Unix

Adam Boult (www.bou.lt)

February 10, 2023

Contents

Preface

This is a live document, and is full of gaps, mistakes, typos etc.

Part I

File systems

Chapter 1

File systems

1.1 Introduction

1.1.1 Introduction

What they are. tree? heap? concept of mapping from path to file

1.1.2 Garbage management of files

The link count is stored per file. If there are zero links to a file then the file system manager knows that the file can be deleted.

1.1.3 inodes

Bad blocks are noted in inode1.

Root is inode 2.

inode 0 is null.

Within a partition, each file or folder has a unique inode.

Each partition divided into blocks. numbered from 0. Blocks are the minimum size for readable or writable operations. Changing a block means needing to read the whole block, making a change and then rewriting the block.

Files can be stored across many blocks. Block don't have to be next to each other.

Because a block is the mimimum size of any operation, there can only be one file per block, and each file takes up at least one block.

File names are the property of folders.

1.1.4 ext

1.1.5 ext2

1.1.6 FAT12

1.1.7 FAT32

Chapter 2

Journalling file systems

2.1 Introduction

2.1.1 Introduction

2.1.2 ext3

2.1.3 ext4

2.1.4 NTFS

Chapter 3

Partitioning including MBR and GPT

3.1 Introduction

3.1.1 MBR and GPT

Drives are divided into sectors. Each sector contains a fixed number of bytes, eg 4096 bytes.

File systems can be partitioned using either MBR or GPT:

- Master Boot Record (MBR)
- Globally Unique IDentifiers (GUID) Partition Table (GPT)

GPT is newer than MBR.

3.1.2 Boot sectors

The first sector of a disk is the boot sector. This applies to both MBR and GPT.

GPT does not use the boot sector, and it is just kept for compatibility reasons. An EFI system partition is used instead, as discussed later.

Part II

Boot loaders

Chapter 4

First-stage boot loaders, including BIOS, UEFI and core/libreboot

4.1 Introduction

4.1.1 Introduction

The motherboard has the first-stage boot loader. This if firmware with its own read-only memory.

4.1.2 Basic Input/Output System (BIOS)

4.1.3 Unified Extensible Firmware Interface (UEFI)

Supports Secure Boot.

If /sys/firmware/efi/ exists, the system is an EFI computer. Modern systems are UEFI rather than BIOS.

UEFI stores data in .efi file located in a hard drive, not a rom like in bios.

UEFI file stored in EFI system partition (ESP).

UEFI runs in 32/64 bit. bIOS in 16 bit. means uefi can support mouse and GUI.

UEFI supports disks over 2TB.

4.1.4 Coreboot and Libreboot

4.1.5 Android boot loaders

Chapter 5

Second-stage boot loaders including GRUB

5.1 Introduction

5.1.1 Introduction

The first-stage bootloader, eg BIOS, looks for a second-stage bootloader to load on a disk.

The second-stage bootloader loads the linux kernel then runs "init".

5.1.2 GRand Unified Bootloader (GRUB)

GRUB is a second-stage bootloader.

- /etc/default/grub
- /etc/grub.d/

memtest: run from grub

5.1.3 systemd-boot

Alternative to GRUB which supports UEFI.

5.1.4 EFISTUB

Part III

Character devices

Chapter 6

Printers and keyboards

6.1 Introduction

6.1.1 Introduction

Part IV

Tools for partitioning and formatting drives

Chapter 7

Partitioning and formatting drives

7.1 Partitioning drives

7.1.1 Introduction

three options for partitioning are fdisk, gdisk and parted. parted generally seems the better option.

7.1.2 fdisk

fdisk is designed with MBR in mind, but later versions have some GPT support:

- fdisk -l (list things in /dev/) (or can use lsblk)
- fdisk /dev/sda (or whatever correct device is)
- this opens dialogue:
  - "d" to delete partitions
  - create a new table, using MBR or GPT
  - create partitions (can press "n" for new)
  - make one bootable
  - "w" to write"

7.1.3 parted

partitioning using parted:

- supports MBR and GPT

- different to fdisk? needed if drives over 2TB?
- parted -l (list things in /dev/) (or can use lsblk)
- parted /dev/sda (or whatever correct device is)
- this opens dialogue:
  - see status with "print"
  - type "quit" when done
  - make gpt using "mklabel gpt"
  - make mbr using "mklabel msdos"
  - make partitions: "mkpart". is interactive
  - make one bootable? "set ¡partition¿ boot on"

7.1.4 gdisk

gdisk is similar to fdisk but aimed at GPT.

7.1.5 What partitions can be used?

root partition if uefi, efi system partition (boot partition). also need boot partition if doing LVM or encryption on BIOS swap, though this is discussed later

7.2 Formatting drives

7.2.1 Introduction

once partitions have been made, they show up on /dev/

7.2.2 Making ext4 partitions

```
mkfs.ext4 /dev/<textless_root_partition>
```

7.2.3 Making FAT partitions

also for grub boot?

```
mkfs.fat -F 32 /dev/<efi_system_partition>
```

7.2.4 Swap

swap, though this is discussed later

Part V

Linux kernel

Chapter 8

Directory layout on Linux

8.1 Introduction

8.1.1 Introduction

8.1.2 /boot

8.1.3 /bin

/bin is where main binaries are stored.

8.1.4 /proc

/proc has kernal files?

8.1.5 /tmp

8.1.6 /lib

/lib has libraries for /bin.

8.1.7 /etc

/etc has conf?

8.1.8 /tmp

Part VI

OS basics

Chapter 9

Batch processing

9.1 Introduction

9.1.1 Introduction

multiple programs set to run one after another. virtual memory (and pages) here? something on segmentation faults

DOS is like this maybe?

Chapter 10

Interrupts

10.1 Introduction

10.1.1 Introduction

Swap between processes (eg if user says to swap during, waiting for input, or priting). multi process needed for system managment if even running 1 job? thread safety. address spcae layout randomisation. privilege. memory protection. avaoiding deadlocks. job scheduler

Chapter 11

Concurrency control

11.1 Introduction

11.1.1 Introduction

separate to parallel or multi threading. overlapping lifetimes of programs can cause

Part VII

GNU

Chapter 12

File systems commands

12.1 Introduction

12.1.1 Introduction

12.1.2 whereis

12.1.3 man

12.1.4 whatis

one line version of man

Chapter 13

sed and tr

13.1 Introduction

13.1.1 sed

minor scripting options but not central to concept. is regex thing.

13.1.2 tr

Part VIII

GNU coreutils

Chapter 14

GNU Core Utilities (files)

14.1 Introduction

14.1.1 Introduction

14.1.2 pwd

14.1.3 cd

14.1.4 ls

14.1.5 du

sizes of files in folder.

14.1.6 stat

Chapter 15

GNU Core Utilities (files) (writing)

15.1 Introduction

15.1.1 Introduction

15.1.2 cp

15.1.3 mv

15.1.4 rm

15.1.5 mkdir

15.1.6 rmdir

15.1.7 touch

15.1.8 ln

Chapter 16

GNU Core Utilities (text)

16.1 Introduction

16.1.1 Introduction

16.1.2 cat

16.1.3 head and tail

head(first x lines)

tail(last x lines)

16.1.4 cut

Part IX

GNU cont.

Chapter 17

Pipes and xargs

17.1 Introduction

17.1.1 Pipes

17.1.2 xargs

Chapter 18

more, less, ed and vi

18.1 Introduction

18.1.1 Introduction

Chapter 19

Check sums

19.1 Introduction

19.1.1 Introduction

sha1sum, sha256sum, sha512sum, crc32sum

19.1.2 md5sum

19.1.3 sha1sum

19.1.4 sha256sum

19.1.5 sha512sum

19.1.6 crc32sum

Chapter 20

Compression

20.1 Introduction

20.1.1 Introduction

tar, gzip, zipbomb

Chapter 21

Jails

21.1 Introduction

21.1.1 Introduction

chroot (changes apparent root for processes, chroot jail?)

Part X

The shell

Chapter 22

Basic shells

22.1 Introduction

22.1.1 Introduction

Chapter 23

Keyboards and locales

23.1 Introduction

23.1.1 Introduction

loadkeys

locale-gen function to eg set languages. see locale using "locale"

Part XI

Users and permissions

Part XII

Systemd

Part XIII

Compiling

Chapter 24

GNU Compiler Collection (gcc)

24.1 Introduction

24.1.1 Introduction

compile to objhect, to assembly, links and runtime libraries?or is in make?, objdump; returning stderr and stdout. raise

Chapter 25

C headers for linux

25.1 Introduction

25.1.1 Introduction

25.1.2 getopt

```
#include <unistd.h>

int getopt(int argc, char *const argv[],
const char *optstring);

extern char *optarg;
extern int optind, opterr, optopt;
```

allows

```
myScript -a s
```

25.1.3 Longer

```
#include <getopt.h>

int getopt_long(int argc, char *const argv[],
           const char *optstring,
           const struct option *longopts, int *longindex);
int getopt_long_only(int argc, char *const argv[],
           const char *optstring,
           const struct option *longopts, int *longindex);
```

allows

```
myScript --a something
```

25.1.4 stdio

stdio allows arguments but not as flags. allows

```
myScript something
```

25.1.5 Static

use of "static" to say not to be exported

Chapter 26

GNU Debugger (gdb)

26.1 Introduction

26.1.1 Introduction

debugger. can step through, stop on specific functions etc

Chapter 27

make

27.1 Introduction

27.1.1 Introduction

Makefile; target:, .PHONY clean; clean: (means that doesn't look to see if clean exists because declared as phony.)

Chapter 28

Libraries and namespaces in C

28.1 Introduction

28.1.1 Libraries

Chapter 29

Macros and the C preprocessor

29.1 Macros

29.1.1 Macros

Not separate, code is replaced before it is run

```
#define PI 3.14
```

These are then expanded before compiling.

29.1.2 Parametised macros

We can also replace using functions.

For example we could do:

```
print(3 + 1)
print(2 + 1)
```

or

```
#define plusOne(x) (x+1)
print(plusOne(x))
print(plusOne(x))
```

Part XIV

Scripting

Chapter 30

sh

30.1 Introduction

30.1.1 Introduction

shebang at top.

30.1.2 Pipes

`|`

30.1.3 Multiple jobs

`multiple commands (&), trailing &,`

30.1.4 Control flow

`&&`
`||`

control flow in sh (do while, case, for loop).

30.1.5 Writing to files

`write to file with > (overwrite) and >> (append),`
`direct stderr to stdout with 2>&1.`

raise error?

30.1.6 Getting interactive input

getting input from user as part of script. doing so in password way to hide input.

30.1.7 Variables

defining variables.

env.

30.1.8 Functions

functions.

30.1.9 Passing variables to shell scripts

passing variables to sh script (-, –?)

30.1.10 xargs

30.1.11 Stream and batch data

stream vs batch data here or elsewhere?

30.1.12 Other commmands

exit. sleep? timer?

Chapter 31

Bourne-Again Shell (bash)

31.1 Introduction

31.1.1 Introduction

.bashrc file. .bashprofile file

Chapter 32

awk

32.1 Introduction

32.1.1 Introduction

compile to objhect, to assembly, links and runtime libraries?or is in make?, objdump; returning stderr and stdout. raise

Chapter 33

Perl

33.1 Introduction

33.1.1 Introduction

perl compatible regular expressions

Part XV

Serialisation

Chapter 34

Comma Separated Values (CSV)

34.1 Introduction

34.1.1 Introduction

Chapter 35

Extensible Markup Language (XML)

35.1 Introduction

35.1.1 Introduction

billion laughs attack. prevent with limited parsers

Chapter 36

JavaScript Object Notation (JSON)

36.1 Introduction

36.1.1 Introduction

Chapter 37

Yet Another Markup Language (YAML)

37.1 Introduction

37.1.1 Introduction

Part XVI

Encryption

Chapter 38

Hard drive encryption

38.1 Introduction

38.1.1 Introduction

Chapter 39

RAM encryption

39.1 Introduction

39.1.1 Introduction

Part XVII

Analysing programs

Chapter 40

Static analysis

40.1 Introduction

40.1.1 Introduction

frama-C?, formal verification

Chapter 41

Dynamic analysis

41.1 Introduction

41.1.1 Introduction

unit tests (inc asserts) and integration tests

Chapter 42

Hex analysis

42.1 Introduction

42.1.1 Introduction

hexdump, xxd. can search for english strings? if want to know how part of code used, can search for references to it?

Chapter 43

Ghidra

43.1 Introduction

43.1.1 Introduction