

Liana + Anna Lecture!

Heavily edited photo of Margaret Hamilton, 1969



WEEEEEEEEEEK NINE !!

All functionality and extensions through assign6 are out! Style grades in the works.

Assignment 7 due on Tuesday 5PM (on-time) or Thursday 5PM (in grace)

We'll re-test every night; the last deadline for re-submits is Sunday March 12th at 5PM

Project labs are this week and next week

WEEEEEEEEEEK NINE !!

All functionality and extensions through assign6 are out! Style grades in the works.

Assignment 7 due on Tuesday 5PM (on-time) or Thursday 5PM (in grace)

We'll re-test every night; the last deadline for re-submits is Sunday March 12th at 5PM

Project labs are this week and next week

WEEEEEEEEEEK NINE !!

All functionality and extensions through assign6 are out! Style grades in the works.

Assignment 7 due on Tuesday 5PM (on-time) or Thursday 5PM (in grace)

We'll re-test every night; the last deadline for re-submits is Sunday March 12th at 5PM

Project labs are this week and next week

WEEEEEEEEEEK NINE !!

All functionality and extensions through assign6 are out! Style grades in the works.

Assignment 7 due on Tuesday 5PM (on-time) or Thursday 5PM (in grace)

We'll re-test every night; the last deadline for re-submits is Sunday March 12th at 5PM

Project labs are this week and next week

Today

Anna's favorite thing: file systems

Liana's favorite thing: ethics

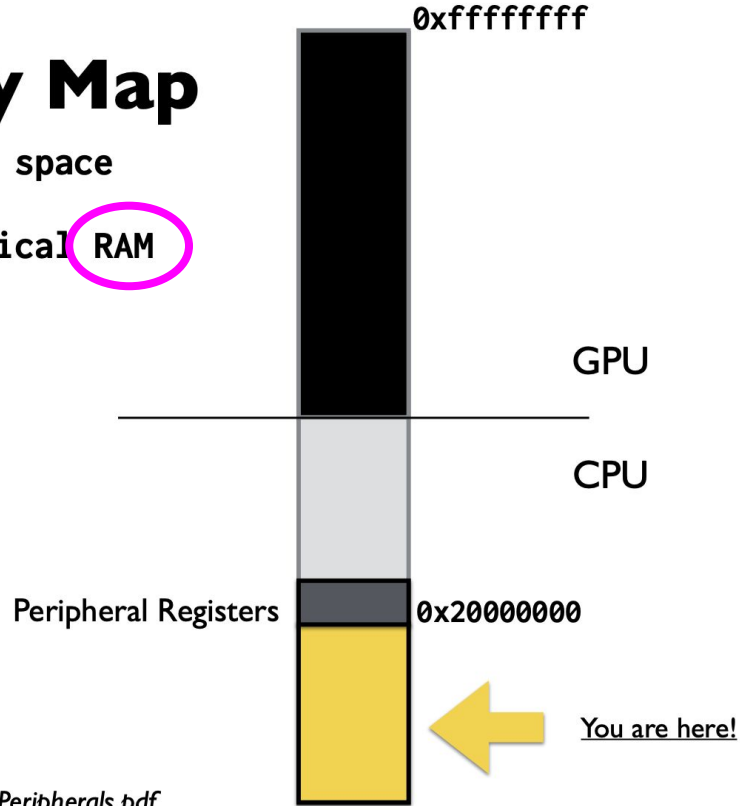
Final projects !!

File systems

Memory Map

32-bit address space

512 MB of physical **RAM**



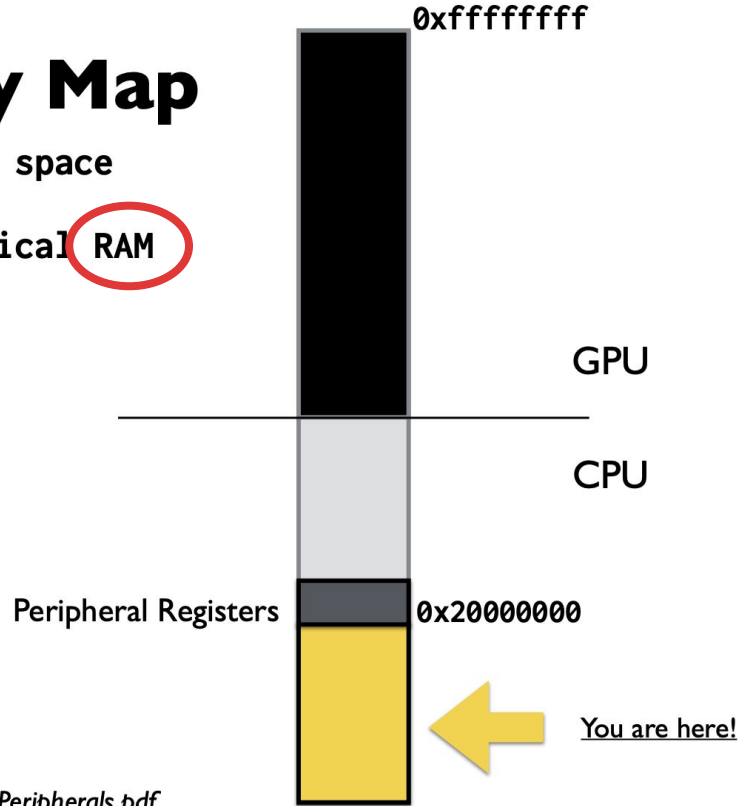
Ref: BCM2835-ARM-Peripherals.pdf



Memory Map

32-bit address space

512 MB of physical **RAM**



Ref: BCM2835-ARM-Peripherals.pdf

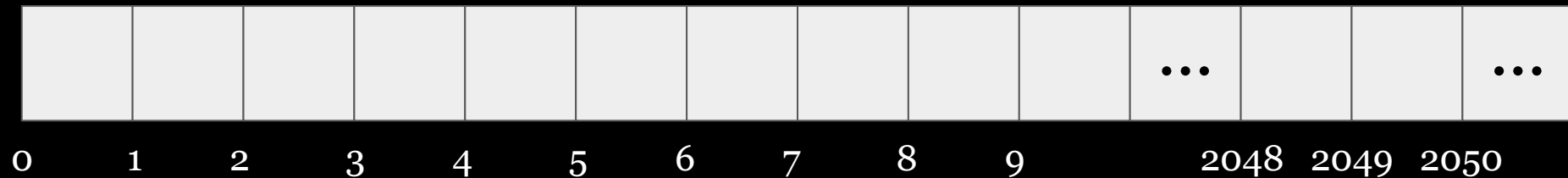
What about the HARD DRIVE?



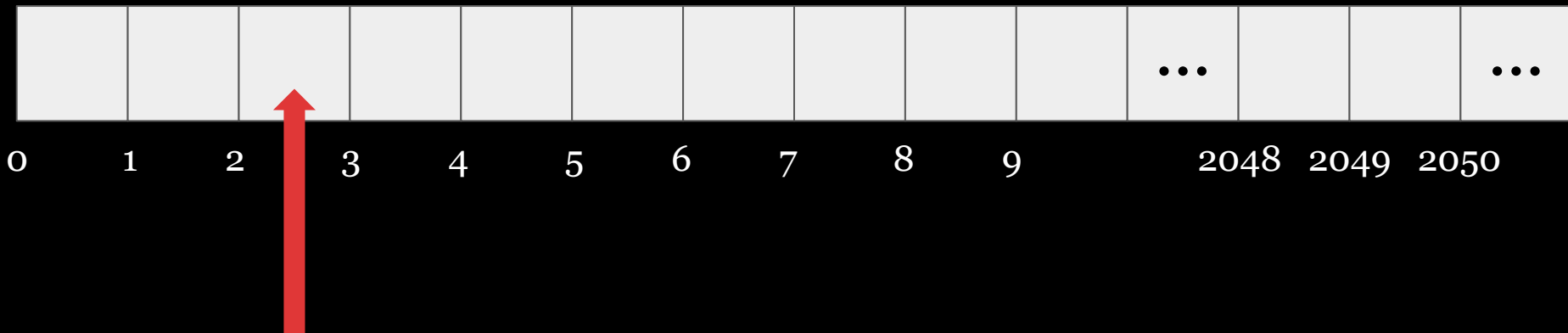
Why file systems are
the coolest thing you'll
ever see



The hard drive is not byte-addressable



The hard drive is not byte-addressable



Each sector is
512 bytes*

* The size of a sector depends on the hardware, but here we'll assume a sector represents 512 bytes

The hard drive is not byte-addressable



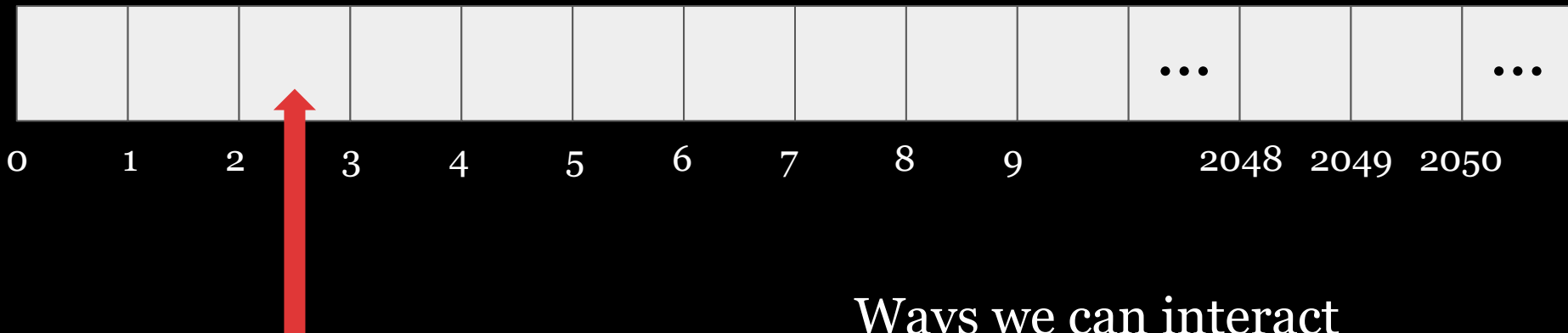
Each sector is
512 bytes*

Ways we can interact
with the hard drive:

`readSector`
`writeSector`

* The size of a sector depends on the hardware, but here we'll assume a sector represents 512 bytes

The hard drive is not byte-addressable



Each sector is
512 bytes*

Ways we can interact
with the hard drive:

```
readSector(sectorNum, buf);  
writeSector(sectorNum, buf);
```

* The size of a sector depends on the hardware, but here we'll assume a sector represents 512 bytes

The hard drive is not byte-addressable



Each sector is
512 bytes*

Ways we can interact
with the hard drive:

```
readSector(sectorNum, buf);  
writeSector(sectorNum, buf);  
char buf[512];
```

* The size of a sector depends on the hardware, but here we'll assume a sector represents 512 bytes

So how do we store files
in these sectors?



printf.c

(1024 bytes)

```
code code code code  
code buf[256] lots  
of code %s if else  
va_list code code  
code snprintf code  
code code code code
```

printf.c

(1024 bytes)

```
code code code code  
code buf[256] lots  
of code %s if else  
va_list code code  
code snprintf code  
code code code code
```

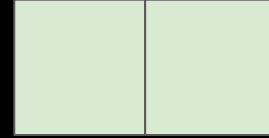
printf.c

size: 1024

file type: regular

permissions: ...

printf.c data

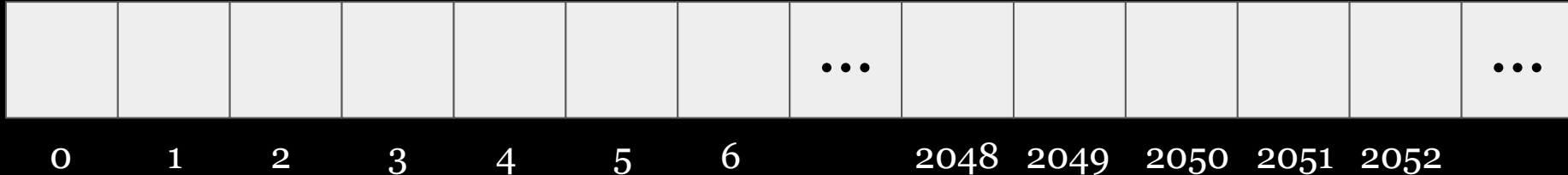
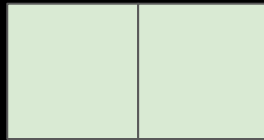


printf.c (1024 bytes)

```
code code code code  
code buf[256] lots  
of code %s if else  
va_list code code  
code snprintf code  
code code code code
```

printf.c
size: 1024
file type: regular
permissions: ...

printf.c data



printf.c (1024 bytes)

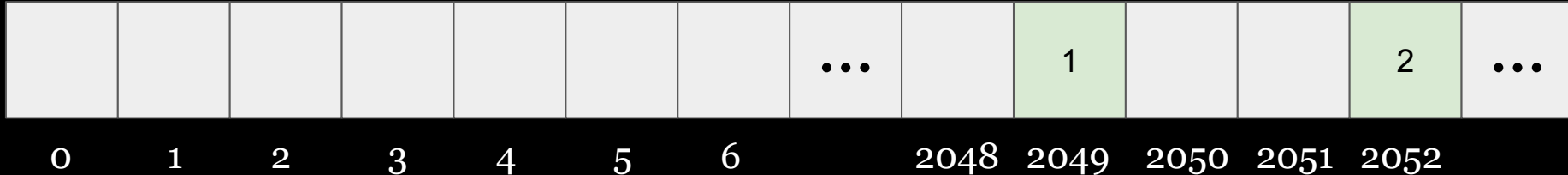
```
code code code code  
code buf[256] lots  
of code %s if else  
va_list code code  
code snprintf code  
code code code code
```

printf.c
size: 1024
file type: regular
permissions: ...

sectors:
2049, 2052

printf.c data

1	2
---	---



printf.c (1024 bytes)

```
code code code code  
code buf[256] lots  
of code %s if else  
va_list code code  
code snprintf code  
code code code code
```

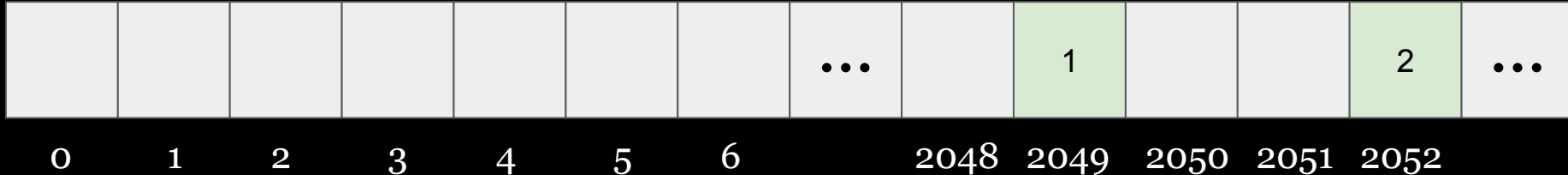
printf.c
size: 1024
file type: regular
permissions: ...

sectors:
2049, 2052

printf.c data

1	2
---	---

What if our file size isn't a
perfect multiple of 512?





inode

printf.c

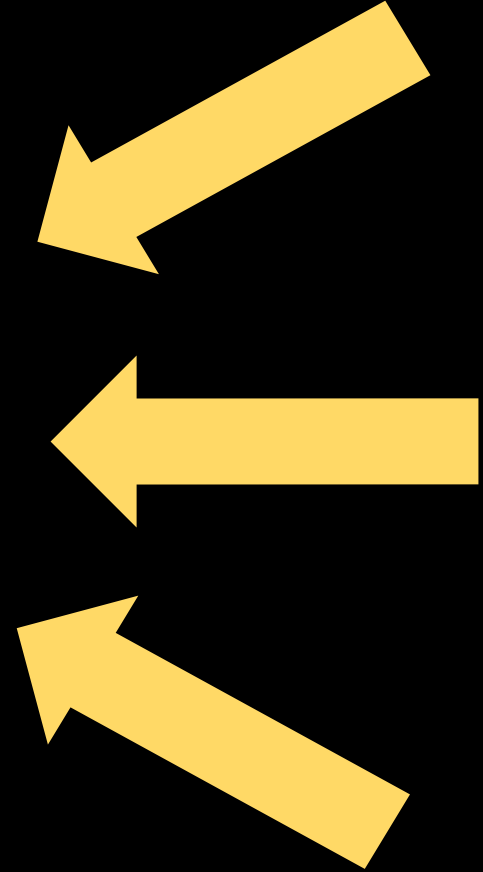
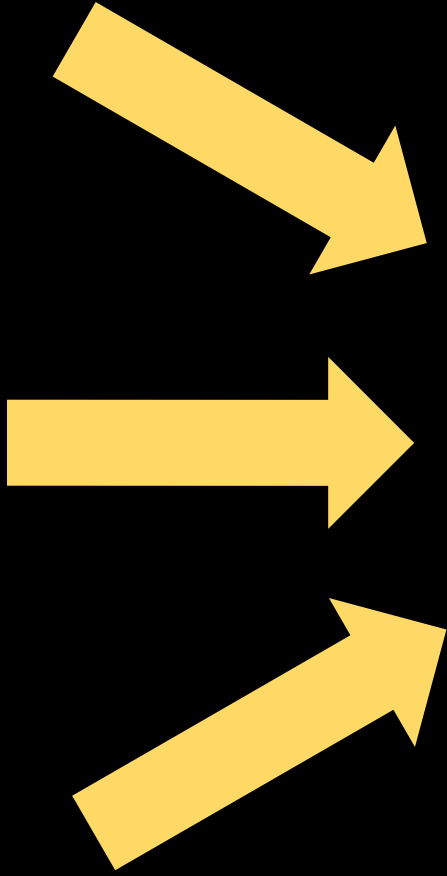
size: 1024

file type: regular

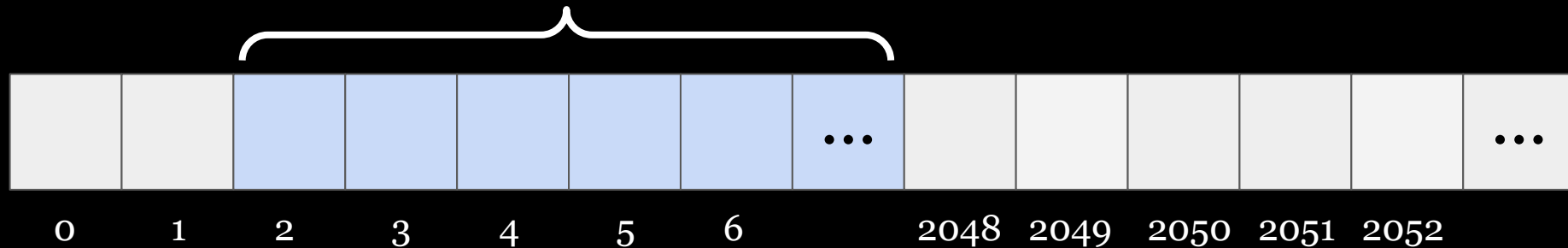
permissions: ...

sectors:

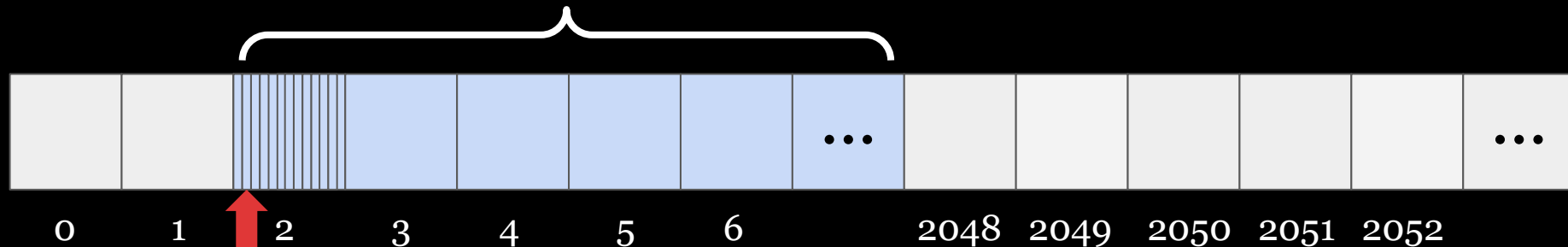
2049, 2052



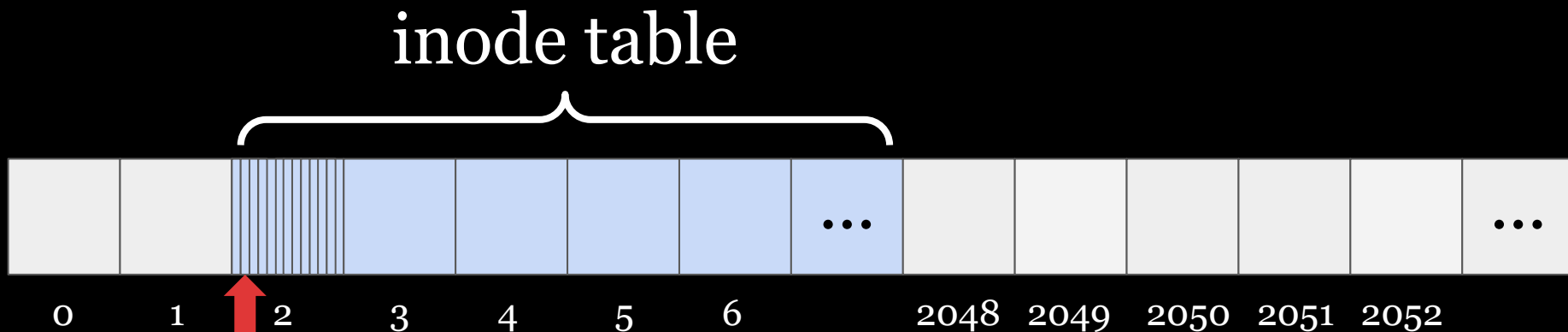
inode table



inode table

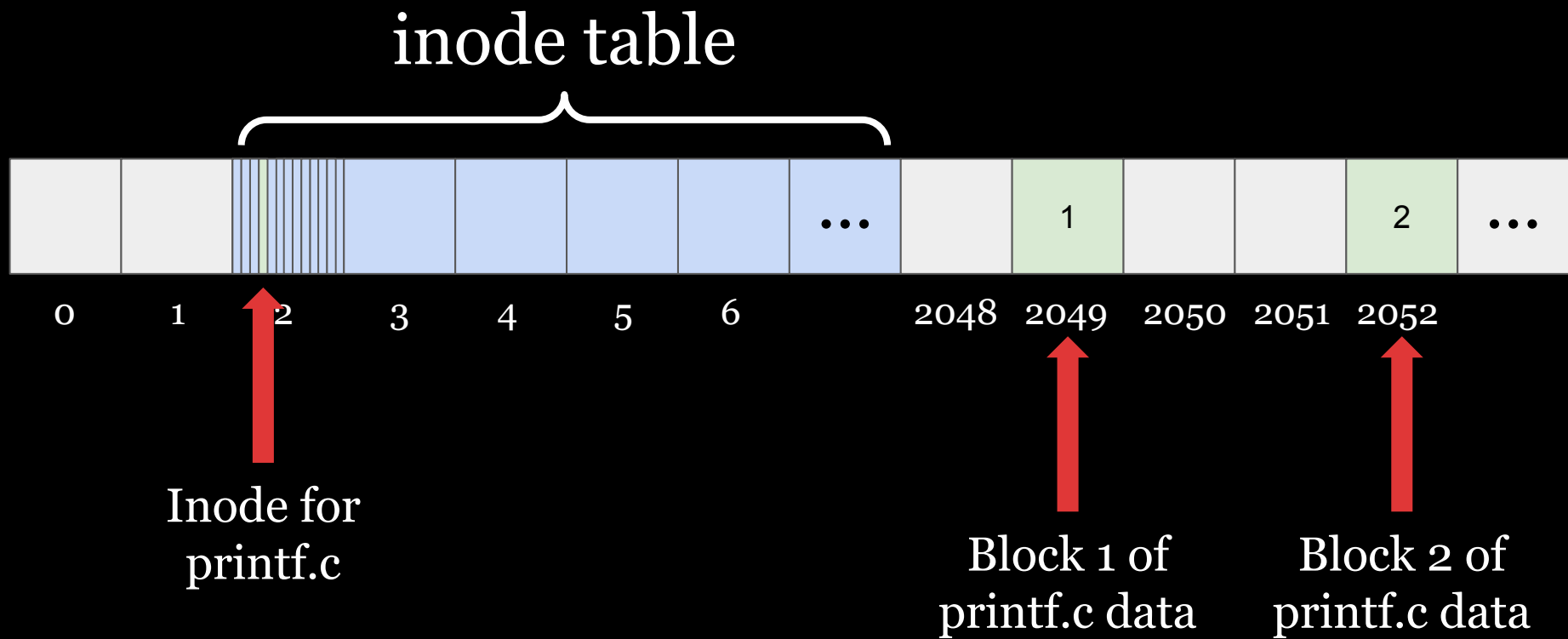


Each sector
holds 16 inodes



Each sector
holds 16 inodes

Inode numbering:
Sector 2 has inodes
1-16, Sector 3 has
inodes 17-32, etc.



Directories!



An inode can represent a...

regular file

printf.c

size: 1024

file type: regular

permissions: ...

sectors:

2049, 2052

directory

/

size: 64

file type: directory

permissions: ...

sectors:

2048

/
size: 64
file type: directory
permissions: ...

sectors:
2048

Sector 2048:

.	1
..	1
printf.c	4
trombone.mp3	8

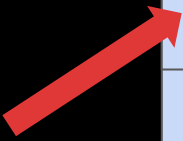
/
size: 64
file type: directory
permissions: ...

sectors:
2048

Sector 2048:

.	1
..	1
printf.c	4
trombone.mp3	8

File name*



*A file can mean a regular file or a directory!

/
size: 64
file type: directory
permissions: ...

sectors:
2048

Sector 2048:

.	1
..	1
printf.c	4
trombone.mp3	8

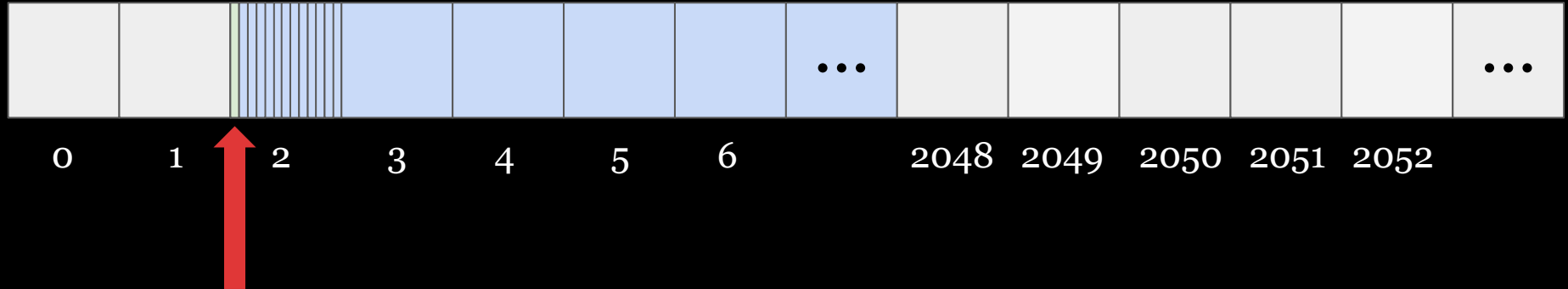
File name*

The inode
number for
the file

*A file can mean a regular file or a directory!

Diagram illustrating a 256-bit vector V . The vector is divided into segments, with indices 0, 1, 2, 3, 4, 5, 6, ..., 2048, 2049, 2050, 2051, 2052, and ... shown below. Segments 3 through 6 are highlighted in light blue.

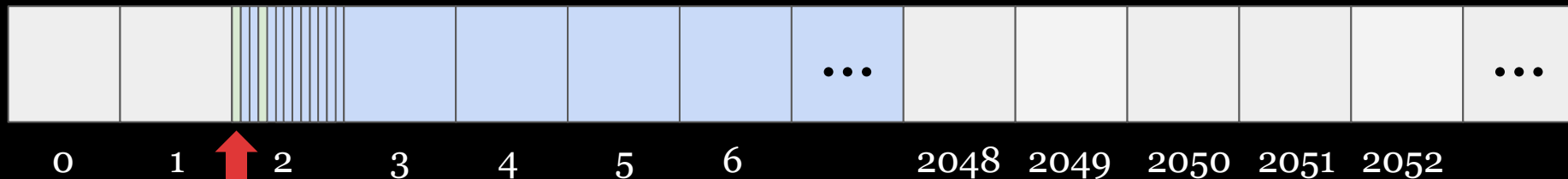
Let's walk through!!



Root inode

The inode representing the root folder
/
is always stored in Inode #1
(sector 2, offset 0)

Let's walk through!!

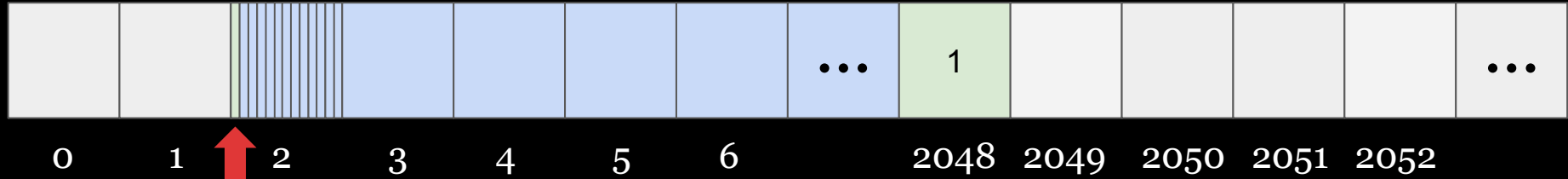


Root inode

```
/
size: 64
file type: directory
permissions: ...
```

sectors:
2048

Let's walk through!!



Root inode

/

size: 64

file type: directory

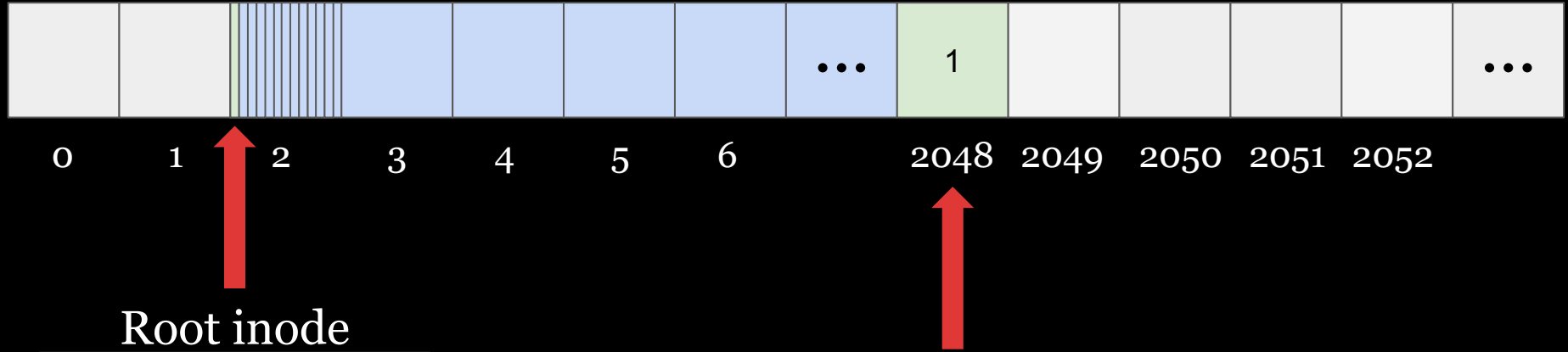
permissions: ...

sectors:

2048

Information about what's
in the root directory

Let's walk through!!



Root inode

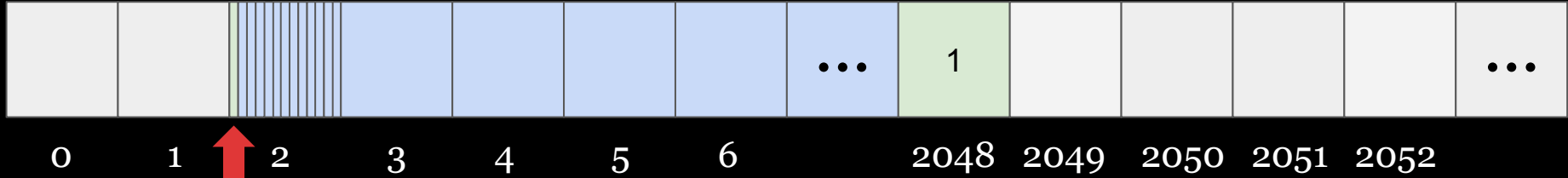
/

size: 64
file type: directory
permissions: ...

sectors:
2048

.	1
..	1
printf.c	4
trombone.mp3	8

Let's walk through!!



Root inode

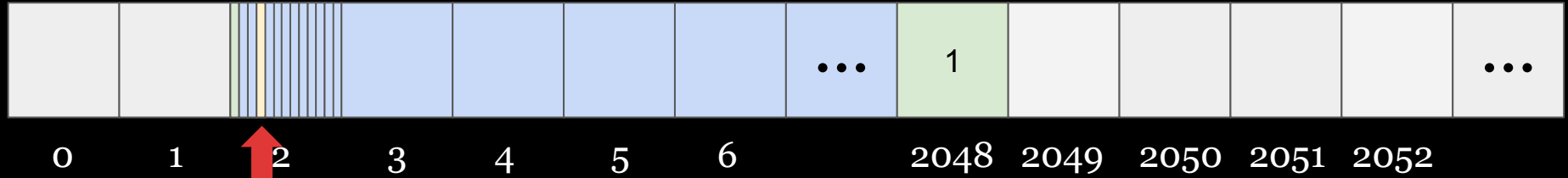
/
size: 64
file type: directory
permissions: ...

sectors:
2048

.	1
..	1
printf.c	4
trombone.mp3	8

If we want to
get to printf.c,
we need to
check Inode #4

Let's walk through!!



printf.c inode

printf.c

size: 1024

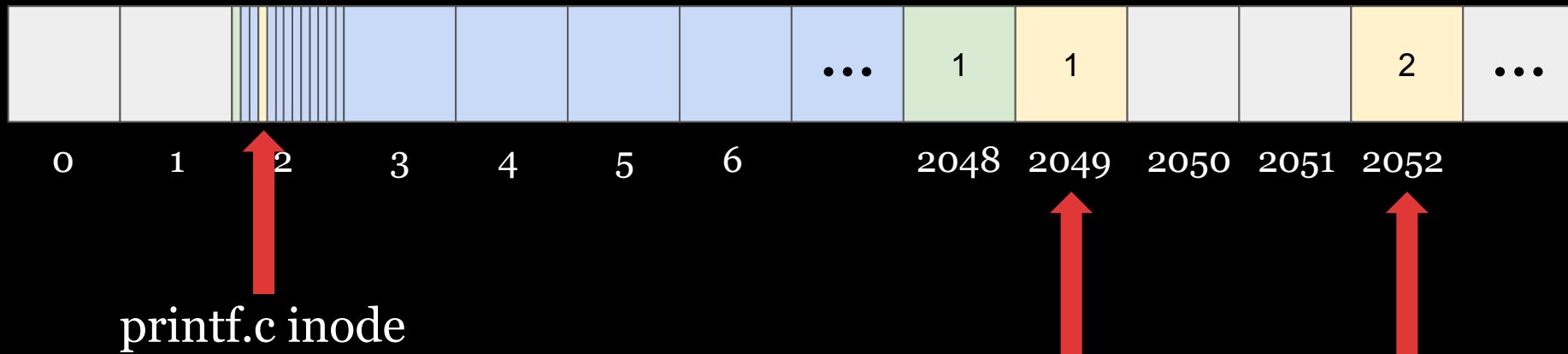
file type: regular

permissions: ...

sectors:

2049, 2052

Let's walk through!!



printf.c inode

printf.c

size: 1024

file type: regular

permissions: ...

sectors:

2049, 2052

printf.c data
(can be read like a
normal file!)

So you start at Inode #1
(root folder) and you
can get anywhere from
there!





TAKE A DEEP BREATH because
that's how file systems work

inode

printf.c

size: 1024

file type: regular

permissions: ...

sectors:

2049, 2052

There is not
space to store a
ton of sector
numbers here!



inode

printf.c
size: 4098
file type: regular
permissions: ...

sectors:
2049, 2052, 2057,
4923, 2938, 2293,
2939, 2999, ????

This is $512 * 8 + 2$
(so we need 9
blocks)

What if we have
more than 8
sectors we want
to store??



Here is where things
get CRAZY



printf.c

size: 4098

file type: regular

permissions: ...

sectors:

3013

Sector 3013:

2049	2052	2057	4923	2938
2293	2939	2999	8392	...

This one sector tells us
where we're storing our
sector list!

think this stuff is cool?



Systems

Systems is the study of the design and implementation of computer systems such as compilers, databases, networks, and operating systems. Topics include the hardware/software interface, the networking stack, digital architecture, memory models, optimization, concurrency, privacy, security, distributed and large-scale systems, reliability and fault tolerance, and related algorithms and theoretical topics.

(In CS 111, there are 5 lectures on file systems, so if it felt a bit too fast-paced here, don't worry!!)

classes:

1

CS 111: Operating Systems Principles

Explores operating system concepts including concurrency, synchronization, scheduling, processes, virtual memory, I/O, file systems, and protection. Available as a substitute for CS110 that fulfills any requirement satisfied by CS110. Prerequisite: CS107.

2

CS 140: Operating Systems and Systems Programming

Operating systems design and implementation. Basic structure; synchronization and communication mechanisms; implementation of processes, process management, scheduling, and protection; memory organization and management, including virtual memory; I/O device management, secondary storage, and file systems. Prerequisite: CS110.

?

CS 140E: Operating systems design and implementation

Students will implement a simple, clean operating system (virtual memory, processes, file system) in the C programming language, on a raspberry pi computer and use the result to run a variety of devices and implement a final project. All hardware is supplied by the instructor, and no previous experience with operating systems, raspberry pi, or embedded programming is required.