MP7 : Simple File System (Vasudha Devarakonda)

Changes :

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file.C
        modified:   file.H
        modified:   file_system.C
        modified:   file_system.H
        modified:   kernel.C

Untracked files:
```

File System

Maintain static linked list for inode, super block, maintained in disk block 0 and free block to maintain for empty data blocks available in the disk block 1

```
Inode *FileSystem::head_pointer = nullptr;
Inode *FileSystem::tail_pointer = nullptr;
unsigned char *FileSystem::super_block = new unsigned char[SimpleDisk::BLOCK_SIZE];
unsigned char *FileSystem::free_data_block = new unsigned char[SimpleDisk::BLOCK_SIZE];
```

FileSystem -> constructor

Initialize the linked list to null which will be filled eventually while creating the file

Deconstructor : ~FileSystem

Disassociate the disk by making it null for the file system

Mount

Associate the passed disk to file system’s disk

```
/*----------------------------------------------------------------------

bool FileSystem::Mount(SimpleDisk *_disk)
{
    Console::puts("mounting file system\n");
    disk = _disk;
    return true;
}
```

Format :

Intialse super block to disk block 0 and free block to disk block 1. These 2 are bitmaps. Make first 2 MSBs 1 as they are occupied by free block and super block

```
bool FileSystem::Format(SimpleDisk *_disk, unsigned int _size)
{ // static!
    Console::puts("formatting disk\n");
    unsigned int numBlocks = _size / SimpleDisk::BLOCK_SIZE;
    for (int i = 0; i < SimpleDisk::BLOCK_SIZE; i++)
    {
        free_data_block[i] = 0x0;
    }
    _disk->write(0, super_block);
    free_data_block[0 / 8] = free_data_block[0 / 8] | shift_after_write(0); // 1000000000000 -> free block bitmap
    free_data_block[1 / 8] = free_data_block[1 / 8] | shift_after_write(1); // 1100000000000 -> free block bitmap updated
    _disk->write(1, free_data_block);                                       // 2nd block is reserved by bitmap for free data reg
    // please note that free block for inode is not maintained because we are mainiting inode as a linked list and not an array
    return true;
}
```

Lookup

Check if the file_id exists in the file system by iterating through inode's linked lits which keeps track of the file id

```
Inode *FileSystem::LookupFile(int _file_id)
{
    Console::puts("looking up file with id = ");
    Console::puti(_file_id);
    Console::puts("\n");
    if (head_pointer == NULL)
        return NULL;
    Inode *current = head_pointer;
    while (current != NULL)
    {
        if (current->id == _file_id)
            return current;
        current = current->next;
    }
    return NULL;
}
```

CreateFile

Create Inode and add to linked list. A block number is assigned to each file for its read and write operations.

BONUS : In order to accommodate files >512 B or more than BLOCK_SIZE and maximum 64KB, 128 contiguous blocks are assigned to each file as shown in the screenshot below. However, since the block will get exhausted eventually it is checked and if the block numbers are exhausted the allocation starts from 2 again i.e overwrite

```
bool FileSystem::CreateFile(int _file_id)
{
    Console::puts("creating file with id:");
    Console::puti(_file_id);
    Console::puts("\n");

    if (this->LookupFile(_file_id))
        return false;

    Inode *new_inode = new Inode();
    if (head_pointer == NULL)
    {

        head_pointer = tail_pointer = new_inode;
        head_pointer->id = _file_id;
        head_pointer->next = NULL;
    }
    else
    {
        tail_pointer->next = new_inode;
        tail_pointer = tail_pointer->next;
        tail_pointer->id = _file_id;
    }
    tail_pointer->block_no = current_block;
    current_block = current_block + 128;
    // Console::puts("Printing current block");
    // Console::puti(current_block);
    // Console::puts("\n");
```

Bonus Screenshot :

```
    }
    tail_pointer->block_no = current_block;
    current_block = current_block + 128;
    // Console::puts("Printing current block");
    // Console::puti(current_block);
    // Console::puts("\n");
    if (current_block > (10 * 1024 * 1024) / SimpleDisk::BLOCK_SIZE)
    {
        current_block = 2;
    }
    return true;
}
```

Delete File

Remove the corresponding inode from linked list. Data block need not be deleted because file fetches its allocated block number from linked list only if inode exists

```
bool FileSystem::DeleteFile(int _file_id)
{
    Console::puts("Deleting file with ID:");
    Console::puti(_file_id);
    Console::puts("\n");
    Inode *fileToDelete = this->LookupFile(_file_id);

    if (fileToDelete == NULL)
    {
        Console::puts("File does not exist.\n");
        return false;
    }
    else
    {
        Console::puts("Existing file found.\n");
                Inode *current
        Inode *current = head_pointer;
        Inode *prev = NULL;

        // Locate the file by its ID in the inode list
        while (current != NULL && current->id != _file_id)
        {
            prev = current;
            current = current->next;
        }
```

File. C and File.H

Constructor -> Open File

Fetch the start of the assigned block from the corresponding inode. Set position of read and write as 0 add the file to corresponding inode

```
/*--------------------------------------------------------------------------*/

File::File(FileSystem *_fs, int _id)
{
    Console::puts("Opening file.\n");
    file_id = _id;
    size = 0;
    position = 0;
    inode = _fs->LookupFile(file_id);
    my_block_number = inode->block_no;
    Console::puts("My block number as assigned by file system");
    Console::puti(my_block_number);
    Console::putch('\n');
    inode->file = this;
    memset(block_cache, 0, SimpleDisk::BLOCK_SIZE);
}
```

Deconstructor -> Close File
Set position to 0

```
File::~File()
{
    Console::puts("Closing file.\n");
    // FILE_SYSTEM->disk->write(current_block, block_cache);
    position = 0;
```

Read

From size of read characters, get the number of blocks to read. Set position =0 for each block read and iterate through each block

```
int File::Read(unsigned int _n, char *_buf)
{

    int my_blocks = (_n + SimpleDisk::BLOCK_SIZE - 1) / SimpleDisk::BLOCK_SIZE;
    int read_size = 0;
    int i = my_blocks;
    int start_block = my_block_number;
    while (i >0)
    {
        if (start_block <= 1)
        {
            Console::puts("File not initialized\n");
            assert(false)
        }
        memset(buf, 0, SimpleDisk::BLOCK_SIZE);
        FILE_SYSTEM->disk->read(start_block, buf);
        position = 0;
        while (!EoF() && _n >0)
        {
            _buf[read_size++] = (char)buf[position++];
            _n--;
        }
        start_block ++;
        i--;
    }
    return read_size;
}
```

Write

Write iteratively through blocks assigned. After writing to corresponding blocks, update free block bitmap

file.C > File(FileSystem *, int)

SYSTEM_DISK_SIZE Aa ab .* No results

```
93  int File::Write(unsigned int _n, const char *_buf)
94  {
95      int block = 0;
96      int number_blocks = (_n + SimpleDisk::BLOCK_SIZE - 1) / SimpleDisk::BLOCK_SIZE;
97      Console::puts("\n");
98      int i = my_block_number;
99      while (number_blocks > 0)
00      {
01          if (i <= 1)
02          {
03              Console::puts("File not initialized\n");
04              assert(false)
05          }
06          position = 0;
07          memset(buf, 0, SimpleDisk::BLOCK_SIZE);
08          while (position < SimpleDisk::BLOCK_SIZE & _n > 0)
09          {
10              buf[position++] = _buf[block++];
11              if (position == SimpleDisk::BLOCK_SIZE || _n <=0)
12                      break;
13              _n--;
14          }
15          FILE_SYSTEM->disk->write(i, buf);
16          FILE_SYSTEM->free_data_block[i / 8] = FILE_SYSTEM->free_data_block[i / 8] | shift_after_write(i);
17          FILE_SYSTEM->disk->write(1, FILE_SYSTEM->free_data_block);
18          i++;
19          number_blocks--;
20      }
21      Console::puts("Number OF BLOCKS Written");
22      Console::puti(block);
23      Console::puts("\n");
24      return block;
25  }
```

Reset

Set position to 0

```
void File::Reset()
{
    Console::puts("resetting file\n");
    position = 0;
}
```

EOF

Position should not be more than 512 Bytes ie per block

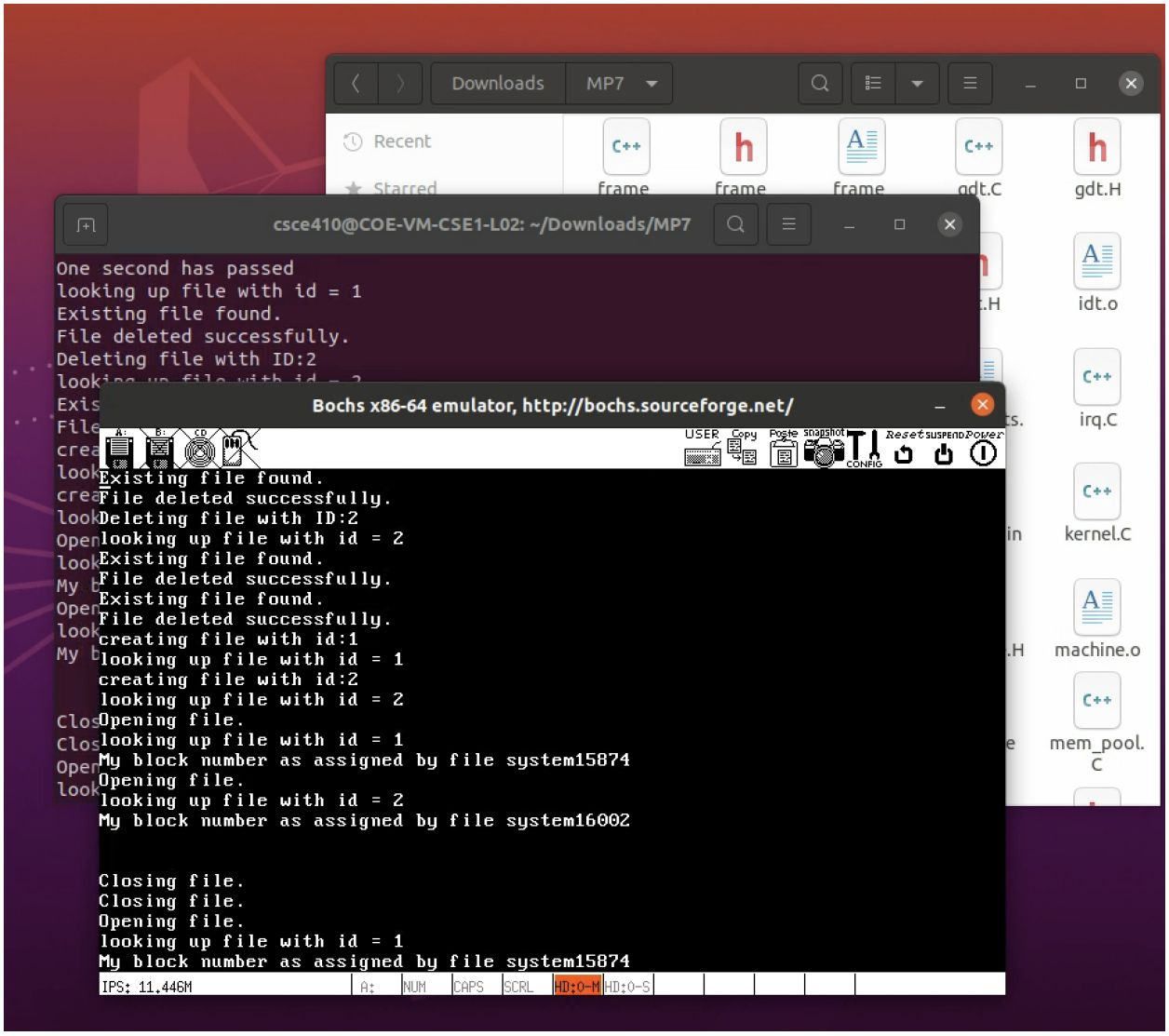
```
bool File::EoF()
{
    return position >= SimpleDisk::BLOCK_SIZE;
}
```

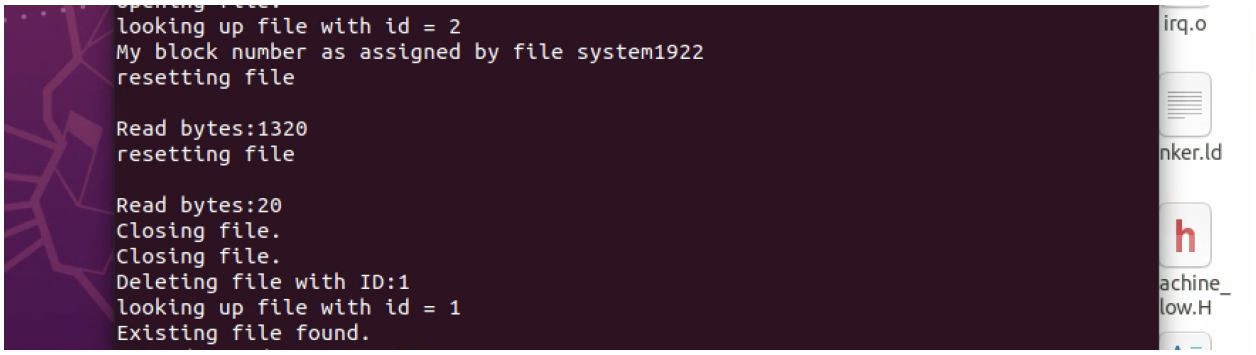
Kernel.C is updated to **accommodate bonus question** >512 Bytes of file .

File 1 has 1320 Bytes of data and File 2 has 20 Bytes of data

```
void exercise_file_system(FileSystem * _file_system) {

    const char * STRING1 = "012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
    const char * STRING2 = "abcdefghijabcdefghij";

    /* -- Create two files -- */

    assert(_file_system->CreateFile(1));
    assert(_file_system->CreateFile(2));

    /* -- "Open" the two files -- */

    {
        File file1(_file_system, 1);

        File file2(_file_system, 2);

        /* -- Write into File 1 -- */
        file1.Write(1320, STRING1);

        /* -- Write into File 2 -- */

        file2.Write(20, STRING2);
```

Test Case :


Downloads
MP7
Recent
Starred
frame
frame
frame
gdt.C
gdt.H
idt.o
irq.C
kernel.C
machine.o
mem_pool.C
csce410@COE-VM-CSE1-L02: ~/Downloads/MP7
One second has passed
looking up file with id = 1
Existing file found.
File deleted successfully.
Deleting file with ID:2
Bochs x86-64 emulator, http://bochs.sourceforge.net/
Existing file found.
File deleted successfully.
Deleting file with ID:2
looking up file with id = 2
Existing file found.
File deleted successfully.
Existing file found.
File deleted successfully.
creating file with id:1
looking up file with id = 1
creating file with id:2
looking up file with id = 2
Opening file.
looking up file with id = 1
My block number as assigned by file system15874
Opening file.
looking up file with id = 2
My block number as assigned by file system16002
Closing file.
Closing file.
Opening file.
looking up file with id = 1
My block number as assigned by file system15874
IPS: 11.446M


looking up file with id = 2
My block number as assigned by file system1922
resetting file
Read bytes:1320
resetting file
Read bytes:20
Closing file.
Closing file.
Deleting file with ID:1
looking up file with id = 1
Existing file found.
irq.o
nker.ld
achine_
low.H