CSc 352

C Programming
GDB, files

Benjamin Dicken

Announcements

Exam 1 Next week!

- Variety of question types (programming, explanation, FIB, etc)
- Will post a topic list, but not a study guide

No PA due on Friday the 25th

Key options for gdb

break - sets a stopping points within the code

run - starts the program running

next / step - walk through the program

bt - backtrace

frame - show information for a stack frame (**`info frame X`**)

print - display the value of a variable / expr

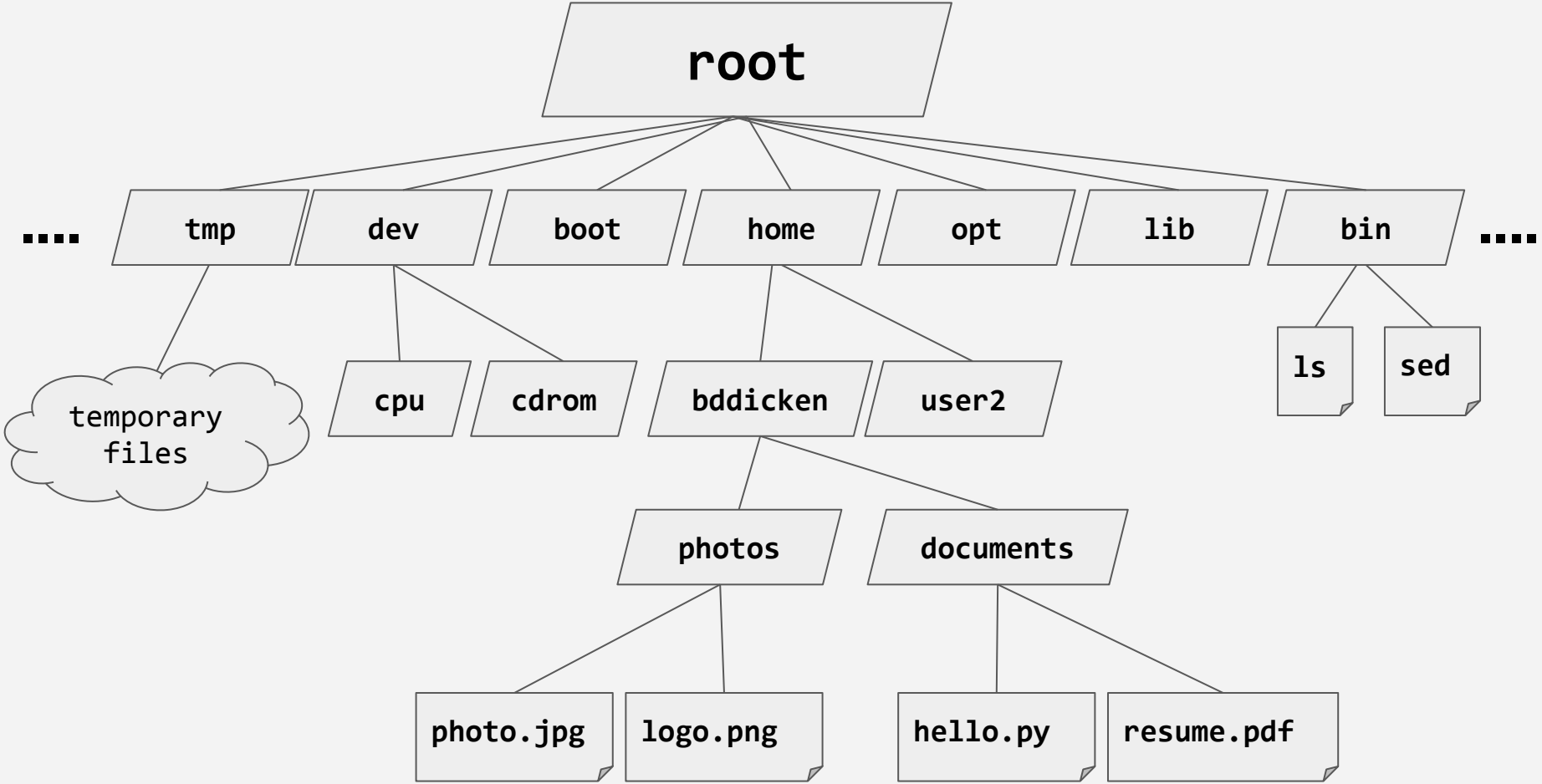
Debug

- Download **`code.c`** and **`makefile`** from the class website
- Without modifying the makefile or C file, determine:
  - What could cause this program to crash?
  - Why?
  - Use GDB
- I'll give you 5-7 minutes to download, test, explore with GDB, then we can discuss

The UNIX File System

- The file system is a core component to a UNIX operating system
- There are different specific implementations, but there are shared general-principles
  - UFS, EXT2, EXT3, EXT4, ZFS, etc, etc
- We will focus on the general principles

https://en.wikipedia.org/wiki/Unix_filesystem


root
....
tmp
dev
boot
home
opt
lib
bin
....
temporary files
cpu
cdrom
bddicken
user2
ls
sed
photos
documents
photo.jpg
logo.png
hello.py
resume.pdf

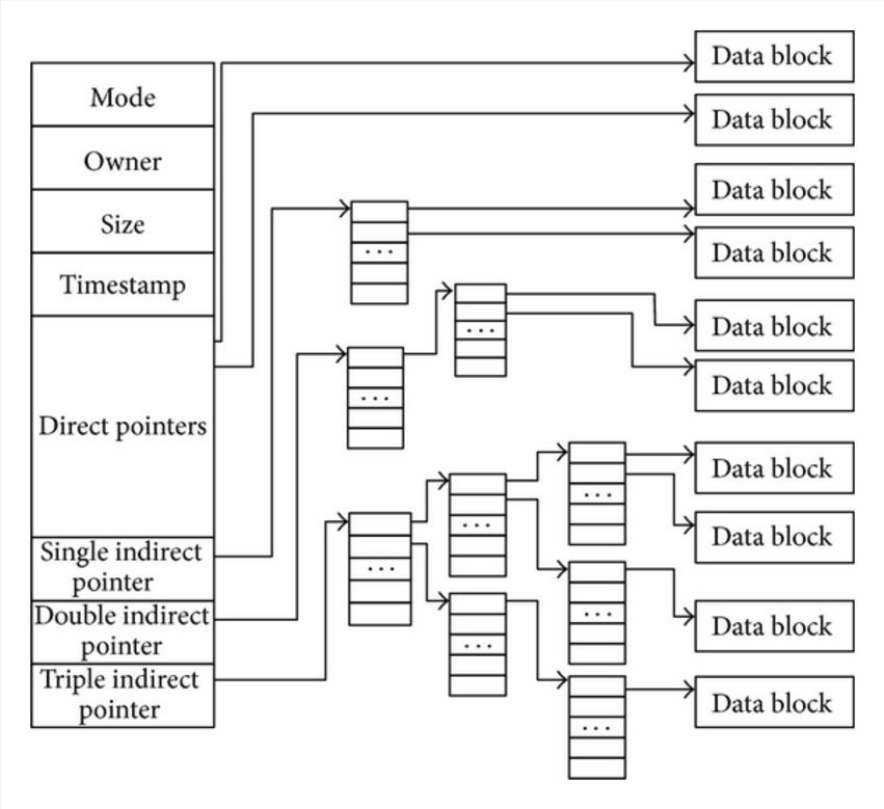
Files vs Directories

- A “regular” file (.txt, .c, .out, etc) and a directory are both just files
- A directory files contains a list of inodes including itself, its parent, and its child inodes
- Try: **`$ ls -l`**

https://en.wikipedia.org/wiki/Inode

iNodes

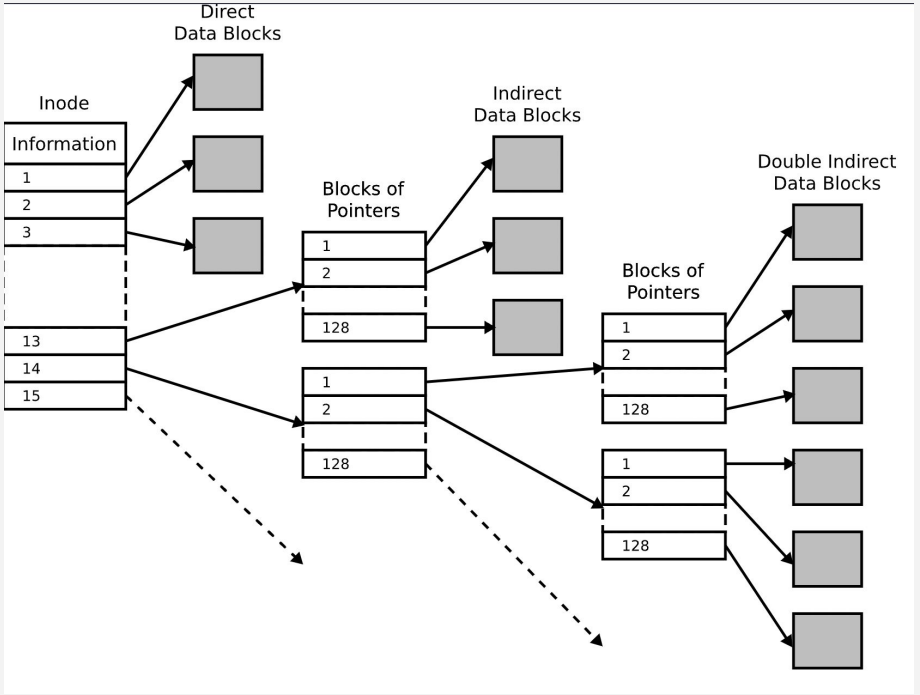
- Behind the scenes, a **file** is really a node containing a collection of **metadata** and pointers to blocks of the actual **data**
- These nodes are called **inodes**
- The file systems stores a table or a tree of **inodes** on the actual hard drive



https://www.researchgate.net/figure/The-architecture-of-an-inode-in-EXT3-file-system_fig2_258396310

iNodes pointer structure

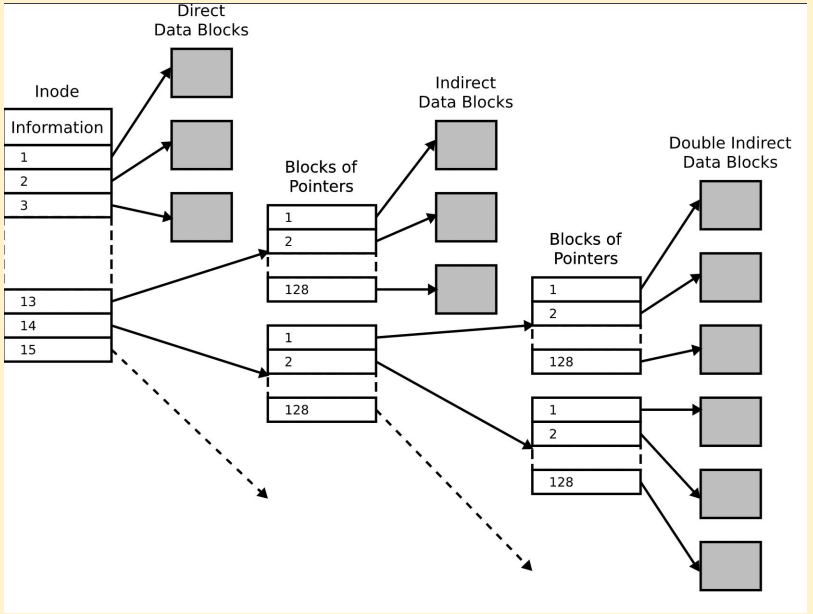
- For small files, can store data within the blocks from the direct pointers
- For larger files, use some of the indirect pointers
- Find block size: **`$ stat -f /`**



https://en.wikipedia.org/wiki/Inode_pointer_structure
http://www.linuxintro.org/wiki/Blocks,_block_devices_and_block_sizes

How big will the file be?

How many block pointers will be required for a text file with 100,000 ascii characters with a block size of 4096 bytes and a block pointer size of 64 bits?

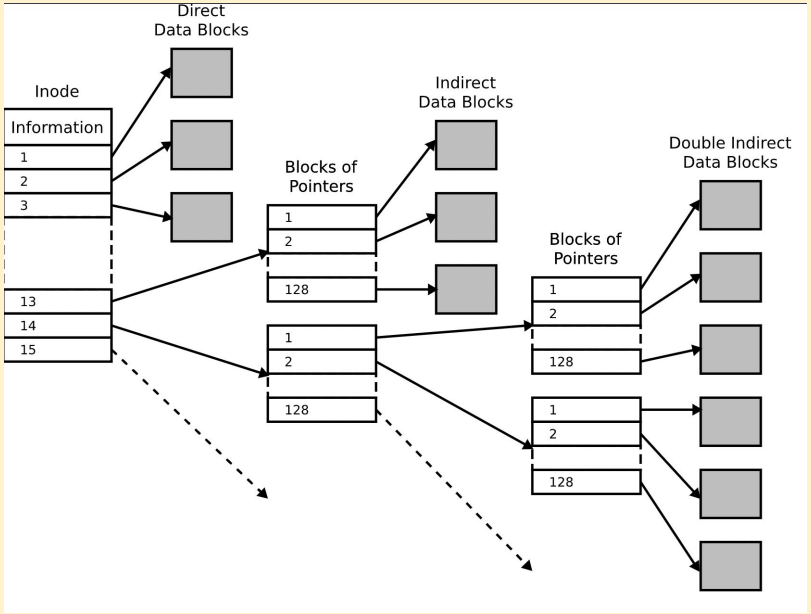


How big will the file be?

How many block pointers will be required for a text file with 100,000 ascii characters with a block size of 4096 bytes and a block pointer size of 64 bits?

26

(12 in the inode, 1 indirect, 13 in the block of pointers)



File-related Commands

```
stat

df

ls -i
```

Text File I/O in C

- Can read and write text to and from files
- Similar to reading/writing to stdin/stdout
- stdio/stderr are basically just “files” that have already been opened for you

```c
#include <stdio.h>
#include <errno.h>

int main() {
  FILE* test_file;
  test_file = fopen("file.txt", "w");
  if (test_file == NULL) {
    fprintf(stderr, "Opening file failed with code %d.\n", errno);
    return 1;
  }
  fprintf(test_file, "Number: %d\n", 25);
  fflush(test_file);
  fclose(test_file);
  return 0;
}
```

```c
#include <stdio.h>
#include <errno.h>

int main() {
  FILE* test_file;
  test_file = fopen("file.txt", "w");
  if (test_file == NULL) {
    fprintf(stderr, "Opening file failed with code %d.\n", errno);
    return 1;
  }
  fprintf(test_file, "Number: %d\n", 25);
  fflush(test_file);
  fclose(test_file);
  return 0;
}
```

What is a FILE* ?

Many different possible modes (see man pages)

Same function, different locations to send the output to

See man pages for fopen, fprintf, fflush, fclose, fscanf

What is a FILE?

A structure containing the necessary information to manage that particular file

See the standard!

http://port70.net/~nsz/c/c11/n1570.html

What is a FILE?

Investigate on lectura. You can use:

```
$ locate stdio.h

$ echo '#include <stdio.h>' | cpp -H -o /dev/null 2>&1 | head -n1
```

Can you figure out what a FILE actually is?

What is a FILE?

/usr/include/stdio.h

/usr/include/x86_64-linux-gnu/bits/types/struct_FILE.h

Implement Sum

Write a C program that

1. Prompts the user for a file name
2. Opens this file
3. Reads through each line of file, assuming each line will have exactly 1 integer number
4. Sum the numbers, save the result to sum.txt