carefully considered storage strategy.
- Limit shell access to centralized log servers to trusted system administrators and personnel involved in addressing compliance and security issues.

Ref 1: Pg. (324)

3.7 Storage

3.7.1. Storage Hardware and Interfaces

- Hard Disks
  - How hard disks work?
  - Hard disk reliability.
  - Failure modes and metrics
  - Drive types
    - Value drives, Mass-market performance drives, NAS drives, Enterprise drives
  - Warranties and retirement

3.7 Storage

3.7.1. Storage Hardware and Interfaces

- Solid State Disks
  - Rewritability limits
  - Flash memory and controller types
  - Page clusters and pre-erasing
  - SSD reliability

3.7 Storage

3.7.1. Storage Hardware and Interfaces

- Why storage devices shifted to 4KB blocks either by native or emulation?

3.7 Storage

3.7.1. Storage Hardware and Interfaces

- Storage Hardware Interfaces
    - The SATA Interface
    - The PCI Express Interface
    - The SAS Interface
    - USB

- Compare each interface in terms of the performance.

3.7 Storage

3.7.1. Storage Hardware and Interfaces

- **lsblk** commands can print out a list of the disks that the system is aware of in a Linux system.
- A newly added disk is represented by device files in **/dev**. For example, **/dev/sda** on Linux is the first drive managed by the sd driver. The next drive would be **/dev/sdb**, and so on
- Disk names are assigned in sequence as the kernel enumerates the various inter- faces and devices on the system.
- **parted -l** lists the sizes, partition tables, model numbers, and manufacturers of every disk on the system.

3.7 Storage

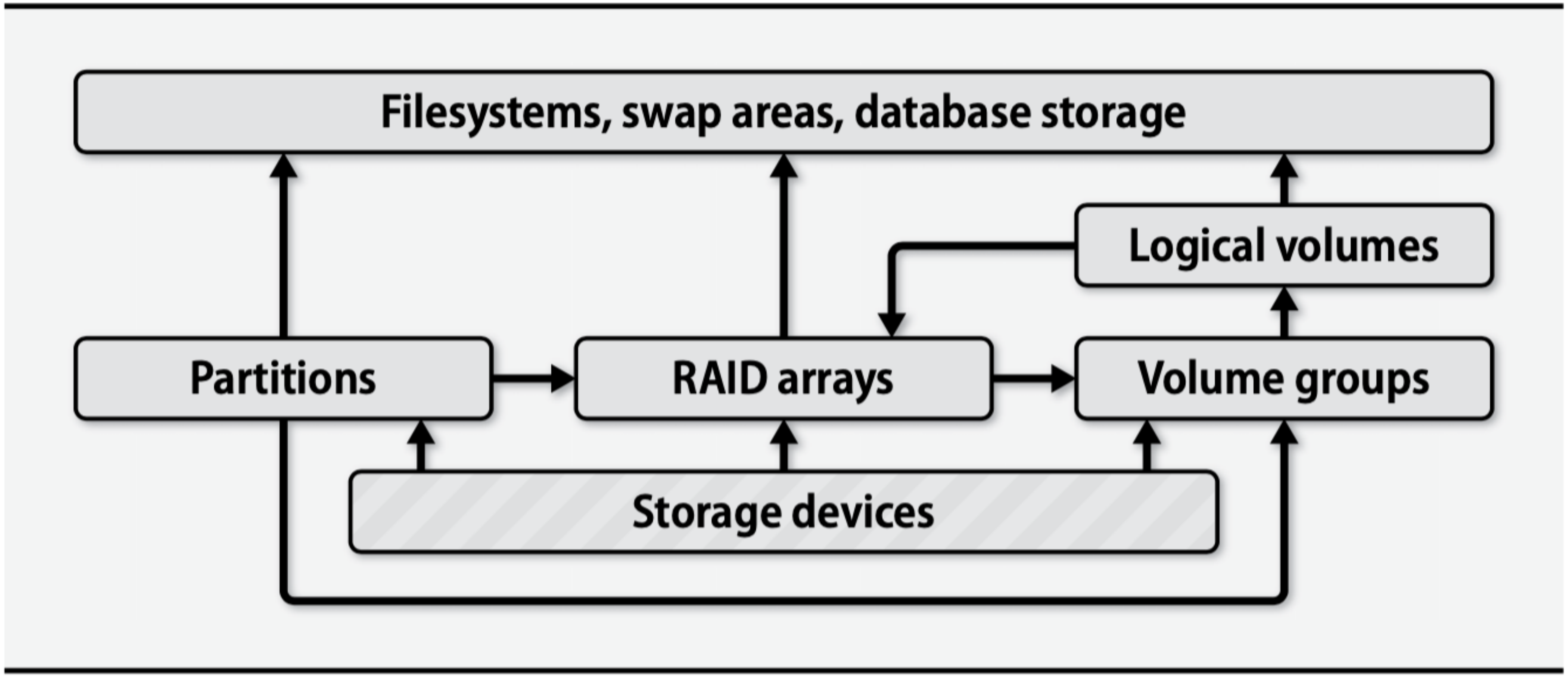
3.7.1. Storage Hardware and Interfaces

- Functions on Hard Disk Drives
  - Formatting vs Initializing
  - Secure erase - **shred**
  - Set disk and interface parameters – **hdparam**
  - Hard disk monitoring with SMART

3.7 Storage

3.7.1. Storage Hardware and Interfaces

- Elements of a storage management system
  - Storage management layers

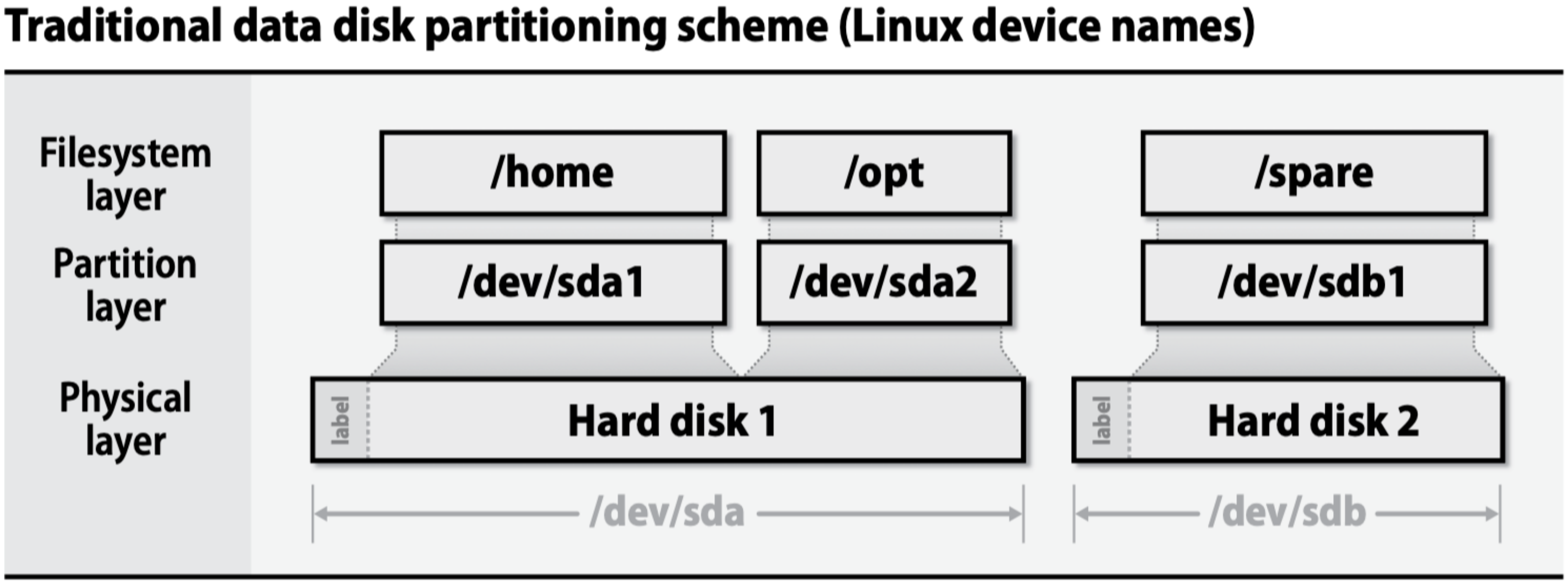


What is the Linux device mapper?

Source: Ref 01

3.7 Storage

3.7.2. Disk Partitioning



Source: Ref 01

3.7 Storage

3.7.2. Disk Partitioning

- General guides – reason out why?
  - Putting **/tmp** on a separate filesystem
  - Log files are kept in **/var/log**
  - Having a separate partition for **/home**

- The Center for Internet Security publishes configuration guidelines for a variety of operating systems at www.cisecurity.org/cis-benchmarks.

3.7 Storage

3.7.2. Disk Partitioning

MBR vs GPT partitioning

- Linux partitioning with **gparted**.

3.7 Storage

3.7.3. Logical Volume Management

- Operations a volume manager lets you carry out
  - Move logical volumes among different physical devices
  - Grow and shrink logical volumes on the fly
  - Take copy-on-write “snapshots” of logical volumes
  - Replace on-line drives without interrupting service
  - Incorporate mirroring or striping in your logical volumes

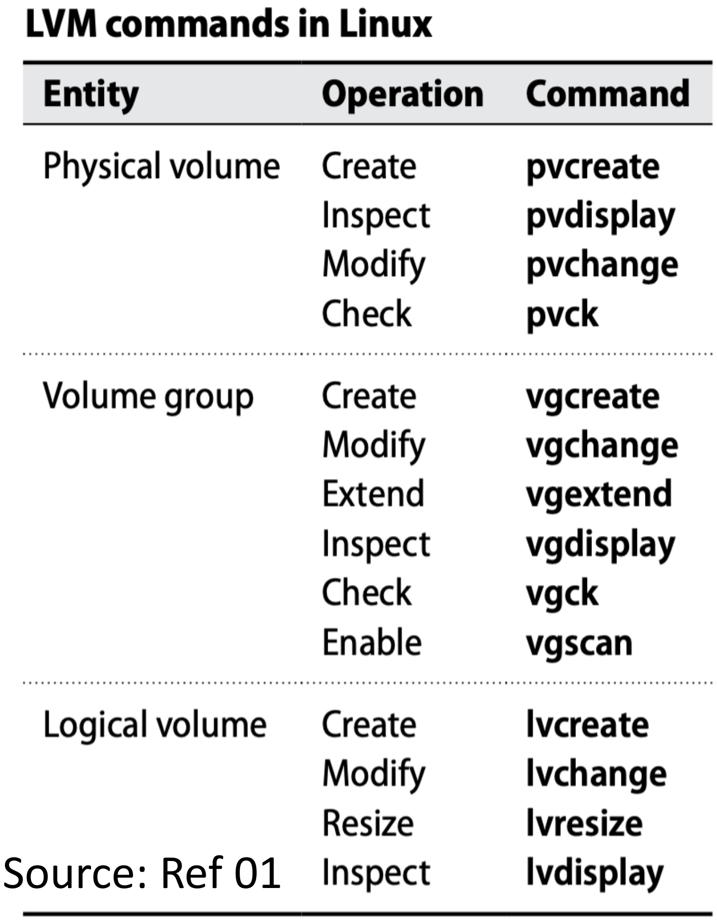
3.7 Storage

3.7.3. Logical Volume Management

- Linux logical volume management
- Hands on experience

- A Linux LVM configuration procedure:
  - Creating (defining, really) and initializing physical volumes
  - Adding the physical volumes to a volume group
  - Creating logical volumes on the volume group

- Learn the Linux way of doing LVM:
  - Volume snapshots
  - Filesystem resizing



LVM commands in Linux

Entity	Operation	Command
Physical volume	Create	**pvcreate**
	Inspect	**pvdisplay**
	Modify	**pvchange**
	Check	**pvck**
Volume group	Create	**vgcreate**
	Modify	**vgchange**
	Extend	**vgextend**
	Inspect	**vgdisplay**
	Check	**vgck**
	Enable	**vgscan**
Logical volume	Create	**lvcreate**
	Modify	**lvchange**
	Resize	**lvresize**
	Inspect	**lvdisplay**

Source: Ref 01

3.7 Storage

3.7.4. Redundant Arrays of Inexpensive Disks

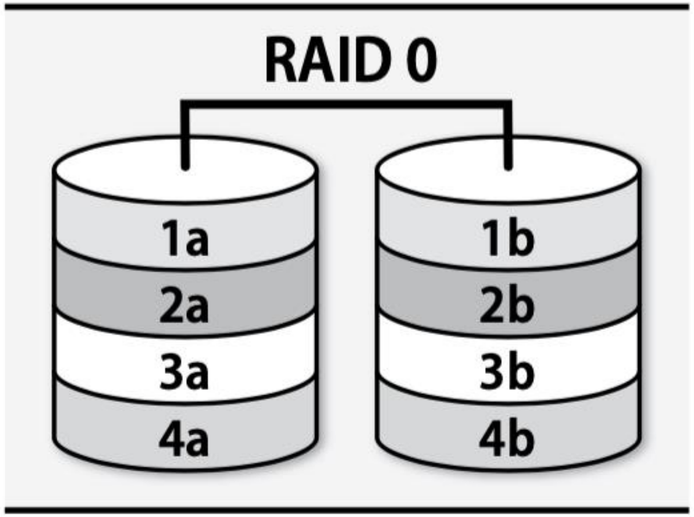
- it can improve performance
- it can replicate data across multiple drives, decreasing the risk associated with a single failed disk

Software vs Hardware RAID

3.7 Storage

3.7.4. Redundant Arrays of Inexpensive Disks

- RAID Level - 0

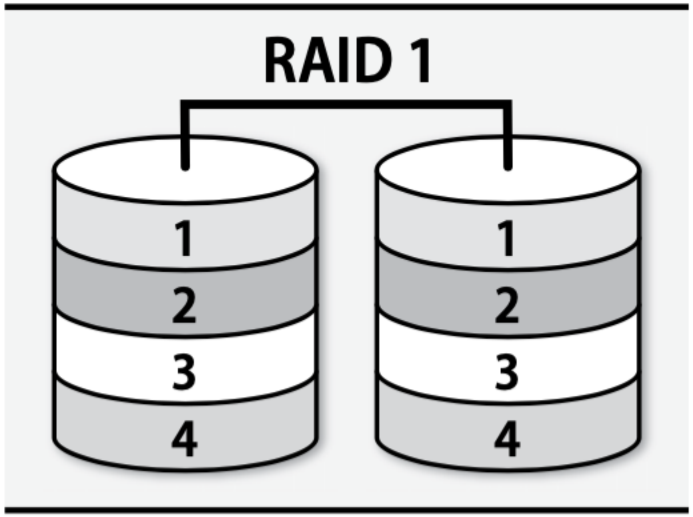


Source: Ref 01

3.7 Storage

3.7.4. Redundant Arrays of Inexpensive Disks

- RAID Level - 1

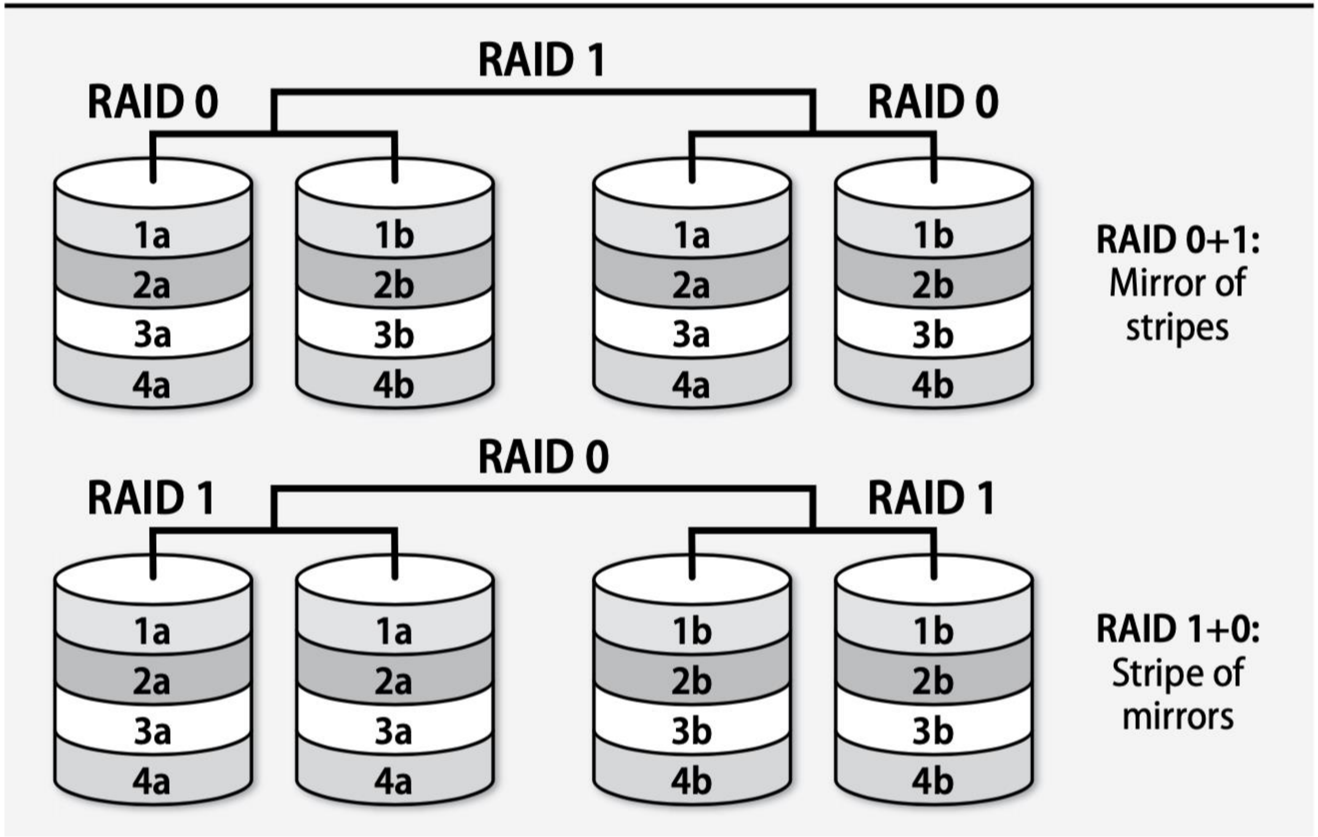


Source: Ref 01

3.7 Storage

3.7.4. Redundant Arrays of Inexpensive Disks

- RAID Level – 0+1 , 1+0

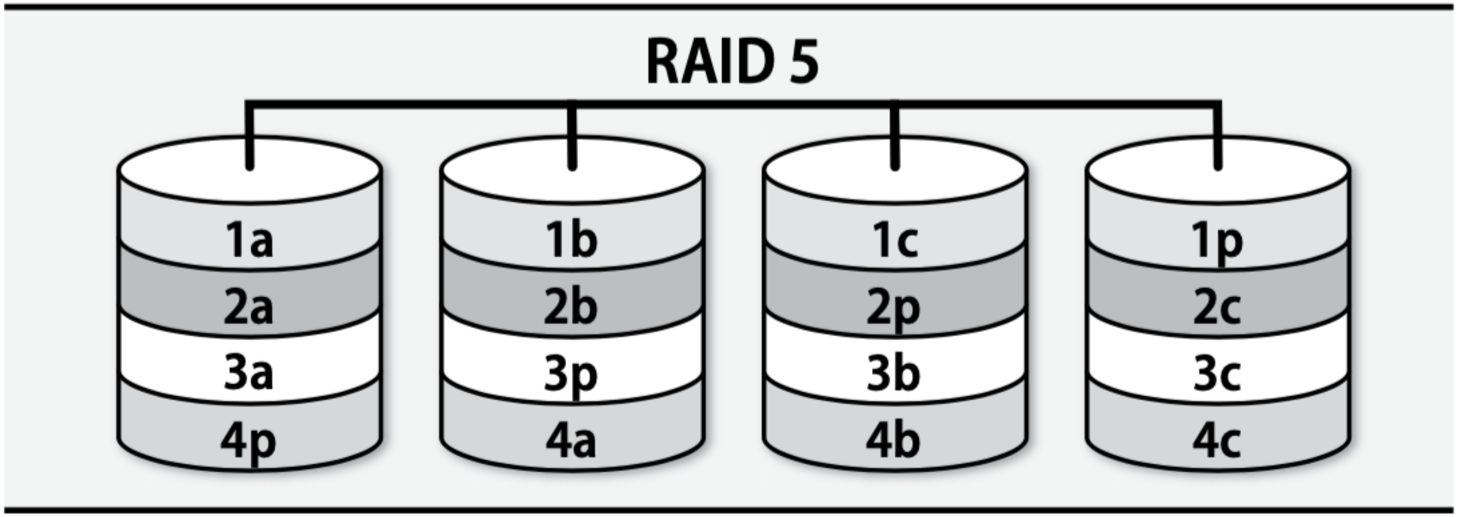


Source: Ref 01

3.7 Storage

3.7.4. Redundant Arrays of Inexpensive Disks

- RAID Level - 5

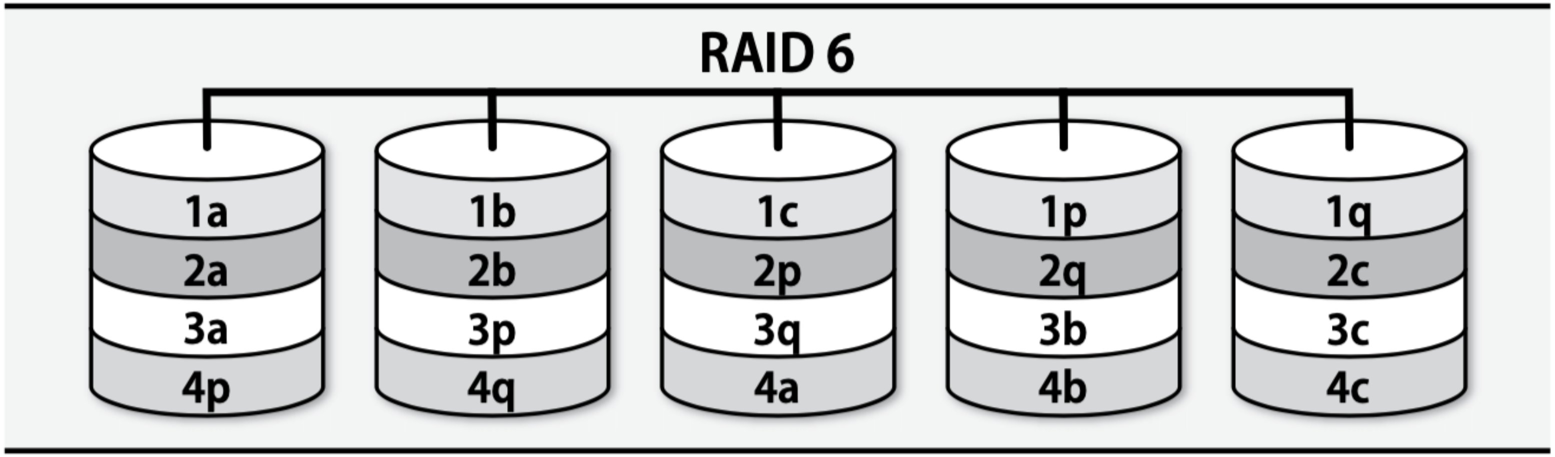


Source: Ref 01

3.7 Storage

3.7.4. Redundant Arrays of Inexpensive Disks

- RAID Level - 6



Source: Ref 01

3.7 Storage

3.7.4. Redundant Arrays of Inexpensive Disks

- Drawbacks of RAID 5
  - Does not replace regular off-line backups
  - Does not have the greatest write performance
  - It is not fool proof from corruption

3.7 Storage

3.7.4. Redundant Arrays of Inexpensive Disks

- **mdadm**: Linux software RAID – Hands on experience
  - Creating an array
  - **mdadm.conf**: document array configuration
  - Simulating a failure

3.7 Storage

3.7.5. Traditional File Systems

- Filesystem terminology
- Filesystem polymorphism
- Filesystem formatting - **mkfs**
- Check and repair filesystems – **fsck**, **tune2fs**
- Filesystem mounting - **mount**
- Setup for automatic mounting - **fstab**
- Swapping recommendations – **mkswap, swapon**

3.7 Storage

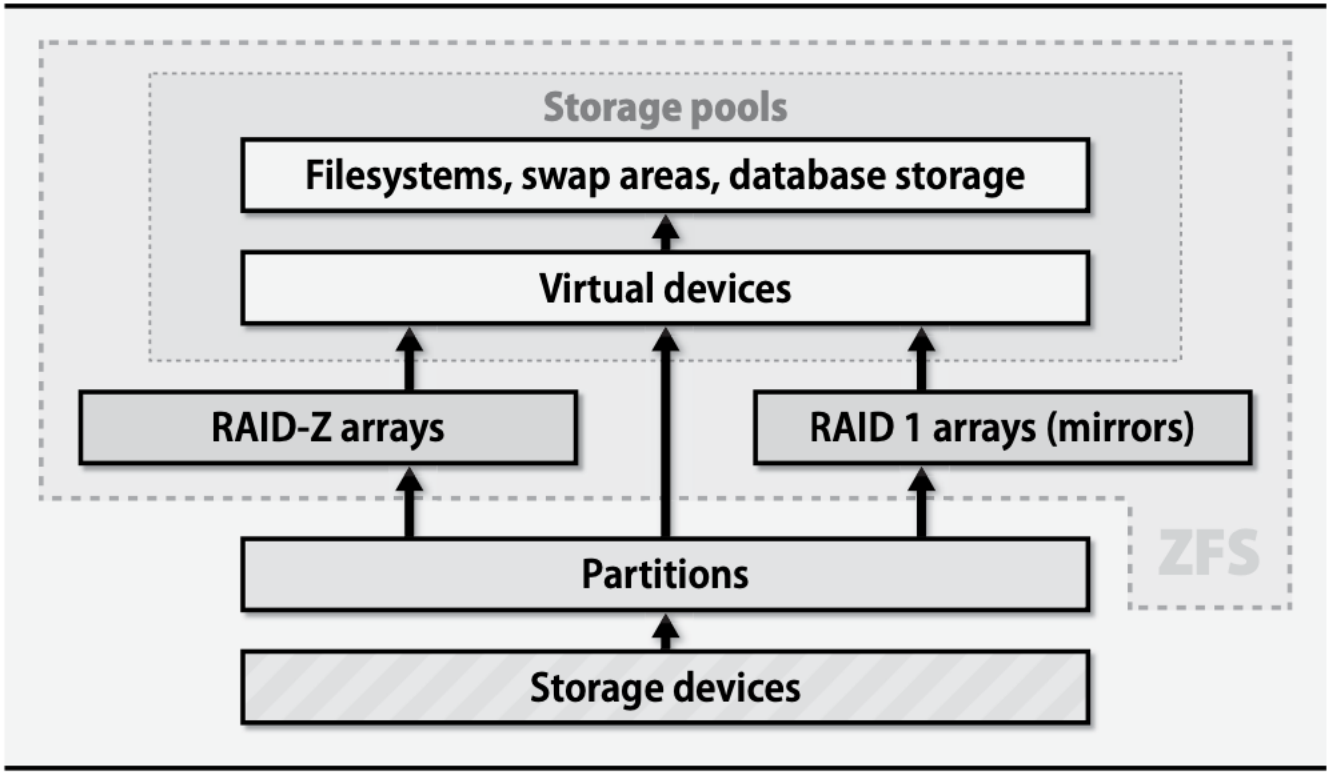
3.7.6. Next Generation File Systems

- Comparison of ZFS and BTRFS
  - Copy on write
  - Error detection
  - Performance

3.7 Storage

3.7.6. Next Generation File Systems

- ZFS Architecture



Source: Ref 01

3.7 Storage

3.7.6. Next Generation File Systems

- ZFS on Linux – Hands on experience
  - Disk addition
  - File systems and properties
  - Property inheritance
  - One filesystem per user
  - Snapshots and clones
  - Raw volumes
  - Storage pool management

3.7 Storage

3.7.6. Next Generation File Systems

- BTRFS for Linux – Hands on experience
  - Setup and storage conversion
  - Volumes and subvolumes
  - Volume snapshots
  - Shallow copies

3.7 Storage

3.7.7. Data Backup Strategies

- Ways that can lose data:
    - Failure of a single facility or piece of hardware
    - Security breach
    - Infection by ransomware
    - Data alteration or corruption though the physical layer

3.7 Storage

3.7.7. Data Backup Strategies

- Overall strategy: Briefly discuss
  - What data is to be backed up?
  - What system or technology will perform the backups?
  - Where will backup data be stored?
  - Will backups be encrypted? If so, where will encryption keys be stored?
  - How much will it cost to store backups over time?

3.7 Storage

3.7.7. Data Backup Strategies

- Timelines: Briefly discuss
  - How often will backups be performed?
  - How often will backups be validated and restore-tested?
  - How long will backups be retained?

3.7 Storage

3.7.7. Data Backup Strategies

- People: Briefly discuss
  - Who will have access to backup data?
  - Who will have access to the encryption keys that protect backup data?
  - Who will be in charge of verifying the execution of backups?
  - Who will be in charge of validating and restore-testing backups?

3.7 Storage

3.7.7. Data Backup Strategies

- Use and protection: Briefly discuss
  - How will backup data be accessed or restored in an emergency?
  - How will you ensure that neither a hacker nor a bogus process can corrupt, modify, or delete backups? (That is, how will you achieve immutability?)
  - How will backup data be protected against being taken hostage by an adversarial cloud provider, vendor, or government?

References

- Ref 1. Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley and Dan Mackin "UNIX and Linux System Administration Handbook " (5th Edition), Pearson Education, Inc., 2018.