Project Outcome

Project outcome explained easily

Here’s the project in one sentence:

You’re building two tiny tools that make and modify a **binary disk image** of a mini filesystem—first create a blank but valid FS image (mkfs\_builder), then add a real file into its root directory and save a new image (mkfs\_adder). The “output” is those **.img files** being byte-correct (superblock, bitmaps, inodes, directory entries, checksums) and showing the added file.

## **What “done” looks like (outcomes)**

* **Program outputs:**
  + fs.img — fresh MiniVSFS image with just the root directory (/) containing . and ...
  + fs2.img — the same image **plus** your chosen file placed into /.
* **How it’s validated:** with simple byte checks (xxd/hexdump) or a tiny dumper:  
  + Superblock magic is MVSF.
  + Bitmaps flip bits when you allocate an inode or data block.
  + Inode #1 (root) has correct links/size and a data block.
  + A new inode appears for your file, and a directory entry shows its name.

## **Your specific output (what you showed = correct)**

* fs.img vs fs2.img are both 4.0 MiB — expected for a 1024-block image.
* **Magic:** 46 53 56 4d at block 0 → “MVSF” ✅
* **Inode bitmap (block 1):** 03 → inode #1 (root) **and** inode #2 (your file) allocated ✅
* **Data bitmap (block 2):** 03 → 1 block for root directory **+** 1 block for file\_13.txt ✅
* minivsfs\_ls confirms:  
  + fs.img: root links=2, size=128 bytes, entries . and .. only.
  + fs2.img: root links=3, size=192 bytes, entries . .. **and** file\_13.txt (type=1, ino=2). ✅

That’s exactly the expected outcome: your builder made a valid image, and your adder created a new inode, wrote the file’s data, added a dir entry in /, bumped root’s link count, and saved a new image.

## **What to hand in (typical)**

* Your two C sources (mkfs\_builder.c, mkfs\_adder.c).
* A tiny README with compile/run examples and a couple of the hexdump outputs you already captured.
* (Optionally) the sample fs.img / fs2.img you produced.

Short version: your images look good; your pipeline did what the assignment asks.

README.md

# **MiniVSFS — Quick Start**

Two C programs:

* **mkfs\_builder**: creates a valid MiniVSFS disk image (.img) with just the root directory (/).
* **mkfs\_adder**: takes an existing MiniVSFS image and a real host file, adds that file into /, and writes a **new** image.

## **Files in this folder**

* mkfs\_builder.c — builder tool
* mkfs\_adder.c — adder tool
* *(optional for debugging)* minivsfs\_ls.c — tiny read-only lister to print /

## **Requirements (Linux)**

* gcc (C17 OK)
* xxd, hexdump, dd (usually installed)

## **Build**

gcc -O2 -std=c17 -Wall -Wextra mkfs\_builder.c -o mkfs\_builder

gcc -O2 -std=c17 -Wall -Wextra mkfs\_adder.c -o mkfs\_adder

# optional helper

# gcc -O2 -std=c17 -Wall -Wextra minivsfs\_ls.c -o minivsfs\_ls

**Optional Makefile**

CC=gcc

CFLAGS=-O2 -std=c17 -Wall -Wextra

all: mkfs\_builder mkfs\_adder

mkfs\_builder: mkfs\_builder.c

$(CC) $(CFLAGS) $< -o $@

mkfs\_adder: mkfs\_adder.c

$(CC) $(CFLAGS) $< -o $@

clean:

rm -f mkfs\_builder mkfs\_adder

## **Usage**

### **1) Create a filesystem image**

./mkfs\_builder --image fs.img --size-kib 4096 --inodes 256

* --size-kib: 180..4096 (multiple of 4)
* --inodes: 128..512

This writes fs.img with:

* Block 0: superblock (magic MVSF), checksum set
* Block 1: inode bitmap
* Block 2: data bitmap
* Blocks 3..N: inode table
* Remaining blocks: data region
* Root inode (#1) with . and .. in its first data block

### **2) Add a real file to / and produce a new image**

# the file must already exist on your host filesystem

./mkfs\_adder --input fs.img --output fs2.img --file file\_13.txt

* First-fit allocation for a free **inode** and **data blocks**
* If root’s first block is full, it **extends** root with another block
* Max file size: **49,152 bytes** (12 direct pointers × 4096)

## **Typical workflow (copy-paste)**

# 1) Build tools

gcc -O2 -std=c17 -Wall -Wextra mkfs\_builder.c -o mkfs\_builder

gcc -O2 -std=c17 -Wall -Wextra mkfs\_adder.c -o mkfs\_adder

# 2) Make a 4 MiB image with 256 inodes

./mkfs\_builder --image fs.img --size-kib 4096 --inodes 256

# 3) Add an existing text file

echo "hello MiniVSFS" > file\_13.txt

./mkfs\_adder --input fs.img --output fs2.img --file file\_13.txt

## **Quick validations**

### **Sizes**

ls -lh fs.img fs2.img

# both should be same size (e.g., 4.0M if size-kib=4096)

### **Superblock magic (block 0)**

xxd -g 1 -l 4 -s 0 fs2.img

# Expect: 46 53 56 4d → “MVSF”

### **Bitmaps (block 1 = inode, block 2 = data)**

xxd -g 1 -l 32 -s $((4096\*1)) fs2.img # inode bitmap

xxd -g 1 -l 32 -s $((4096\*2)) fs2.img # data bitmap

# After adding one file, both often start with 03 (root + one file)

### **(Optional) List / entries**

# if you compiled the helper

./minivsfs\_ls fs.img

./minivsfs\_ls fs2.img

# fs.img: shows '.' and '..'

# fs2.img: shows '.', '..', and your file name

## **Constraints & details (spec highlights)**

* Block size = **4096 B**; inode size = **128 B**
* **12 direct** pointers per inode (no indirects)
* Only the **root directory** is supported
* Inodes are **1-indexed** (root is inode **1**)
* Checksums:  
  + Superblock: CRC32 stored in its last 4 bytes
  + Inode: CRC32 stored in last 8 bytes (low 4 bytes carry the CRC)
  + Dirent (64B): XOR of bytes 0..62

## **Troubleshooting**

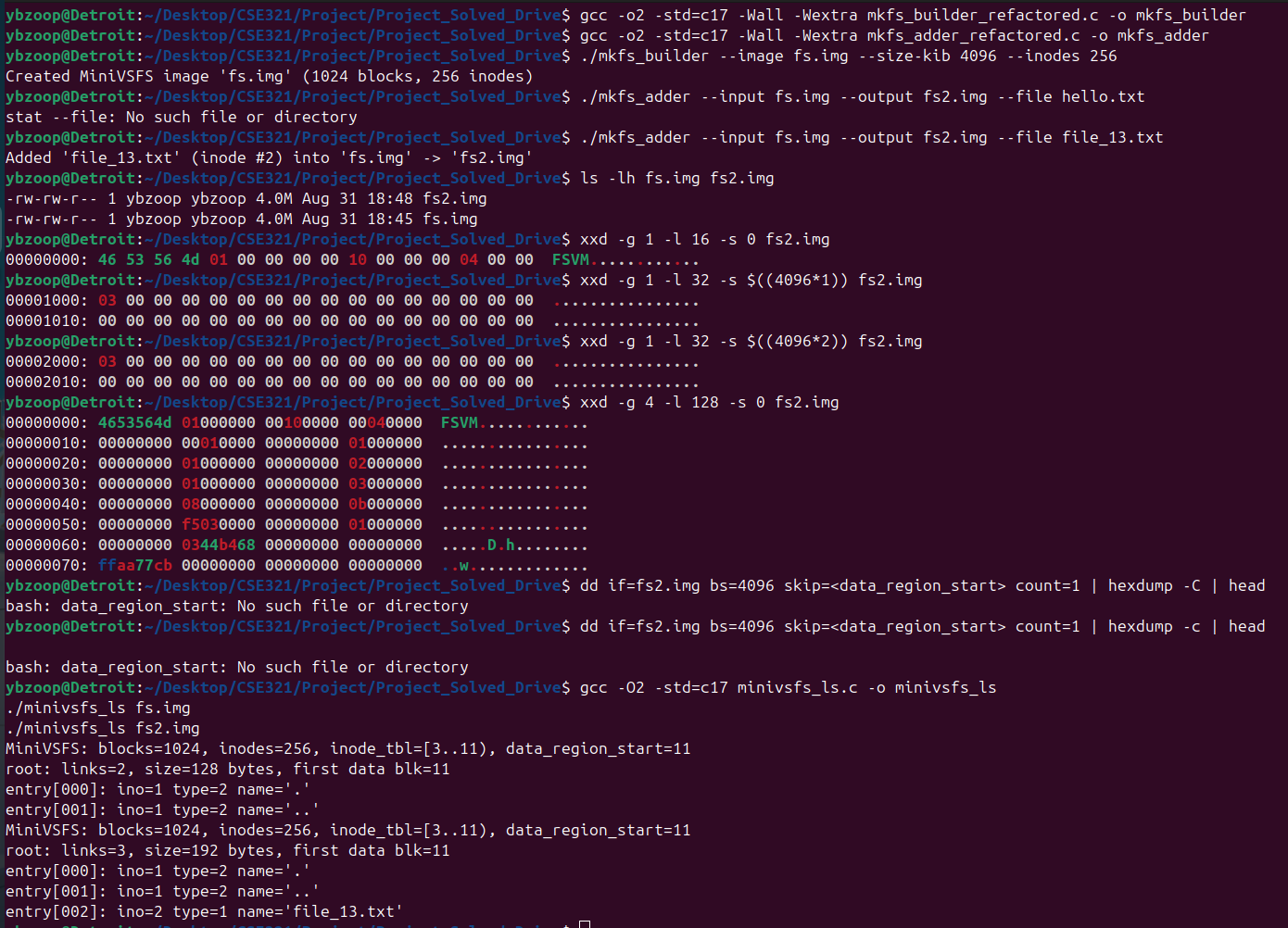
* **“Not a MiniVSFS image”** → Rebuild with this mkfs\_builder; don’t mix with old formats.
* **“File too large for 12 direct blocks”** → Keep files ≤ 49,152 bytes.
* **Dir entry missing** → Ensure the host file exists; check you used a new --output.
* **No free blocks** → Create a larger image (--size-kib) and try again.

That’s it—compile, create fs.img, add files with mkfs\_adder, and inspect with the quick checks above.

GPT Link

Link: <https://chatgpt.com/share/68b453c1-c034-8011-95e7-c58cda83b18f>

Terminal SS



Terminal Text

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ gcc -o2 -std=c17 -Wall -Wextra mkfs\_builder\_refactored.c -o mkfs\_builder

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ gcc -o2 -std=c17 -Wall -Wextra mkfs\_adder\_refactored.c -o mkfs\_adder

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ ./mkfs\_builder --image fs.img --size-kib 4096 --inodes 256

Created MiniVSFS image 'fs.img' (1024 blocks, 256 inodes)

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ ./mkfs\_adder --input fs.img --output fs2.img --file hello.txt

stat --file: No such file or directory

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ ./mkfs\_adder --input fs.img --output fs2.img --file file\_13.txt

Added 'file\_13.txt' (inode #2) into 'fs.img' -> 'fs2.img'

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ ls -lh fs.img fs2.img

-rw-rw-r-- 1 ybzoop ybzoop 4.0M Aug 31 18:48 fs2.img

-rw-rw-r-- 1 ybzoop ybzoop 4.0M Aug 31 18:45 fs.img

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ xxd -g 1 -l 16 -s 0 fs2.img

00000000: 46 53 56 4d 01 00 00 00 00 10 00 00 00 04 00 00 FSVM............

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ xxd -g 1 -l 32 -s $((4096\*1)) fs2.img

00001000: 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

00001010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ xxd -g 1 -l 32 -s $((4096\*2)) fs2.img

00002000: 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

00002010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ xxd -g 4 -l 128 -s 0 fs2.img

00000000: 4653564d 01000000 00100000 00040000 FSVM............

00000010: 00000000 00010000 00000000 01000000 ................

00000020: 00000000 01000000 00000000 02000000 ................

00000030: 00000000 01000000 00000000 03000000 ................

00000040: 00000000 08000000 00000000 0b000000 ................

00000050: 00000000 f5030000 00000000 01000000 ................

00000060: 00000000 0344b468 00000000 00000000 .....D.h........

00000070: ffaa77cb 00000000 00000000 00000000 ..w.............

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ dd if=fs2.img bs=4096 skip=<data\_region\_start> count=1 | hexdump -C | head

bash: data\_region\_start: No such file or directory

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ dd if=fs2.img bs=4096 skip=<data\_region\_start> count=1 | hexdump -c | head

bash: data\_region\_start: No such file or directory

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$ gcc -O2 -std=c17 minivsfs\_ls.c -o minivsfs\_ls

./minivsfs\_ls fs.img

./minivsfs\_ls fs2.img

MiniVSFS: blocks=1024, inodes=256, inode\_tbl=[3..11), data\_region\_start=11

root: links=2, size=128 bytes, first data blk=11

entry[000]: ino=1 type=2 name='.'

entry[001]: ino=1 type=2 name='..'

MiniVSFS: blocks=1024, inodes=256, inode\_tbl=[3..11), data\_region\_start=11

root: links=3, size=192 bytes, first data blk=11

entry[000]: ino=1 type=2 name='.'

entry[001]: ino=1 type=2 name='..'

entry[002]: ino=2 type=1 name='file\_13.txt'

ybzoop@Detroit:~/Desktop/CSE321/Project/Project\_Solved\_Drive$