Homework #6: File System (100 points)

Submit a compressed folder containing **FS.h**, **FS.cpp**, & **hw6.ino** to Canvas

For this assignment, you must implement a *simple* **file system** using the Arduino Uno and EEPROM chip (Microchip 24LC256). Specifically, your program must meet the following requirements:

- Your *file system* (FS class) must have the following capabilities:
  - **reformat** the EEPROM chip (clear free-space list and file directory)
  - **initialize** file system (bring free-space list and file directory into memory)
  - **create** a *named* file
  - **open** a *named* file
  - **write** bytes to a file
  - **read** bytes from a file
  - **seek** to *beginning* of a file (only)
  - **close** a file
  - **delete** a *named* file
  - **list** all files in file system (name and page-aligned size)

- Your file system must implement its *free-space list* as a **bit vector**
  - 1 bit for each 64-byte *block* of EEPROM memory
    - EEPROM chip as *512* × 64-byte blocks of memory
      - 1 block of EEPROM should hold *entire bit vector*
      - => 64 bytes / block × 8 bits / byte = 512 bits / block

- Your file system must implement its *directory structure* as a **linear list**
  - no more than **32** files in directory / file system
  - *File directory* should consume 1 block of EEPROM (e.g., block 1)
    - 32 pointers to FCB blocks × 2 bytes / pointer = 64 bytes = 1 block

- Your file system must implement **indexed allocation**
  - simplified *File Control Block* (FCB) structure, e.g.,
    - file name
    - current file offset
    - **16** pointers to data blocks (initialized to 0 or NULL)
      - files can contain no more than **1024 bytes** of data
  - file data blocks should only be allocated when **needed**
    - when file write operation *exceeds current block*, requires *new block*
    - no pre-allocation…

- You must implement *error-checking* in your file system, **e.g.,**
  - creating a file that already exists
  - **etc.**

- You must submit an Arduino sketch (**hw6.ino**) that *validates* your file system

<u>HINTS:</u>

- This is a **non-trivial** assignment… **start early!!!**
    - Work incrementally
    - Test often

- Understand the file system design **<u>before</u>** you write *any* code
    - Read chapter 11 in book
    - Draw diagrams
    - Write pseudocode
    - Write non-Arduino test code in C++

- EEPROM chip contains 32,768 bytes of memory
    - 512 × 64 byte blocks
    - All EEPROM chip I/O should be in 64-byte *blocks* (not single bytes)

- *Bit vector* should consume 1 block of EEPROM (e.g., block 0)
    - 1 := data block empty
    - 0 := data block occupied

- Bitwise operators to manipulate *bit vector*, e.g.,
    - (0x02 >> 1) & 0x01 = ??
    - 0xFF & ~(0x80) = ??
    - 0x00 | 0x40 = ??

- Each FCB should consume 1 block of EEPROM (64 bytes)
    - Simplifies writing FCB to EEPROM
    - Simplifies reading FCB from EEPROM
    - **avoid malloc()**

- E.g., steps to create a file:
    1) scan directory structure, check if file exists (error)
    2) find empty slot in directory structure
    3) find empty data block in free-space list for FCB
    4) fill in default FCB values
    5) write everything to EEPROM

- Maintain copies in main memory
    - free-space bit vector (64 bytes)
    - directory structure (64 bytes)
    - current file data (64 bytes)

- Minimize EEPROM writes
    - time consuming (up to 5 ms per page!)
    - limited # of writes (1,000,000 per page)

<u>EXAMPLES:</u> Serial output shown.

// list files, format, list files:
```
listing files...
File: test1.txt, 0 bytes
File: test2.txt, 0 bytes
File: test3.txt, 0 bytes
formatting EEPROM...
listing files...
```

// create 32 files (test00.txt, test01.txt, ..., test31.txt) and list files
```
File: test_00.txt, 0 bytes
File: test_01.txt, 0 bytes
File: test_02.txt, 0 bytes
File: test_03.txt, 0 bytes
File: test_04.txt, 0 bytes
File: test_05.txt, 0 bytes
File: test_06.txt, 0 bytes
File: test_07.txt, 0 bytes
File: test_08.txt, 0 bytes
File: test_09.txt, 0 bytes
File: test_10.txt, 0 bytes
File: test_11.txt, 0 bytes
File: test_12.txt, 0 bytes
File: test_13.txt, 0 bytes
File: test_14.txt, 0 bytes
File: test_15.txt, 0 bytes
File: test_16.txt, 0 bytes
File: test_17.txt, 0 bytes
File: test_18.txt, 0 bytes
File: test_19.txt, 0 bytes
File: test_20.txt, 0 bytes
File: test_21.txt, 0 bytes
File: test_22.txt, 0 bytes
File: test_23.txt, 0 bytes
File: test_24.txt, 0 bytes
File: test_25.txt, 0 bytes
File: test_26.txt, 0 bytes
File: test_27.txt, 0 bytes
File: test_28.txt, 0 bytes
File: test_29.txt, 0 bytes
File: test_30.txt, 0 bytes
File: test_31.txt, 0 bytes
```

// try to create 33rd file:
```
create_file ERROR: no space in FCB directory
```

// open files (i.e., test_00.txt -> test_31.txt), write *custom* message, close files, *list files*

```
File: test_00.txt, 64 bytes
File: test_01.txt, 64 bytes
File: test_02.txt, 64 bytes
File: test_03.txt, 64 bytes
File: test_04.txt, 64 bytes
File: test_05.txt, 64 bytes
File: test_06.txt, 64 bytes
File: test_07.txt, 64 bytes
File: test_08.txt, 64 bytes
File: test_09.txt, 64 bytes
File: test_10.txt, 64 bytes
File: test_11.txt, 64 bytes
File: test_12.txt, 64 bytes
File: test_13.txt, 64 bytes
File: test_14.txt, 64 bytes
File: test_15.txt, 64 bytes
File: test_16.txt, 64 bytes
File: test_17.txt, 64 bytes
File: test_18.txt, 64 bytes
File: test_19.txt, 64 bytes
File: test_20.txt, 64 bytes
File: test_21.txt, 64 bytes
File: test_22.txt, 64 bytes
File: test_23.txt, 64 bytes
File: test_24.txt, 64 bytes
File: test_25.txt, 64 bytes
File: test_26.txt, 64 bytes
File: test_27.txt, 64 bytes
File: test_28.txt, 64 bytes
File: test_29.txt, 64 bytes
File: test_30.txt, 64 bytes
File: test_31.txt, 64 bytes
```

// open each file, *lseek to position 0*, read message, print message, close file

```
Contents of file: test_00.txt =
      Hello CS 444, this is file 00!!!

Contents of file: test_01.txt =
      Hello CS 444, this is file 01!!!

(not all output shown)

Contents of file: test_31.txt =
      Hello CS 444, this is file 31!!!
```

// open each file, write *another* short message, close file, list files

```
File: test_00.txt, 128 bytes
File: test_01.txt, 128 bytes

(not all output shown)

File: test_31.txt, 128 bytes
```

// open each file, *lseek to position 0*, read full message, print message, close file

```
Contents of file: test_00.txt =
      Hello CS 444, this is file 00!!!
      ... another message in file 00!!!

Contents of file: test_01.txt =
      Hello CS 444, this is file 01!!!
      ... another message in file 01!!!

(not all output shown)

Contents of file: test_31.txt =
      Hello CS 444, this is file 31!!!
      ... another message in file 31!!!
```

//delete files test_10.txt –> test_31.txt, list files

```
File: test_00.txt, 128 bytes
File: test_01.txt, 128 bytes
File: test_02.txt, 128 bytes

(not all output shown)

File: test_09.txt, 128 bytes
```

//open file test_06.txt, lseek to 0, write 1024 random letters, close file, list files
//   (write 256 bytes at a time)

```
writing chars:
HOGOKPAIYBVUXWIBPHEWXACVNHNNAGDIMMIKYDPPPQNIOLLBXHHJHGBYMRALGOTFVOVFMOVGQWJXO
KRBWYCCYTXJBEMXIQAKIFGFKCAFFYEJTFELKAKEHSQQBWQPWTAJQHUVVQFJCNNGBWRUDGXBVLRAIG
IVRUDLRRLKBNUSYOFGSQSJNMQUCEUUOCFEROHYOSNKDFQGPDTHIKGYPDSTQDNLPXYHWDSFUPXRCFT
GFNVFHAFPUYKEXLWUHHKMHJUY to file test_06.txt
RNUTHUGWMLJAVMRLSHQUHBCMDYBGMDRJMSSNSEKCBTVDYHTALWJWOUFSOONNAIUOFGSEDDGQCPFOT
YYMANHJMYWVJSTYVHAIGWLKMJLYVFHJGGBTDXEEQOVTWKHFPWXFBCAMJGGVKGSMWATHQJLQRMMGNM
JKQEFDIRAEGYQKQKVDNIGWRMPFBYVXHEDAHVWLAPFMJEPPVKQPNGNEPLDKMGJOJQSJTXRSDCYQIOA
OTNDGBCUXWDNQGYPBSOQDYDYY to file test_06.txt
TASUFODDOYWWHUUGPNCQYGFVXDNKCVMWARUVXDGBFIRTJGJPAEKYVKLMJHLUDJJVLUPGUXPJRDKUB
YELOKTUQURUTNWTTUXIRNLTMYBOOVDVCESEWYWCGDALLCTLYLRTHCTRSODMRHLKDPTWHFKRKRJJMT
KTWMJEGCVFNYSCYDHPEKIROAGBLYRQRYLACGGQPLTKUORSCXFDJRHLFVUGQRUMMLUSQTHUYDCBJRK
DNCBWXRVLTTPVJFXLRCNRXBNR to file test_06.txt
HHKOHDNPAMECIUSKOCSSAHMXSGQODNYCWHTFGJPKNIDHABGXUNORHJJBVXYJSSWYYYRRXMEUPJHDU
QOPFSUHHYKTBCTMRATAWCXAXALPSTNBFJHDAPIFOWLWHHQQLIEFVFWPKWCFUNCANLYRLWQXDRYUPB
JMBQVHOIOLUSCIMEYSHHHMTLYMHJEXTVMSYCAHHMPLBLXHMPQJNVHUQUVHDUKDNPLMFDFFAMSGRSC
KAFQUJBHBWNXDWECEWXNUBEOH to file test_06.txt

File: test_00.txt, 128 bytes
File: test_01.txt, 128 bytes
File: test_02.txt, 128 bytes
File: test_03.txt, 128 bytes
File: test_04.txt, 128 bytes
File: test_05.txt, 128 bytes
File: test_06.txt, 1024 bytes
File: test_07.txt, 128 bytes
File: test_08.txt, 128 bytes
File: test_09.txt, 128 bytes
```

//open file test_06.txt, lseek to position 0, read 1024 bytes, print, close file
// (read / print 256 bytes at a time)
// NOTE: characters match above

```
reading chars:
HOGOKPAIYBVUXWIBPHEWXACVNHNNAGDIMMIKYDPPPQNIOLLBXHHJHGBYMRALGOTF6OVFMOVGQWJXO
KRBWYCCYTXJBEMXIQAKIFGFKCAFFYEJTFELKAKEHSQQBWQPWTAJQ6UVVQFJCNNGBWRUDGXBVLRAIG
IVRUDLRRLKBNUSYOFGSQSJNMQUCEUUOCFEROHYOS6KDFQGPDTHIKGYPDSTQDNLPXYHWDSFUPXRCFT
GFNVFHAFPUYKEXLWUHHKMHJUY from file test_06.txt
RNU6HUGWMLJAVMRLSHQUHBCMDYBGMDRJMSSNSEKCBTVDYHTALWJWOUFSOONNAIUOFGSE6DGQCPFOT
YYMANHJMYWVJSTYVHAIGWLKMJLYVFHJGGBTDXEEQOVTWKHFPWXFBCAMJ6GVKGSMWATHQJLQRMMGNM
JKQEFDIRAEGYQKQKVDNIGWRMPFBYVXHEDAHVWLAPFMJE6PVKQPNGNEPLDKMGJOJQSJTXRSDCYQIOA
OTNDGBCUXWDNQGYPBSOQDYDYY from file test_06.txt
TASUFOD6OYWWHUUGPNCQYGFVXDNKCVMWARUVXDGBFIRTJGJPAEKYVKLMJHLUDJJVLUPGUXPJ6DKUB
YELOKTUQURUTNWTTUXIRNLTMYBOOVDVCESEWYWCGDALLCTLYLRTHCTRSODMR6LKDPTWHFKRKRJJMT
KTWMJEGCVFNYSCYDHPEKIROAGBLYRQRYLACGGQPLTKUORSCX6DJRHLFVUGQRUMMLUSQTHUYDCBJRK
DNCBWXRVLTTPVJFXLRCNRXBNR from file test_06.txt
HHKOHDNPAME6IUSKOCSSAHMXSGQODNYCWHTFGJPKNIDHABGXUNORHJJBVXYJSSWYYYRRXMEUPJHD6
QOPFSUHHYKTBCTMRATAWCXAXALPSTNBFJHDAPIFOWLWHHQQLIEFVFWPKWCFUNCAN6YRLWQXDRYUPB
JMBQVHOIOLUSCIMEYSHHHMTLYMHJEXTVMSYCAHHMPLBLXHMPQJNV6UQUVHDUKDNPLMFDFFAMSGRSC
KAFQUJBHBWNXDWECEWXNUBEOH from file test_06.txt
```

//open file test_03.txt, lseek to 0, write 1024 random characters, close file, list files
// (write 1 character at a time… this is SLOW!!)

```
writing :
HOGOKPAIYBVUXWIBPHEWXACVNHNNAGDIMMIKYDPPPQNIOLLBXHHJHGBYMRALGOTFVOVFMOVGQWJXO
KRBWYCCYTXJBEMXIQAKIFGFKCAFFYEJTFELKAKEHSQQBWQPWTAJQHUVVQFJCNNGBWRUDGXBVLRAIG
IVRUDLRRLKBNUSYOFGSQSJNMQUCEUUOCFEROHYOSNKDFQGPDTHIKGYPDSTQDNLPXYHWDSFUPXRCFT
GFNVFHAFPUYKEXLWUHHKMHJUYRNUTHUGWMLJAVMRLSHQUHBCMDYBGMDRJMSSNSEKCBTVDYHTALWJW
OUFSOONNAIUOFGSEDDGQCPFOTYYMANHJMYWVJSTYVHAIGWLKMJLYVFHJGGBTDXEEQOVTWKHFPWXFB
CAMJGGVKGSMWATHQJLQRMMGNMJKQEFDIRAEGYQKQKVDNIGWRMPFBYVXHEDAHVWLAPFMJEPPVKQPNG
NEPLDKMGJOJQSJTXRSDCYQIOAOTNDGBCUXWDNQGYPBSOQDYDYYTASUFODDOYWWHUUGPNCQYGFVXDN
KCVMWARUVXDGBFIRTJGJPAEKYVKLMJHLUDJJVLUPGUXPJRDKUBYELOKTUQURUTNWTTUXIRNLTMYBO
OVDVCESEWYWCGDALLCTLYLRTHCTRSODMRHLKDPTWHFKRKRJJMTKTWMJEGCVFNYSCYDHPEKIROAGBL
YRQRYLACGGQPLTKUORSCXFDJRHLFVUGQRUMMLUSQTHUYDCBJRKDNCBWXRVLTTPVJFXLRCNRXBNRHH
KOHDNPAMECIUSKOCSSAHMXSGQODNYCWHTFGJPKNIDHABGXUNORHJJBVXYJSSWYYYRRXMEUPJHDUQO
PFSUHHYKTBCTMRATAWCXAXALPSTNBFJHDAPIFOWLWHHQQLIEFVFWPKWCFUNCANLYRLWQXDRYUPBJM
BQVHOIOLUSCIMEYSHHHMTLYMHJEXTVMSYCAHHMPLBLXHMPQJNVHUQUVHDUKDNPLMFDFFAMSGRSCKA
FQUJBHBWNXDWECEWXNUBEOH to test_03.txt

File: test_00.txt, 128 bytes
File: test_01.txt, 128 bytes
File: test_02.txt, 128 bytes
File: test_03.txt, 1024 bytes
File: test_04.txt, 128 bytes
File: test_05.txt, 128 bytes
File: test_06.txt, 1024 bytes
File: test_07.txt, 128 bytes
File: test_08.txt, 128 bytes
File: test_09.txt, 128 bytes
```

//open file test_03.txt, lseek to position 0, read 1024 characters, close file
// (read 1 character at a time)
// MATCHES!

```
reading :
HOGOKPAIYBVUXWIBPHEWXACVNHNNAGDIMMIKYDPPPQNIOLLBXHHJHGBYMRALGOTFVOVFMOVGQWJXO
KRBWYCCYTXJBEMXIQAKIFGFKCAFFYEJTFELKAKEHSQQBWQPWTAJQHUVVQFJCNNGBWRUDGXBVLRAIG
IVRUDLRRLKBNUSYOFGSQSJNMQUCEUUOCFEROHYOSNKDFQGPDTHIKGYPDSTQDNLPXYHWDSFUPXRCFT
GFNVFHAFPUYKEXLWUHHKMHJUYRNUTHUGWMLJAVMRLSHQUHBCMDYBGMDRJMSSNSEKCBTVDYHTALWJW
OUFSOONNAIUOFGSEDDGQCPFOTYYMANHJMYWVJSTYVHAIGWLKMJLYVFHJGGBTDXEEQOVTWKHFPWXFB
CAMJGGVKGSMWATHQJLQRMMGNMJKQEFDIRAEGYQKQKVDNIGWRMPFBYVXHEDAHVWLAPFMJEPPVKQPNG
NEPLDKMGJOJQSJTXRSDCYQIOAOTNDGBCUXWDNQGYPBSOQDYDYYTASUFODDOYWWHUUGPNCQYGFVXDN
KCVMWARUVXDGBFIRTJGJPAEKYVKLMJHLUDJJVLUPGUXPJRDKUBYELOKTUQURUTNWTTUXIRNLTMYBO
OVDVCESEWYWCGDALLCTLYLRTHCTRSODMRHLKDPTWHFKRKRJJMTKTWMJEGCVFNYSCYDHPEKIROAGBL
YRQRYLACGGQPLTKUORSCXFDJRHLFVUGQRUMMLUSQTHUYDCBJRKDNCBWXRVLTTPVJFXLRCNRXBNRHH
KOHDNPAMECIUSKOCSSAHMXSGQODNYCWHTFGJPKNIDHABGXUNORHJJBVXYJSSWYYYRRXMEUPJHDUQO
PFSUHHYKTBCTMRATAWCXAXALPSTNBFJHDAPIFOWLWHHQQLIEFVFWPKWCFUNCANLYRLWQXDRYUPBJM
BQVHOIOLUSCIMEYSHHHMTLYMHJEXTVMSYCAHHMPLBLXHMPQJNVHUQUVHDUKDNPLMFDFFAMSGRSCKA
FQUJBHBWNXDWECEWXNUBEOH from test_03.txt
```

//power cycle Arduino (unplug / plug back in), list files

```
File: test_00.txt, 128 bytes
File: test_01.txt, 128 bytes
File: test_02.txt, 128 bytes
File: test_03.txt, 1024 bytes
File: test_04.txt, 128 bytes
File: test_05.txt, 128 bytes
File: test_06.txt, 1024 bytes
File: test_07.txt, 128 bytes
File: test_08.txt, 128 bytes
File: test_09.txt, 128 bytes
```

// There are countless examples. E.g.,
// - creating / deleting files
// - reading / writing files
// - error checking:
// -> file exceeds 1024 bytes
// -> no empty blocks in EEPROM
// -> etc.