CS 447/647

Filesystem(s)

References

Love, Robert. *Linux Kernel Development*. Addison-Wesley, 2015.

Nemeth, Evi, et al. *UNIX and LINUX System: Administration Handbook*. Addison-Wesley/Pearson, 2018.

Overview

- Philosophy
- Virtual File System
- File Types

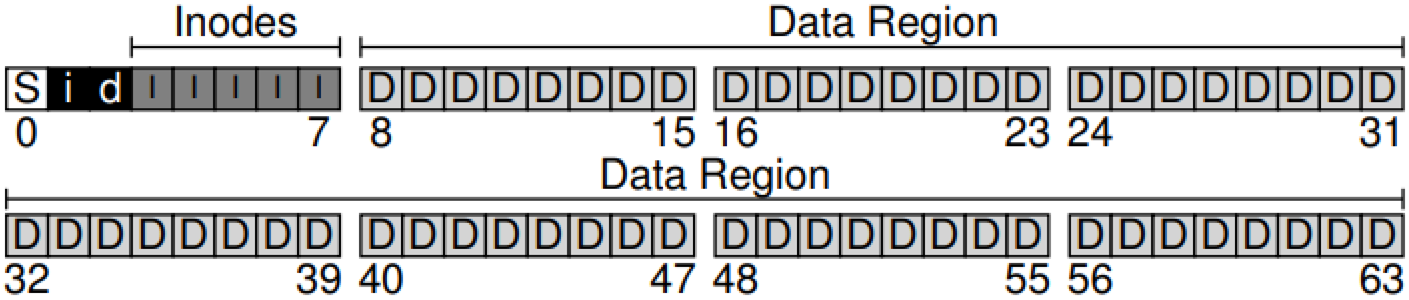
Intro

- Processes - /proc
- Audio devices, Graphics Cards- /dev
- Kernel data structures and tuning parameters - /sys
- Interprocess communication channels - /run (sockets)
- Directories

```
cat /dev/urandom > /dev/dsp #Old
cat /dev/urandom | tr -dc "0-9a-z" | aplay #New
```

What is a filesystem?

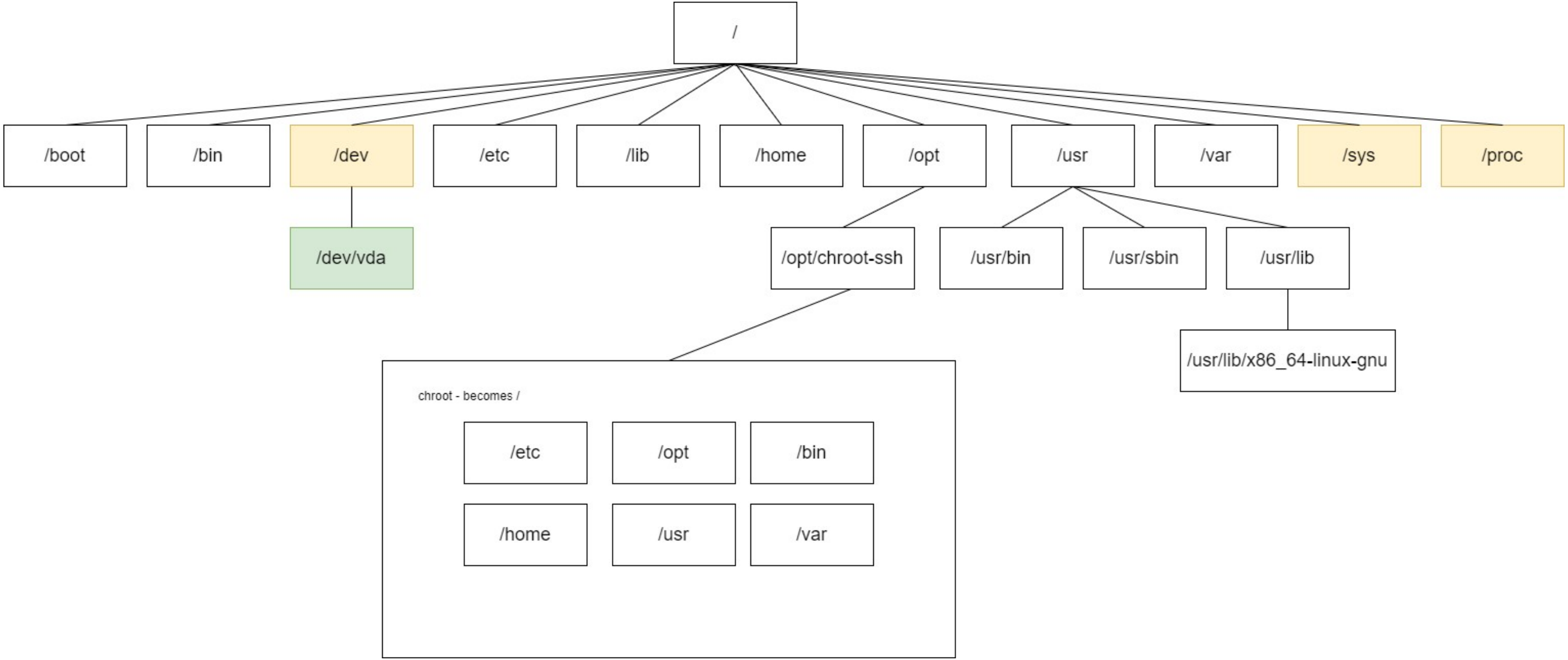
- A namespace
  - Organize items into a hierarchy
- An API
  - A set of system calls for navigation and manipulation
- Security Model
  - Protecting, hiding and sharing
- An implementation
  - Logical Model to the hardware

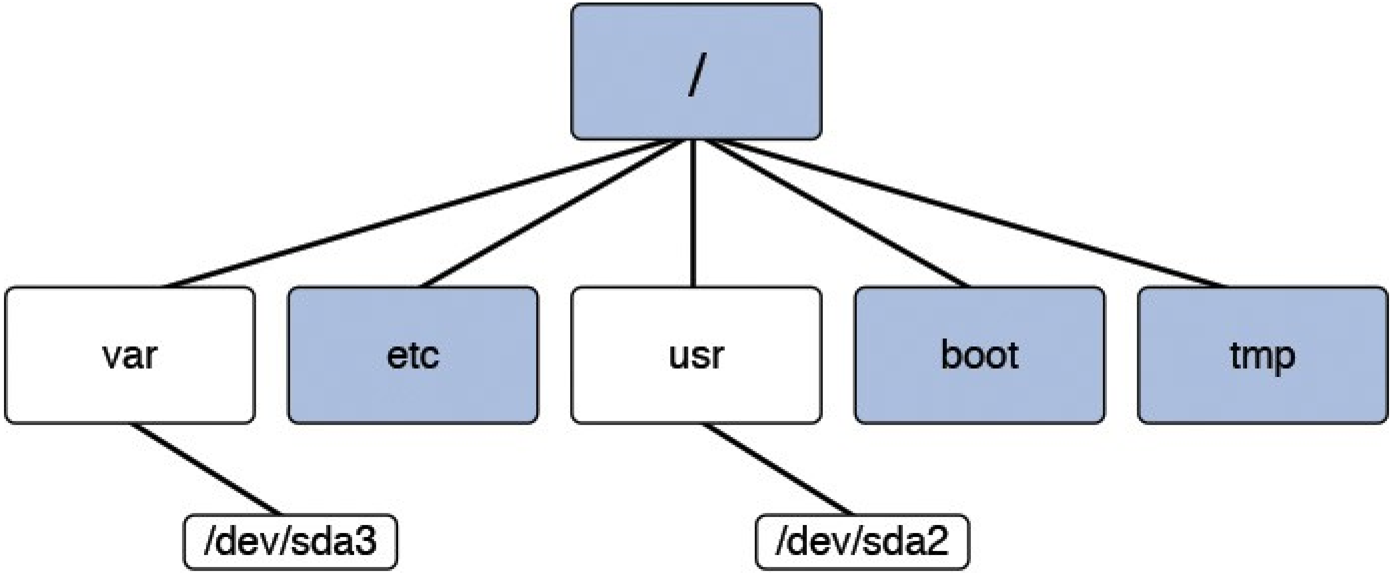

Inodes
Data Region
S i d I I I I I
D D D D D D D D
0
7
8
15
16
23
24
31
Data Region
32
39
40
47
48
55
56
63

Filesystem Philosophy

- Everything is a file
  - /dev/nvidia0
  - /proc/1
- Single Tree
  - “/”
- Files are a big bag of bytes
  - Streams of bytes with little metadata
  - Not so true with xattrs now. > 2.5 Kernel
  - ioctl

man hier


/
/boot
/bin
/dev
/etc
/lib
/home
/opt
/usr
/var
/sys
/proc
/dev/vda
/opt/chroot-ssh
/usr/bin
/usr/sbin
/usr/lib
/usr/lib/x86_64-linux-gnu
chroot - becomes /
/etc
/opt
/bin
/home
/usr
/var



Shaded boxes are in the filesystem
that is stored on /dev/sda1

Pathname	Contents
/bin	Core operating system commands
/boot	Boot loader, kernel, and files needed by the kernel
/compat	On FreeBSD, files and libraries for Linux binary compatibility
/dev	Device entries for disks, printers, pseudo-terminals, etc.
/etc	Critical startup and configuration files
/home	Default home directories for users
/lib	Libraries, shared libraries, and commands used by **/bin** and **/sbin**
/media	Mount points for filesystems on removable media
/mnt	Temporary mount points, mounts for removable media
/opt	Optional software packages (rarely used, for compatibility)
/proc	Information about all running processes
/root	Home directory of the superuser (sometimes just /)
/run	Rendezvous points for running programs (PIDs, sockets, etc.)
/sbin	Core operating system commands[a]
/srv	Files held for distribution through web or other servers
/sys	A plethora of different kernel interfaces (Linux)
/tmp	Temporary files that may disappear between reboots
/usr	Hierarchy of secondary files and commands
/usr/bin	Most commands and executable files
/usr/include	Header files for compiling C programs
/usr/lib	Libraries; also, support files for standard programs
/usr/local	Local software or configuration data; mirrors **/usr**
/usr/sbin	Less essential commands for administration and repair
/usr/share	Items that might be common to multiple systems
/usr/share/man	On-line manual pages
/usr/src	Source code for nonlocal software (not widely used)
/usr/tmp	More temporary space (preserved between reboots)
/var	System-specific data and a few configuration files
/var/adm	Varies: logs, setup records, strange administrative bits
/var/log	System log files
/var/run	Same function as **/run**; now often a symlink
/var/spool	Spooling (that is, storage) directories for printers, mail, etc.
/var/tmp	More temporary space (preserved between reboots)

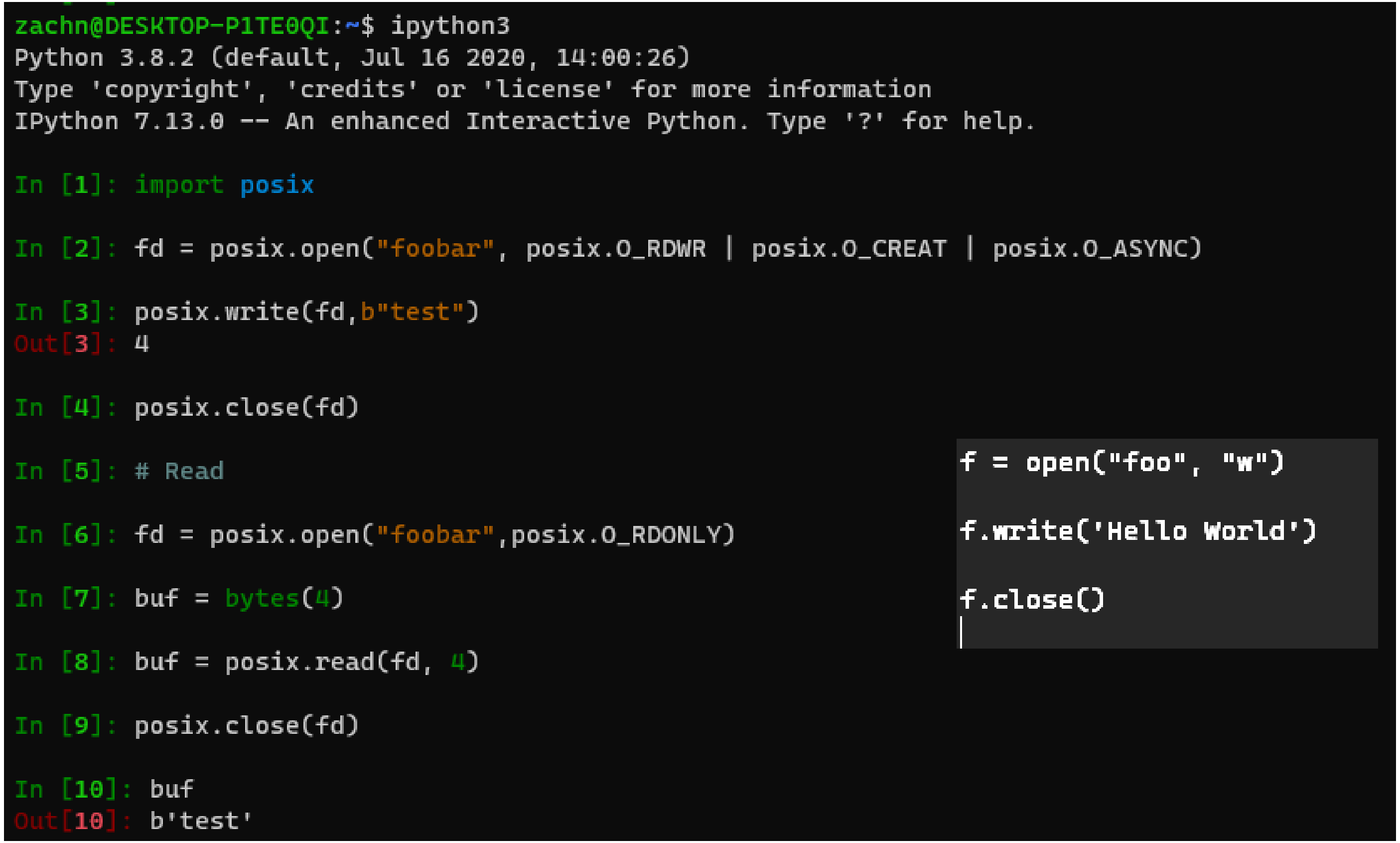
[a] The distinguishing characteristic of **/sbin** was originally that its contents were statically linked and so had fewer dependencies on other parts of the system. These days, all binaries are dynamically linked and there is no real difference between **/bin** and **/sbin**.

Size Requirements

- /boot
  - 100 MB (modern recommendation is 1 GB). Keep kernels under the 1024-cylinder limit.
- swap
  - 1 GB, depending on RAM.
- /
  - 500 MB (minimum).
- /usr
  - 4 GB. All of the executables in /usr are shared to workstations via read-only NFS.
- /var
  - 2 GB. Since log files are in their own partition, they won’t threaten system stability if the filesystem is full.
- /tmp
  - 500 MB. Since temporary files are in their own partition, they won’t threaten system stability if the filesystem is full.
- /home
  - 90 GB. This big partition takes up the vast bulk of available space, offered to users for their home directories and data.

Kernel Virtual File System

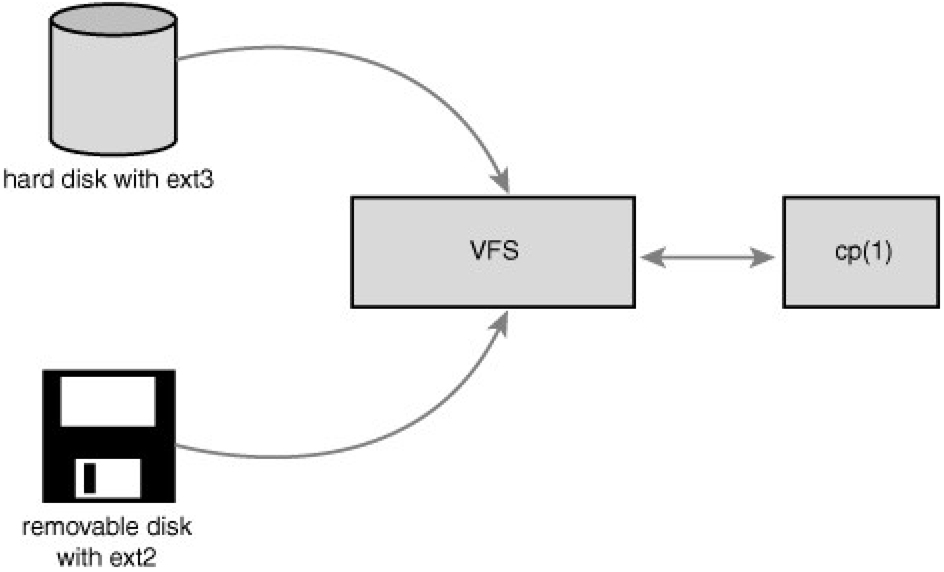
- Subsystem that implements file and filesystem related interfaces to user-land
- Almost all filesystems rely on VFS
- Allows all applications to use standard *nix system system calls
  - `open(2)`
  - `stat(2)`
  - `read(2)`
  - `write(2)`
  - `chmod(2)`
- Works between filesystems and devices



```
zachn@DESKTOP-P1TE0QI:~$ ipython3
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import posix

In [2]: fd = posix.open("foobar", posix.O_RDWR | posix.O_CREAT | posix.O_ASYNC)

In [3]: posix.write(fd,b"test")
Out[3]: 4

In [4]: posix.close(fd)

In [5]: # Read

In [6]: fd = posix.open("foobar",posix.O_RDONLY)

In [7]: buf = bytes(4)

In [8]: buf = posix.read(fd, 4)

In [9]: posix.close(fd)

In [10]: buf
Out[10]: b'test'
```

```
f = open("foo", "w")

f.write('Hello World')

f.close()
```

VFS



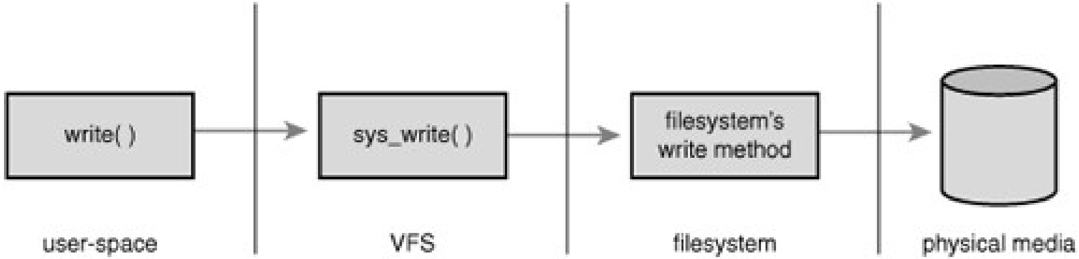
Linux Kernel Development, 3rd Edition

VFS - Abstraction Layer

- Abstracts the low-level filesystem interface (ext{2,3,4})
- Allows different filesystems to interact
  - Compression
  - Encryption
  - ZFS
- Biased to Unix-style filesystems
  - Support for NTFS, FAT, exFAT, SAMBA

VFS - Abstraction Layer

- Defines interfaces and data structures
- VFS Defines
  - How to open files
  - This is what a directory is
- Results in support for many types of file systems easily
  - All of these filesystems implement the VFS interfaces and structures
- The Kernel doesn’t need to know the underlying implementation



Linux Kernel Development, 3rd Edition

*nix Filesystems

- Filesystem abstractions
  - Files - Ordered string of bytes, human readable name
  - Directory Entries - analogous to a folder of files. Each part of a path is a directory entry (dentry)
  - Inodes - represents an object within the file system
  - Mount Points - superblock
- Filesystem is hierarchical storage adhering to a structure

VFS superblock

- A superblock represents a filesystem
- Defined in <linux/fs.h> struct super_block
- Properties
  - Mount point
  - Block size
  - Quotas
  - Mount flags (rw, ro, sync, async)
  - Filesystem type

VFS inode

- An inode stores metadata about a file
- Defined in <linux/fs.h> struct inode
- Properties
    - Bytes consumed
    - User ID
    - Group ID
    - Last access time
    - Last modify time
    - Last change time
    - Permissions

```
find . -name "*" -printf "%u %p %k \n"
```

VFS dentry

- Parts of a path associated with an inode
  - /bin/vim ->/ bin / vim
- Properties
  - Parent
  - Name
  - inode
  - Subdirectories

VFS file

- In-memory representation of a file
- Content has no inherit structure
- Defined in <linux/fs.h> struct inode
- Properties
  - dentry
  - mode
- Functions
  - Read
  - Write
  - Open
  - Close
  - Lock

Filesystems in the kernel

62 total

fscache
reiserfs
ext4
ext2
jbd2
cramfs
squashfs
ramfs
hugetlbfs
coda

minix
fat
bfs
isofs
hfsplus
hfs
ecryptfs
freevxfs
nfs
exportfs
cifs

ntfs
ufs
fuse
overlayfs
jfs
xfs
autofs
tmpfs

File Types

Regular files

- Bag of bytes
- No structure
- Text, binary, executables, shared libraries

Portable Network Graphic

- Image format
- Designed for the WWW
- *25 years old*
- Made up of Chunks
  - Can be streamed
- Designed to be simple, legally unencumbered, compressed, interchangeable, flexible and robust.
- Pronounced “ping”

PNG Binary Format

Self describing chunks

Starts with a file signature

b'\x89**PNG**\r\n\x1a\n'

\r\n - Return, Newline

\x1a - CTRL-Z, DOS SUB character, EOF character. Hides the binary data.

PNG Chunks

Typical Image

1. Signature
2. IHDR
3. IDAT
4. IEND

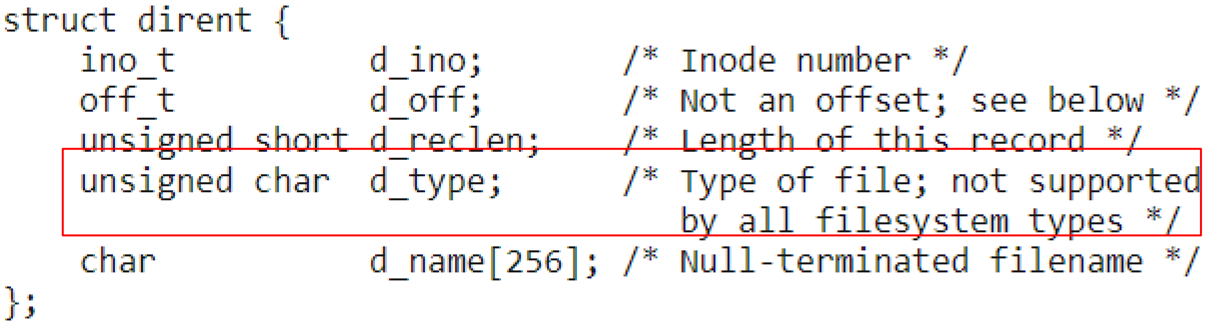
Chunks: gAMA, tEXT, tRNS, cHRM, sRGB, iCCP, bKGB, etc.

Directories

A file that points to other files. Limited to 255 characters.

Each file is a `struct dirent`

dirent stream is read with `getdents64(2) or readdir(3)`



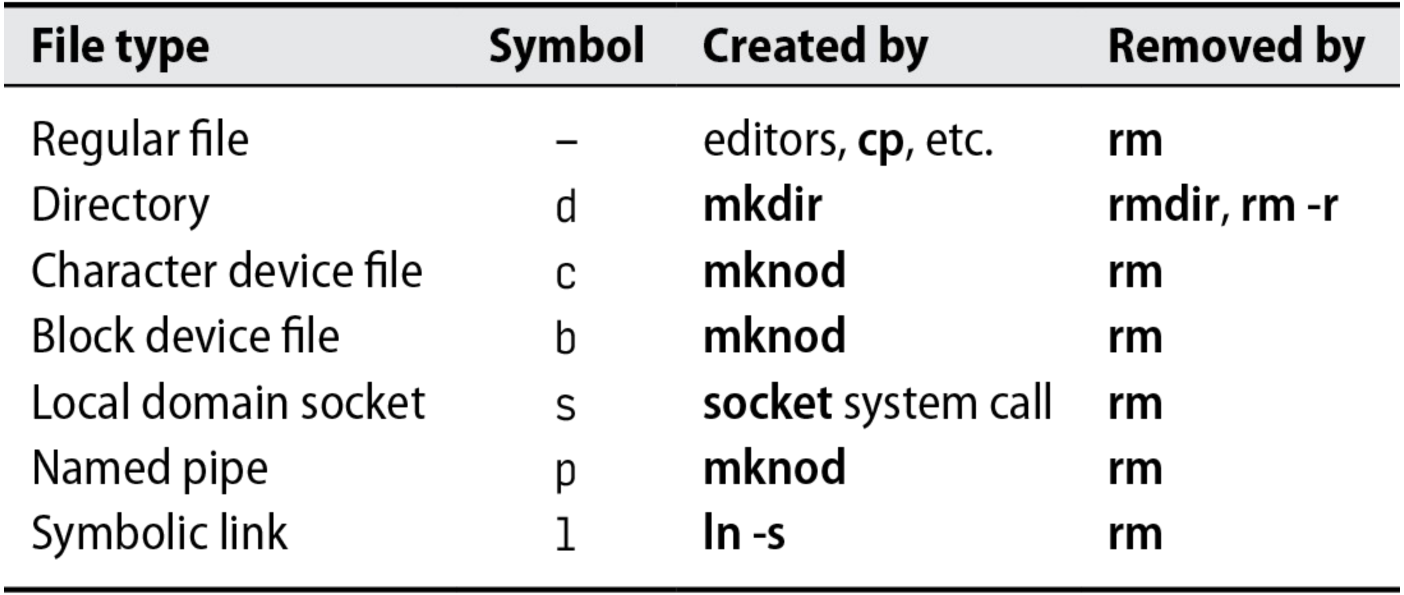
```
struct dirent {
    ino_t          d_ino;       /* Inode number */
    off_t          d_off;       /* Not an offset; see below */
    unsigned short d_reclen;    /* Length of this record */
    unsigned char  d_type;      /* Type of file; not supported
                                   by all filesystem types */
    char           d_name[256]; /* Null-terminated filename */
};
```

Files in a directory

d_type

DT_BLK	This is a block device.
DT_CHR	This is a character device.
DT_DIR	This is a directory.
DT_FIFO	This is a named pipe (FIFO).
DT_LNK	This is a symbolic link.
DT_REG	This is a regular file.
DT_SOCK	This is a UNIX domain socket.
DT_UNKNOWN	The file type could not be determined.

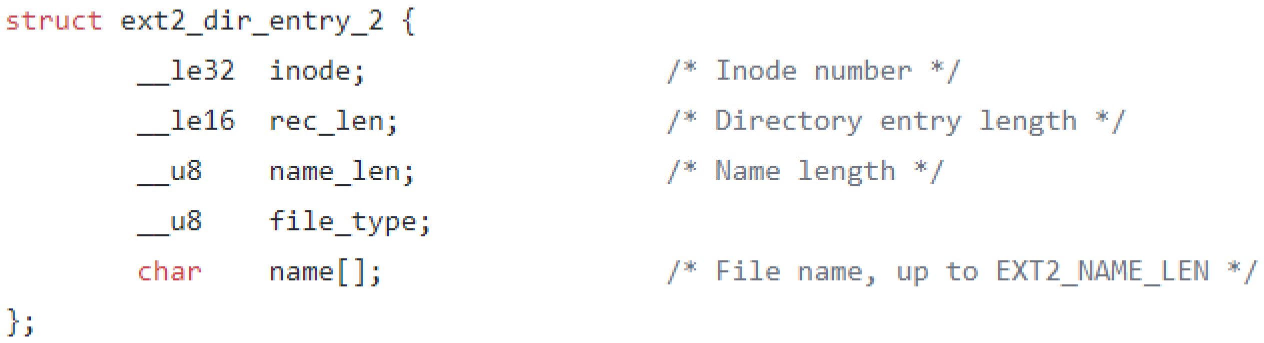
File Types



File type	Symbol	Created by	Removed by
Regular file	–	editors, **cp**, etc.	**rm**
Directory	d	**mkdir**	**rmdir**, **rm -r**
Character device file	c	**mknod**	**rm**
Block device file	b	**mknod**	**rm**
Local domain socket	s	**socket** system call	**rm**
Named pipe	p	**mknod**	**rm**
Symbolic link	l	**ln -s**	**rm**

```
ls -lha /dev
find . -type d -exec file {} \;
```

ext2 implementation

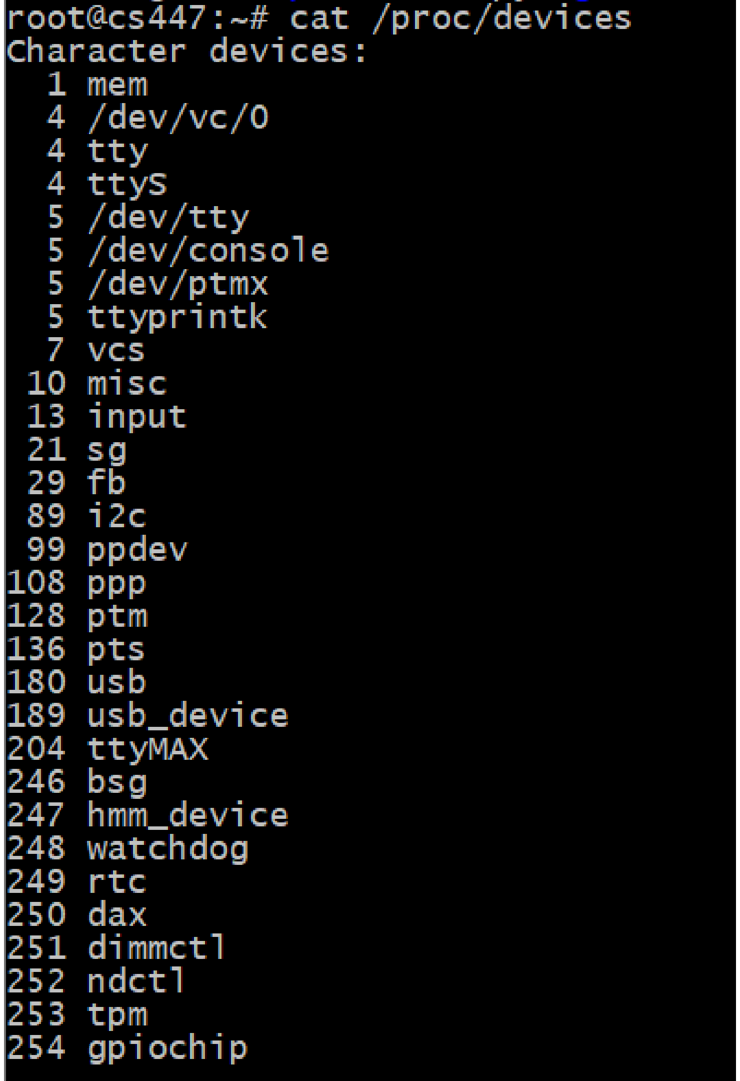


```
struct ext2_dir_entry_2 {
        __le32  inode;                  /* Inode number */
        __le16  rec_len;                /* Directory entry length */
        __u8    name_len;               /* Name length */
        __u8    file_type;
        char    name[];                 /* File name, up to EXT2_NAME_LEN */
};
```

Character device file

- Made with `mknod(1)`
- Allows you to communicate with hardware
  - GPUs
  - Serial ports - /dev/ttyUSB0
  - Audio
  - /dev/random
  - i2c
- Major and Minor numbers, 252:0

```
find /dev -type c
```

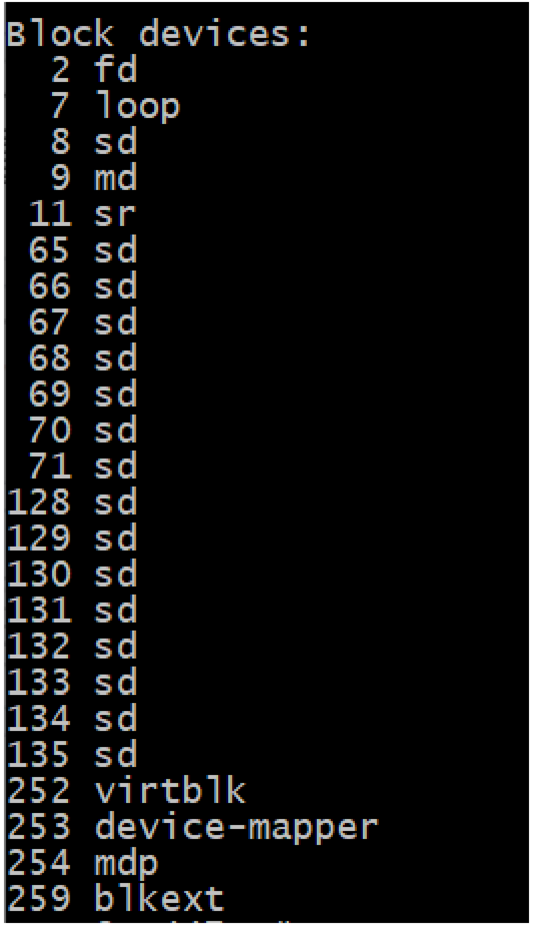


```
root@cs447:~# cat /proc/devices
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  5 ttyprintk
  7 vcs
 10 misc
 13 input
 21 sg
 29 fb
 89 i2c
 99 ppdev
108 ppp
128 ptm
136 pts
180 usb
189 usb_device
204 ttyMAX
246 bsg
247 hmm_device
248 watchdog
249 rtc
250 dax
251 dimmctl
252 ndctl
253 tpm
254 gpiochip
```

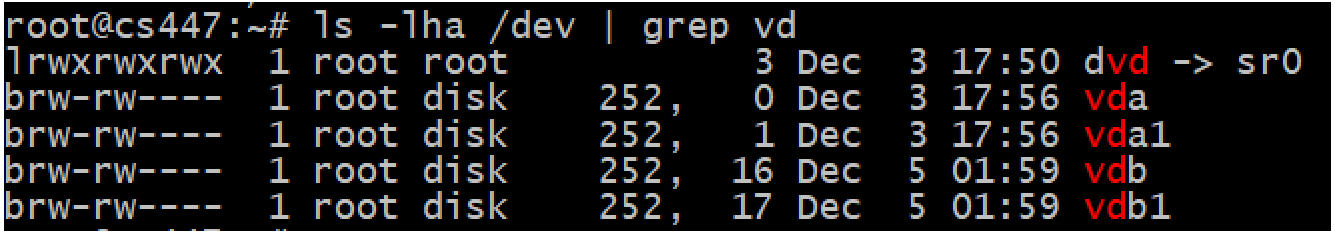
Block device file

- Represents a block device
    - Storage
- Similar to Character device
- Writes and reads in fixed sizes. 512 bytes (default)

```
dd if=/dev/vda of=first_block.img count=1

find /dev -type b;
```



```
Block devices:
  2 fd
  7 loop
  8 sd
  9 md
 11 sr
 65 sd
 66 sd
 67 sd
 68 sd
 69 sd
 70 sd
 71 sd
128 sd
129 sd
130 sd
131 sd
132 sd
133 sd
134 sd
135 sd
252 virtblk
253 device-mapper
254 mdp
259 blkext
```



```
root@cs447:~# ls -lha /dev | grep vd
lrwxrwxrwx  1 root root           3 Dec  3 17:50 dvd -> sr0
brw-rw----  1 root disk    252,   0 Dec  3 17:56 vda
brw-rw----  1 root disk    252,   1 Dec  3 17:56 vda1
brw-rw----  1 root disk    252,  16 Dec  5 01:59 vdb
brw-rw----  1 root disk    252,  17 Dec  5 01:59 vdb1
```

Local domain socket file

- AKA Unix socket
- Used for interprocess communication
- Bi-directional
- Created with the socket(2) function
- bind(2) assigns an address
  - Filename
  - IP + Port
- Client-Server

```
find /dev -type s;
man 7 unix
```

Local domain socket file

- What are they used for?
  - UWSGI - Application Container
  - syslog - Logging
  - systemd - Linux init process
  - SQL Databases - MySQL, PostgreSQL
  - Mail - postfix, dovecot, mailx, alpine
  - Virtualization - QEMU management interfaces
  - Graphical User Interfaces - Xorg, GNOME, etc.

Named pipe file

- AKA First-In First-Out (FIFO) file
- Unidirectional file for interprocess communication
- Created with mknod(1)
- Not used much compared to AF_UNIX or AF_INET
- Blocks until read

Hard link

- Filename is stored within the parents directory
- A file can be in more than one directory
    - Directories “link” the file aka hardlink
- Hard links are not a file type
    - Filesystems allow >= 1 directory entries.
- Create with `ln(1)`
- Deleted with `rm(1)`

Symbolic Link

- Makes a link between files
  - Soft - Points to file path (dentry)
  - Hard - Points to inode

```
ln [OPTION]... [-T] TARGET LINK_NAME
ln /data/file.txt /var/www/html/file.txt    # Hard
ln -s /data /var/www/html/data              # Soft

find /etc -type l; # Find symbolic links
```