CS3026 Assessment 02 – Virtual Disk

Run Code

In each folder there includes the same Makefile.

Make commands:

- Make clean – deleted compilated files in the Directory
- Make – compiles program inside Directory
- Make run – runs the program in the Directory

CGS D3-D1

Shell.c

```
int main()
{
    // test format
    format();
    // call print block
    printBlock(0);
    writedisk("virtualdiskD3_D1");
    return 0 ;
}
```

Calls format printBlock 0 then writes to disk

Format

- The writeblock pre-existing function to write the created blocks to virtual disk
- The purpose of format was to create a structure for the virtual disk.
- It does this by Creating block 0, The FAT and the Root Dir block in the virtual disk.

```
diskblock_t block ; // block 0
   for(int i =0; i < BLOCKSIZE; ++i)
   {
      block.data[i] = '\0';
   }
   strcpy((char*)block.data, "CS3026 Operating Systems Assessment
2023");
   writeblock(&block,0);
```

- FAT table consists of two blocks of size 512 each because each entry is 2 bytes of space
- The FAT table value for block 0 , The FAT table itself (block 1 and 2) and root block , block 3
- All FAT tables are set to UNUSED before adding default blocks in
- The FAT table blocks point to each other the

```
diskblock_t block_1;
   diskblock_t block_2;
```

```
  // all FAT entries are UNUSED
   for(int i = 0; i < BLOCKSIZE; ++i)
   {
      FAT[i] = UNUSED; // 1024 entries set to UNUSED
   }
   FAT[0] = ENDOFCHAIN; // block 0
   FAT[1] = 2; // fat block 1
   FAT[2] = ENDOFCHAIN; // fat block 2
   FAT[3] = ENDOFCHAIN; // root
   // 4-1023 entries == UNUSED
   for(int i=0;i<FATENTRYCOUNT; ++i)
   {
      block_1.fat[i] = FAT[i]; // fatblock 0 -> 512 stores FAT 0 -> 512
entries
   }
   for (int i = FATENTRYCOUNT; i < BLOCKSIZE; ++i)
   {
      block_2.fat[i-512] = FAT[i]; //fatblock 0 -> 512 stores FAT 512 -
> 1023 entries
   }
   // write fat to disk
   writeblock(&block_1,1);
   writeblock(&block_2,2);
```

- All Root block data initially set to '\0'
- Isdir to identify as directory as such
- Nextentry set to 0 as default

```
 diskblock_t root_Block;
   // fill with \0
   for(int i=0;i<BLOCKSIZE;++i)
   {
      root_Block.data[i] = '\0';
   }
   //is a directory
   root_Block.dir.isdir = TRUE;
   root_Block.dir.nextEntry = 0; // starts at 0
   rootDirIndex = 3;
   // write root to block
   writeblock(&root_Block, 3);
```

hexdump -C virtualdiskD3_D1
HEXDUMP
Virtualdiskd3_d1

```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  ff ff ff ff ff ff ff ff  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00100000
```

CGS C3-C1

- Added helper functions to re-use code easily

```
//added functions
int findUNUSEDfatentry () ;
void addfatentry ( int blokno) ;
void addtofatentry (int blokno , int newblokno) ;
int findfilebyname (dirblock_t * current, const char * filename);
```

- findUNUSEDfatentry gets a fat entry that is UNUSED in the virtualdisk and returns the index. If not found returns EOC.

```
// find an UNUSED fat entry in FAT
int findUNUSEDfatentry ()
{
   // 4 - 1023
   for (int i=4;i<BLOCKSIZE;++i)
   {
      // finds fat entry set to UNUSED
      if (FAT[i] == UNUSED){
         return i; //return index of fat block
      }
   }
   //return EOC if fat not found
   return ENDOFCHAIN;
}
```

- addfatentry adds the fat entry (blokno) to FAT table and equals it to ENDOFCHAIN

```
/*add fat entry to block_1 or block_2 of fat table
   */
void addfatentry (int blokno)
{

   //check if blokno size fits in block 1 or block 2
   if (blokno > 1024)
   {
```

```
      printf("block no is outside range of fat table\n"); // blokno outside
FAT table range
   }
   // from 4 to 511 add to block 1
   if ( blokno < 511)
   {
      FAT[blokno] = ENDOFCHAIN;
      virtualDisk[1].fat[blokno] = ENDOFCHAIN;

   }
   else
   {
      // or from 512 - 1024 add to block 2
      FAT[blokno] = ENDOFCHAIN;
      virtualDisk[2].fat[blokno] = ENDOFCHAIN;
   }
}
```

- addtofatentry adds fat entry(newblokno) to the end of the previous chain (blokno) in FAT table

```
/*adds a fat entry to existing FAT chain
   */
void addtofatentry (int blokno, int newblokno)
{

   // checks within range
   if (blokno > 1024)
   {
      printf("block no is outside range of fat table\n"); // blokno outside
FAT table range
   }
   //checks less than 512 add block 1
   if (blokno < 511)
   {
      FAT[blokno] = newblokno;
      virtualDisk[1].fat[blokno] = newblokno;
   }
   else
   {
      // if greater than 512 add to block 2
      FAT[blokno] = newblokno;
      virtualDisk[2].fat[blokno] = newblokno;
   }
}
```

- findfilebyname (finds the name of a file inside of the dirblock parameter current) and returns the index . if not found returns EOF

```
int findfilebyname (dirblock_t * current ,const char* filename)
{
   for (int i=0; i < DIRENTRYCOUNT; ++i)
   {
      //checks if filename exists then returns the index of the file
      if (strcmp(current->entrylist[i].name, filename) == 0)
      {
         //printf("file found at index %d\n", i);
         return i;
      }
   }
   // could not find name
   return EOF;
}
```

- findUNUSEDdirentry finds dir entry that has unused set to TRUE meaning it not in use currently and returns the index . if not found return EOF

```
/* find an unused in the specific dir
   */
int findUNUSEDdirentry (dirblock_t *dir)
{
   for (int i=0;i<DIRENTRYCOUNT;++i)
   {
      if (dir->entrylist[i].unused == TRUE && dir-
>entrylist[i].entrylength == 0)
      {
         //printf("file found at index %d\n", i);
         return i;
      }
   }
   return EOF;
}
```

```
// check mode is in write or read
   if (strcmp("w", mode ) !=0  &&  strcmp("r", mode) != 0)
   {
      printf("file not opened in the appropriate mode\n");
      return NULL; // return nothing
   }
```

Function checks if mode is set to w or r meaning read or write mode

- If set to either mode the function checks if the file already exists and if it does returns the file

```
// if mode set to wtite
   if (strcmp("w", mode) == 0)
   {
      // get dir entry from root dir
      int dirIndex = findfilebyname(root, filename);
```

```
      //check file can be found in disk
      if (dirIndex == EOF)
      {
         // if entry name not found create start creating the file
         printf("Creating File...\n");
      }

      else // get exsiting file if name is found
      {

         file_ptr->blockno =
virtualDisk[rootDirIndex].dir.entrylist[dirIndex].firstblock;
         file_ptr->buffer = virtualDisk[file_ptr->blockno];
         //file_ptr->buffer.dir.entrylist[0] =
virtualDisk[rootDirIndex].dir.entrylist[dirIndex];
         return file_ptr;
      }
```

This not the direct structure of myfopen

```
// if opened in read mode
   if (strcmp(mode, "r") == 0)
   {
      // get the dir entry of the file
      int dirIndex = findfilebyname(root,filename);
      //check file name can be found in disk
      if (dirIndex == EOF)
      {
         printf("FileNotFoundError!\n");
         return NULL;
      }
      // get exsiting file
      file_ptr->blockno =
virtualDisk[rootDirIndex].dir.entrylist[dirIndex].firstblock;
      file_ptr->buffer = virtualDisk[file_ptr->blockno];
      file_ptr->pos = 0;
      // mode message
      printf("File opened in '%c' mode\n",*file_ptr->mode);
      return file_ptr;
   }
```

- If set to read mode and file doesn't already existing then it will return a pointer allocated memory but equals nothing so (NULL) which can checking in shell.c

```
if (ptr_file == NULL)
    {
      printf("FILE NOT OPENED");
      return 0;
    }
```

- However if the file does not already exist it will the be create afterward if mode set to write

```
// set the blockNo
      int UNUSED_fatentry = findUNUSEDfatentry(); // find a block number set
to UNUSED

      //set blokno to founs index
      file_ptr->blockno = UNUSED_fatentry; // is still FAT[file_ptr->blockno]
= UNUSED
      // set position
      file_ptr->pos = 0;


      //get unused directory in root
      dirIndex = findUNUSEDdirentry(root);
      if (dirIndex == EOF)
      {
         printf("All entry used up in directory");
         return NULL;
      }
      // add unused dir entry to root
      virtualDisk[rootDirIndex].dir.entrylist[dirIndex].entrylength = 0;//
entry length is 0 currently
      virtualDisk[rootDirIndex].dir.entrylist[dirIndex].filelength = 0;//
nothing in file so length 0
      virtualDisk[rootDirIndex].dir.entrylist[dirIndex].isdir = FALSE ;// is a
file
      virtualDisk[rootDirIndex].dir.entrylist[dirIndex].unused = FALSE;// set
to used
      virtualDisk[rootDirIndex].dir.entrylist[dirIndex].firstblock = file_ptr-
>blockno;// set firstblock
      //virtualDisk[rootDirIndex].dir.entrylist[dirIndex] = file_ptr-
>buffer.dir.entrylist[0]; // set root dir nextEntry to entry above
      strncpy(virtualDisk[rootDirIndex].dir.entrylist[dirIndex].name,
filename, MAXNAME); // set name

      virtualDisk[rootDirIndex].dir.nextEntry++;

      addfatentry(file_ptr->blockno);// add to fat block and FAT table
```

- Using helper functions to get unused dir entry inside root and fat entry, to add the FAT table and create entry in the root dir.
- Myfputc

```
/ check if file mode is set to write mode
   if ( strcmp(stream-> mode, "w") != 0)
   {
      //output error if not in "w" mode
      printf("MyFILE mode not set to 'w' mode \n");
```

```
   }
   else
```

- will not write to buffer data un less mode is set to write

```
// checks if the pos is >= to 1023
   if (stream->pos == BLOCKSIZE - 1 )
   {
      printf("buffer is full\n");
      // write buffer to if buffer is full to current block number location
      writeblock(current, stream->blockno);
```

- if buffer full write the buffer to the virtual disk then create new buffer to write on

```
// get UNUSED fat entry
      int newfatentry = findUNUSEDfatentry();

      // checks fat entry  does not equal EOC
      if (newfatentry == ENDOFCHAIN)
      {
         // if returned EOC ouput error
         printf("There are no more fat entries left\n");
      }

      //add new entry to the fat block
      addfatentry(newfatentry);// new fat entry = EOC

      // add new entry to the end of firstblock
      addtofatentry(stream->blockno, newfatentry);
```

- then set block and file blockno so it write in correct position

```
stream->blockno = newfatentry;// set new blokno to new fat entry found
      stream->pos = 0; //reset position to 0
      memset(stream->buffer.data, 0, BLOCKSIZE); // reset memory location to 0
```

- if not full write parameter (int b) to buffer data

```
// add the data b to the buffer data at the current pos of stream(file)
   current->data[stream->pos] = (Byte) b;
   stream->pos++;//increase pos
```

- mfgetc

```
// Check if file mode is set to read mode
   if (strcmp(stream->mode, "r") != 0) {
      // Output error if not in "r" mode
      printf("MyFILE mode not set to 'r' mode\n");
      return EOF;
   }
   else
```

- Will not run unless in read mode

```
if (stream->pos == BLOCKSIZE - 1)
   {
```

```
      //print the current block to terminal
      printBlock(stream->blockno);
      // if eoc is reached then return eof
      if (FAT[stream->blockno] == ENDOFCHAIN)
      {
         return EOF;
      }
      // traverse block chain
      stream->blockno = FAT[stream->blockno];
      stream->pos = 0; // reset pos
```

- If the end of the buffer is reached print the block then traverse to the next block in the FAT chain and set the new blockno and pos

```
int character;
   //stream->buffer = virtualDisk[stream->blockno]; // get buffer of current
block
   character = stream->buffer.data[stream->pos]; // get each character of the
buffer
   stream->pos++; // pos++
```

- While end not reach then set character equal to file pos in the buffer data the return character
- Myfclose

```
/*  myfclose function
 */
void myfclose ( MyFILE * stream )
{
   writeblock(&stream->buffer, stream->blockno);
   free(stream);
   printf("file is now closed\n");
}
```

- Write the filebuffer block to disk then frees the file pointer indicating it is now closed

```
/ test myfgetc
    int character = myfgetc(ptr_file);
    FILE * realfile = fopen("testfileC3_C1_copy.txt", "w"); // open file to
copy content
    while (character != EOF)
    {
      fprintf(realfile,"%c",character); // copy content
      character = myfgetc(ptr_file);
    }
    fclose(realfile);
    // close again
    myfclose(ptr_file);
```

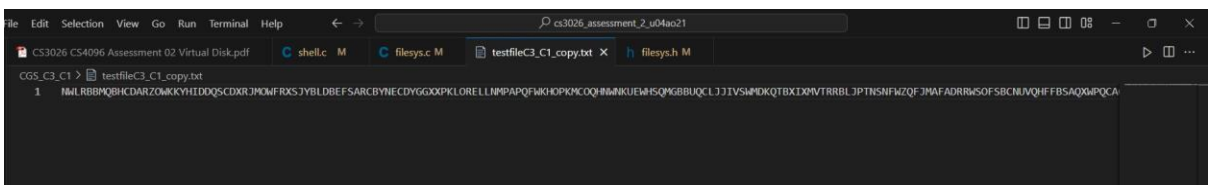
- Content of the myfgetc is also copied to real file in the directory
- Using make run > tracefileC3_C1.txt the output in the terminal is redirect to this text file
- Hexdump -C virtualdiskC3_C1

HEXDUMP

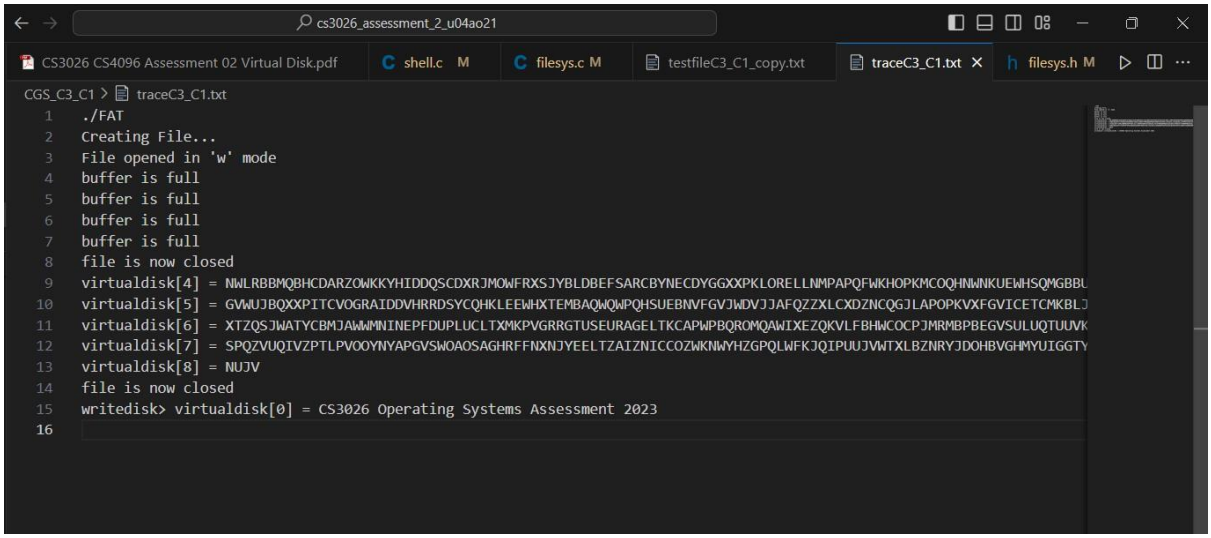
Virtualdiskc3_c1

```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  05 00 06 00 07 00 08 00  |................|
00000410  00 00 ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000420  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 74 65  |..............te|
00000c20  73 74 66 69 6c 65 2e 74  78 74 00 00 00 00 00 00  |stfile.txt......|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  4e 57 4c 52 42 42 4d 51  42 48 43 44 41 52 5a 4f  |NWLRBBMQBHCDARZO|
00001010  57 4b 4b 59 48 49 44 44  51 53 43 44 58 52 4a 4d  |WKKYHIDDQSCDXRJM|
00001020  4f 57 46 52 58 53 4a 59  42 4c 44 42 45 46 53 41  |OWFRXSJYBLDBEFSA|
00001030  52 43 42 59 4e 45 43 44  59 47 47 58 58 50 4b 4c  |RCBYNECDYGGXXPKL|
00001040  4f 52 45 4c 4c 4e 4d 50  41 50 51 46 57 4b 48 4f  |ORELLNMPAPQFWKHO|
00001050  50 4b 4d 43 4f 51 48 4e  57 4e 4b 55 45 57 48 53  |PKMCOQHNWNKUEWHS|
00001060  51 4d 47 42 42 55 51 43  4c 4a 4a 49 56 53 57 4d  |QMGBBUQCLJJIVSWM|
00001070  44 4b 51 54 42 58 49 58  4d 56 54 52 52 42 4c 4a  |DKQTBXIXMVTRRBLJ|
00001080  50 54 4e 53 4e 46 57 5a  51 46 4a 4d 41 46 41 44  |PTNSNFWZQFJMAFAD|
00001090  52 52 57 53 4f 46 53 42  43 4e 55 56 51 48 46 46  |RRWSOFSBCNUVQHFF|
000010a0  42 53 41 51 58 57 50 51  43 41 43 45 48 43 48 5a  |BSAQXWPQCACEHCHZ|
000010b0  56 46 52 4b 4d 4c 4e 4f  5a 4a 4b 50 51 50 58 52  |VFRKMLNOZJKPQPXR|
000010c0  4a 58 4b 49 54 5a 59 58  41 43 42 48 48 4b 49 43  |JXKITZYXACBHHKIC|
000010d0  51 43 4f 45 4e 44 54 4f  4d 46 47 44 57 44 57 46  |QCOENDTOMFGDWDWF|
000010e0  43 47 50 58 49 51 56 4b  55 59 54 44 4c 43 47 44  |CGPXIQVKUYTDLCGD|
000010f0  45 57 48 54 41 43 49 4f  48 4f 52 44 54 51 4b 56  |EWHTACIOHORDTQKV|
00001100  57 43 53 47 53 50 51 4f  51 4d 53 42 4f 41 47 55  |WCSGSPQOQMSBOAGU|
00001110  57 4e 4e 59 51 58 4e 5a  4c 47 44 47 57 50 42 54  |WNNYQXNZLGDGWPBT|
00001120  52 57 42 4c 4e 53 41 44  45 55 47 55 55 4d 4f 51  |RWBLNSADEUGUUMOQ|
00001130  43 44 52 55 42 45 54 4f  4b 59 58 48 4f 41 43 48  |CDRUBETOKYXHOACH|
00001140  57 44 56 4d 58 58 52 44  52 59 58 4c 4d 4e 44 51  |WDVMXXRDRYXLMNDQ|
00001150  54 55 4b 57 41 47 4d 4c  45 4a 55 55 4b 57 43 49  |TUKWAGMLEJUUKWCI|
00001160  42 58 55 42 55 4d 45 4e  4d 45 59 41 54 44 52 4d  |BXUBUMENMEYATDRM|
00001170  59 44 49 41 4a 58 4c 4f  47 48 49 51 46 4d 5a 48  |YDIAJXLOGHIQFMZH|
```

TestfileC3_C1copy.txt



TracefileC3_C1.txt



CGS B3-B1

- For this section Add a directory hierarchy to your virtualdisk that allows the creation of subdirectories
- create a directory “/myfirstdir/myseconddir/mythirddir” in the virtual disk
  - call mylistdir(“/myfirstdir/myseconddir”): print out the list of strings returned by this function
  - write out virtual disk to “virtualdiskB3_B1_a”
  - create a file “/myfirstdir/myseconddir/testfile.txt” in the virtual disk
  - call mylistdir(“/myfirstdir/myseconddir”): print out the list of strings returned by this function
  - write out virtual disk to “virtualdiskB3_B1_b

```
// declare pointer to pointers
    char ** listdirs;
    //test mymkdir
    mymkdir("/myfirstdir/myseconddir/mythirddir");
    // test mylistdir
    listdirs = mylistdir("/myfirstdir/myseconddir");
    //prints all dir in path
    printf("Path contents ... \n");
    for (int i = 0; i < DIRENTRYCOUNT; i++)
    {
        if(strcmp(listdirs[i], "\0") != 0)
        {
            printf("%s\n", listdirs[i]);
        }
    }

    writedisk("virtualdiskB3_B1_a");

    // test myfopen in path
    MyFILE * ptr_file = myfopen("/myfirstdir/myseconddir/testfile1.txt", "w");
```

```
    //test mylistdir
    listdirs = mylistdir("/myfirstdir/myseconddir");
    printf("Path contents ... \n");
    //prints all dir in path
    for (int i = 0; i < DIRENTRYCOUNT; i++)
    {
        if(strcmp(listdirs[i], "\0") != 0)
        {
            printf("%s\n", listdirs[i]);
        }
    }
    myfclose(ptr_file);
    writedisk("virtualdiskB3_B1_b");
```

- For create a path using mymkdir

```
char *token, *rest; // tokenize path and save pointer
   char *pathCopy = strdup(path); // copy path
   diskblock_t *currentParent = &virtualDisk[rootDirIndex]; // root
original parent direcotry
   currentDirIndex = rootDirIndex;//get root block index
   token = strtok_r(pathCopy, "/", &rest); // tokenize path
```

- This part token will contain tokenize strings of the path defined with strotk_r() split by '/' delimiter to search for the directory name individually
- pathCopy copies the path string
- rest is the save pointer used in the strtok_r() function
- currentParent diskblock is equal to the root block at the start of the function
- If token can not find a string after '/' it will equal NULL because strtok_r() will have returned NULL

```
while (token != NULL)
   {
      // find directory index in current directory
      int dirIndex = findfilebyname(&currentParent->dir, token);
      //printf("current directory is %s\n", token);
      // if found
      if (dirIndex != EOF) // return an index
      {
         // update the current parent to next dir
         currentDirIndex = currentParent-
>dir.entrylist[dirIndex].firstblock; // get fat index
         currentParent = &virtualDisk[currentDirIndex]; // update
currentparent
      }
      else // the index was not found
```

- While token doesn't equal NULL dirIndex will find the directory(token) by name using findfilebyname function I created earlier. If findfilebyname return EOF that means the name was not found in the currentParent entrylist. So if it dirIndex doesn't equal EOF then get the firstblock of that entry found firstblock = FAT block no .

- Equal global variable currentDirIndex to firstblock and the get the diskblock from disk with currentDirIndex and set currentParent to this block. This just a way traverse to the location of the last directory in the path

```
dirIndex = findUNUSEDdirentry(&currentParent->dir); // find unused dir in
current parent
         if (dirIndex  == EOF)
         {
            printf("All entries used up! \n");
         }
         else
         {
            int fatIndex = findUNUSEDfatentry(); // find an unused fat entry
            if (fatIndex == ENDOFCHAIN) // end not found
            {
               printf("FAT table is full!\n");
            }
            else
            {
```

- After if a directory is not found then we create on inside the currentParent dir block
- We do this be finding a unused dir entry and unused fat entry used findUNUSEDdirentry and findUNUSEDfatentry if both return a valid index then continue

```
               if ( path[0] == '/')
               {
                  printf("Creating directory...\n");
                  // initialise the next level directory
                  currentParent->dir.entrylist[dirIndex].firstblock =
fatIndex; // set firstblock to found entry
                  currentParent->dir.entrylist[dirIndex].isdir = TRUE; // is
dir
                  currentParent->dir.entrylist[dirIndex].unused = FALSE;//
used
                  strncpy(currentParent->dir.entrylist[dirIndex].name, token,
MAXNAME); // set name
                  writeblock(currentParent, currentDirIndex); // write the
current parent block

                  addfatentry(fatIndex); // add entry to fat table
                  currentDirIndex = fatIndex; // update current Index
                  currentParent = &virtualDisk[currentDirIndex]; // give it a
block
                  currentParent->dir.isdir = TRUE; // new block is dir
                  currentParent->dir.nextEntry = 0; // set to 0

                  // set all entry in current to unused
                  for (int i = 0; i< DIRENTRYCOUNT;++i)
                  {
                     currentParent->dir.entrylist[i].unused = TRUE;
```

```
                }
                writeblock(currentParent, currentDirIndex);
            }
```

- If the first string in the path was equal to "/" then that means its absolute and we should create the path we initialize the new directory in the currentparent then we add it FAT then we equal currentDirIndex to FAT block no and set currentParent to that block no in virtualDisk. The dir block of the block with be set to isdir=TRUE to indicate it's a dir. All entries is equal to unused.

MYLIST

- Uses a list of pointer Charlist to store the names of directories found in the path

```
// find directory index in current directory
      int dirIndex = findfilebyname(&currentParent->dir, token);
      // if found
      if (dirIndex != EOF)
      {
         // update the current parent to next dir
         currentDirIndex = currentParent->dir.entrylist[dirIndex].firstblock;
// get fat index
         currentParent = &virtualDisk[currentDirIndex]; // update
currentparent
         for (int i = 0; i < DIRENTRYCOUNT; i++)
         {
            // allocate memory to each pointer in the list
            CharList[i] = malloc(sizeof(char)*MAXNAME);
            // set the name of each entry (i) in the current parent
(entrylist) into the list at index i
            strcpy(CharList[i], currentParent->dir.entrylist[i].name);
         }
```

- This is done by copying the name every entry from the currentParent entrylist to the same an index in the Charlist each index is simultaneously allocated memory large enough to hold this entry name
- After this loop terminates the CharList will contain the names of all the directories in the path

MyFopen

- Some code in myfopen was changed to support calling the path as a parameter

```
/* seperate the directory from the file name*/
   char * lastSlash = strrchr(filename, '/');
   if (lastSlash != NULL)
   {
      size_t newlength = lastSlash - filename; // the size of dir path

      char newpath[newlength]; // create string

      strncpy (newpath, filename, newlength); // copy dir path into string
```

```
      newpath[newlength] = '\0';

      mymkdir(newpath); // call mymkdir to make the directories or get the
currentDirIndex

      filename = lastSlash + 1;  // this will be the name of the file only
   }
```

- This was done using the strrchr function the returns the index of the string where the last time the '/' was found in the string
- I used this because I separate the /firstdir/secondir / from the actual file name testfile.txt
- Strrchr will return NULL if delimiter '/' etc was not found so why lastSlash does not equal NULL. Newlength will equal the exact length of the path from the 0 to the index – 1 the lasr '/' was found the an array newPath is recreate with the the newlength allowing me to copy the from the path the exact length of the path that doesn't hold the actual file name
- Then mymkdir is called to create this dir path if it doesn't not exist of cd this path to currentDirIndex. Filename is then cut from the lastSlash index to the end of the path ("testfile.txt"). filename can be used all through the function so it was the easier way to change like this
- If a path isn't specified in when calling myfopen eg.

```
myfopen("testfile2.txt", "w");
```

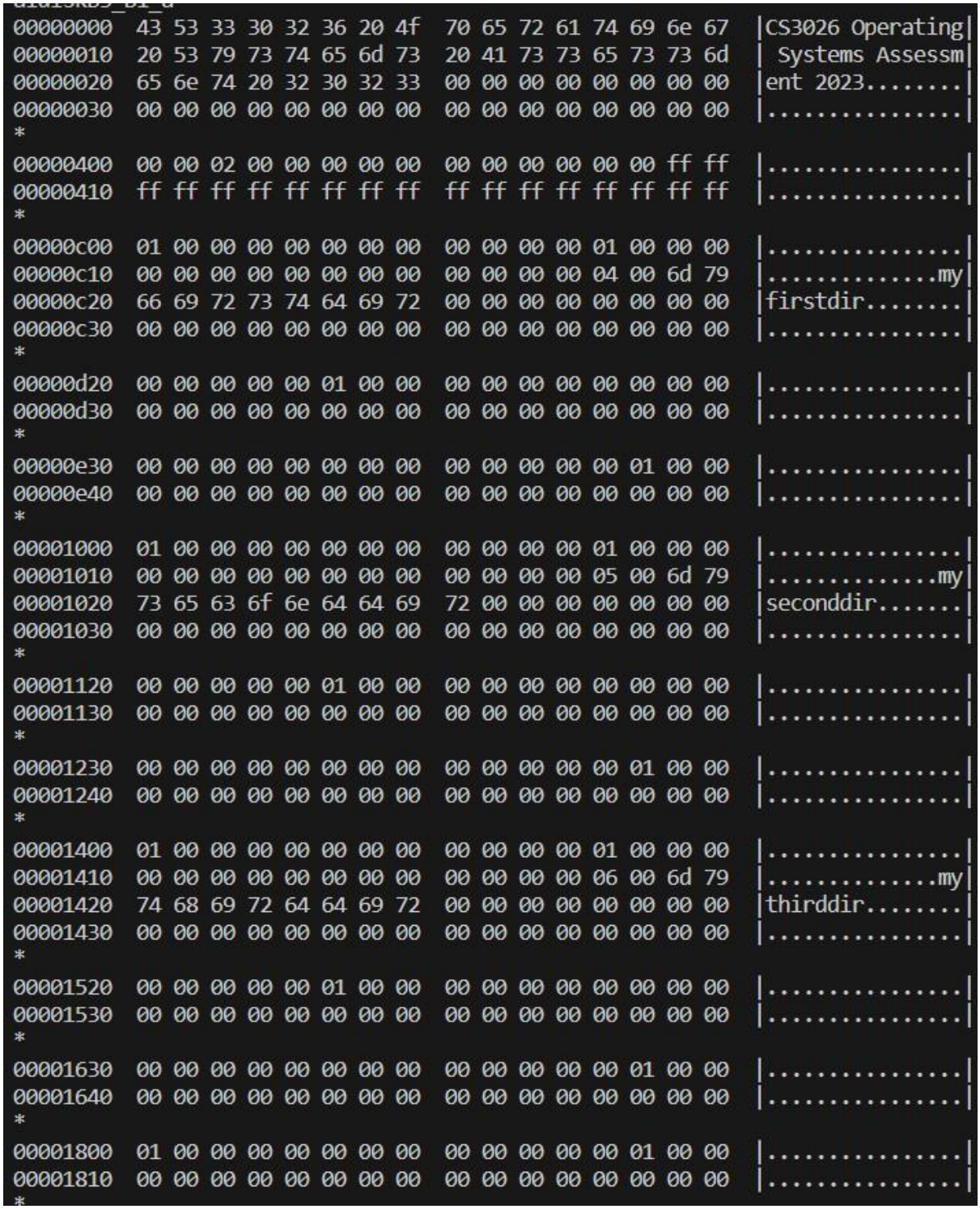
- Then file will be create inside the currentDirIndex block

```
diskblock_t * currentParent = &virtualDisk[currentDirIndex];
```

- Further parts of the code wasn't necessary to be changed

HEXDUMP

Virtualdiskb3_b1_a

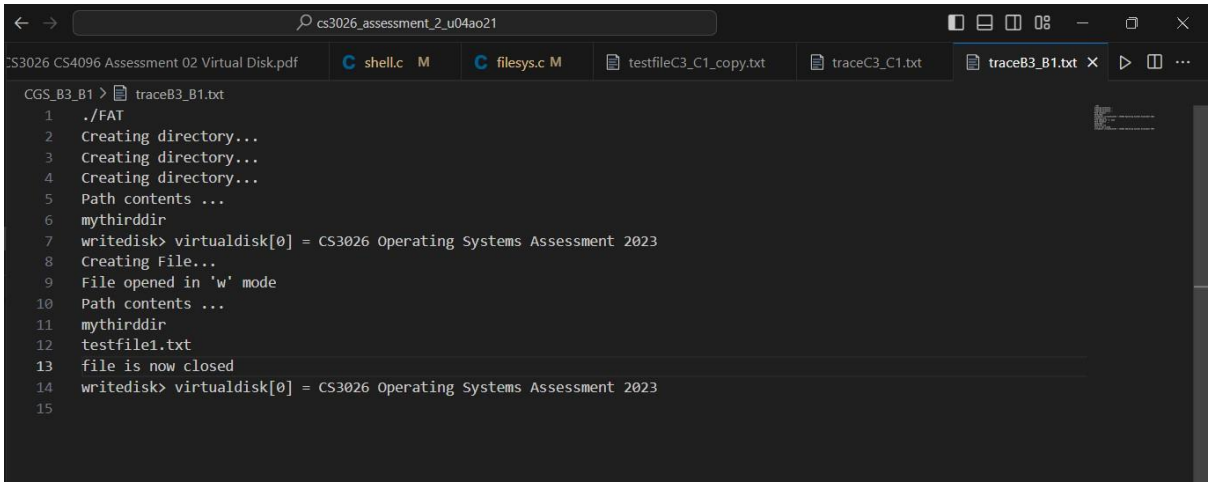


```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 00 00 ff ff  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 6d 79  |..............my|
00000c20  66 69 72 73 74 64 69 72  00 00 00 00 00 00 00 00  |firstdir........|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001010  00 00 00 00 00 00 00 00  00 00 00 00 05 00 6d 79  |..............my|
00001020  73 65 63 6f 6e 64 64 69  72 00 00 00 00 00 00 00  |seconddir.......|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 06 00 6d 79  |..............my|
00001420  74 68 69 72 64 64 69 72  00 00 00 00 00 00 00 00  |thirddir........|
00001430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001520  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001530  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001630  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001640  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001800  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001810  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

virtualdiskb3_b1_b

```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 6d 79  |..............my|
00000c20  66 69 72 73 74 64 69 72  00 00 00 00 00 00 00 00  |firstdir........|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001010  00 00 00 00 00 00 00 00  00 00 00 00 05 00 6d 79  |..............my|
00001020  73 65 63 6f 6e 64 64 69  72 00 00 00 00 00 00 00  |seconddir.......|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 06 00 6d 79  |..............my|
00001420  74 68 69 72 64 64 69 72  00 00 00 00 00 00 00 00  |thirddir........|
00001430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001530  00 00 00 00 07 00 74 65  73 74 66 69 6c 65 31 2e  |......testfile1.|
00001540  74 78 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |txt.............|
00001550  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001630  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001640  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001800  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001810  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

TracefileB3_B1.txt



CGS A5-A1

- mychdir(char * path), using the global variable “currentDir” as specified in filesys.c: a change into a
  directory will change the variable “currentDir”
  myremove(char * path) removes a file; the path can be absolute or relative
  myrmdir(char * path) removes a directory, if it is empty; the path can be absolute or relative

mylistDir

```
// ------------- Code Changed ----------------- ||
   // first check that path is self referenced
   if (path[0]== '.')
   {

      // the current block using currentDirIndex
      currentParent = &virtualDisk[currentDirIndex];
      for (int i = 0; i < DIRENTRYCOUNT; i++)
      {
         // allocate memory to each pointer in the list
         CharList[i] = malloc(sizeof(char)*MAXNAME);
         // set the name of each entry (i) in the current parent (entrylist)
into the list at index i
         strcpy(CharList[i], currentParent->dir.entrylist[i].name);
      }
      return CharList; // return List
   }
```

- With this change mylistdir(“.”) can be used to return the name of directories and files inside the currentDirIndex entrylist.

```
if (path[0] == '/')
   {
      currentParent = &virtualDisk[rootDirIndex];
      currentDirIndex = rootDirIndex;
   }
```

- Also if it start with “/” then currentDirIndex is set to rootDirIndex which is enabled globally and currentParent set to root dir block.

```
for (int i = 0; i < DIRENTRYCOUNT; i++)
   {
      // allocate memory to each pointer in the list
      CharList[i] = (char*) malloc(sizeof(char)*MAXNAME);
      // set the name of each entry (i) in the current parent (entrylist) into
the list at index i
      strcpy(CharList[i], currentParent->dir.entrylist[i].name);
   }
```

- Also this for loop was taken out of while loop to make the code run after and because it was necessary to be in the loop as it was only needing to copy the contents inside the end directory as was previously doing it for all of them
- Added helper function deletefat() to remove fat entry from the FAT table the it just the opposite of addfatentry().

```
void deletefat (int blokno)
{

   //check if blokno size fits in block 1 or block 2
   if (blokno > 1024)
   {
      printf("block no is outside range of fat table\n"); // blokno outside
FAT table range
   }
   // from 4 to 511 remove from  block 1
   if ( blokno < 512)
   {
      FAT[blokno] = UNUSED;
      virtualDisk[1].fat[blokno] = UNUSED;

   }
   else
   {
      // or from 512 - 1024 remove from block 2
      FAT[blokno] = UNUSED;
      virtualDisk[2].fat[blokno] = UNUSED;
   }
}
```

Mychdir

```
   if (path == '/')
   {
      currentDirIndex = rootDirIndex;
      currentParent = &virtualDisk[currentDirIndex];
   }
```

- If the path parameter as a whole equal only this '/' string then the currentDirIndex is equal to rootDirIndex

```
if (path == "..")
   {
      // update current
      currentDirIndex = parentDirIndex;
   }
```

- If the path == ".." then the currentDirIndex is equal to parentDirIndex which is th prev currentDirIndex
- `int parentDirIndex = rootDirIndex; // parenntDirIndex = prev level Dir index`

```
if (dirIndex != EOF)
      {
         parentDirIndex = currentDirIndex;
         currentDirIndex = currentParent->dir.entrylist[dirIndex].firstblock;
         currentParent = &virtualDisk[currentDirIndex];
      }
```

- Inside the loop the parentDirIndex is equal to the previous value of currentDirIndex

Myremove

```
char * lastSlash = strrchr(path, '/');

   if (lastSlash != NULL)
   {
      size_t newlength = lastSlash - path + 1; // the size of dir path

      char newpath[newlength]; // create string

      strncpy (newpath, path, newlength); // copy dir path into string

      newpath[newlength] = '\0';

      pathCopy = strdup(newpath); // dir path copied

      path = lastSlash + 1;  // this will be the name of the file only
```

- Like the method used to separate the file name from the dir path
- pathCopy copies the newpath which is the dir path
- and path parameter changed to file's name
- The loop only executes if a path or "/" is used but since no name who be found after "/"

```
mychdir(pathCopy); // change to directory
      currentParent = &virtualDisk[currentDirIndex];
      int dirIndex = findfilebyname(&currentParent->dir, path);
      if (dirIndex != EOF)
      {
```

- This Condition inside this loop wont execute

```
if (currentParent->dir.entrylist[dirIndex].isdir == FALSE)
```

```
        {
```

- Followed by a check if what is deleting is file or directory; will only delete file

```
// un-initialise the file dir
            currentParent->dir.entrylist[dirIndex].unused = TRUE;
            currentParent->dir.entrylist[dirIndex].filelength = 0;
            currentParent->dir.entrylist[dirIndex].entrylength = 0;
            for (int i=0;i<MAXNAME;++i)
            {
               currentParent->dir.entrylist[dirIndex].name[i] = '\0';
            }
```

- Removes file from the directory it is in

```
int nextblock = currentParent->dir.entrylist[dirIndex].firstblock;
            while (nextblock != ENDOFCHAIN)
            {
               // pointer instead of re writing to block
               diskblock_t * buffer = &virtualDisk[nextblock];
               for (int i = 0;i<BLOCKSIZE;++i)
               {
                  buffer->data[i] = '\0';
               }
               // save the number ---> go to next block ---> delete previous
block number
               int saveblkno = nextblock; nextblock = FAT[nextblock];
deletefat(saveblkno);
            }
            currentParent->dir.entrylist[dirIndex].firstblock = UNUSED;
```

- Nextblock = firstblock in chain
- Traverse through FAT chain equal all the data in the buffer to '\0'
- Save the block no , traverse through to the nextblock and delete the prev block no from the chain
- Set the firstblock in the chain to UNUSED in the currentParent directory

```
// if just the testfile name
   currentParent = &virtualDisk[currentDirIndex];
   int dirIndex = findfilebyname(&currentParent->dir, path);
   if (dirIndex != EOF)
   {
         if (currentParent->dir.entrylist[dirIndex].isdir == FALSE)
         {
            // un-initialise the file dir
            currentParent->dir.entrylist[dirIndex].unused = TRUE;
            currentParent->dir.entrylist[dirIndex].filelength = 0;
            currentParent->dir.entrylist[dirIndex].entrylength = 0;
            for (int i=0;i<MAXNAME;++i)
            {
               currentParent->dir.entrylist[dirIndex].name[i] = '\0';
            }
            currentParent->dir.nextEntry = 0;
```

```
            // remove the file buffer(s) from the FAT table
            int nextblock = currentParent->dir.entrylist[dirIndex].firstblock;
            while (nextblock != ENDOFCHAIN)
            {
               // pointer instead of re writing to block
               diskblock_t* buffer = &virtualDisk[nextblock];
               for (int i = 0;i<BLOCKSIZE;++i)
               {
                  buffer->data[i] = '\0';
               }
               // save the number ---> go to next block ---> delete previous
block number
               int saveblkno = nextblock; nextblock = FAT[nextblock];
deletefat(saveblkno);
            }
            currentParent->dir.entrylist[dirIndex].firstblock = UNUSED;
            printf("File is now deleted!\n");
         }
         else
         {
            printf("Can not delete a directory\n");
         }
   }
      else
      {
         printf("FileNotFoundError!\n");
      }
```

- Outside lastSlash if statement the code is repeated so that if a file is removed with specifying a path with it then it would remove the file from the current directory if it is found

Myrmdir

```
char *token, *rest; // tokenize path and save pointer
   char *pathCopy = strdup(path); // copy path
   diskblock_t *currentParent = &virtualDisk[currentDirIndex];
   fatentry_t prevDirIndex = 0;
   int dirIndex;
   token = strtok_r(pathCopy, "/", &rest);
   while (token != NULL)
   {
      dirIndex = findfilebyname(&currentParent->dir, token); // find directory
by name
      //currentDir = &currentParent->dir.entrylist[dirIndex]; // change into
path directory !!
      if (dirIndex != EOF)
      {
         prevDirIndex = currentDirIndex;
         currentDirIndex = currentParent->dir.entrylist[dirIndex].firstblock;
         currentParent = &virtualDisk[currentDirIndex];
```

```
    }
    else
    {
       printf("Not found path \n");
    }
    token = strtok_r(NULL, "/",&rest);
}
```

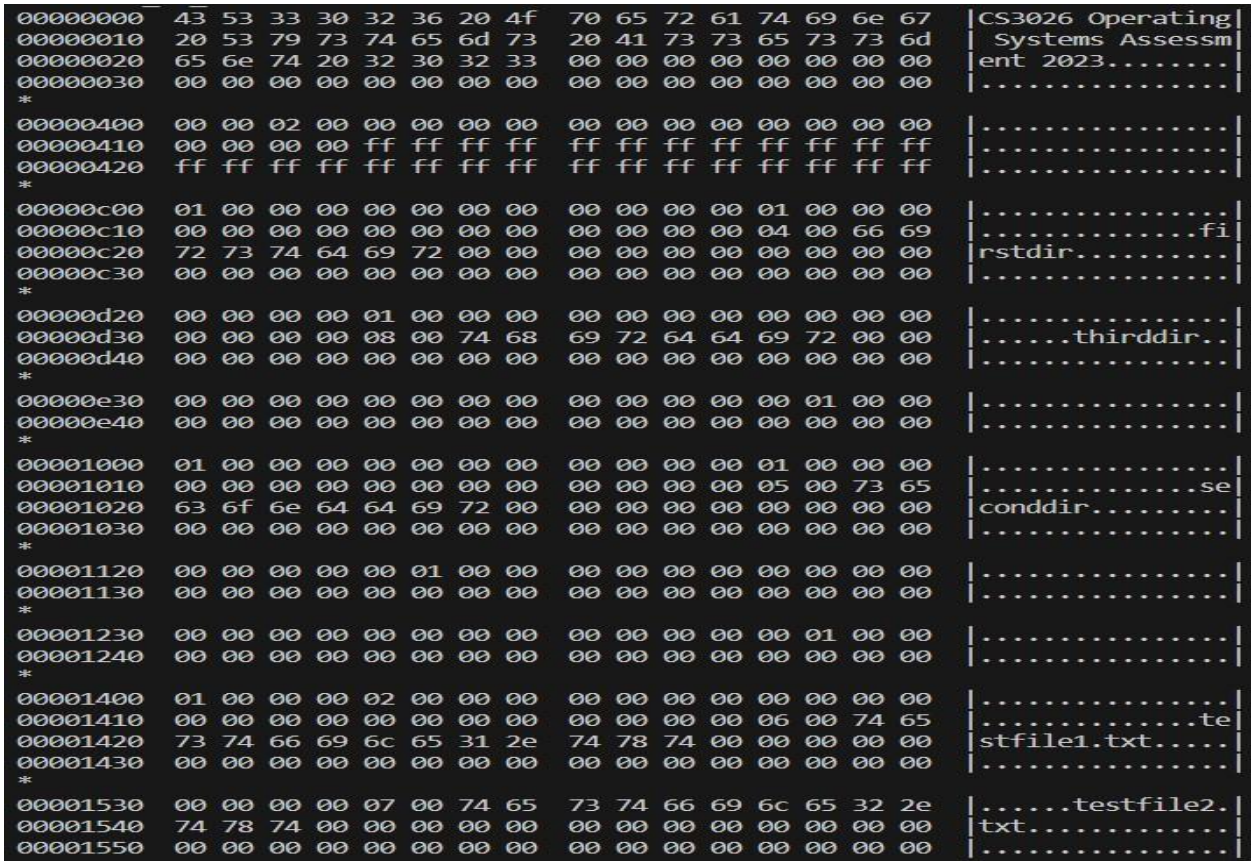
- prevDirIndex stored directory fat indexes
- dirIndex declared outside while loop so it can be used outside loop
- prevDirIndex equal to previous value of currentDirIndex each iteration

```
diskblock_t * prevParent = &virtualDisk[prevDirIndex];
   direntry_t p;
   for (int i = 0; i < MAXNAME; ++i)
   {
      p.name[i] = '\0';
   }
   p.unused = TRUE;
   p.firstblock = NULL;
   p.filelength = 0;
   p.entrylength = 0;
   p.isdir= FALSE;
   if (dirIndex != EOF)
   {
      prevParent->dir.entrylist[dirIndex] = p;
      //prevParent->dir.entrylist[dirIndex].unused = TRUE;
      deletefat(currentDirIndex);
      printf("deleted\n");
   }
```

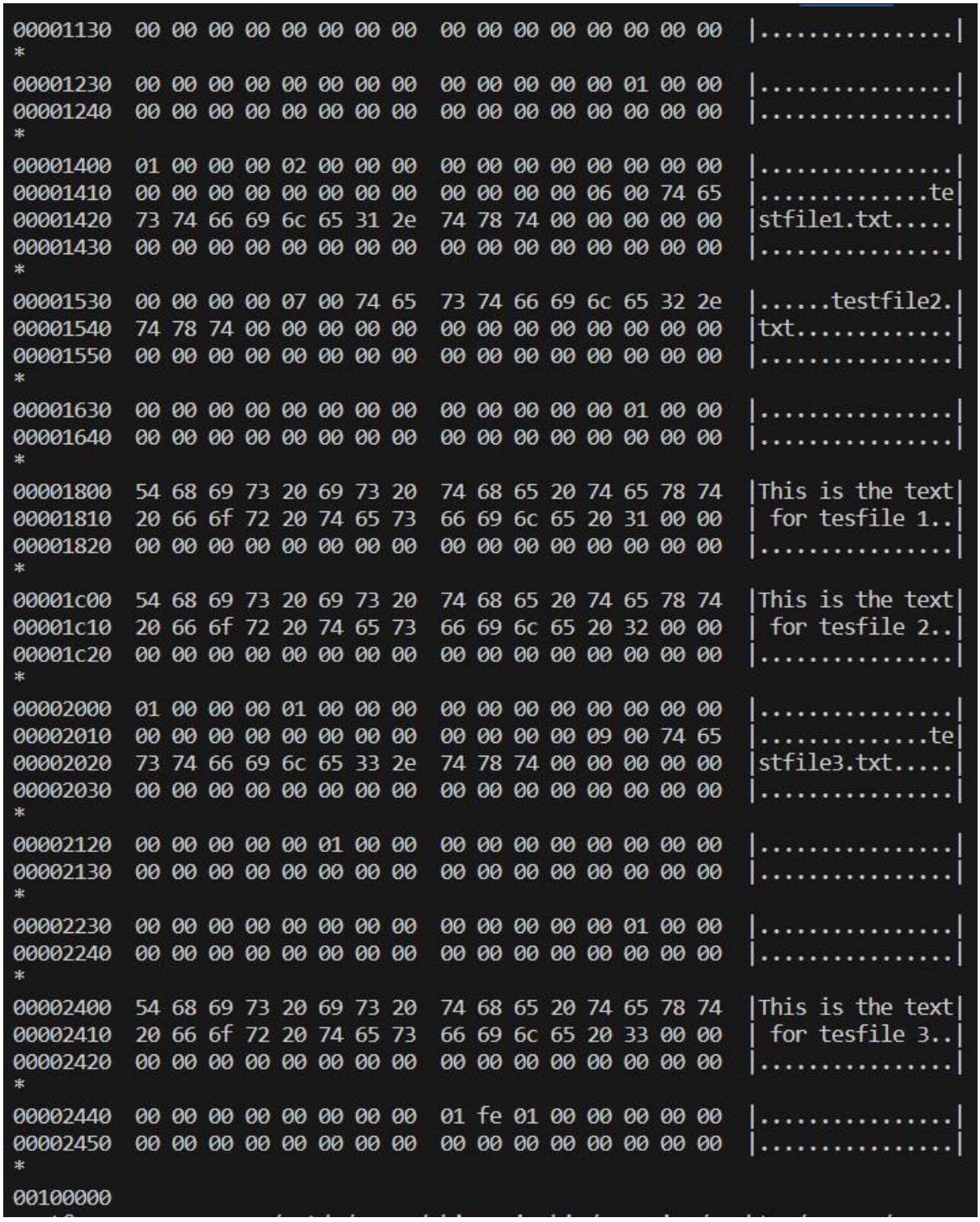
- creates a null dir entry ‘p’ and block that equal the previous block of currentParent
- dir entry ‘p’ has it’s variables initialized to equal NULL/UNUSED/0/FALSE to clear to clear entry values
- if dirIndex is not EOF then set name equal to ‘\0’
- clear dir entry from the prevParent using dir entry p
- set the entry to unused so it maybe used again
- delete currentDirIndex from FAT using delete fat function

HEXDUMP

VirtualdiskA5_A1_a



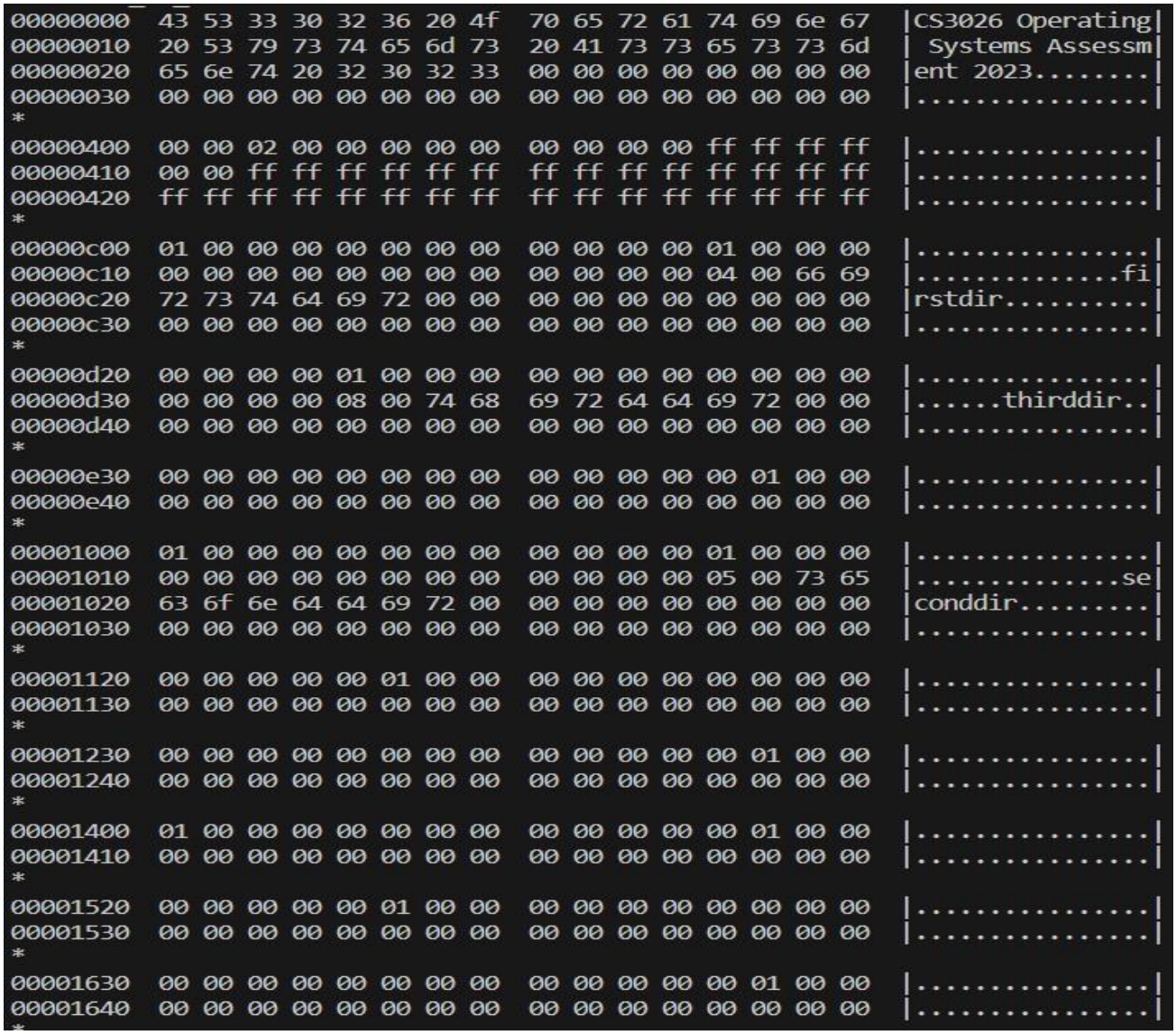
```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000410  00 00 00 00 ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000420  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 66 69  |..............fi|
00000c20  72 73 74 64 69 72 00 00  00 00 00 00 00 00 00 00  |rstdir..........|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 08 00 74 68  69 72 64 64 69 72 00 00  |......thirddir..|
00000d40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001010  00 00 00 00 00 00 00 00  00 00 00 00 05 00 73 65  |..............se|
00001020  63 6f 6e 64 64 69 72 00  00 00 00 00 00 00 00 00  |conddir.........|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 02 00 00 00  00 00 00 00 00 00 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 06 00 74 65  |..............te|
00001420  73 74 66 69 6c 65 31 2e  74 78 74 00 00 00 00 00  |stfile1.txt.....|
00001430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001530  00 00 00 00 07 00 74 65  73 74 66 69 6c 65 32 2e  |......testfile2.|
00001540  74 78 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |txt.............|
00001550  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```



virtualdiskA5_A1_b

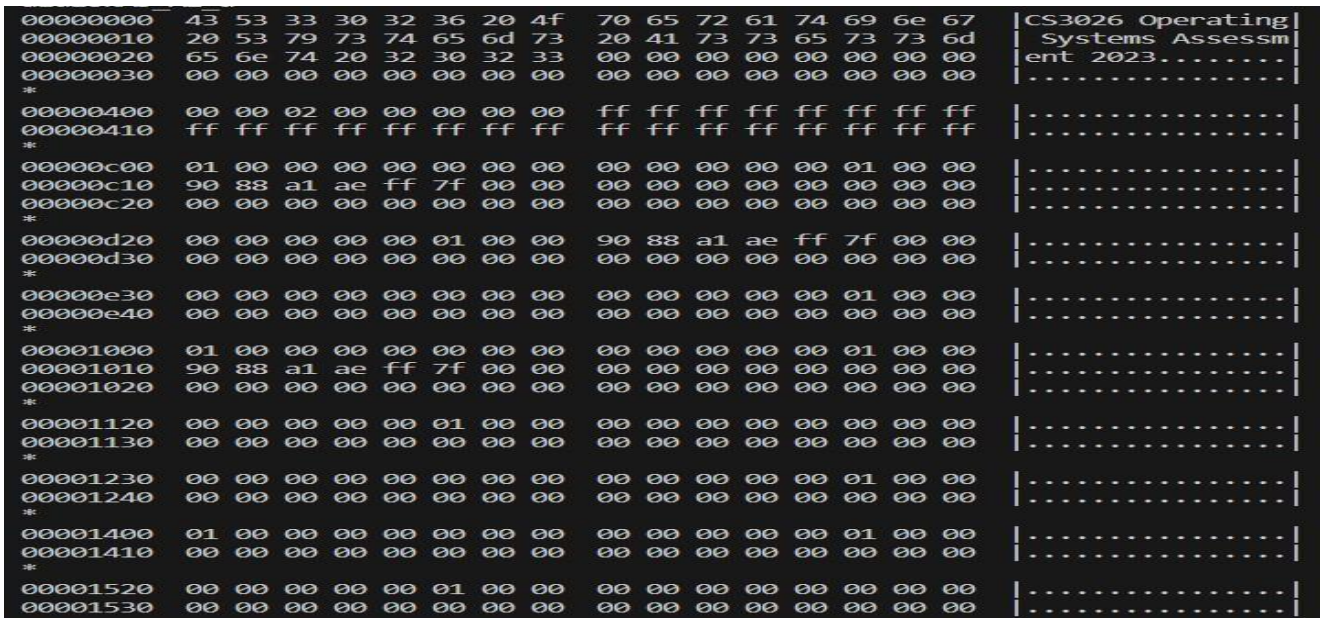
```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 ff ff ff ff  |................|
00000410  00 00 00 00 ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000420  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 66 69  |..............fi|
00000c20  72 73 74 64 69 72 00 00  00 00 00 00 00 00 00 00  |rstdir..........|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 08 00 74 68  69 72 64 64 69 72 00 00  |......thirddir..|
00000d40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001010  00 00 00 00 00 00 00 00  00 00 00 00 05 00 73 65  |..............se|
00001020  63 6f 6e 64 64 69 72 00  00 00 00 00 00 00 00 00  |conddir.........|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001520  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001530  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001630  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001640  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00002000  01 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00002010  00 00 00 00 00 00 00 00  00 00 00 00 09 00 74 65  |..............te|
00002020  73 74 66 69 6c 65 33 2e  74 78 74 00 00 00 00 00  |stfile3.txt.....|
00002030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

virtualdiskA5_A1_c



```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 ff ff ff ff  |................|
00000410  00 00 ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000420  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 04 00 66 69  |..............fi|
00000c20  72 73 74 64 69 72 00 00  00 00 00 00 00 00 00 00  |rstdir..........|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000d30  00 00 00 00 08 00 74 68  69 72 64 64 69 72 00 00  |......thirddir..|
00000d40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00001010  00 00 00 00 00 00 00 00  00 00 00 00 05 00 73 65  |..............se|
00001020  63 6f 6e 64 64 69 72 00  00 00 00 00 00 00 00 00  |conddir.........|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001520  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001530  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001630  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001640  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

virtualdiskA5_A1_d



```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  ff ff ff ff ff ff ff ff  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000c10  90 88 a1 ae ff 7f 00 00  00 00 00 00 00 00 00 00  |................|
00000c20  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  90 88 a1 ae ff 7f 00 00  |................|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001010  90 88 a1 ae ff 7f 00 00  00 00 00 00 00 00 00 00  |................|
00001020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001120  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001230  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001520  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001530  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

A3

A3: Try to write a copy function that allows you to copy files from your real hard disk into your virtual
disk and vice versa.

CopyToMyFILE:

- This function reads the contents of a real file in hard disk and copy the text into a file made in virtual disk
- This is done using built in function fread and fseek to get the filesize
- The memory is allocated to a char array of size = file size + 1
- Then the content of the file is read to Text array
- Afterward to is put into the virtualdisk file using mfputc function

CopyToRealFILE:

- This function copy contents to a real file the same way executed in C3_C1
- Using mfgetc to read each character in the buffer and returning to a character and while that character doesn't reach EOF to will write the character to real file and get new character recalling mfgetc

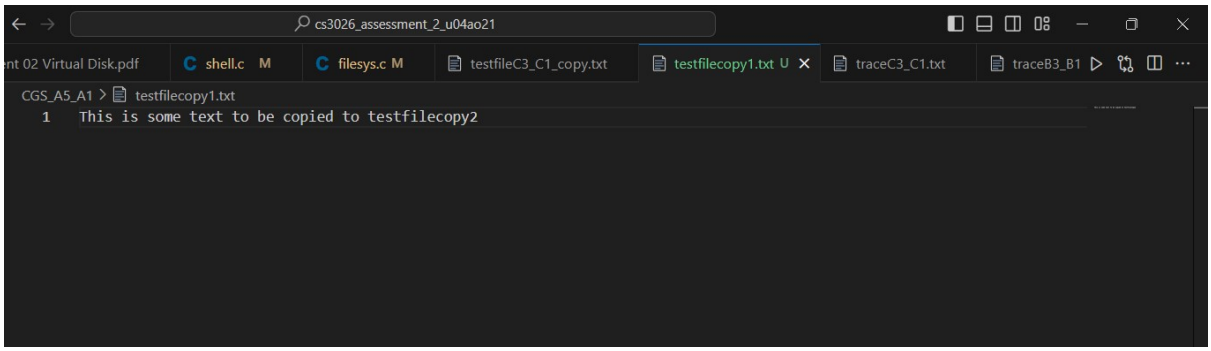
```
/* A3 Copy function
   */
void CopyToMyFILE ( FILE * realfile, MyFILE * fakefile)
{
   // read real file to string
   fseek(realfile, 0, SEEK_END);
    long int file_size = ftell(realfile);
    fseek(realfile, 0, SEEK_SET);
     char *Text = (char *)malloc(file_size + 1);
   fread(Text, 1, file_size, realfile);
   Text[file_size] = '\0';
   // write string to fake file
   for (int i=0; i < file_size;++i)
   {
      myfputc(Text[i], fakefile);
   }
   printf("Real file copied to disk\n");
}
void CopyToRealFILE ( MyFILE * fakefile, FILE * realfile)
{
   // copy fake file into real file
   int character = myfgetc(fakefile);
   // while character isnt EOF
   while(character != EOF)
   {
      // write to file each character
      fprintf(realfile, "%c", character);
```

```
        // get new character
        character = myfgetc(fakefile);
    }
}
```

The shell.c changes

```
/* A3 Copy functions test
    */
    //COPY from real file in hard disk to virtual disk
    /* IF THERE IS ALREADY TEXT IN testfilecopy2.txt DELETE THE CONTENTS
        THEN RUN THE CODE BELOW */
    mychdir("/");
    FILE * realfile = fopen("testfilecopy1.txt", "r");
    MyFILE * fakefile = myfopen("testfile5.txt", "w");
    CopyToMyFILE(realfile, fakefile);
    fclose(realfile);
    myfclose(fakefile);
    // Copy Vice versa function
    mychdir("/");
    FILE * realfile2 = fopen("testfilecopy2.txt", "w");
    fakefile = myfopen("testfile5.txt", "r");
    CopyToRealFILE(fakefile, realfile2);
    fclose(realfile);
    myfclose(fakefile);
    // deleting files
    mychdir("/");
    myremove("testfile5.txt");
    writedisk("virtualdiskA3");
```

Testfilecopy1



Testfilecopy2

HEXDUMP

VirtualdiskA3

```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 ff ff ff ff ff ff  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000c10  00 e9 24 91 fe 7f 00 00  00 00 00 00 04 00 74 65  |..$...........te|
00000c20  73 74 66 69 6c 65 35 2e  74 78 74 00 00 00 00 00  |stfile5.txt.....|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 01 00 00  00 e9 24 91 fe 7f 00 00  |..........$.....|
00000d30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  54 68 69 73 20 69 73 20  73 6f 6d 65 20 74 65 78  |This is some tex|
00001010  74 20 74 6f 20 62 65 20  63 6f 70 69 65 64 20 74  |t to be copied t|
00001020  6f 20 74 65 73 74 66 69  6c 65 63 6f 70 79 32 00  |o testfilecopy2.|
00001030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  01 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001410  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001520  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  |................|
00001530  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001630  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00001640  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

A2

A2: Try to implement a copy and a move function that relocates files within your virtual disk.

```
/* A2 Copy and move function
        */
    mychdir("/");
```

```
    MyFILE * newfile1 = myfopen("testfile6.txt", "w");
    char text6 [32] = "This is the text for tesfile 6";
    for (int i=0;i<sizeof(text6);++i)
    {
        myfputc(text6[i], newfile1);
    }
    myfclose(newfile1);
    newfile1 = myfopen("testfile6.txt", "r");
    MyFILE * newfile2 = myfopen("testfile7.txt", "w");
    movefile("testfile6.txt", "testfile7.txt");
    writedisk("virtualdiskA2");
```

- First I wrote some text in text6 into testfile6.txt I opened in virtualdisk and closed the file
- Then opened a second file testfile7.txt
- Then reopened same testfile6 and called movefile function contents of first file into second file

MOVEFILE PART 1

- Traverse to the index of the file you want to copy from
- Create char array of size =BLOCKSIZE(1024)
- Then copy content of buffer to the array

```
// get contents of file1
   char *token, *rest; // tokenize path and save pointer
   char *pathCopy = strdup(file1); // copy path of file you want to copy from
   diskblock_t *buffer= &virtualDisk[rootDirIndex];
   token = strtok_r(pathCopy, "/", &rest);
   while (token != NULL)
   {
      int dirIndex = findfilebyname(&buffer->dir, token); // find directory by
name
      //currentDir = &currentParent->dir.entrylist[dirIndex]; // change into
path directory !!
      if (dirIndex != EOF)
      {
         // update buffer and index
         currentDirIndex = buffer->dir.entrylist[dirIndex].firstblock;
         buffer = &virtualDisk[currentDirIndex];
      }
      else
      {
         printf("PLZ imput correct path!\n");
      }
      token = strtok_r(NULL, "/",&rest);
   }
   char * content[BLOCKSIZE]; // char array that hold content of buffer
   // copy contents of buffer data to array
   for (int i =0;i<BLOCKSIZE;++i)
```

```
   {
      content[i] = buffer->data[i];
   }
```

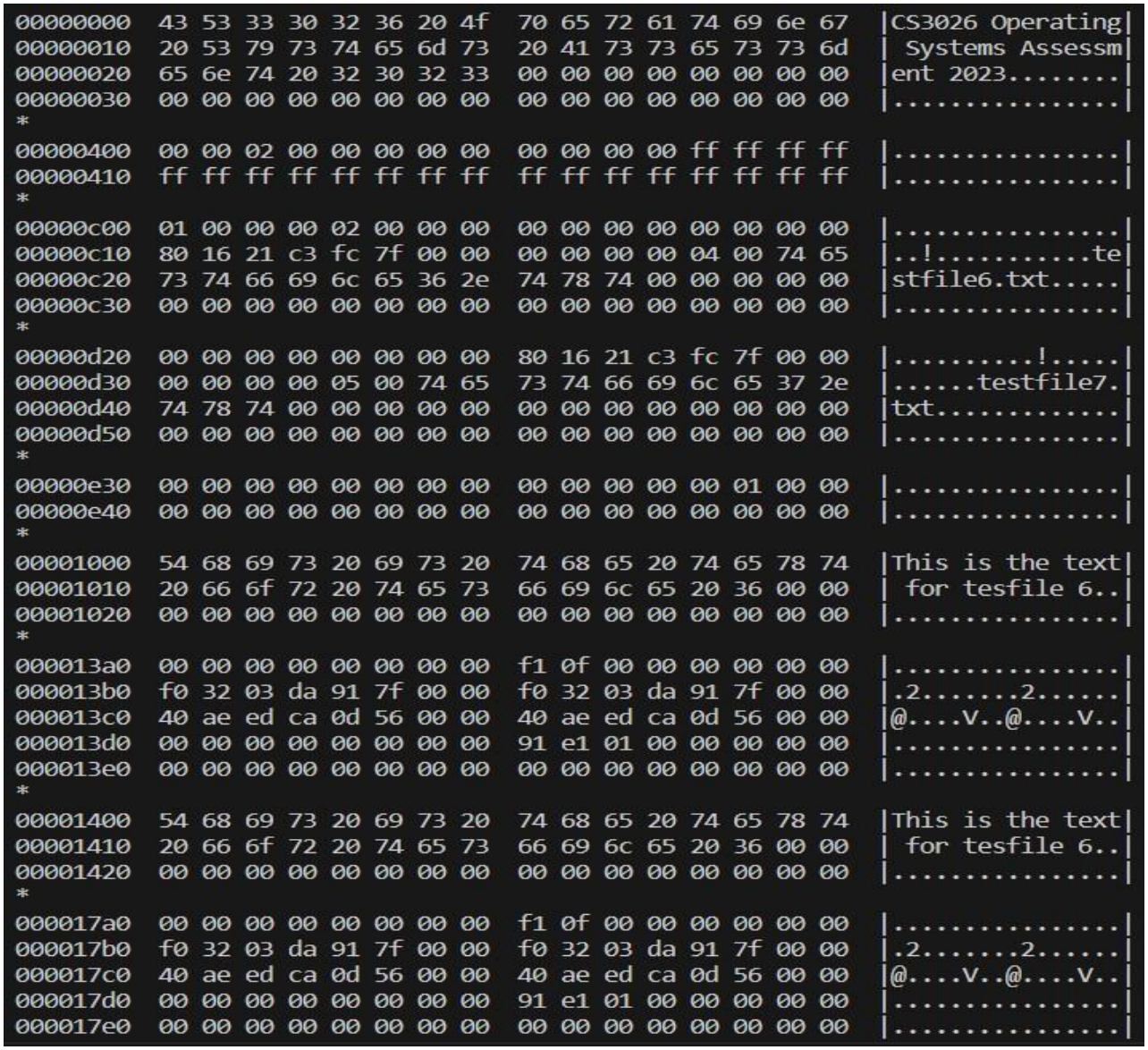
MOVEFILE PART 2

- Copy the path of the second file and traverse directory till you get the index
- The copy contents of the array to the buffer

```
// create file2 and copy contents to file2
   char * rest2;
   pathCopy = strdup(file2); // copy path of the file you want to copy to
   buffer= &virtualDisk[rootDirIndex]; // reset buffer
   token = strtok_r(pathCopy, "/", &rest2);
   while (token != NULL)
   {
      int dirIndex = findfilebyname(&buffer->dir, token); // find directory by
name
      //currentDir = &currentParent->dir.entrylist[dirIndex]; // change into
path directory !!
      if (dirIndex != EOF)
      {
         //update buffer and index
         currentDirIndex = buffer->dir.entrylist[dirIndex].firstblock;
         buffer = &virtualDisk[currentDirIndex];
      }
      else
      {
         printf("PLZ imput correct path!\n");
      }
      token = strtok_r(NULL, "/",&rest2);
   }
   // write to buffer of file2 the content that was copied
   for (int i =0;i<BLOCKSIZE;++i)
   {
      buffer->data[i] = content[i];
   }
```

HEXDUMP

VirtualdiskA2

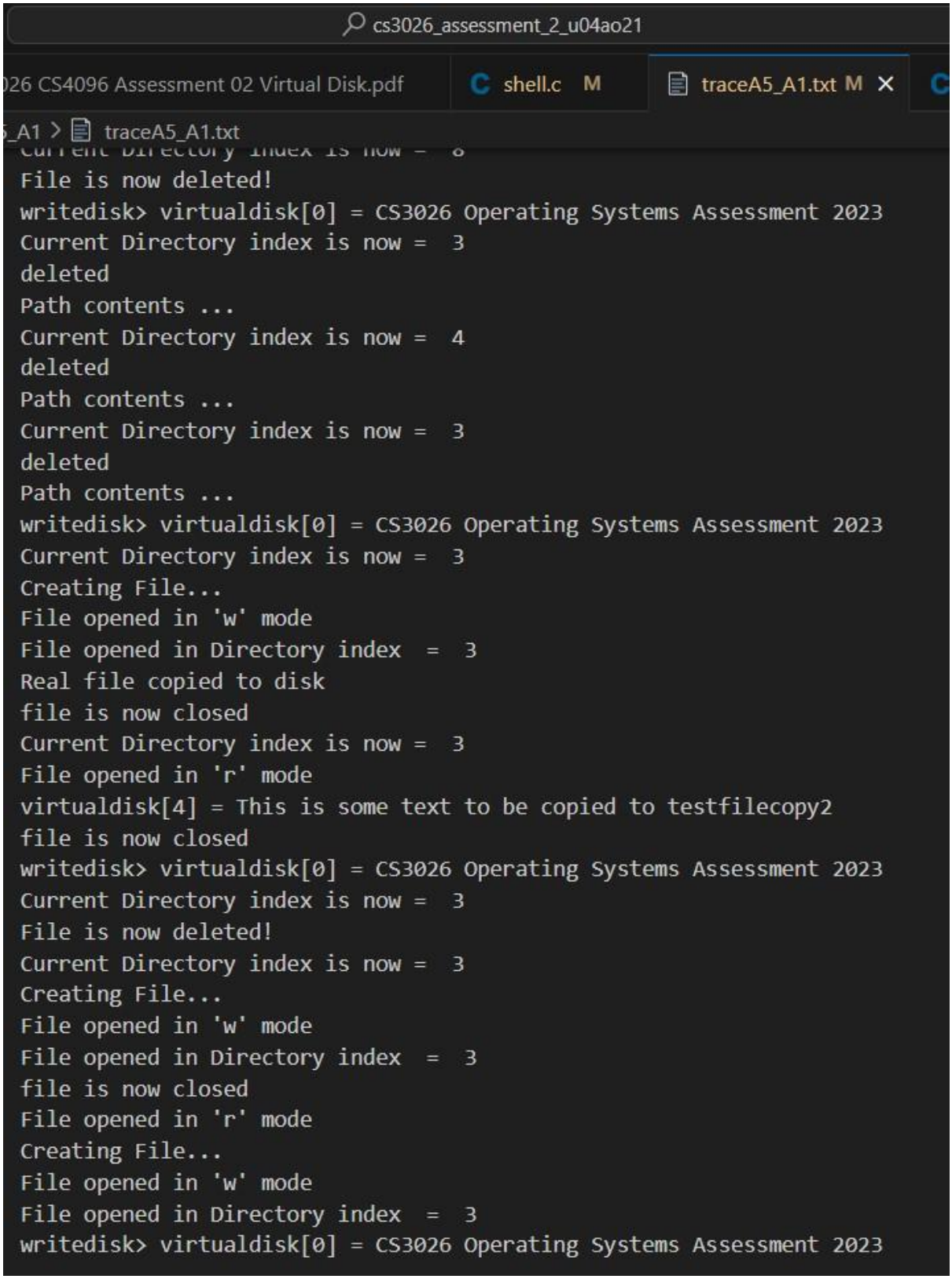


```
00000000  43 53 33 30 32 36 20 4f  70 65 72 61 74 69 6e 67  |CS3026 Operating|
00000010  20 53 79 73 74 65 6d 73  20 41 73 73 65 73 73 6d  | Systems Assessm|
00000020  65 6e 74 20 32 30 32 33  00 00 00 00 00 00 00 00  |ent 2023........|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 02 00 00 00 00 00  00 00 00 00 ff ff ff ff  |................|
00000410  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00000c00  01 00 00 00 02 00 00 00  00 00 00 00 00 00 00 00  |................|
00000c10  80 16 21 c3 fc 7f 00 00  00 00 00 00 04 00 74 65  |..!...........te|
00000c20  73 74 66 69 6c 65 36 2e  74 78 74 00 00 00 00 00  |stfile6.txt.....|
00000c30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000d20  00 00 00 00 00 00 00 00  80 16 21 c3 fc 7f 00 00  |..........!.....|
00000d30  00 00 00 00 05 00 74 65  73 74 66 69 6c 65 37 2e  |......testfile7.|
00000d40  74 78 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |txt.............|
00000d50  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000e30  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  54 68 69 73 20 69 73 20  74 68 65 20 74 65 78 74  |This is the text|
00001010  20 66 6f 72 20 74 65 73  66 69 6c 65 20 36 00 00  | for tesfile 6..|
00001020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000013a0  00 00 00 00 00 00 00 00  f1 0f 00 00 00 00 00 00  |................|
000013b0  f0 32 03 da 91 7f 00 00  f0 32 03 da 91 7f 00 00  |.2.......2......|
000013c0  40 ae ed ca 0d 56 00 00  40 ae ed ca 0d 56 00 00  |@....V..@....V..|
000013d0  00 00 00 00 00 00 00 00  91 e1 01 00 00 00 00 00  |................|
000013e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400  54 68 69 73 20 69 73 20  74 68 65 20 74 65 78 74  |This is the text|
00001410  20 66 6f 72 20 74 65 73  66 69 6c 65 20 36 00 00  | for tesfile 6..|
00001420  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000017a0  00 00 00 00 00 00 00 00  f1 0f 00 00 00 00 00 00  |................|
000017b0  f0 32 03 da 91 7f 00 00  f0 32 03 da 91 7f 00 00  |.2.......2......|
000017c0  40 ae ed ca 0d 56 00 00  40 ae ed ca 0d 56 00 00  |@....V..@....V..|
000017d0  00 00 00 00 00 00 00 00  91 e1 01 00 00 00 00 00  |................|
000017e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

TraceA5_A1.txt

CGS_A5_A1 > traceA5_A1.txt

```
./FAT
Creating directory...
Creating directory...
Creating File...
File opened in 'w' mode
File opened in Directory index  =  5
file is now closed
Path contents ...
Current Directory index is now =  5
testfile1.txt
Current Directory index is now =  5
Path contents ...
testfile1.txt
Creating File...
File opened in 'w' mode
File opened in Directory index  =  5
file is now closed
Path contents ...
testfile1.txt
testfile2.txt
Creating directory...
Creating File...
File opened in 'w' mode
File opened in Directory index  =  8
file is now closed
Path contents ...
testfile3.txt
writedisk> virtualdisk[0] = CS3026 Operating Systems Assessment 2023
Current Directory index is now =  5
Path contents ...
testfile1.txt
testfile2.txt
File is now deleted!
File is now deleted!
Path contents ...
writedisk> virtualdisk[0] = CS3026 Operating Systems Assessment 2023
Current Directory index is now =  8
```


cs3026_assessment_2_u04ao21
026 CS4096 Assessment 02 Virtual Disk.pdf
shell.c M
traceA5_A1.txt M
_A1 > traceA5_A1.txt
File is now deleted!
writedisk> virtualdisk[0] = CS3026 Operating Systems Assessment 2023
Current Directory index is now = 3
deleted
Path contents ...
Current Directory index is now = 4
deleted
Path contents ...
Current Directory index is now = 3
deleted
Path contents ...
writedisk> virtualdisk[0] = CS3026 Operating Systems Assessment 2023
Current Directory index is now = 3
Creating File...
File opened in 'w' mode
File opened in Directory index = 3
Real file copied to disk
file is now closed
Current Directory index is now = 3
File opened in 'r' mode
virtualdisk[4] = This is some text to be copied to testfilecopy2
file is now closed
writedisk> virtualdisk[0] = CS3026 Operating Systems Assessment 2023
Current Directory index is now = 3
File is now deleted!
Current Directory index is now = 3
Creating File...
File opened in 'w' mode
File opened in Directory index = 3
file is now closed
File opened in 'r' mode
Creating File...
File opened in 'w' mode
File opened in Directory index = 3
writedisk> virtualdisk[0] = CS3026 Operating Systems Assessment 2023