CSCI3150 Tutorial on Assignment 2

Mingshen Sun

April 11&12, 2016

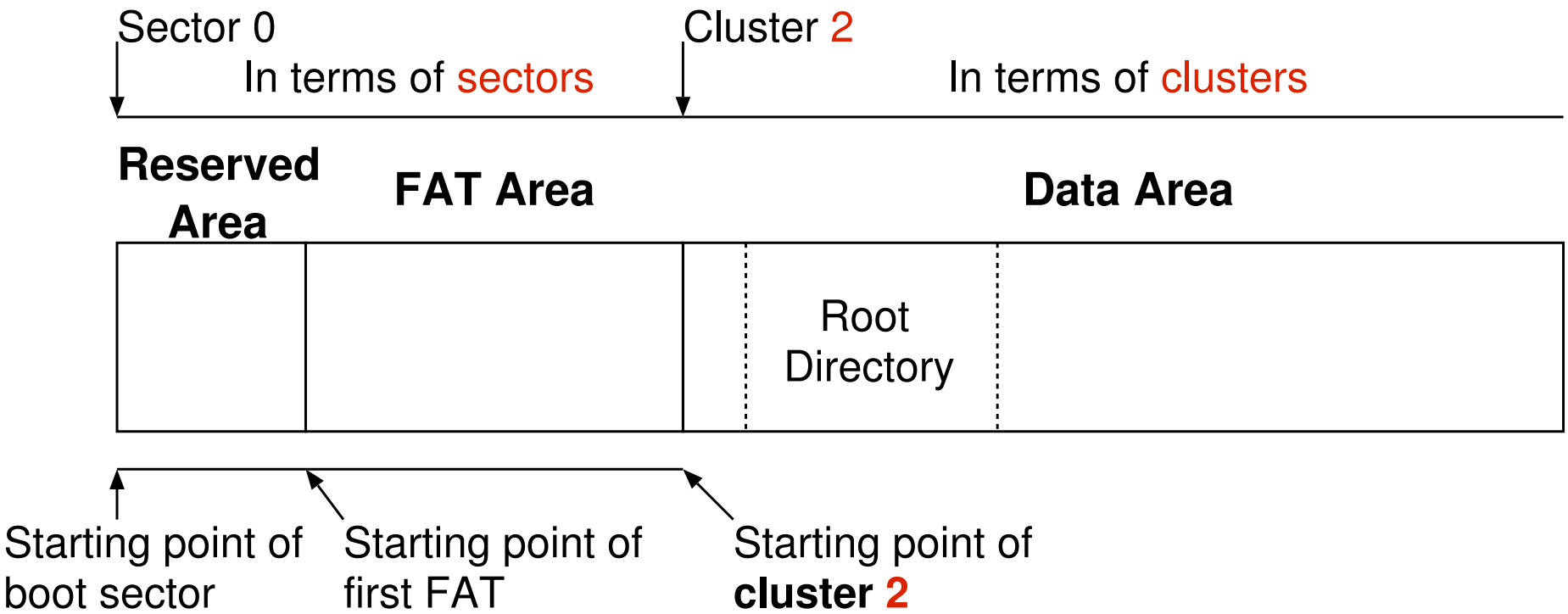
Outline

- FAT32 Overview.
- Reserved Area (Boot Sector).
- FAT Area.
- Reserved Area (Directory Entry).

FAT32 Overview

- Reserved area.
  - Storing important information about the file system. Boot sector is stored in sector 0.
- FAT area.
  - Storing a number of FATs (File Allocation Tables).
- Data area.
  - Storing root directory, as well as other files and directories.

A General Idea



Reserved Area

- How to access?
  - Located at Sector 0.
  - Just read from the very begining of the file system!
  - `fread()`
  - `fseek()`
- Get boot sector:
  - The data structure of boot sector is **pre-defined**
  - Declare a same data structure in your code.
  - Read in the data structure from the very begining of the file system.
  - Now, you can extract useful information.

Boot Sector (0-35 Bytes)

```c
#pragma pack(push, 1)
struct BootEntry {
  uint8_t BS_jmpBoot[3];   /* Assembly instruction to jump to boot code */
  uint8_t BS_OEMName[8];   /* OEM Name in ASCII */
  uint16_t BPB_BytsPerSec; /* Bytes per sector. Allowed values include 512,
                              1024, 2048, and 4096 */
  uint8_t BPB_SecPerClus;  /* Sectors per cluster (data unit). Allowed values
                              are powers of 2, but the cluster size must be
                              32KB or smaller */
  uint16_t BPB_RsvdSecCnt; /* Size in sectors of the reserved area */
  uint8_t BPB_NumFATs;     /* Number of FATs */
  uint16_t BPB_RootEntCnt; /* Maximum number of files in the root directory for
                              FAT12 and FAT16. This is 0 for FAT32 */
  uint16_t BPB_TotSec16;   /* 16-bit value of number of sectors in file system */
  uint8_t BPB_Media;       /* Media type */
  uint16_t BPB_FATSz16;    /* 16-bit size in sectors of each FAT for FAT12 and
                               FAT16. For FAT32, this field is 0 */
  uint16_t BPB_SecPerTrk;  /* Sectors per track of storage device */
  uint16_t BPB_NumHeads;   /* Number of heads in storage device */
  uint32_t BPB_HiddSec;    /* Number of sectors before the start of partition */
  uint32_t BPB_TotSec32;   /* 32-bit value of number of sectors in file system.
                              Either this value or the 16-bit value above must be
                              0 */
```

Boot Sector (36-89 Bytes)

```
  uint32_t BPB_FATSz32;      /* 32-bit size in sectors of one FAT */
  uint16_t BPB_ExtFlags;     /* A flag for FAT */
  uint16_t BPB_FSVer;        /* The major and minor version number */
  uint32_t BPB_RootClus;     /* Cluster where the root directory can be
                                found */

  uint16_t BPB_FSInfo;       /* Sector where FSINFO structure can be
                                found */
  uint16_t BPB_BkBootSec;    /* Sector where backup copy of boot sector is
                                located */
  uint8_t BPB_Reserved[12]; /* Reserved */
  uint8_t BS_DrvNum;         /* BIOS INT13h drive number */
  uint8_t BS_Reserved1;      /* Not used */
  uint8_t BS_BootSig;        /* Extended boot signature to identify if the
                                next three values are valid */
  uint32_t BS_VolID;         /* Volume serial number */
  uint8_t BS_VolLab[11];     /* Volume label in ASCII. User defines when
                                creating the file system */
  uint8_t BS_FilSysType[8]; /* File system type label in ASCII */
};
#pragma pack(pop)
```

- `fread(&boot_entry, 1, sizeof(struct BootEntry), fp);`

FAT Area

- There may be more than one FATs.
- How to access?
  - First FAT: right after reserved area.
  - n-th FAT: right after the $(n-1)$-th FAT.
  - Use the information in boot sector and do a simple math.
- FAT entry: 4-byte variable, stores information about file organization.

Data Area

- How to access?
  - Locate after the last FAT.
  - You know all the information from boot sector, do the math.
- Data area is organized in terms of clusters, not sectors.
- Clusters
  - Basic unit of file access.
  - A group of contiguous sectors.
  - The starting cluster is Cluster 2!
  - There are no Cluster 0 and Cluster 1.

Root Directory

- The location of root directory can be get from boot sector.
  - If BPB RootClus = 2, then the root directory is located in Cluster 2.
- The root directory is a directory.

Directory

- Directory
  - Can be viewed a file storing a number of directory entries.
- Directory entry
  - Contains important information of the files/subdirs in the directory.
  - e.g., filename, file length, starting cluster, etc.
  - The data structure of directory entry is also pre-defined.

Directory Entry

```c
#pragma pack(push, 1)
struct DirEntry {
  uint8_t DIR_Name[11];     /* File name */
  uint8_t DIR_Attr;         /* File attributes */
  uint8_t DIR_NTRes;        /* Reserved */
  uint8_t DIR_CrtTimeTenth; /* Created time (tenths of second) */
  uint16_t DIR_CrtTime;     /* Created time (hours, minutes, seconds) */
  uint16_t DIR_CrtDate;     /* Created day */
  uint16_t DIR_LstAccDate;  /* Accessed day */
  uint16_t DIR_FstClusHI;   /* High 2 bytes of the first cluster address */
  uint16_t DIR_WrtTime;     /* Written time (hours, minutes, seconds */
  uint16_t DIR_WrtDate;     /* Written day */
  uint16_t DIR_FstClusLO;   /* Low 2 bytes of the first cluster address */
  uint32_t DIR_FileSize;    /* File size in bytes. (0 for directories) */
};
#pragma pack(pop)
```

- `fread(&dir_entry, 1, sizeof(struct DirEntry), fp);`

Directory Entry – filename

- `DIR Name[11]`.
- Capital letters only.
- Bytes 0..7 store filename, bytes 8..10 store extension.
- If a byte is NA, padded with spaces.

Directory Entry – filename

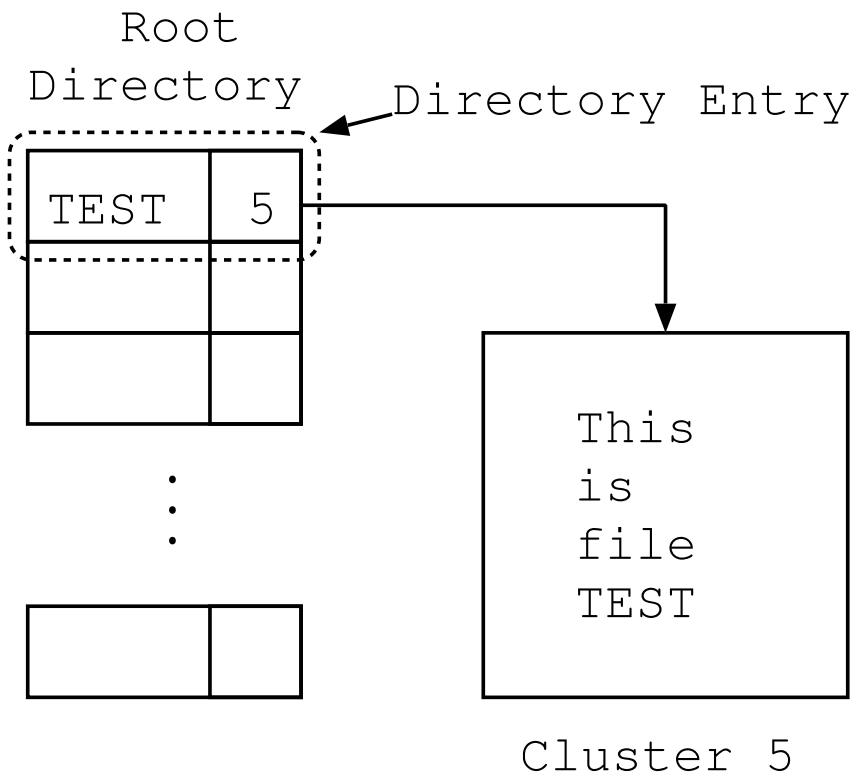
Filename	DIR Name[]
FOO	`FOO␣␣␣␣␣␣␣␣`
FOO.BAR	`FOO␣␣␣␣␣BAR`
FOO.	`FOO␣␣␣␣␣␣␣␣`
CSCI3150	`CSCI3150␣␣␣`
CSCI3150.ASG	`CSCI3150ASG`
.EXT	illegal!

File Attributes

- 8-bit value. Each bit represents one property.
  - `0000 0001`: Read only
  - `0000 0010`: Hidden
  - `0000 0100`: System
  - `0000 1000`: Volume label
  - `0001 0000`: Directory
  - `0010 0000`: Archive
  - `0000 1111`: Long file name.
- A file can has multiple properties:
- `0001 0011 = 0001 0000 | 0000 0010 | 0000 0001`
- A hidden read-only directory

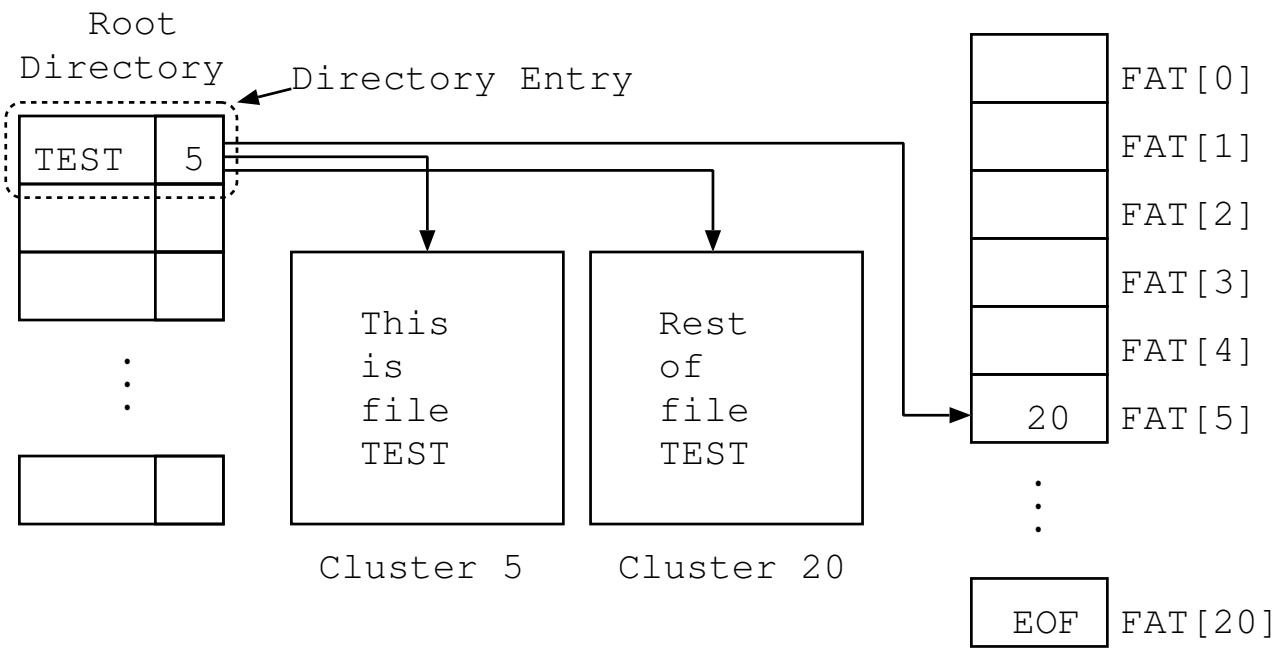
How to Access a File?

- Get the starting cluster from the direcotry entry and go to the corresponding cluster.



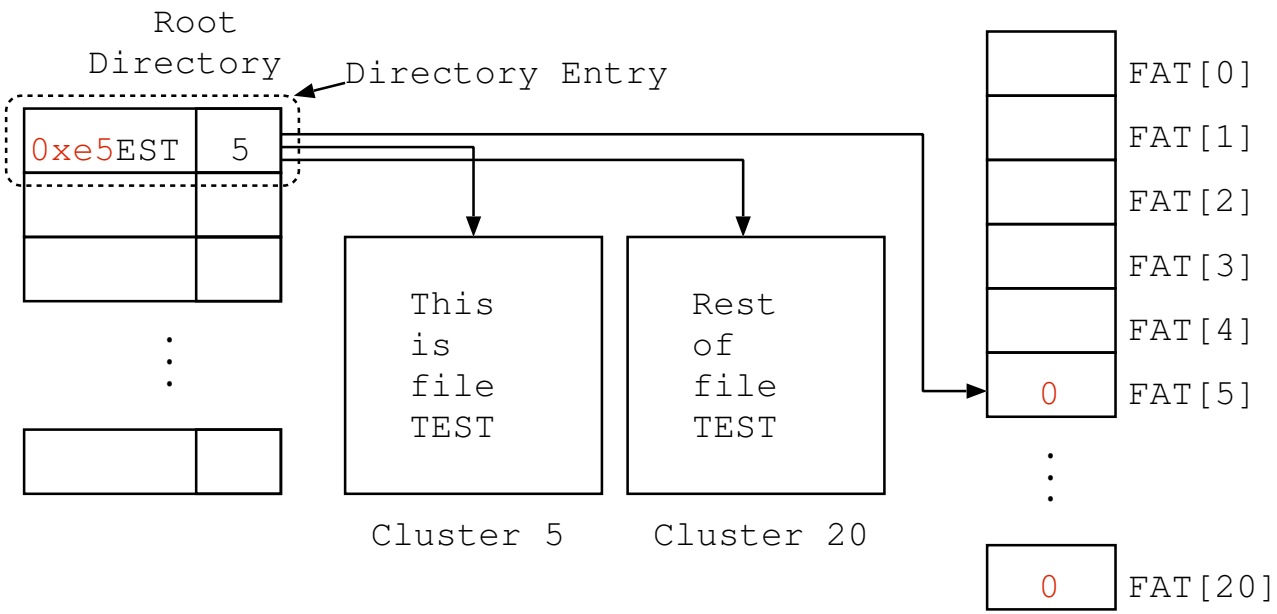
How to Access a Large File?

- Use FAT.
- Hint: directory is also a file, except that it stores directory entries.



After Deletion

- First character of DIR Name is changed to 0xe5.
- Related entry in FAT is changed to 0.
- Everything else is unchanged.



How to Get the Content of Deleted File?

- Compare the user-provided filename to locate the corresponding directory entry.
- Care for ambiguous filename. Filename of BEST and TEST are both changed to 0xe5EST after deletion.
- We assume the size of deleted file is less than one cluster, thus get the starting cluster and read out the content.

When and How the FAT32 File System Store File Using LFN?

- Recall that we talked about 8.3 format.
- Any non-8.3-format filename will be stored as LFN.
- There are multiple directory entries store information about the file.
  - One ”real” entry stores file attribute, file length starting cluster and other information.
  - One or more entries store filename.

FLN Structure

long file name

```
LFN Directory Entry

0100400: 4174 0065 0073 0074 002e 000f 00cb 6500  At.e.s.t......e.
0100410: 7800 7400 0000 ffff ffff 0000 ffff ffff  x.t.............

0100420: 5445 5354 2020 2020 4558 5420 0000 637a  TEST    EXT ..cz
0100430: 6b43 6b43 0000 637a 6b43 0500 0c03 0000  kCkC..czkC......

Real Directory Entry


Directory entries representing test.ext
```

Figure: Directory entries representing filename test.ext

FLN Structure

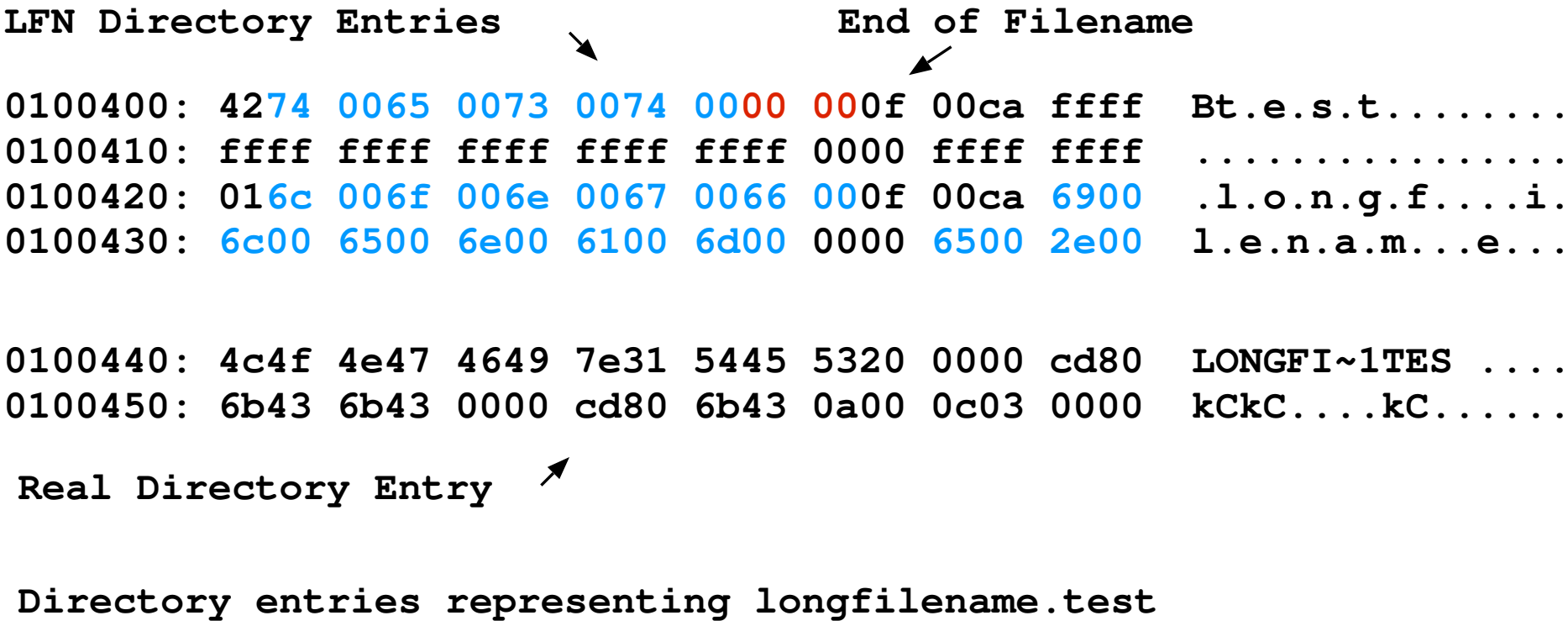


Figure: Directory entries representing filename longfilename.test

Index of LFN Directory Entries

- LFN is stored in a stack of LFN directory entries.
- The first byte of a LFN directory entry is the index.
- Suppose there are n LFN entries
  - The index of first n − 1 entry is n − 1
  - The index of the last entry is n|0x 40

FLN Structure

```
          Last LFN Entry                    LFN Directory Entries
                ↓                                   ↙
0100400: 4461 006d 0065 002e 0074 000f 00b9 6500  Da.m.e...t....e.
0100410: 7300 7400 0000 ffff ffff 0000 ffff ffff  s.t.............
0100420: 0372 0079 0020 006c 006f 000f 00b9 6e00  .r.y. .l.o....n.
0100430: 6700 2000 6600 6900 6c00 0000 6500 6e00  g. .f.i.l...e.n.
0100440: 026c 006c 0079 0020 0072 000f 00b9 6500  .l.l.y. .r....e.
0100450: 6100 6c00 6c00 7900 2000 0000 7600 6500  a.l.l.y. ...v.e.
0100460: 0174 0068 0069 0073 0020 000f 00b9 6900  .t.h.i.s. ....i.
0100470: 7300 2000 6100 2000 7200 0000 6500 6100  s. .a. .r...e.a.

0100480: 5448 4953 4953 7e31 5445 5320 0000 2976  THISIS~1TES ..)v
0100490: 7243 7243 0000 2976 7243 0300 5401 0000  rCrC..)vrC..T...

Real Directory Entry  ↗

Directory entries representing
"this is a really really very long filename.test"
```

Figure: Index of LFN entries

Recovery of LFN File

- The index of LFN directory entries will be changed to 0xe5 after deletion.
- Since the LFN entries are stacked in the root directory, You can always get back the original LFN and get the located cluster.

no need to recover long file name

Q&A

Thank you.

dont use string copy because of \0