Linux File System Administration

©

This course is for you because…

- You are a Linux user with a basic understanding of the Linux operating system and..
- You want to learn advanced operating system topics such as…
- ACLs, quotas, file archive, file links, creating and mounting disk partitions and partition types such as…
- LVM and RAID

You Will Understand

- Linux filesystem concepts and tools
- How to find and search files
- The inode table and file links
- The difference between a hard link and a symbolic link

You Will Be Able To

- Modify file permissions and ACLs
- Use file quotas
- Archive and back up files
- Create and mount disk partitions
- Implement LVMs and RAID

You Should Know and Have

- Know basics of the Linux shell and the Linux filesystem
- Know basic shell commands
- Have access to a Linux installation (I will be using Ubuntu) and…
- Have installed Linux with plenty of unused disk space (we will create several new disk partitions)

File Permissions

File Permissions

- We will discuss:
  - File permission basics
  - Use of `chmod` and `chown`
  - SUID / SGID
  - The often misunderstood writeable directory and the sticky bit
  - Simultaneous file edits

File Permission Basics

File permissions are shown with `ls -l`:

```
student$ ls -l
total 64
-rwxrwxr-x 1 student student   832 Oct  2 16:26 backup.sh
-rwxr-xr-x 1 student student    70 Mar  6  2016 hello.pl
-rwxr-xr-x 1 student student 30832 Mar  6  2016 largefile.txt
-rwxr-xr-x 1 student student    53 Mar  6  2016 quote.txt
-rwxr-xr-x 1 student student   142 Mar  6  2016 records.data
-rw-rw-r-- 1 student student     8 Oct 15 08:54 regex.txt
-rwxr-xr-x 1 student student    53 Mar  6  2016 sed.out
-rwxr-xr-x 1 student student    76 Mar  6  2016 wallquote.txt
```

File Permission Basics

- `r` - read
  `w` - write
  `x` - execute

  The first character can be:
  `-` - regular file
  `d` - directory
  `l` - symbolic link
  `b` - block file
  `c` - character device file
  `p` - named pipe
  `s` - socket

chmod

Change file permissions with `chmod`:

```
student$ ls -l regex.txt
-rw-rw-r-- 1 student student 8 Oct 15 08:54 regex.txt
student$ chmod 600 regex.txt
student$ ls -l regex.txt
-rw------- 1 student student 8 Oct 15 08:54 regex.txt
```

chmod

The numeric value represents:

```
 6    0    0
110  000  000
rw-  ---  ---
```

For 751:

```
 7    5    1
111  101  001
rwx  r-x  --x
```

```
student$ chmod 751 regex.txt
student$ ls -l regex.txt
-rwxr-x--x 1 student student 8 Oct 15 08:54  regex.txt
```

chmod

Symbolic notation can also be used:

```
student$ ls -l regex.txt
-rwxr-x--x 1 student student  8 Oct 15 08:54 regex.txt
student$ chmod -x regex.txt
student$ ls -l regex.txt
-rw-r----- 1 student student  8 Oct 15 08:54 regex.txt
student$ chmod g+w regex.txt
student$ ls -l regex.txt
-rw-rw---- 1 student student  8 Oct 15 08:54 regex.txt
student$ chmod gu-r regex.txt
student$ ls -l regex.txt
--w--w---- 1 student student 8 Oct 15 08:54 regex.txt
```

chmod

```
student$ ls -l regex.txt
--w--w---- 1 student student 8 Oct 15 08:54 regex.txt
student$ chmod g+r,o+x regex.txt
student$ ls -l regex.txt
--w-rw---x 1 student student 8 Oct 15 08:54 regex.txt
student$ chmod u+r,g-w,o-x regex.txt
student$ ls -l regex.txt
-rw-r----- 1 student student 8 Oct 15 08:54  regex.txt
```

chown / chgrp

The `chown` command changes the owner - only `root` can do this:

```
root# ls -l regex.txt
-rw-r----- 1 student student 8 Oct 15 08:54 regex.txt
root# chown jdoe regex.txt
root# ls -l regex.txt
-rw-r----- 1 jdoe student 8 Oct 15 08:54  regex.txt
```

`chown` can also be used to change owner and group:

```
root# chown student:class regex.txt
root# ls -l regex.txt
-rw-r----- 1 student class 8 Oct 15 08:54 regex.txt
```

chown / chgrp

The `chgrp` command changes the group associated with the file:

```
root# chgrp student regex.txt
root# ls -l regex.txt
-rw-r----- 1 student student 8 Oct 15 08:54  regex.txt
```

chown / chgrp

The `chown` and `chgrp` commands can do their work recursively with the `-R` option:

```
root# ls -ld data
drwxrwxr-x 2 student student 4096 Nov  5 13:46 data
root# ls -l data
-rw-rw-r-- 1 student student 0 Nov  5 13:46 a.dat
-rw-rw-r-- 1 student student 0 Nov  5 13:46 b.dat
-rw-rw-r-- 1 student student 0 Nov  5 13:46 c.dat
root# chown -R jdoe data
root# ls -ld data
drwxrwxr-x 2 jdoe student 4096 Nov  5 13:46 data
root# ls -l data
-rw-rw-r-- 1 jdoe student 0 Nov  5 13:46 a.dat
-rw-rw-r-- 1 jdoe student 0 Nov  5 13:46 b.dat
-rw-rw-r-- 1 jdoe student 0 Nov  5 13:46 c.dat
```

SUID and SGID

SUID (*Set User ID*) is an access flag that allows the executable to be executed with the permissions of the executable's owner or group.

SGID (*Set Group ID*) causes files and sub-directories to inherit the GID of the directory.

SUID Executables

SUID is necessary when normal users are executing programs that require elevated privileges, such as the `passwd` utility that modifies `/etc/shadow`:

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 54256 May 16 19:37  /usr/bin/passwd
```

The `s` in place of the owner's `x` permission indicates the program SUID.

SUID Executables

Consider the following C program (`setuid.c`) that will print the real user id and the effective user id:

```
#include <stdio.h>
#include <unistd.h>

int main() {
  int real = getuid();
  int euid = geteuid();
  printf("The REAL UID =      %d\n", real);
  printf("The EFFECTIVE UID = %d\n", euid);
}
```

SUID Executables

Let's execute this program and see how the effective user ID changes to the owner of the program:

```
student$ gcc -o /tmp/setuid setuid.c
student$ ls -l /tmp/setuid
-rwxr-xr-x 1 student student 8172 Oct  16 17:25 /tmp/setuid
student$ /tmp/setuid
The REAL UID =      1001
The EFFECTIVE UID = 1001
student$ su - jdoe
Password:
jdoe$ /tmp/setuid
The REAL UID =      1002
The EFFECTIVE UID = 1002
jdoe$ exit
```

SUID Executables

A file is set to SUID with `chmod`. Note the `x` replaced with `s`:

```
student$ chmod u+s /tmp/setuid
student$ ls -l /tmp/setuid
-rwsr-xr-x 1 student student  8172 Oct  16 17:25 /tmp/setuid
student$ su - jdoe
Password:
jdoe$ /tmp/setuid
The REAL UID =      1002
The EFFECTIVE UID = 1001
jdoe$ exit
```

Using numeric notation (`4xxx`):

```
student$ chmod 4755  /tmp/setuid
```

SGID Directories

Setting the SGID permission on a directory causes new files and subdirectories created within it to inherit its GID. The SGID bit is also inherited. This only affects new directories created, not currently existing directories.

```
student$ mkdir /tmp/test
student$ chmod 777 /tmp/test
student$ ls -l /tmp/test
total 0
student$ ls -ld /tmp/test
drwxrwxrwx 2 student student 4096 Oct 16 18:36 /tmp/test
jsmith$ su - jdoe
Password:
jdoe$ mkdir /tmp/test/foo1
jdoe$ ls -ld /tmp/test/foo1
drwxrwxr-x 2 jdoe jdoe  4096 Mar  4 18:36 /tmp/test/foo1
jdoe$ exit
```

SGID Directories

```
student$ chmod g+s /tmp/test
student$ ls -ld /tmp/test
drwxrwsrwx 3 student student 4096 Oct  16 18:36 /tmp/test
student$ su - jdoe
Password:
jdoe$ mkdir /tmp/test/foo2
jdoe$ ls -ld /tmp/test/*
drwxrwxr-x 2 jdoe jdoe    4096 Oct  16 18:36 /tmp/test/foo1
drwxrwsr-x 2 jdoe student 4096 Oct  16 18:38 /tmp/test/foo2
jdoe$ exit
```

Using numeric notation (2xxx):

```
student$ chmod 2775 /tmp/test
```

Writeable Directory

The writeable directory is often misunderstood.

One can write into and delete files owned by others from a directory that is writeable, even if the file is not readable or writeable:

```
student$ ls -ld /tmp/mydirectory
drwxrwxrwx 2 jdoe jdoe 17 Oct 16 16:17 my directory
student$ ls -l /tmp/mydirectory
-rw------- 1 jdoe jdoe 17 Oct 16 16:19 myfile.txt
student$ cat /tmp/mydirectory/myfile.txt
cat: /tmp/mydirectory/myfile.txt: Permision denied
student$ rm /tmp/mydirectory/myfile.txt
rm: remove write-protected regular file '/tmp/mydirectory/myfile.txt'? y
student$ cat /tmp/mydirectory/myfile.txt
cat: /tmp/mydirectory/myfile.txt: No such file or directory
```

Writeable Directory

Solved with the “sticky bit” created with `chmod +t`. Notice the “other” `x` changes to a `t`:

```
jdoe$ chmod +t /tmp/mydirectory
jdoe$ ls -ld /tmp/mydirctory
drwxrwxrwt 2 jdoe jdoe 4096 Oct 16 17:07 /tmp/mydirctory
jdoe$ touch /tmp/mydirectory/myfile.txt
jdoe$ chmod 600 /tmp/mydirectory/myfile.txt
jdoe$ su - student
Password:
student$ cat /tmp/mydirectory/myfile.txt
cat: /tmp/mydirectory/myfile.txt: Permission denied
student$ rm /tmp/mydirectory/myfile.txt
rm: remove write-protected regular empty file '/tmp/mydirectory/myfile.txt'? y
rm: cannot remove '/tmp/mydirectory/myfile.txt': Operation not permitted
```

`/tmp` is an example of a directory with a sticky bit.

Writeable Directory

Numeric notation can be used to set the sticky bit (`1xxx`):

```
jdoe$ chmod 1777 /tmp/mydirectory
```

Simultaneous File Edits

- files simultaneously edited by two users follow the “last one wins” rule
- if more than one user is editing the same file, the last write overwrites all other writes
- the only way to resolve this issue is by locking files

Exercise

1. Given the following permissions, write the numeric `chmod` argument to create them:

`rwxrw-r--`	`chmod` _____
`r-xr--r--`	`chmod` _____
`rw-rw-rw-`	`chmod` _____
`rw-rw-r--`	`chmod` _____
`r---w---x`	`chmod` _____
`r--------`	`chmod` _____
`r--r--r--`	`chmod` _____

Exercise

2. Given the following permissions, write the symbolic `chmod` argument to create them (cumulative, meaning from the recent permissions to the desired permissions):

`----------.`	
`rw-r--r--`	`chmod _____`
`rw-rw-r--`	`chmod _____`
`rw-rw-rw-`	`chmod _____`
`r---w---x`	`chmod _____`
`r--------`	`chmod _____`
`r--r--r--`	`chmod _____`
`rwxrw-r--`	`chmod _____`
`r-xr--r--`	`chmod _____`

Exercise

3. Compile the program `setuid.c`. Run it as a non-SUID program as another user. Then SUID it, run it again.

4. Create a directory and set the sticky bit (as `root`) with the numeric notation.

File Attributes and ACLs

File Attributes and ACLs

- file attributes
  - the `lsattr` command
  - the `chattr` command
- Access Control Lists (ACLs)
  - the `getfacl` command
  - the `setfacl` command

File Attributes

Linux supports additional file attributes beyond `r`, `w`, `x` and include (see man `chattr` for a complete list):

- `a` - only use append for writing
- `A` - no atime updates
- `c` - automatically compress on disk
- `d` - don't backup with `dump(8)`
- `i` - file cannot be modified (immutable)
- `j` - data journaling
- `s` - when file is deleted, zero out disk

`lsattr` and `chattr`

The `lsattr` command lists the attributes. The `chattr` command changes attributes:

```
root# touch test.txt
root# lsattr test.txt
----------e-- test.txt
root# chattr +i test.txt
root# lsattr test.txt
----i-----e-- test.txt
root# echo hello >> test.txt
-bash: test.txt: Operation not permitted
```

lsattr and chattr

```
root# echo this is a test >> test.txt
-bash: test.txt: Permission denied
root# chattr +a -i test.txt
root# lsattr test.txt
-----a----e-- test.txt
root# echo this is a test > test.txt
-bash: test.txt: Operation not permitted
root# echo this is a test >> test.txt
root# chattr -a test.txt
root# lsattr test.txt
----------e-- test.txt
```

Access Control Lists

Access Control Lists (ACLs) provide more granularity for controls on file and directory permissions, for example:

- remove read/write permissions on a file from the group and others, but
- allow one or two users to have read/write permissions

`getfacl` Command

The `getfacl` command shows a file's ACLs:

```
student$ touch test.txt
student$ chmod 750 test.txt
student$ ls -l test.txt
-rwxr-x--- 1 student student 0 Oct  6 08:54 test.txt
student$ getfacl test.txt
# file: test.txt
# owner: student
# group: student
user::rwx
group::r-x
other::---
```

`setfacl` Command

Now let's give read/write permission to the user `jdoe`, who at this time has no permissions for this file (since that user is in the "other" category):

```
student$ setfacl -m user:jdoe:rw- test.txt
student$ getfacl --omit-header test.txt
user::rwx
user:jdoe:rw-
group::r-x
mask::rwx
other::---
```

`setfacl` Command

Notice the change in the `ls -l` output (the addition of the `+` character):

```
student$ ls -l test.txt
-rwxrwx---+ 1 student student 0 Oct  6 08:54 test.txt
```

`setfacl` Command

Now the user `jdoe` can modify the file:

```
student$ su - jdoe
Password:
jdoe$ echo hi >> /tmp/test.txt
jdoe$ exit
```

`setfacl` Command

We can also apply ACLs to directories. Let's create a directory that `jdoe` cannot write into:

```
student$ mkdir /tmp/acl
student$ chmod 700 /tmp/acl
student$ getfacl --omit-header  /tmp/acl
getfacl: Removing leading '/' from absolute path
names
# file: tmp/acl
# owner: student
# group: student
user::rwx
group::---
other::---
```

`setfacl` Command

Let's see if jdoe can write into it:

```
student$ su - jdoe
Password:
jdoe$ touch /tmp/acl/test.txt
touch: cannot touch `/tmp/acl/test.txt': Permission denied
```

`setfacl` Command

Let's give jdoe read/write permission:

```
student$ setfacl -m user:jdoe:rwx /tmp/acl
student$ getfacl --omit-header /tmp/acl
getfacl: Removing leading '/' from absolute path names
user::rwx
user:jdoe:rwx
group::---
mask::rwx
other::---
student$ ls -ld /tmp/acl
drwxrwx---+ 2 student student  4096 Mar  6 09:26 /tmp/acl
```

setfacl Command

```
student$ su - jdoe
Password:
jdoe$ touch /tmp/acl/test.txt
jdoe$ ls -l /tmp/acl/test.txt
-rw-rw-r-- 1 jdoe jdoe 0 Mar  6 09:26 /tmp/acl/test.txt
jdoe$ exit
```

Exercise

1. Create a file. Using `chattr`, make it immutable. Then try to write to it.

2. Using `chattr`, make the file mutable but only append-able. Then try to write to it. Then try to append to the file.

3. Using `chattr`, make the file automatically compressed. Compare the output of these commands:

```
student$ du test.dat
student$ du --apparent-size test.dat
```

Exercise

4. Create a file. Set the permissions so that only the owner can read and write. Using `setfacl`, allow the user `jdoe` to read and write the file.

5. Create a directory. Set the permissions so only the owner can create files in the directory. Using `setfacl`, allow `jdoe` to create files in the directory, but not able to list its contents.

Locating and Searching Files

Locating and Searching Files

- the `which` command
- the `whatis` command
- the `find` command
- the `locate` command
- `grep` and regular expressions (aka. regexes)

`which` Command

- use `which` to locate an executable

```
student$ which date
/bin/date
```

- `which` searches all directories in the `PATH` variable and prints first found - use `-a` (or `--all`) to print all executables:

```
student$ ls ~/bin/date
~/bin/date
student$ which -a date
/bin/date
~/bin/date
```

`whatis` Command

- a `man` page database can be created with `/usr/sbin/makewhatis` and can be searched with `whatis` - it will display the `man` pages for utilities that exactly match the argument:

```
student$ whatis printf
printf (1) - format and print data
printf (3) - formatted output conversion
student$ whatis prin
prin: nothing appropriate
```

`man -k` Command

- `whatis` finds exact matches - for partial matches, use `man -k` (aka `apropos`)

```
student$ man -k print
banner (6) - print large banner on printer
base64 (1) - base64 encode/decode data and print to standard output
blkid (8)  - command-line utility to locate/print block device attributes
cat (1)    - concatenate files and print on the standard output
date (1)   - print or set the system date and time
```

`man -K` Command

- `man -K` searches the content of man pages (not just titles):

```
student$ man -K Linus
--Man-- next: git-read-tree(1) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]

--Man-- next: git-describe(1) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]

--Man-- next: git-diff-tree(1) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]
```

`find` Command

- `find` searches for files - it has a lot of options, but a simple use is to look in `/usr` for the file `zip`:

```
student$ find /usr -name zip
/usr/bin/zip
/usr/share/doc/zip
```

- if there is standard error output, it can be suppressed with:

```
student$ find /usr -name zip 2>/dev/null
```

`find` Command

- here are some examples of using `find`:

```
# find all files that have been accessed more than
# 1 day ago
find / -atime 1

# find all files created this year
touch --date="12:00 am jan 1 2020" /tmp/ref_time
find / -newer /tmp/ref_time

# get list of all newly updated videos
find /filer/videos -type f -mtime 1
```

`find` Command

```
# strip suid bits from anything in /home
find /home -perm -4000 -exec chmod u-s {}  \;

# fix the permissions on /var/www/htdocs
# make it so that everyone in www-data group can
# create and edit all files, and sgid bit on  dir
# assures that all files are created with www-data
# group ownership as well
chown -R www-data:www-data /var/www/htdocs
find /var/www/htdocs -type d -exec chmod 2775 {} \;
find /var/www/htdocs -type f -exec chmod 664 {}  \;
```

`locate` Command

- the `locate` command quickly searches for files on the system by searching the database created by `updatedb`
- note: `updatedb` should be run by `root` daily and is usually done so through `cron`

```
student$ locate locate
/bin/ntfsfallocate
/etc/alternatives/locate
/etc/alternatives/locate.1.gz
/etc/cron.daily/mlocate
/usr/bin/fallocate
/usr/bin/locate
/usr/bin/mlocate
...
```

`locate` Command

- `locate` has many options:
  - `-i` - case insensitive search
  - `-n` - limit the lines of output to the provided number
  - `-e` - exclude provided directories
  - `-r` - filter output with supplied regex
  - `-d` - path of databases to search

Regular Expressions

- a regular expression (aka *regex*) is a sequence or pattern of characters that is compared against a string of text

- regexes are used to perform text matching, text replacement, text extraction and various other text manipulation operations

- the `grep` command uses regexes to look for text in files; this example displays all lines of the file `a.dat` that match the regex `xyz`:

```
student$ grep xyz a.dat
```

Patterns: Single Characters

- most characters match themselves:
  - `a` - matches “a” anywhere
  - `ab` - matches “ab” anywhere
  - `abc` - matches “abc” anywhere
- there are some special characters:
  - `.` - (the period) matches any character but newline
  - `^` - matches the beginning of the string
  - `$` - matches the end of the string

Patterns: Character Classes

- created using square brackets - matches one occurrence of any character in the class:
  - `[abcde]` - matches one of either “a”, “b”, “c”, “d” or “e”
  - `[a-e]` - the same
  - `[0-9]` - matches one digit
- if the carat (^) is the first character in the class, it matches any character except what is in the class:
  - `[^abcde]` - any character except "a", "b", "c", "d" and "e"
  - `[^a-e]` - the same
  - `[^0-9]` - matches one non-digit

Patterns: Quantifiers

- the number of characters can be specified with *quantifiers*:
  - `*` - 0 or more
  - `+` - 1 or more
  - `?` - 0 or 1
  - `\{m\}` - exactly `m`
  - `\{m,\}` - `m` or more
  - `\{m,n\}` - `m` through `n`, inclusive

Patterns: Quantifiers

Examples:

```
# one or more e in a row in largefile.txt
grep e+ largefile.txt

# exactly 2 e in a row in largefile.txt
grep "e\{2\}" largefile.txt

# 2 or more vowels in a row in largefile.txt
grep "[aeiou]\{2,\}" largefile.txt

# one or more lower or upper e at the beginning
# of the line in largefile.txt
grep ^[eE]+ largefile.txt
```

`grep` Command

The `grep` command attempts to match every line of a file against a regular expression. If the regex matches, the line is printed:

```
student$ grep abc file.txt
```

The `-i` option ignores case:

```
student$ grep -i abc file.txt
```

The `-v` prints all lines that do not match:

```
student$ grep -v abc file.txt
```

`grep` Command

If no file argument is provided, then read from standard input:

```
student$ ls -l | grep ^d
```

The -f option uses the regexes in the provided file:

```
student$ cat regex.txt
thou
^G
student$ grep -f regex.txt largefile.txt
```

Exercise

1. For these words:

```
red
reed
read
dread
bread
head
lead
```

Determine with words match each of these regexes:

```
^ead
^.ead
^.[ea]+d
r[a-z]\{4\}
.[a-z]\{3,4\}
[b-z]\{1,5\}
```

Exercise

2. Create regexes to do the following:

- 0 or 1 "a" followed by any number of “b"
- one or more digits followed by one or more alpha characters

Exercise

3. Search for the following text using `grep`:

- all lines of `largefile.txt` with the text "thee"
- all lines of `largefile.txt` with the text "thee" regardless of case (hint: `-i` is helpful)
- all the blank lines in `largefile.txt`
- all lines of `largefile.txt` with two vowels next to one another, but not 3 vowels

Exercise

4. Find the location of these programs when executed from the shell

- `who`
- `date`
- `ifconfig`

5. Find all files that were modified 2 days ago

Inodes, Links and Low-level File Data

Inodes, Links and Low-level File Data

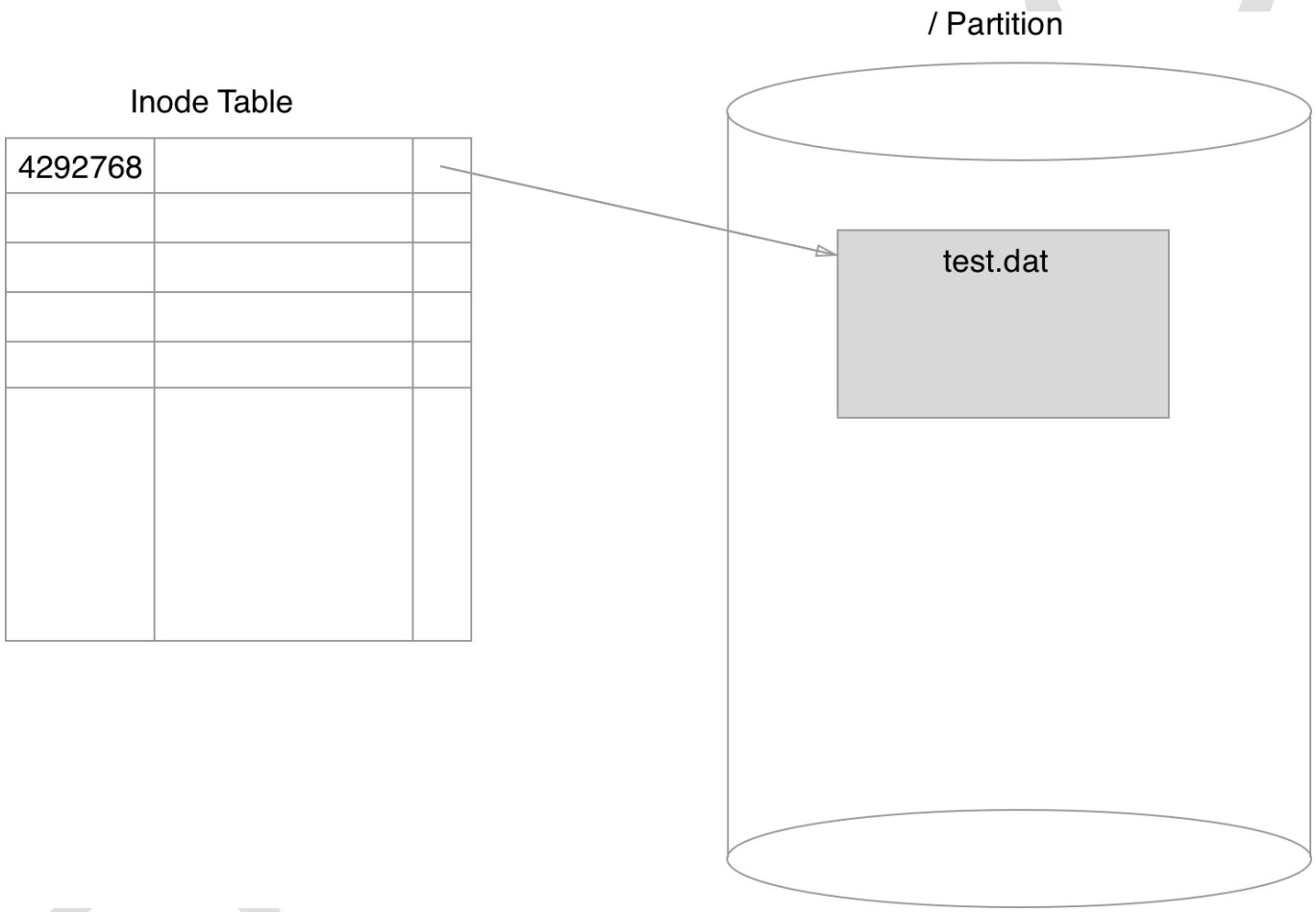
- inodes
- hard links
- symbolic links
- low-level file data (`stat`, `readlink`)

Inodes

An *inode* is a unique number given to every file (or directory, etc.) stored on the disk. To display a file's inode number, use `ls -i`:

```
student$ touch test.dat
student$ ls -il test.dat
4292768 -rw-rw-r-- 1 student student 0 Apr 16 13:07 test.dat
```

Inodes



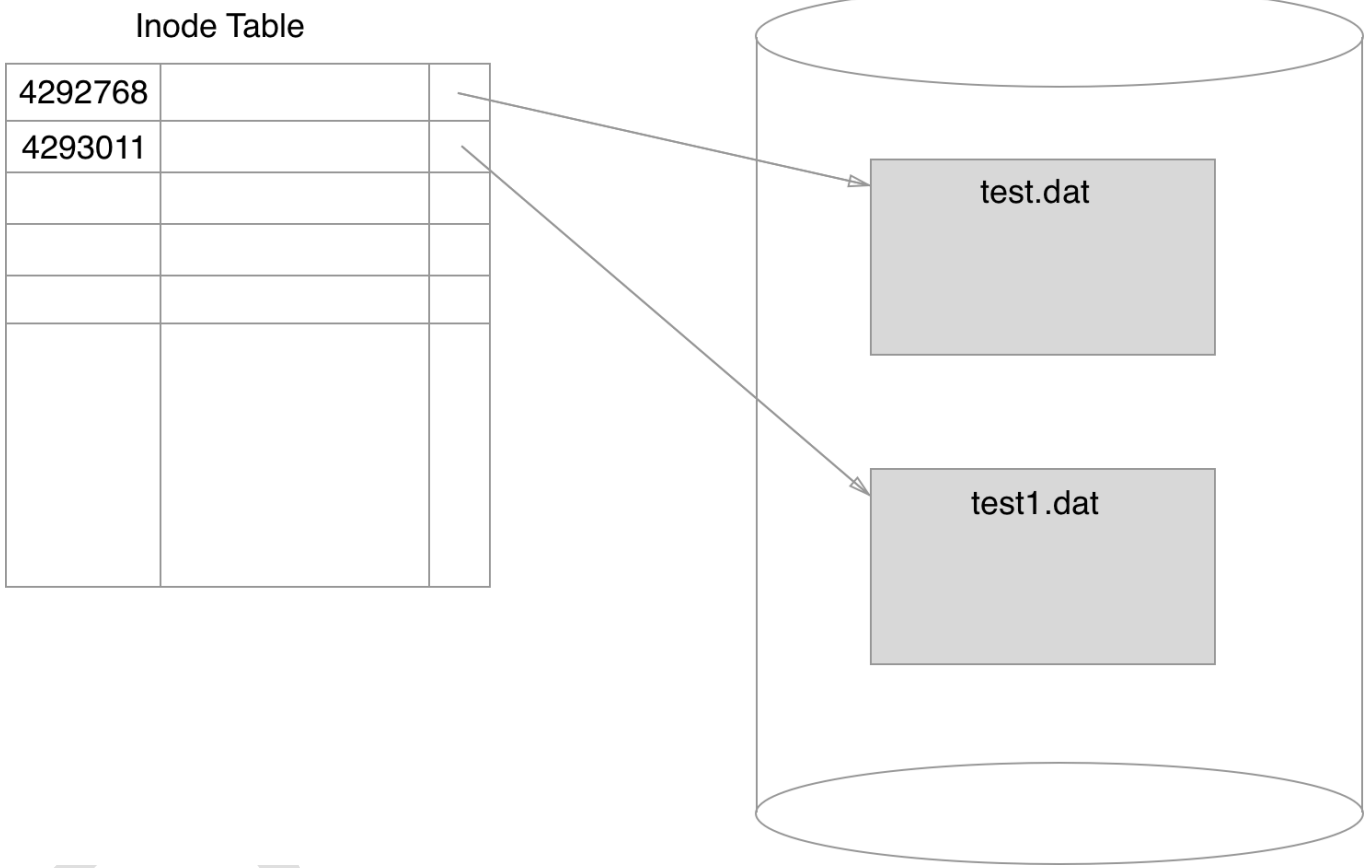
Inodes

When a file is copied, another file is created, with its own unique inode number:

```
student$ cp test.dat test1.dat
student$ ls -il test*.dat
4293011 -rw-rw-r-- 1 student student 0 Oct 6 13:07  test1.dat
4292768 -rw-rw-r-- 1 student student 0 Oct 6 13:07  test.dat
```

When a file is removed, the inode number is released, and if there are no other inodes pointing to it, the file is deleted from disk.

Inodes



Inodes

A directory is a special file that lists other files in the directory and their inode numbers:

```
student$ mkdir inode.dir
student$ ls -lid inode.dir
4293848 drw-rw-r-- 1 student student 0 Oct 6 13:07 inode.dir
student$ touch inode.dir/a.dat
student$ touch inode.dir/b.dat
student$ touch inode.dir/c.dat
```

Inodes

Using `vim` to edit the directory:

`student$` **`vim inode.dir`**

We would see:

```
" ============================================================================
" Netrw Directory Listing                                        (netrw v125)
"   /home/student/inode.dir
"   Sorted by      name
"    Sort sequence: [\/]$,\.h$,\.c$,\.cpp$,*,\.o$,\.obj$,\.info$,\.swp$,\.bak$,\~
"  Quick Help: &lt;F1>:help  -:go up dir  D:delete  R:rename  s:sort-by  x:exec
" ============================================================================
../
a.dat
b.dat
c.dat
```

Links

Linux has two types of links:

- hard links
  - not a new file on disk
  - an entry in its directory that has the same inode as the physical file to which it is linked
  - must reside on the same disk partition as the file to which it is linked

Links

- symbolic links
  - a new file on disk
  - an alias to another file
  - its “content” is the name of the file to which it is linked
  - can reside on a different disk partition as the file to which it is linked

Links

- they are both create using the `ln` command
- advantages
  - can save disk space - one copy of the file exists on the disk
  - if link to the file is edited, the file is modified for all links

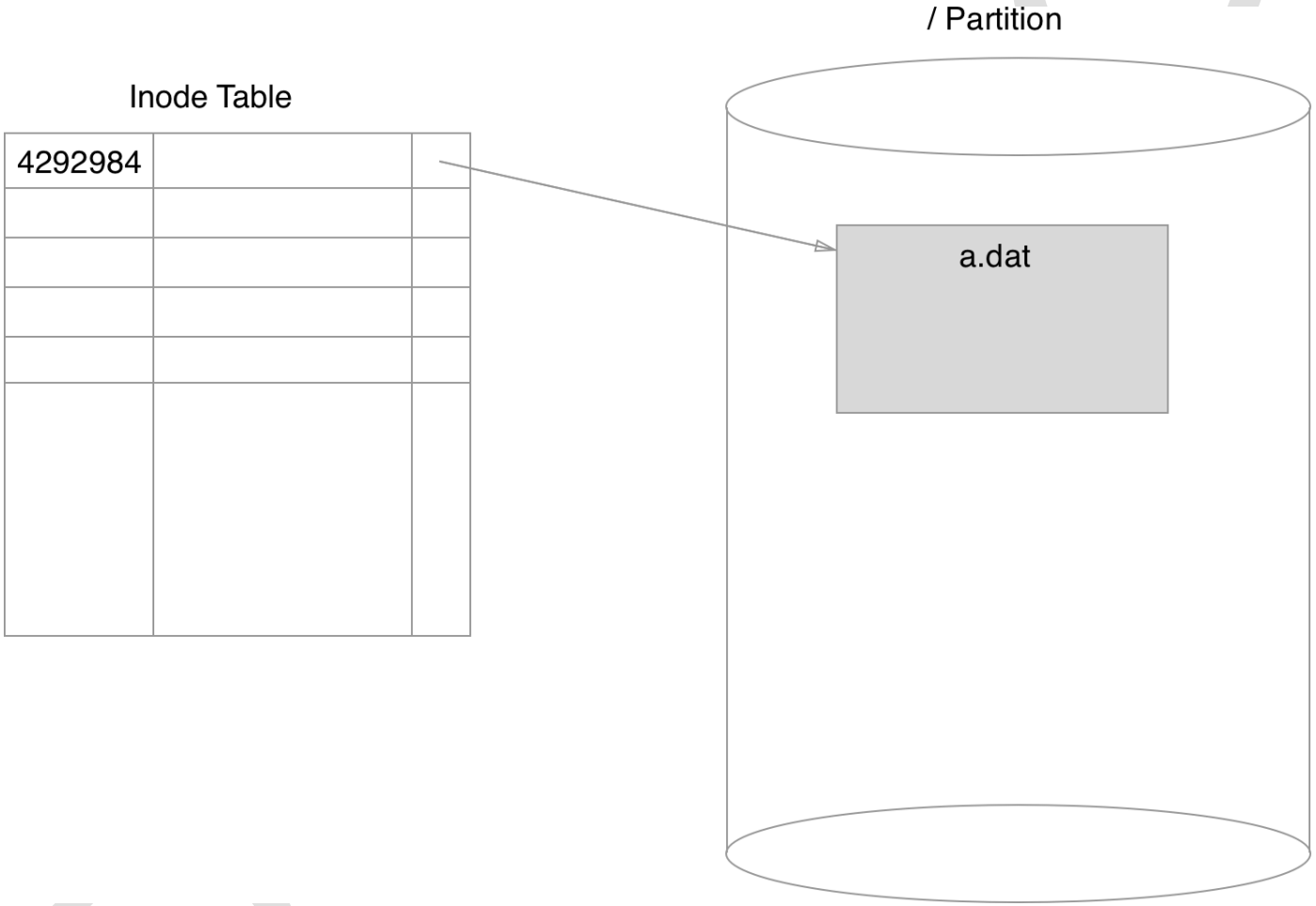
Hard Links

A hard link is another entry in a directory with the same inode as the file to which it links. Consider this file:

```
student$ cat a.dat
this is the file a.dat
student$ ls -li a.dat
4292984 -rw-rw-r-- 1 student student 23 Oct  6 16:02 a.dat
```

This shows that the inode number of this file is `4292984` and the number of hard links to this file (the number just to the left of the username) is `1`. This means that this file's inode number is only listed in one directory (the current directory).

Hard Links



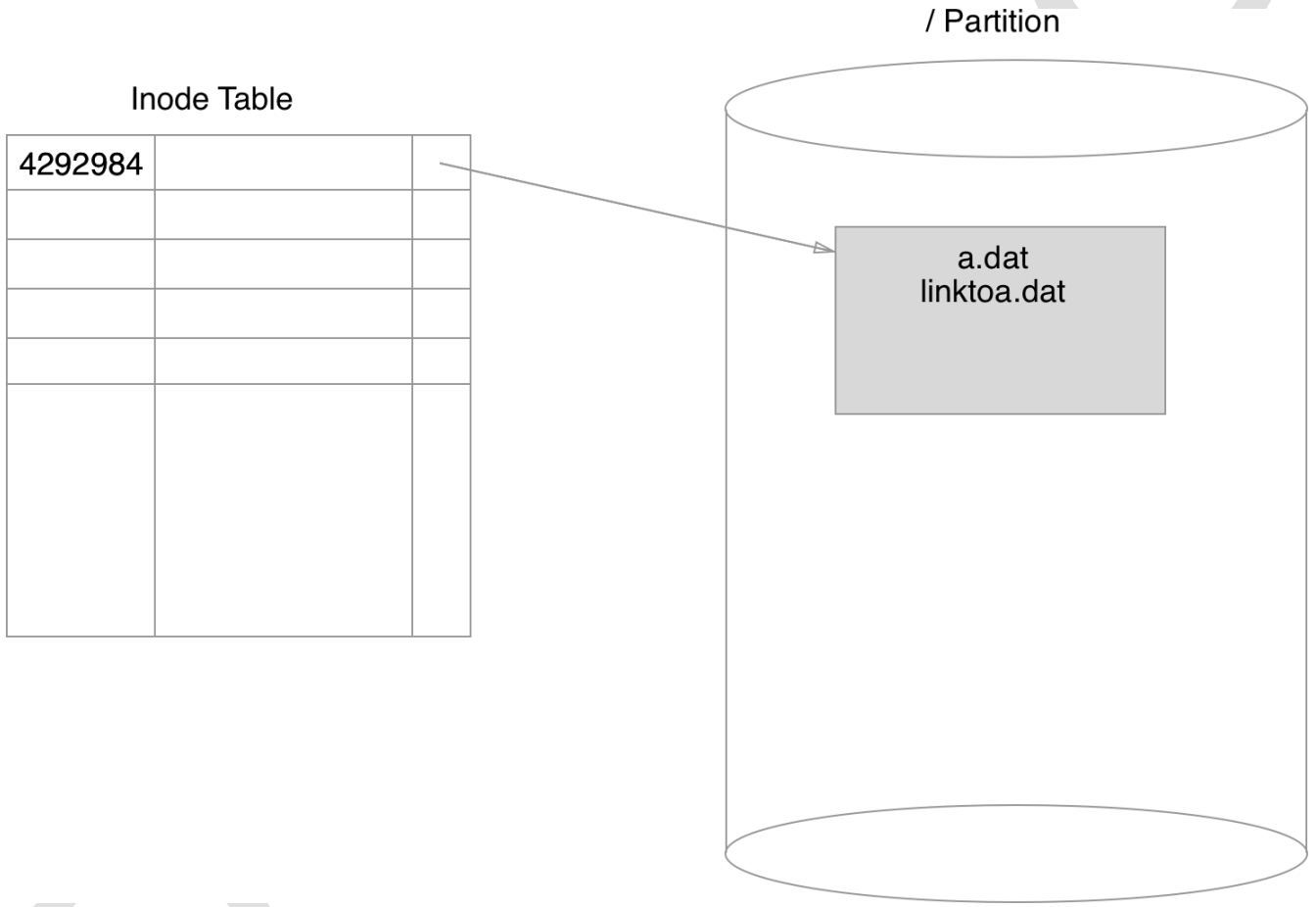
Hard Links

Let's create a hard link to this file - we'll call it `linktoa.dat`:

```
student$ ln a.dat linktoa.dat
student$ ls -li a.dat linktoa.dat
4292984 -rw-rw-r-- 2 student student 23 Oct  6 16:02 a.dat
4292984 -rw-rw-r-- 2 student student 23 Oct  6 16:02 linktoa.dat
```

Notice that the inode numbers of these files are the same (which means they are the same file physically on the disk). Also notice the number to the left of the username is now `2` since there are now 2 links to the file.

Hard Links



Hard Links

Let's show the contents of `linktoa.dat`:

```
student$ cat linktoa.dat
this is the file a.dat
```

Now let's delete the original file `a.dat` and show that the physical file still exists:

```
student$ rm a.dat
student$ ls -li linktoa.dat
4292984 -rw-rw-r-- 1 student student 23 Oct  6 16:02 linktoa.dat
student$ cat linktoa.dat
this is the file a.dat
```

Hard Links

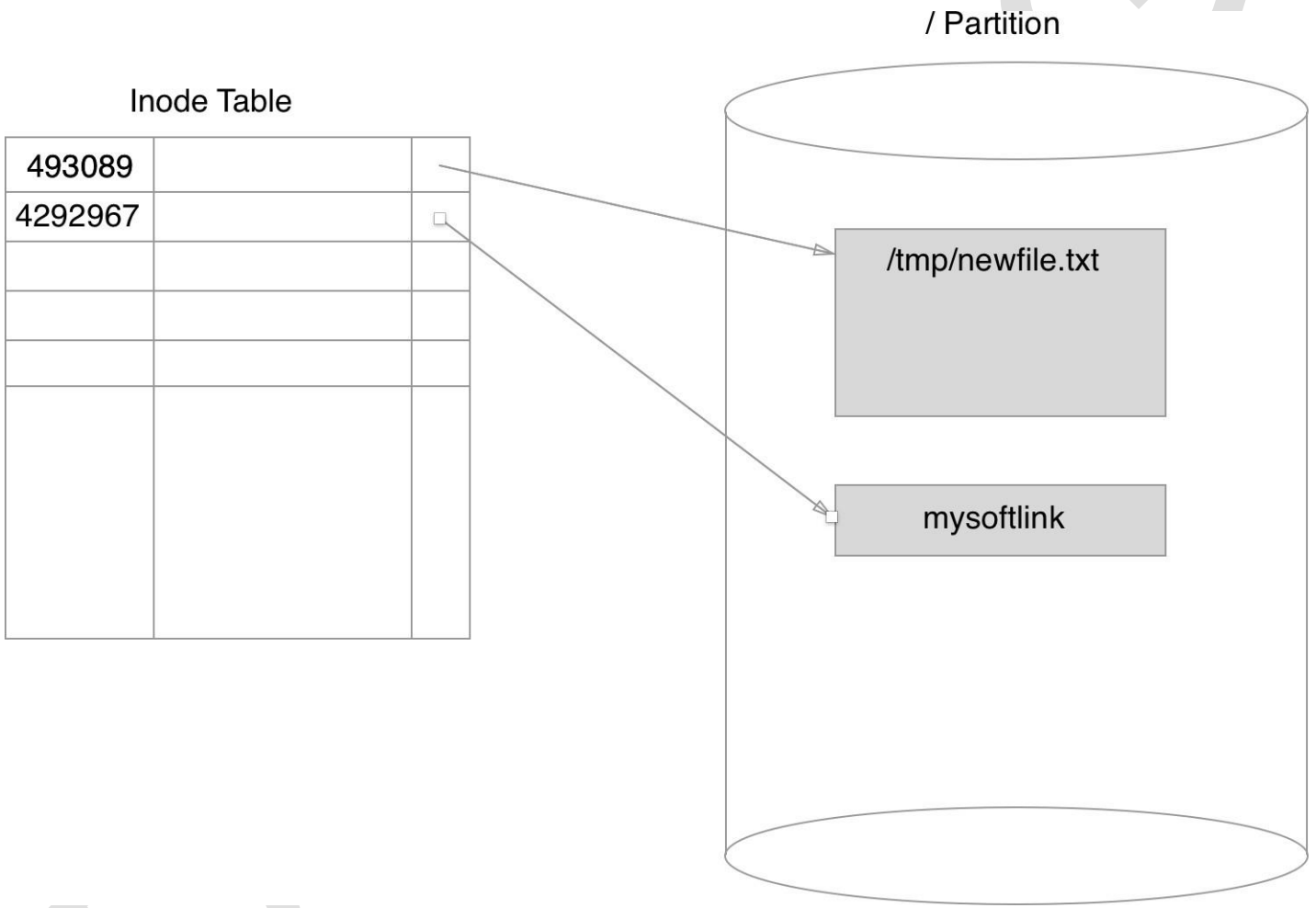
Now the link count is down to `1`. Once the link count decreases to `0`, the file is removed from disk.

Symbolic Links

A symbolic link (aka *soft link*) is entirely new file with its own inode, however it is a special type of file. Its "content" is the name of the file to which it is linked. It is created with `ln -s`:

```
student$ echo this is a test of symbolic link > /tmp/newfile.txt
student$ cat /tmp/newfile.txt
this is a test of symbolic link
student$ ls -il /tmp/newfile.txt
493089 -rw-rw-r-- 1 student student 28 Apr  3 18:02 /tmp/newfile.txt
student$ ln -s /tmp/newfile.txt mysoftlink
student$ ls -il /tmp/newfile.txt mysoftlink
4292967 lrwxrwxrwx 1 student student 16 Apr  3 18:03 mysoftlink -> /tmp/newfile.txt
 493089 -rw-rw-r-- 1 student student 28 Apr  3 18:02 /tmp/newfile.txt
student$ cat mysoftlink
this is a test of symbolic link
```

Symbolic Links



Symbolic Links

Note:

- `/tmp/newfile.txt` has the inode number `493089`
- when a symbolic link is linked to it, the symbolic link has a different inode number (here `4292967`), so it is an entirely new file on disk
- the first character of the permissions is an `l` and not a dash
- the size of the link is `16`, the number of characters in the filename `/tmp/newfile.txt`

Symbolic Links

When the link is `cat`-ed out, it shows the content of the file to which it is linked.

It is possible to delete the file that the sim link is linked to, and the sim link will still exist, although it now links to a non-existent file:

```
student$ rm /tmp/newfile.txt
student$ ls -li mysoftlink
4292967 lrwxrwxrwx 1 student student 16 Oct  6 18:03 mysoftlink -> /tmp/newfile.txt
jdoe$ cat mysoftlink
cat: mysoftlink: No such file or directory
```

Inode Pointer Structure

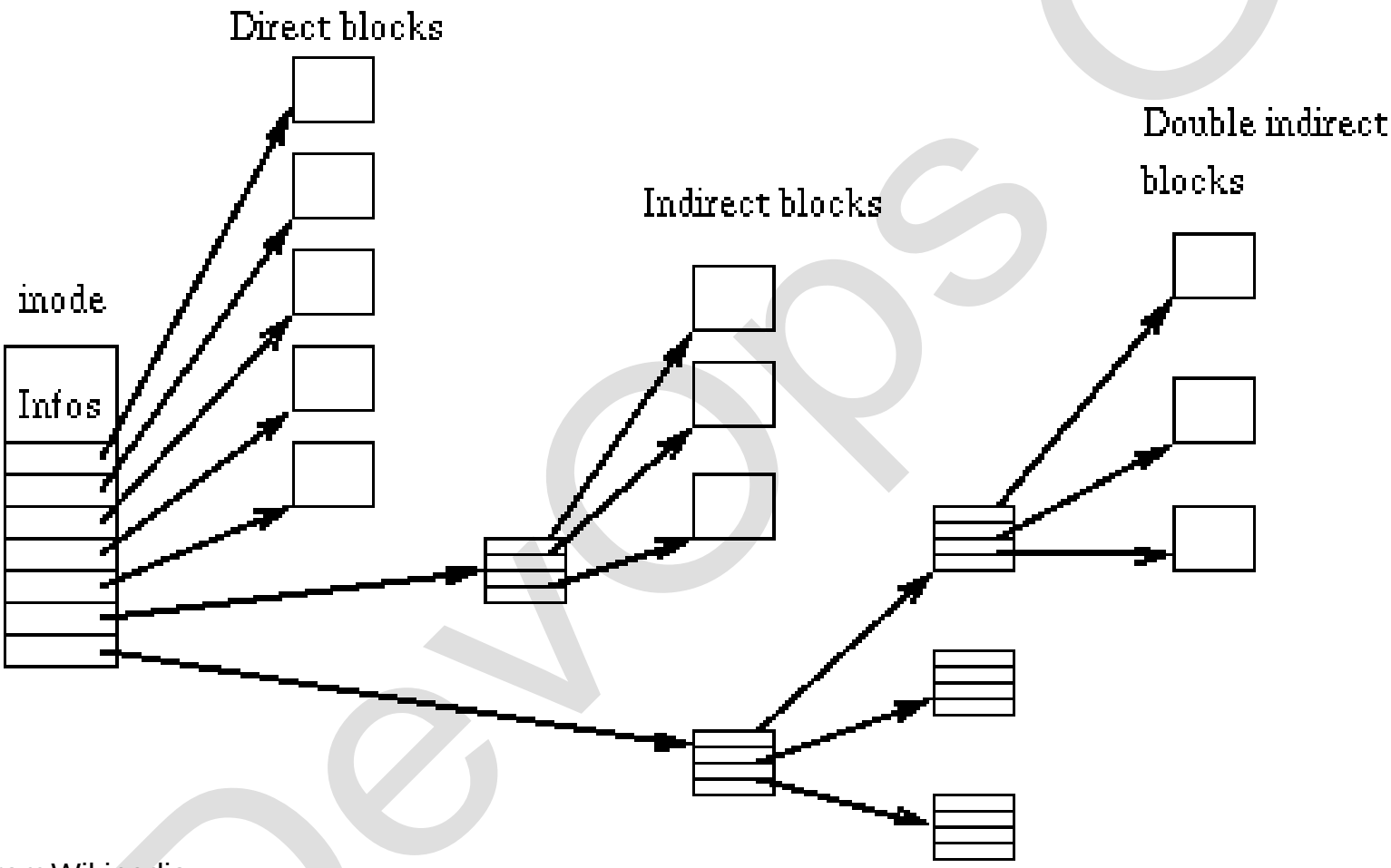


Image from Wikipedia

Low Level File Data

File meta data is available through several low level file commands including:

- `stat`
- `readlink`

stat

The stat command displays a file's system status:

```
student$ stat regex.txt
  File: 'regex.txt'
  Size: 8           Blocks: 8          IO Block: 4096   regular file
Device: 802h/2050d Inode: 436131      Links: 1
Access: (0640/-rw-r-----)  Uid: ( 1000/student)   Gid: ( 1000/student)
Access: 2017-11-05 15:58:01.192000000 -0500
Modify: 2017-10-15 08:54:25.893694474 -0400
Change: 2017-11-05 13:44:30.628000000 -0500
 Birth: -
student$ stat -t regex.txt
regex.txt 8 8 81a0 1000 1000 802 436131 1 0 0 1509915481 1508072065
1509907470 0 4096
```

stat

The `-f` option shows status of a file system:

```
student$ stat -f /dev/sda1
  File: "/dev/sda1"
    ID: 0        Namelen: 255     Type:  tmpfs
Block size: 4096       Fundamental block size: 4096
Blocks: Total: 249974     Free: 249974     Available: 249974
Inodes: Total: 249974     Free: 249516
student$ stat /dev/sda1
  File: '/dev/sda1'
  Size: 0           Blocks: 0          IO Block: 4096   block special file
Device: 6h/6d   Inode: 11567       Links: 1     Device type: 8,1
Access: (0660/brw-rw----)  Uid: (    0/    root)   Gid: (    6/    disk)
Access: 2017-11-06 07:24:05.148000000 -0500
Modify: 2017-11-06 07:24:05.148000000 -0500
Change: 2017-11-06 07:24:05.148000000 -0500
 Birth: -
```

stat

The `--format` constructs a format string (for all options, see `man stat`):

```
student$ stat --format=%g regex.txt
1000
student$ stat --format=%G regex.txt
student
student$ stat --format=%i regex.txt
436131
student$ stat --format="%g %G %i" regex.txt
1000 student 436131
```

readlink

The `readlink` command shows the file a symbolic link links to:

```
student$ ls -l /etc/rc2.d/S02cron
lrwxrwxrwx 1 root root 14 Sep 28 15:38 /etc/rc2.d/S02cron -> ../init.d/cron
student$ readlink /etc/rc2.d/S02cron
../init.d/cron
```

Exercise

1. Create a hard link to `regex.txt`. Show the inode for `regex.txt` and the link. Show the contents of the link. Edit the link, then show the contents of `regex.txt`. Show information about the link and `regex.txt` using `stat`. Remove the link.

2. Create a sim link to `regex.txt`. Edit the sum link, then show the contents of `regex.txt`. Show the size of the link and of `regex.txt`. Show information about the link and `regex.txt` using `stat`. Show the file that is linked with `readlink`. Remove the link.

Exercise

3. Using `vim`, view the contents of the current directory.

Archive and Backups

Archive and Backups

- It is common to archive files and directories for the following reasons:
  - backing up files and directories
  - transporting or copying files and/or directories from one machine to another
- we will discuss these three archive commands:
  - `zip` / `unzip`
  - `gzip` / `gunzip`
  - `tar`

zip/unzip

A common compression utility on Linux is `zip`. It will compress using an algorithm compatible with PKZIP used in the Windows world.

To compress a file:

```
student$ zip largefile.zip largefile.txt
  adding: largefile.txt (deflated 57%)
student$ ls -l largefile.*
-rwxr-xr-x 1 student student  30832 Oct  1 16:56 largefile.txt
-rw-rw-r-- 1 student student  13285 Oct  2 08:23 largefile.zip
```

zip/unzip

To recursively compress the contents of a directory:

```
student$ zip -r temp.zip temp.dir
  adding: temp.dir/ (stored  0%)
  adding: temp.dir/temp.dat (stored 0%)
student$ ls -l temp.zip
-rw-rw-r-- 1 student student 347 Oct  2 08:35 temp.zip
```

See `man zip` for details.

zip/unzip

To uncompress a file that was compressed with `zip`, use `unzip`:

```
student$ mkdir newtemp.dir
student$ cp largefile.zip temp.zip newtemp.dir
student$ cd newtemp.dir
student$ unzip largefile.zip
Archive:  largefile.zip
  inflating: largefile.txt
student$ ls -l largefile.*
-rwxr-xr-x 1 student student 30832 Oct  1 16:56 largefile.txt
-rw-rw-r-- 1 student student 13285 Oct  2 08:46 largefile.zip
```

zip/unzip

```
student$ unzip temp.zip
Archive:  temp.zip
   creating: temp.dir/
 extracting: temp.dir/temp.dat
student$ ls -Rl temp.dir
temp.dir:
total 4
-rw-rw-r-- 1 student student 17 Oct  2 08:35 temp.dat
```

See `man unzip` for details.

gzip/gunzip

Alternatives to `zip` and `unzip` are the GNU programs: `gzip` and `gunzip`.

```
$ ls -l largefile.txt
-rwxr-xr-x 1 student student 30832 Nov  7 14:39 largefile.txt
$ gzip largefile.txt
$ ls -l largefile.txt.gz
-rwxr-xr-x 1 student student 13141 Nov  7 14:39 largefile.txt.gz
$ gunzip largefile.txt.gz
$ ls -l largefile.txt
-rwxr-xr-x 1 student student 30832 Nov  7 14:39 largefile.txt
```

gzip/gunzip

Notice what happens when you recursively zip up a directory:

```
$ gzip -rv data
data/a.dat:         0.0% -- replaced with data/a.dat.gz
data/b.dat:         0.0% -- replaced with data/b.dat.gz
data/c.dat:         0.0% -- replaced with data/c.dat.gz
$ ls data
a.dat.gz  b.dat.gz  c.dat.gz
```

`tar`

- The most common Linux archive utility is `tar` (Tape ARchive) - it will take a collection of files and directories (and their contents) and collect them together into a single file

tar

```
student$ tar cvf /tmp/mystuff.tar *
class/
class/quote.txt
class/sed.out
class/wallquote.txt
class/hello.pl
class/records.data
class/largefile.txt
Desktop/
Documents/
Downloads/
examples.desktop
Music/
Pictures/
Public/
Templates/
Videos/
```

tar

- this command uses the `cvf` options which "creates, verbose, file" `mystuff.tar`

- file details:

```
student$ file /tmp/mystuff.tar
/tmp/mystuff.tar: POSIX tar archive (GNU)
student$ ls -l /tmp/mystuff.tar
-rw-rw-r-- 1 student student 61440 Oct  1 16:56 /tmp/mystuff.tar
```

`tar` Options

- here are some common options:
  - `c` - createfile
  - `x` - extractfile
  - `f` - filename
  - `v` - verbose
  - `z` - zip (compress)
  - `r` - append
  - `t` - list contents

`tar` Append

Here is an example of appending a file to our tarfile `/tmp/mystuff.tar`, and then listing its contents:

```
student$ tar rvf /tmp/mystuff.tar /etc/passwd
tar: Removing leading `/' from member names
/etc/passwd
student$ tar tf /tmp/mystuff.tar
class/
class/quote.txt
class/sed.out
class/wallquote.txt
class/hello.pl
class/records.data
class/largefile.txt
Desktop/
Documents/
Downloads/
examples.desktop
...
etc/passwd
```

`tar` Extract

- we can then extract the tar file

```
student$ mkdir /tmp/tartest
student$ cd /tmp/tartest
student$ tar xvf /tmp/mystuff.tar
class/
class/quote.txt
class/sed.out
class/wallquote.txt
class/hello.pl
class/records.data
class/largefile.txt
...
Videos/
etc/passwd
student$ ls -F
class/  Desktop/  Documents/ Downloads/  etc/    examples.desktop
Music/  Pictures/  Public/  Templates/  Videos/
```

`tar` Compression

- compression is done with the `z` option
- the compressed file is much smaller than the uncompressed file
- the file is usually named either `.tar.gz` or `.tgz`

`tar` Compression

```
student$ tar czf /tmp/mystuff.tar.gz *
student$ tar czf /tmp/mystuff.tgz *

student$ ls -l /tmp/mystuff.*
-rw-rw-r-- 1 student student 61440 Oct  1 16:56 /tmp/mystuff.tar
-rw-rw-r-- 1 student student 18158 Oct  1 16:59 /tmp/mystuff.tar.gz
-rw-rw-r-- 1 student student 18158 Oct  1 16:59 /tmp/mystuff.tgz
```

`tar` Compression

Note: a `.tar.gz` file can be unzipped with `gunzip`:

```
student$ ls -l mystuff.tar.gz
-rw-rw-r-- 1 student student 18158 Oct  1 17:38 mystuff.tar.gz
student$ gunzip mystuff.tar.gz
student$ ls -l mystuff.tar*
-rw-rw-r-- 1 student student 61440 Oct  1 17:38 mystuff.tar
```

Backup Strategies

- Failures happen:
  - hardware failure
  - power failure
  - theft
  - fire / earthquake / etc
  - security breach

Backup Strategies

- Backup to another location:
  - another piece of hardware
  - server in the cloud
  - backup service

Backup Strategies

- Backup regularly:
  - machine on the network
  - server in the cloud
  - backup service

Exercise

1. Zip up the `class` directory (using `zip`). Unzip it in `/tmp` (using `unzip`) and verify that all the files were transferred.

2. Do the same with `tar`.

3. Compare the size of a `.zip` file to a `.tgz` file.

End of Day 1

Day 2 Topics

- filesystem types
- disk partitioning
- swap space
- quotas
- LVM
- RAID

Filesystem Types

Filesystem Types

- ext{2,3,4}
- JFS
- ReiserFS
- Btrfs
- xfs

ext{2,3,4}

ext - the extended file system, was the first filesystem specifically created for the Linux kernel

- created in April 1992
- included in Linux version 0.96c
- inspired by the Unix File System (UFS), designed to overcome Minix limitations (max 64 MB and 14 character filenames)
- could handle filesystems up to 2 Gig and 255 character filenames
- no support of separate timestamps for file access, inode modification, and data modification

ext{2,3,4}

ext2 - the second extended file system

- developed in January 1993 for Linux kernel 0.99, designed with extensibility in mind
- added support for separate timestamps for file access, inode modification, and data modification
- still recommended over journaling file systems (like ext3) on bootable USB flash drives (faster access)
- max partition size: 4TB, max file size: 2GB-2TB (depending on kernel)
- with Large File Support (LFS) - max partition size: 2TB-32TB; max file size: 2GB-2TB

ext{2,3,4}

ext3 - the third extended file system

- released in November 2001 for Linux kernel 2.4.15
- added journaling (keeps track of uncommitted changes in a journal)
  - improves reliability
  - eliminates the need to check the file system after an unclean shutdown
- ext2 can be changed to ext3 by adding journaling with (be sure to modify `/etc/fstab`):

```
tune2fs -j /dev/
```

ext{2,3,4}

ext4 - the fourth extended file system

- released on 25 December 2008 in the 2.6.28 kernel
- features include
  - partition size up to 1 EiB (2**60 bytes); file size up to 16 TiB (Redhat recommends xfs for large files)
  - uses extents - a single extent can map up to 128 MiB of contiguous space with a 4 KiB block size
  - backward compatible with ext2 and ext3 - can mount ext2 or ext3 as ext4
  - persistent pre-allocation and delayed allocation
  - unlimited number of sub-directories

JFS

JFS - Journaled File System - 64 bit journaling filesystem created by IBM. Features include:

- June 2001 kernel 2.4.18pre9-ac4
- fast and reliable with good performance under different kinds of loads
- unlike ext3, journaling was built in from the start
- uses a B+ Tree
- dynamic inode allocation
- extents
- most chose ext4 over JFS

ReiserFS

ReiserFS was developed by Hans Reiser and a team at Namesys. The first Linux version was Reiser3, followed by Reiser4.

- kernel 2.4.18
- metadata-only journaling (block journaling since kernel 2.6.8)
- online resizing
- tail packing to reduce internal fragmentation
- faster than ext2 and ext3

Btrfs

Btrfs is based on the copy-on-write (COW) principle. Initially designed by Oracle for Linux in 2007.

- version 1.0 accepted into kernel in 2009
- ext4 developer Theodor T’so said Btrfs is better than ext4 because "it offers improvements in scalability, reliability, and ease of management”
- default filesystem for SUSE
- deprecated in RHEL 7.4 in August 2017

Btrfs

- features include:
  - copy-on-write (shadowing)
  - auto-defrag
  - online growth/shrinking
  - offline filesystem check
  - data scrubbing - find and auto-fix errors
  - RAID {0,1,10}

xfs

- eXtents File System (xfs) - development begun by Silicon Graphics in 1993
- merged into Linux kernel 2.4.25 in Feb 2004
- default file system for RHEL 7.0 released in June 2014

xfs

- features include:
  - 64-bit file system, max size 8 EiB
  - journaling
  - allocation groups providing scalability and parallelism
  - striped allocation
  - extent based allocation
  - online defragmentation (with `xfs_fsr`) and resizing (with `xfs_growfs`)

Disk Partitioning

Disk Partitioning

- adding a disk partition
- formatting a disk partition
- mounting a disk partition

Disk Partitioning

A disk partition is a portion of a physical disk. The partition can be formatted and mounted. The benefits of having more than one disk partition:

- each partition has a limited number of inodes, so it is possible to run out them - having multiple partitions means there are more inodes
- put similar “files” together
- share a part of the file system on the network

Disk Partitioning

The number of currently mounted partitions are shown with the `df` command:

```
student$ df
Filesystem              1K-blocks    Used Available Use% Mounted on
udev                       999896       0    999896   0% /dev
tmpfs                      204596   21920    182676  11% /run
/dev/sda1                 9481444 4949280   4027488  56% /
...
```

In this example, `/dev/sda1` is mounted on `/`.

Disk Partitioning

The `df` command can show inode information:

```
student$ df -i
Filesystem     Inodes  IUsed  IFree IUse% Mounted on
udev           249974    448 249526    1% /dev
tmpfs          255743    655 255088    1% /run
/dev/sda1      610800 261998 348802   43% /
...
```

Notice that more than 43% of the available nodes on `/` are used. If we create a large number of files on that partition, we will run out inodes.

Create New Partition

We can partition the disk with the f`disk`:

```
root# fdisk -l /dev/sda
Disk /dev/sda: 40.0 GiB, 42949672960 bytes, 83886080 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x74b2e1ea

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *       487424 20019199 19531776  9.3G 83 Linux
/dev/sda2      20019200 24018943  3999744  1.9G 82 Linux swap / Solaris
```

This output shows 2 partitions. The one we have not seen is `/dev/sda2`, our swap space (more on swap later).

Create New Partition

To create a new partition:

- create partition on the disk with `fdisk`
- tell the kernel about the new partition with `partprobe`
- format the new partition with `mkfs.*`
- mount the new partition with `mount`
- add new partition to `/etc/fstab`

Create New Partition

To create a new partition on disk with `fdisk` (`p` prints the partition table):

```
root# fdisk /dev/sda

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk /dev/sda: 40.0 GiB, 42949672960 bytes, 83886080 sectors
…

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *       487424 20019199 19531776  9.3G 83 Linux
/dev/sda2        19531776 23531519  3999744  1.9G 82 Linux swap / Solaris

Command (m for help):
```

Create New Partition

`n` creates a new partition:

```
Command (m for help): n
Partition type
   p   primary (2 primary, 0 extended, 2 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (3,4 default 3): 3
First sector (23531520-83886079, default 23531520): <enter>
Last sector, +sectors or +size{K,M,G,T,P} (23531520-83886079,
default 83886079): +1G

Created a new partition 3 of type 'Linux' and of size 1  GiB.
```

Create New Partition

Let's make sure it was created correctly:

```
Command (m for help): p
Disk /dev/sda: 40.0 GiB, 42949672960 bytes, 83886080 sectors

Device     Boot    Start       End   Sectors  Size Id Type
/dev/sda1  *       487424  20019199 19531776  9.3G 83 Linux
/dev/sda2        19531776  23531519  3999744  1.9G 82 Linux swap / Solaris
/dev/sda3        23531520  25628671  2097152    1G 83 Linux

Command (m for help):
```

Create New Partition

Write the new information:

```
Command (m for help): w

The partition table has been altered.
Syncing disks.
```

Tell the Kernel

We need to tell the kernel about the changes. We can use `partprobe`. Or we can reboot the system.

```
root# partprobe
```

Note: some distributions (such as CentOS) require either executing `partx` (not recommended) or a reboot.

Make a Filesystem

The next step is to format the new disk. We will format as `ext4`, but it can be formatted as other filesystem types. To format at `ext4`, use `mkfs.ext4`:

```
root# mkfs.ext4 /dev/sda3
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with 262144 4k blocks and 65536 inodes
Filesystem UUID: 4214af50-bd61-413f-93d0-8936bad93e6d
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Mount the Partition

To make the partition available, it must be mounted. Many linux systems mount new partitions under `/mnt`. Make a new directory under `/mnt` and then use the `mount` command:

```
root# mkdir /mnt/newpartition
root# mount /dev/sda3 /mnt/newpartition
root# ls -ld /mnt/newpartition
drwxr-xr-x 3 root root 4096 Oct 14 11:01 /mnt/newpartition
root# ls -l /mnt/newpartition
total 16
drwx------ 2 root root 16384 Oct 14 11:01 lost+found
root# touch /mnt/newpartition/test.txt
root# ls /mnt/newpartition
lost+found  test.txt
```

Check Our Work

Let's check `df`:

```
root# df -i
Filesystem     Inodes  IUsed  IFree IUse% Mounted on
/dev/sda1      610800 262004 348796   43% /
/dev/sda3       65536     12  65524    1% /mnt/newpartition
```

Note the number of inodes. This number can be increased by using `mkfs.ext4 -N`.

Mount When Booting

At this point, we have a mounted filesystem, but if the machine is rebooted, it won't be mounted. To make sure the filesystem is mounted on reboot, add a line to `/etc/fstab` (file system table):

```
/dev/sda3 /mnt/newpartition ext4 defaults 0 2
```

The last two numbers on that line:

- enable/disable backing up with `dump`
- the order `fsck` checks the filesystem (`/` should always be `1`, others either `0` (disable) or `2` (after `/`))

Mount When Booting

Some systems use an UUID number (eg. Ubuntu). If that is true about your system, you can replace `/dev/sda3` with `UUID="id"` that is in the output of this command:

```
root# blkid /dev/sda3
```

Mount When Booting

Reboot to verify:

```
root# reboot
```

When the system boots, log in and check the directory:

```
root# ls /mnt/newpartition
```

Swap Space

Swap space is physical disk space used to store memory, usually when the RAM fills to capacity and the OS needs a place to put the content. Linux supports swap through either a disk partition or a file on disk.

We saw our swap space with `fdisk`:

```
root# fdisk -l
...
/dev/sda2       20019200 24018943  3999744  1.9G 82 Linux swap / Solaris
...
```

Swap Space

We can see our swap space with `swapon`:

```
root# swapon -s
Filename   Type       Size     Used  Priority
/dev/sda2  partition 1999868  0     -1
```

Swap Space

Let's create a new swap disk partition. Note: Because we have only one primary partition left, we will first create an extended partition with the rest of the disk, and we can then create partitions within it:

```
root# fdisk /dev/sda
Command (m for help): n
Partition type
   p   primary (3 primary, 0 extended, 1 free)
   e   extended (container for logical partitions)
Select (default e): e

Selected partition 4
First sector (25628672-83886079, default 25628672): <enter>
Last sector, +sectors or +size{K,M,G,T,P} (25628672-83886079, default 83886079): <enter>

Created a new partition 4 of type 'Extended' and of size 27.8 GiB.

Command (m for help):
```

Swap Space

Let’s see the new extended partition:

```
Command (m for help): p
Disk /dev/sda: 40.0 GiB, 42949672960 bytes, 83886080 sectors

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *      487424 20019199 19531776  9.3G 83 Linux
/dev/sda2       19531776 23531519  3999744  1.9G 82 Linux swap / Solaris
/dev/sda3       23531520 25628671  2097152    1G 83 Linux
/dev/sda4       25628672 83886079 58257408 27.8G  5 Extended

Command (m for help):
```

Swap Space

Now that we have the extended partition created, we can create partitions within it. We will make our new partition 1G in size:

```
Command (m for help): n
All primary partitions are in use.
Adding logical partition 5
First sector (25630720-83886079, default 25630720): <enter>
Last sector, +sectors or +size{K,M,G,T,P} (24020992-43741183,
default 43741183): +1G

Created a new partition 5 of type 'Linux' and of size 1 GiB.

Command (m for help): p
...
Device     Boot    Start      End  Sectors  Size Id Type
...
/dev/sda5       25630720 27727871  2097152    1G 83 Linux
```

Swap Space

We have to change the type (`t`) of the disk `82`, the value for swap.

```
Command (m for help): t
Partition number (1-5, default 5): 5
Hex code (type L to list all codes): 82

Changed type of partition 'Linux' to 'Linux swap / Solaris'

Command (m for help): p
...
Device     Boot    Start      End  Sectors  Size Id Type
...
/dev/sda5       25630720 27727871  2097152    1G 82 Linux swap / Solaris
```

Swap Space

Write the new information. This time, a reboot is required:

```
Command (m for help): w

The partition table has been altered.
Calling ioctl() to re-read partition table.
Re-reading the partition table failed.: Device or resource busy

The kernel still uses the old partitions. The new table will be used at
the next reboot.
Syncing disks.

root# reboot
```

Swap Space

The `mkswap` command makes swap. Then we can turn it on with `swapon`:

```
root# mkswap /dev/sda5
Setting up swapspace version 1, size = 1024
MiB (1073737728 bytes)
no label, UUID=a14f218f-041d-4abc-a739-
d588f4988539
root# swapon /dev/sda5
root# swapon -s
Filename   Type        Size      Used  Priority
/dev/sda2  partition   1999868   0     -1
/dev/sda5  partition   1048572   0     -2
```

Swap Space

Add to `/etc/fstab` (the UUID can also be used):

```
/dev/sda5 none swap sw 0 0
```

Reboot, then make sure swap got turned on:

```
root# swapon -s
Filename    Type        Size      Used  Priority
/dev/sda2   partition   1999868   0     -1
/dev/sda5   partition   1048572   0     -2
```

Swap Space

A file can also be used for swap. Use the `dd` command to create it, make it with `mkswap`, then `swapon` to turn it on:

```
root# mkdir /swap
root# dd if=/dev/zero of=/swap/swap1 bs=512 count=4000
4000+0 records in
4000+0 records out
2048000 bytes (2.0 MB, 2.0 MiB) copied, 0.00563459 s, 363  MB/s
root# ls -l /swap/swap1
-rw-r--r-- 1 root root 2048000 Oct 14 11:58 /swap/swap1
```

Swap Space

```
root# mkswap /swap/swap1
Setting up swapspace version 1, size = 2 MiB (2043904 bytes)
no label, UUID=74811884-69d5-46de-b7aa-8e9d68fa4a67
root# chmod 600 /swap/swap1
root# swapon /swap/swap1
root# swapon -s
Filename        Type            Size    Used    Priority
/dev/sda2       partition       1999868 0       -1
/dev/sda5       partition       1048572 0       -2
/swap/swap1     file            1996    0       -3
```

Swap Space

Add to `/etc/fstab` to have the swap space survive a reboot:

```
/swap/swap1 none swap sw 0 0
```

Exercise

1. Create a new partition (you might have to first create an extended partition) of size 500M.

2. Format that partition as `ext4`.

3. Mount the partition at `/mnt/ex1`.

4. Add an entry to `/etc/fstab`, reboot, verify the partition mounted.

Exercise

5. Create a new partition size 500M as type swap.

6. Make the swap, turn it on.

7. Add an entry to `/etc/fstab`, reboot, verify the swap is turned on.

Quotas

Quotas

- Quotas overview
- Configuring quotas
- Managing quotas

Quotas

- limit resources for a user or a group (eg. amount of disk space or number of files)
- two types of limits:
  - soft - when hit, the user is warned, and after a time period elapses without resolution, they will be locked out (denied the ability to create the resource)
  - hard - the user is locked out when the limit is hit

Quotas

- limits are set per disk partition - typically, only certain partitions have quotas:
  - `/home`
  - `/tmp`
  - `/var/spool/lpd` - print spools
  - `/var/spool/mail` - mail spools
  - `/var/www` - web sites

Quotas and SELinux

Note: SELinux must be turned off for quotas to be enabled:

```
root# setenforce 0
```

The `quota` package must be installed. For Debian:

```
root# apt install quota
```

For RHEL:

```
root# yum install quota
```

Turning On Quotas

- step one: mount a partition with quota options (`usrquota` and `grpquota`):

```
root# umount /mnt/newpartition
root# mount /dev/sda3 /mnt/newpartition -o  usrquota,grpquota
root# mount | grep newpartition
/dev/sda3 on /mnt/newpartition type ext4  (usrquota,grpquota)
```

- or:

```
root# mount -o remount usrquota,grpquota /dev/sda3
```

Turning On Quotas

- step two:
  - execute `quotacheck` to create needed files:

```
root# quotacheck -acuvgm
```

  - or create two files on the partition that are readable only by `root`:, then executed `quotacheck`:

```
root# touch /mnt/newpartition/aquota.user
root# touch /mnt/newpartition/aquota.group
root# chmod 0600 /mnt/newpartition/aquota.*
root# quotacheck -auvgm
```

Turning On Quotas

- step four: turn on quotas for the partition:

```
root# quotaon /dev/sda3
```

Setting Quotas

- to set quotas, edit the quotas for a user with `edquota`:

  ```
  root# edquota student
  ```

- this command brings up an editor with this content:

```
Disk quotas for user student (uid 502):
  Filesystem     blocks    soft    hard    nodes    soft    hard
  /dev/sda3           0       0       0        0       0       0
```

Setting Quotas

- if a limit has a value of 0, there is no quota - let's give `student` some limits:

```
Disk quotas for user student (uid 502):
  Filesystem     blocks    soft    hard   nodes    soft    hard
  /dev/sda3           0      20      40       0      20      40
```

Setting Quotas

- we can create quotas for a group:

```
root# edquota -g thisgroup
Disk quotas for user thisgroup (gid 509):
  Filesystem     blocks    soft    hard    nodes    soft    hard
  /dev/sda3           0       0       0        0       0       0
```

Setting Quotas

- setting the grace periods:

```
root# edquota -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes or seconds
  Filesystem      Block grace period     Inode grace period
  /dev/sda3               7days                  7days
```

Testing Quotas

- create a directory for student, switch to that user, and copy some files over:

```
root# mkdir /mnt/newpartition/forstudent
root# chown student:student /mnt/newpartition/forstudent
root# su - student
student$ cd /mnt/newpartition/forstudent
student$ cp -r /etc/pam.d .
cp: error writing './pam.d/chsh': Disk quota exceeded
cp: error writing './pam.d/cron': Disk quota exceeded
cp: error writing './pam.d/cups': Disk quota exceeded
cp: error writing './pam.d/gnome-screensaver': Disk quota  exceeded
cp: error writing './pam.d/lightdm': Disk quota exceeded
cp: error writing './pam.d/lightdm-autologin': Disk quota  exceeded
cp: error writing './pam.d/lightdm-greeter': Disk quota exceeded
cp: error writing './pam.d/login': Disk quota exceeded
cp: error writing './pam.d/newusers': Disk quota exceeded
```

More On Quotas

- we can apply one user's quotas to other users:

```
root# edquota -p student jdoe jsmith
```

- to make quotas permanent, edit `/etc/fstab` and add the `usrquota,grpquota` options:

```
/dev/sda3 /mnt/newpartition ext4 usrquota,grpquota 0  2
```

Exercise

1. Turn on quotas for `/mnt/ex1`.
2. Set up quotas for blocks (10 soft, 40 hard) and inodes (10 soft, 40 hard).
3. Copy enough data for the user to exceed one of the soft limits.
4. Copy enough data for the user to exceed one of the hard limits.
5. Create a new user, give that new user the same quotas.
6. Turn off quotas for `/mnt/ex1`.

Logical Volume Manager

LVM

LVM is a Logical Volume Manager - it manages disk drives, usually large ones. Some common uses are:

- similar to RAID 0, make a single logical volume out of multiple physical volumes, allowing for dynamic resizing
- add, replace, copy disks without disrupting service - good for large disk farms
- easily resize partitions as needed
- make backups (snapshots)

LVM

- LVM is a layer of software on top of the disk and partitions, creating a logical, continuous disk that is easy to resize

- Any partition, with the exception of `/boot`, can be part of an LVM

LVM

- many distributions support LVM during the installation:
  - Ubuntu - can select LVM
    - `/boot` partition as ext4
    - `/` partition as LVM
    - swap partition as LVM
  - Centos - default is LVM
    - `/boot` partition as xfs
    - `/` partition as LVM
    - swap partition as LVM

LVM

- basic functionality:
  - create one or more Physical Volumes (PVs) using one or more disk partitions (on one or more disks)
  - create a Volume Group (VG) by combining one or more PVs
  - create one or more Logical Volumes (LVs) in a VG
  - VGs can be extended by adding additional PVs
  - LVs can be resized by adding or removing extents

Working with LVMs

- It might be necessary to install the required software
  - Debian

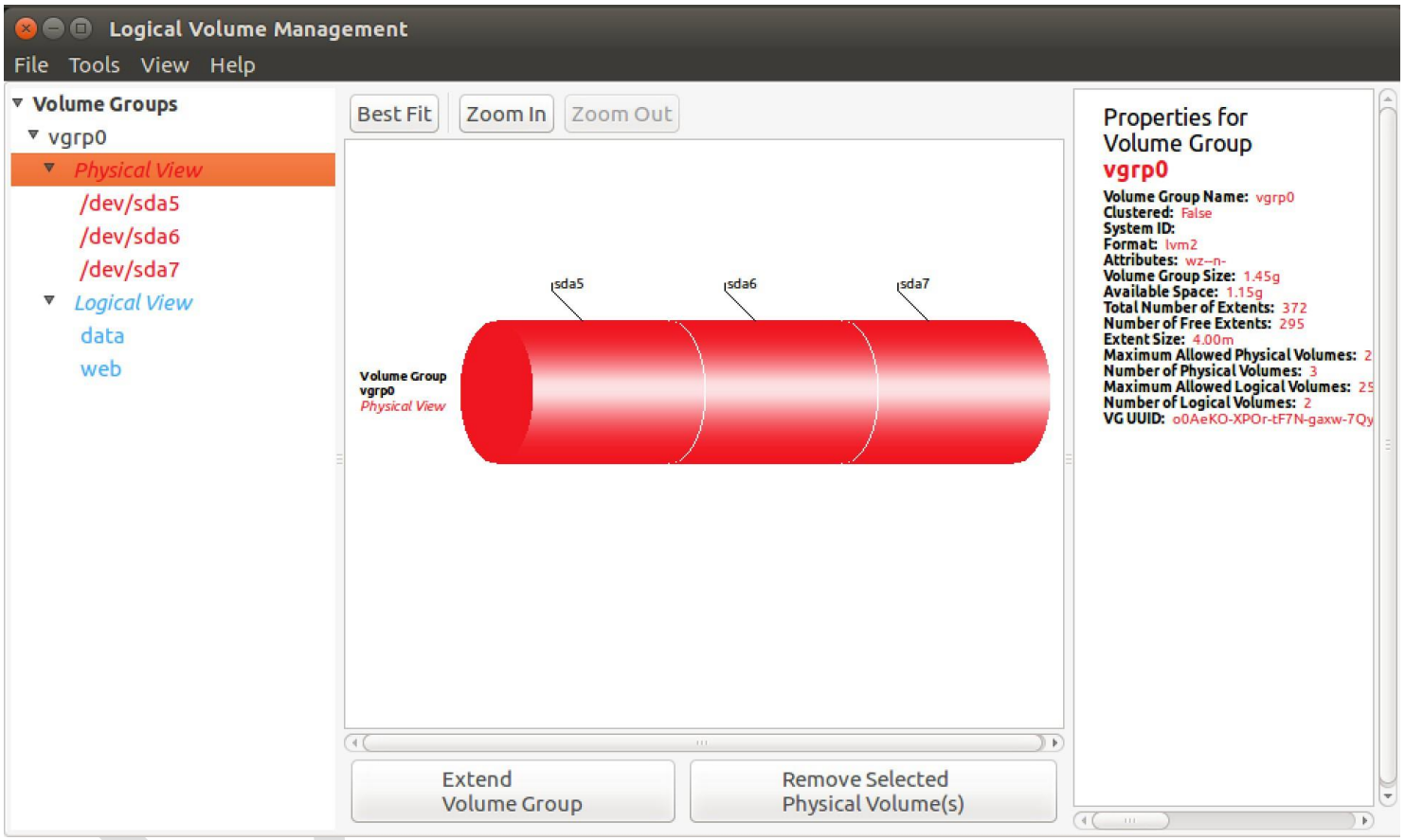
    ```
    root# apt install lvm2 system-config-lvm
    ```

  - RHEL

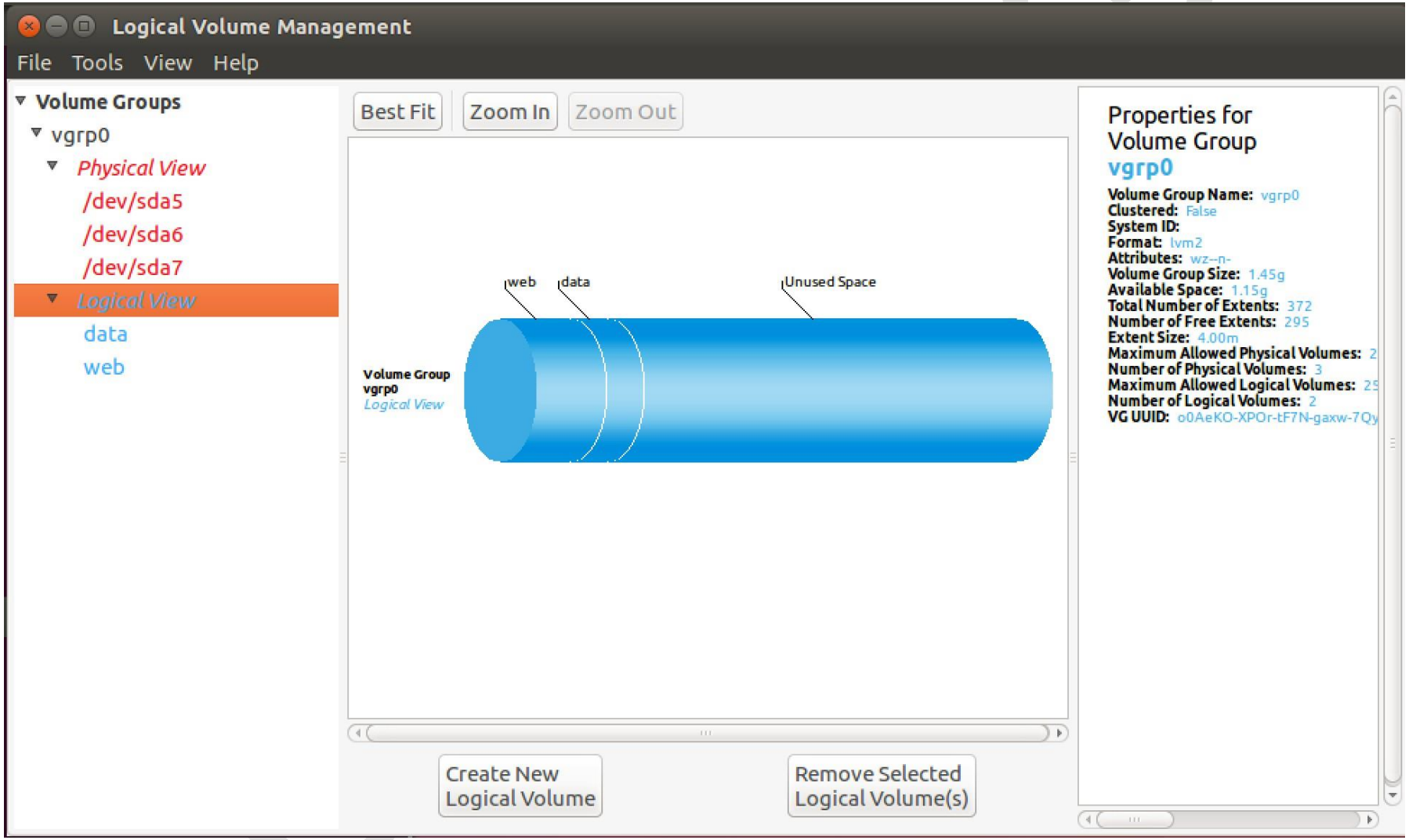
    ```
    root# yum install lvm2 system-config-lvm
    ```

Graphical Tools

- `system-config-lvm`



Graphical Tools



Working with LVMs

To create an LVM:

- create partitions and give them a type of `8e`
- initialize each physical volume with `pvcreate`
- create a volume group with `vgcreate`
- create mountable logical volumes with `lvcreate`
- format and mount the logical volumes

Create LVM Partitions

The first step in creating an LVM is to create disk partitions that will be part of the volume group. These partitions can be on the same physical disk, or they can be on different physical disks. Create the partitions with type 8e, Linux LVM.

```
root# fdisk -l | grep LVM
/dev/sda6        24020992 25044991  1024000  500M 8e Linux LVM
/dev/sda7        25047040 26071039  1024000  500M 8e Linux LVM
/dev/sda8        26073088 27097087  1024000  500M 8e Linux LVM
root# partprobe
```

Initialize LVM PV

- The next step in creating an LVM is to initialize each physical volume with `pvcreate`

```
root# pvcreate /dev/sda{6,7,8}
  Physical volume "/dev/sda6" successfully created
  Physical volume "/dev/sda7" successfully created
  Physical volume "/dev/sda8" successfully created
```

Initialize LVM PV

Let's look at one of the partitions with with `pvdisplay` (with no argument, this command shows all PVs)

```
root# pvdisplay /dev/sda6
  "/dev/sda6" is a new physical volume of "500.00 MiB"
  --- NEW Physical volume ---
  PV Name               /dev/sda6
  VG Name
  PV Size               500.00 MiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               nno7Lt-9R3g-eKw4-kt66-IuDv-8q9Y-ahVsXn
```

Create LVM VG

The next step in creating an LVM is to create a volume group that contains the physical volumes. The `-s` option is the size of the extents

```
root# vgcreate vgrp0 /dev/sda{6,7,8}
  Volume group "vgrp0" successfully created
```

Create LVM VG

```
root# vgdisplay vgrp0
  --- Volume group ---
  VG Name               vgrp0
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               1.45 GiB
  PE Size               4.00 MiB
  Total PE              372
  Alloc PE / Size       0 / 0
  Free  PE / Size       372 / 1.45 GiB
  VG UUID               o0AeKO-XPOr-tF7N-gaxw-7Qyp-Tb83-4ku7PC
```

Create LVM LV

The next step in creating an LVM is to create a mountable logical volume group (LVG) in an existing VG with `lvcreate`. This is a mountable volume.

```
root# lvcreate -L 200M -n web vgrp0
  Logical volume "web" created.
```

Create LVM LV

```
root# lvdisplay /dev/vgrp0/web
  --- Logical volume ---
  LV Path                /dev/vgrp0/web
  LV Name                web
  VG Name                vgrp0
  LV UUID                eHoiBQ-hRMH-auG7-eLCl-oYjG-HY7K-A1xWOW
  LV Write Access        read/write
  LV Creation host, time ubuntu2, 2017-09-30 11:49:45 -0400
  LV Status              available
  # open                 0
  LV Size                200.00 MiB
  Current LE             50
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

Create LVM LV

```
root# mkfs.ext4 /dev/vgrp0/web
root# mkdir /mnt/web
root# mount /dev/vgrp0/web /mnt/web
root# blkid | grep web
/dev/mapper/vgrp0-web:
UUID="4253b8b2-626d-4366-8e1c-15144a011236" TYPE="ext4"

# add to /etc/fstab
/dev/vgrp0/web /mnt/web ext4 defaults 0 2
```

Create LVM LV

Let's create another LV:

```
# make 20 extents
root# lvcreate -l 20 -n data vgrp0
root# mkfs.ext4 /dev/vgrp0/data
root# mkdir /mnt/data
root# mount /dev/vgrp0/data /mnt/data

# add to /etc/fstab
/dev/vgrp0/data /mnt/data ext4 defaults 0 2
```

Resizing LogicalVolumes

Logical volumes can be resized without needing to reboot. They can be expanded without having to unmount (if the size is to be reduced, then it must be unmounted). The `lvextend` will increase the size of the volume, and `lvreduce` will decrease its size.

The `lvextend` option `-L` extends the volume by a given size, the `-l` option extends by the number of extents.

Resizing LogicalVolumes

```
root# lvextend -L +10M /dev/vgrp0/data
  Rounding size to boundary between physical extents: 12.00 MiB
  Size of logical volume vgrp0/data changed from 80.00 MiB (20
extents) to 92.00 MiB (23 extents).
  Logical volume data successfully resized.
root# lvextend -l +4 /dev/vgrp0/data
  Size of logical volume vgrp0/data changed from 96.00 MiB (24
extents) to 112.00 MiB (28 extents).
  Logical volume data successfully resized.
```

Resizing LogicalVolumes

To reduce the size of a logical volume:

```
root# umount /mnt/data
root# lvreduce -L -5M /dev/vgrp0/data
  Rounding size to boundary between physical extents: 4.00 MiB
  WARNING: Reducing active logical volume to 108.00 MiB
  THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce data? [y/n]: y
  Size of logical volume vgrp0/data changed from 112.00 MiB
(28 extents) to 108.00 MiB (27 extents).
  Logical volume data successfully resized.
root# mount /dev/vgrp0/data /mnt/data
```

Removing Logical Volumes

Logical volumes can be removed with `lvremove`:

```
root# umount /mnt/data
root# lvremove /dev/vgrp0/data
```

Adding Physical Volumes

- first, create a new disk partition (in this example, 9):

```
root# fdisk -l | grep LVM
/dev/sda6       24020992 25044991  1024000  500M 8e Linux LVM
/dev/sda7       25047040 26071039  1024000  500M 8e Linux LVM
/dev/sda8       26073088 27097087  1024000  500M 8e Linux LVM
/dev/sda9       27099136 29147135  2048000 1000M 8e Linux LVM
root# partprobe
```

Adding Physical Volumes

- next, create a physical volume:

```
root# pvcreate /dev/sda9
  Physical volume "/dev/sda9" successfully created
```

- next, add the partition to the volume group with `vgextend`:

```
root# vgextend vgrp0 /dev/sda9
  Volume group "vgrp0" successfully extended
```

Display LVM Info

As a review, we can display information about our LVMs:

- physical volumes

```
root# pvdisplay /dev/sda6
```

- volume groups

```
root# vgdisplay vgrp0
```

- logical volumes

```
root# lvdisplay /dev/vgrp0/web
```

RAID

Software RAID

- RAID (Redundant Array of Inexpensive Disks) came about because (as the story goes) an organization with little money and a lot of old hard drives needed some storage space. So the idea to combine the inexpensive disks into a single disk was born. These days, they are better known as Redundant Array of Independent Disks.

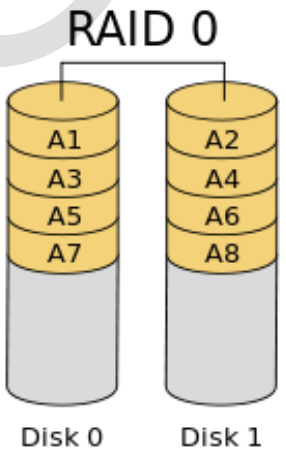
- RAID is often used as a term describing a way to divide and replicate data among multiple disks.

Software Raid

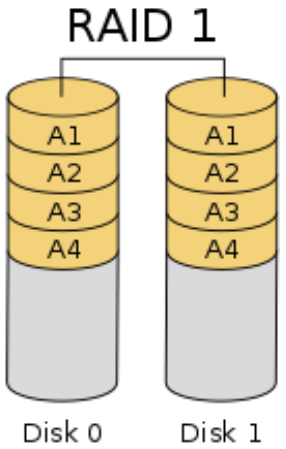
- RAID redundancy is achieved through:
  - mirroring - writing the same data to multiple drives
  - parity data - extra data across multiple disks - for fault tolerance, data is computed on one or more disks and stored in another location

Types of RAID Drives

- Different types of RAID drives:
  - RAID 0 - striped disks - distribute data across all disks - no parity, redundancy or fault tolerance (for speed)



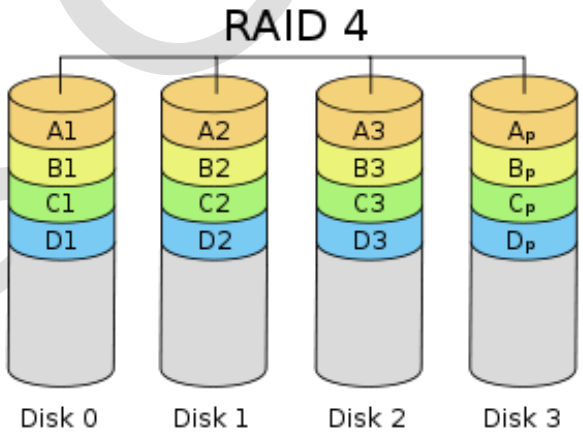
  - RAID 1 - mirrored disks - duplicates data across all disks in the array



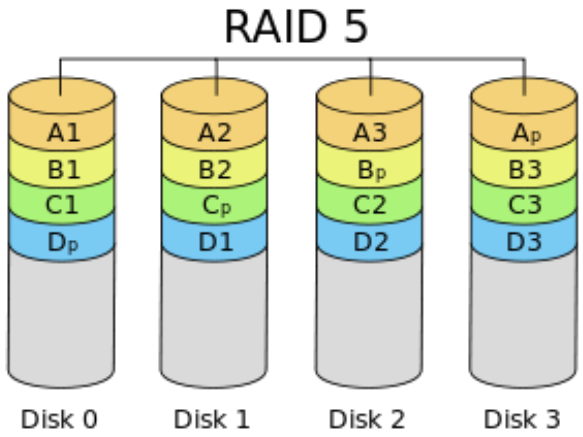
all RAID images from Wikipedia

Types of RAID Drives

- RAID 2 and 3 - rarely used
- RAID 4 - striping with a dedicated parity disk

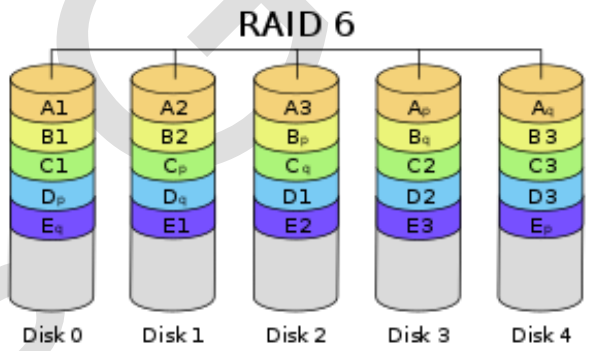


- RAID 5 - striped disks with parity - combines 3 or more disks so that loss of one disk will not lose any data

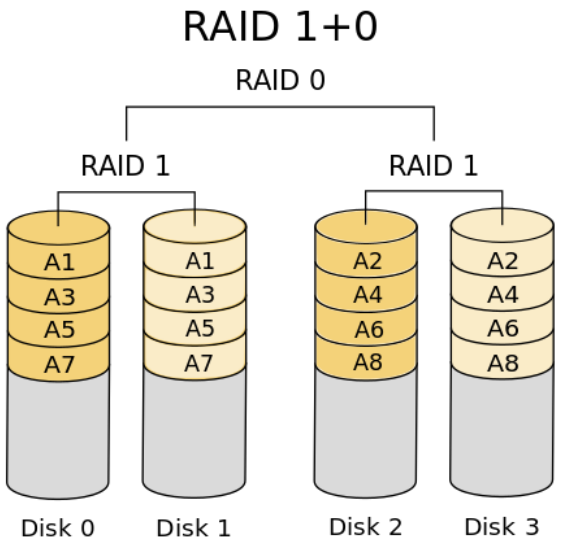


Types of RAID Drives

- RAID 6 - striped disks with dual parity - can recover from the loss of two disks (uncommon)



- RAID 10 (1 + 0) - striped and mirrored disks - uses both striping and mirroring



Software RAID

- Software RAID is when the various RAID levels are implemented within the kernel:
  - cheapest solution
- To create a software RAID:
  - create partitions
  - create virtual device file
  - that device file is then mounted

Creating a Software RAID

- install packages if necessary

```
root# apt install mdadm
root# yum install mdadm
```

- create 2 or more disk partitions of type `fd`:

```
root# fdisk /dev/sda
...
root# fdisk -l /dev/sda | grep fd
/dev/sda10       24020992 25044991  1024000  500M fd Linux raid autodetect
/dev/sda11       25047040 26071039  1024000  500M fd Linux raid autodetect
```

- tell the kernel about the new disk partitions:

```
root# partprobe
```

Creating a Software RAID

- create the virtual device (device is `/dev/md0`, RAID 1, two disks):

```
root# mdadm --create --verbose  /dev/md0 --level=1 \
      --raid-devices=2 /dev/sda10 /dev/sda11
mdadm: array /dev/md0 started.
```

- format the RAID:

```
root# mkfs.ext4 /dev/md0
```

- create a mount point and mount the partition:

```
root# mkdir /mnt/RAID
root# mount /dev/md0 /mnt/RAID
```

Creating a Software RAID

- store new detail in configuration file

```
root# mdadm --detail --scan | \
      tee -a /etc/mdadm/mdadm.conf
```

- update the initial RAM file system so the array will be available during the early boot process

```
root# update-initramfs -u
```

- add a mount point to `/etc/fstab`:

```
root# echo '/dev/md0 /mnt/RAID ext4 \
      defaults,nofail,discard 0 0' | \
      tee -a /etc/fstab
```

RAID Info

To show information about the RAID device

```
root# mdadm -D /dev/md0

/dev/md0:
        Version : 1.2
  Creation Time : Fri Sep 29 17:24:11 2017
     Raid Level : raid1
     Array Size : 511680 (499.77 MiB 523.96 MB)
  Used Dev Size : 511680 (499.77 MiB 523.96 MB)
   Raid Devices : 2
  Total Devices : 2
    Persistence : Superblock is persistent
```

RAID Info

```
    Update Time : Fri Sep 29 17:47:20 2017
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           Name : ubuntu2:0  (local to host ubuntu2)
           UUID : 9077a64b:ee2fcfc0:08fc26bd:5e9f1bd8
         Events : 21

    Number   Major   Minor   RaidDevice State
       0       8        5        0      active sync   /dev/sda10
       1       8        6        1      active sync   /dev/sda11
```

Removing a RAID

- To remove a RAID:
  - unmount the device:

```
root# umount /dev/md0
```

  - stop and remove the RAID:

```
root# mdadm --stop /dev/md0
root# mdadm --remove /dev/md0
```

Removing a RAID

- remove the line added at the bottom of `mdadm.conf`:

  `root#` **`vi /etc/mdadm/mdadm.conf`**

- update the initial RAM file system:

  `root#` **`update-initramfs -u`**

- remove entry from `/etc/fstab`

Exercise

1. Create an LVM:

    - 3 partitions of size 1024M
    - 1 volume group
    - 2 logical volumes: vol1 (512M) and vol2 (100 extents)
    - mount the volumes at `/mnt/vol1` and `/mnt/vol2`
    - add them to `/etc/fstab`
    - reboot to confirm they mount at boot

Exercise

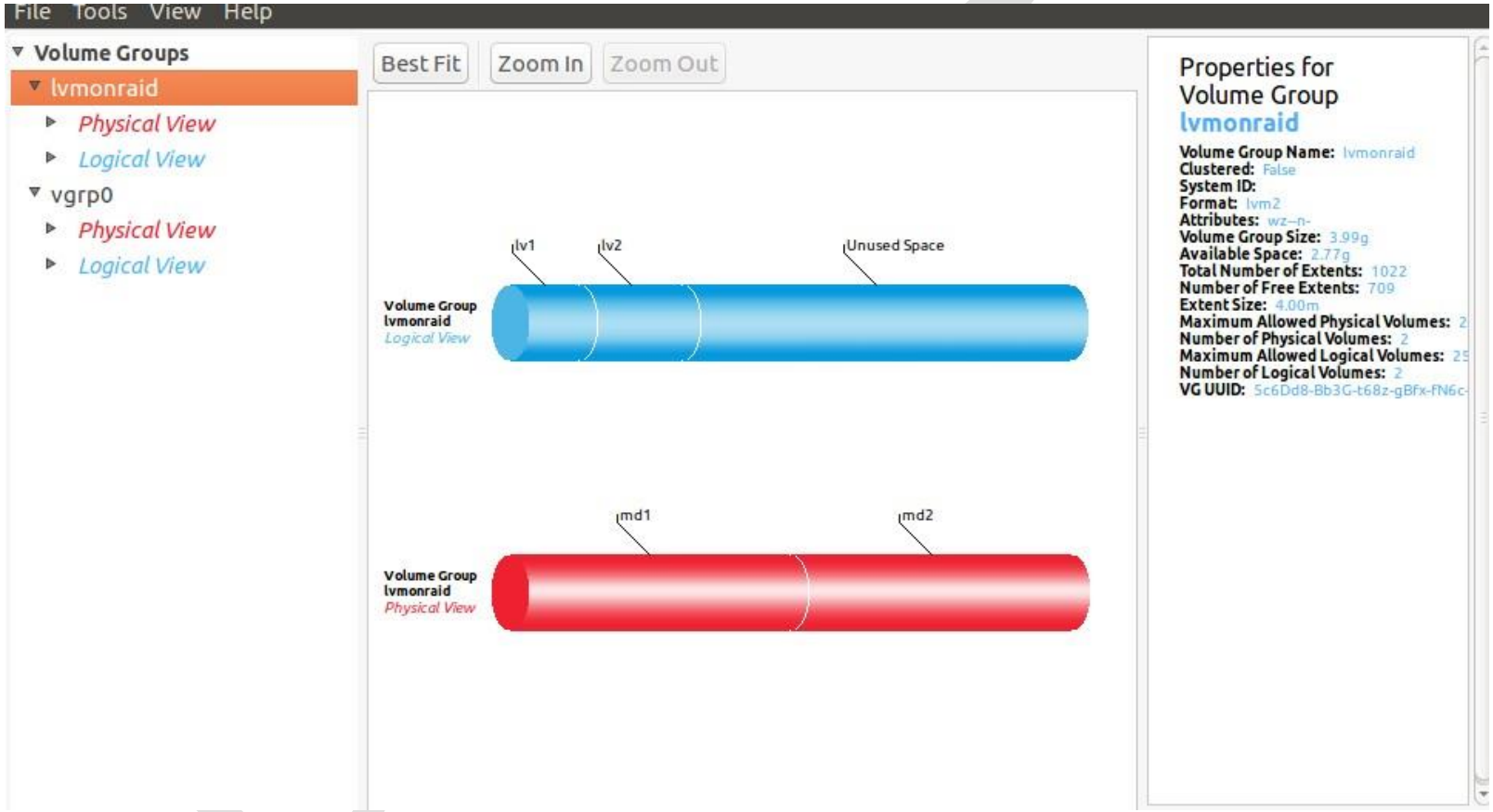
2. Create a RAID:

- 3 partitions of size 1024M
- RAID 0
- block size 2048
- stride 32
- mount it at `/mnt/RAID`
- add it to `/etc/fstab`
- reboot to confirm they mount at boot

Supplemental: LVM on RAID

LVM on RAID

Let’s create an LVM from two partitions, each partition is a RAID with redundancy.



LVM on RAID

1. Create partitions - use `fdisk` and create 4 2G partitions:

```
# fdisk -l /dev/sda
...
/dev/sda12       40278016 44472319  4194304    2G fd Linux raid autodetect
/dev/sda13       44474368 48668671  4194304    2G fd Linux raid autodetect
/dev/sda14       48670720 52865023  4194304    2G fd Linux raid autodetect
/dev/sda15       52867072 57061375  4194304    2G fd Linux raid autodetect
```

2. Create 2 RAIDS from these 4 partitions, 2 partitions each:

```
# mdadm --create --verbose /dev/md1 --level=1 \
    --raid-devices=2 /dev/sda12 /dev/sda13
# mdadm --create --verbose /dev/md2 --level=1 \
    --raid-devices=2 /dev/sda14 /dev/sda15
```

LVM on RAID

3. Create create an LVM from these raids - first create PV:

```
# pvcreate /dev/md1 /dev/md2
```

4. Create a VG:

```
# vgcreate lvmonraid /dev/md1 /dev/md2
```

5. Create some LVs:

```
# lvcreate -L 500M -n lv1 lvmonraid
# lvcreate -L 750M -n lv2 lvmonraid
```

LVM on RAID

6. Format these LVs:

```
# mkfs.ext4 /dev/lvmonraid/lv1
# mkfs.ext4 /dev/lvmonraid/lv2
```

7. Mount them:

```
# mkdir /mnt/lv1
# mkdir /mnt/lv2
# mount /dev/lvmonraid/lv1 /mnt/lv1
# mount /dev/lvmonraid/lv2 /mnt/lv2
```

LVM on RAID

The result: