OPERATING SYSTEMS
TUTORIAL 10

Useful Structures

```
// Super block
struct __attribute__((__packed__)) superblock_t {
  uint8_t   fs_id [8];
  uint16_t block_size;
  uint32_t file_system_block_count;
  uint32_t fat_start_block;
  uint32_t fat_block_count;
  uint32_t root_dir_start_block;
  uint32_t root_dir_block_count;
};
```

```
// Time and date entry
struct __attribute__((__packed__)) dir_entry_timedate_t {
  uint16_t year;
  uint8_t  month;
  uint8_t  day;
  uint8_t  hour;
  uint8_t  minute;
  uint8_t  second;
};
```

```
// Directory entry
struct __attribute__((__packed__)) dir_entry_t {
  uint8_t                  status;
  uint32_t                 starting_block;
  uint32_t                 block_count;
  uint32_t                 size;
  struct   dir_entry_timedate_t create_time;
  struct   dir_entry_timedate_t modify_time;
  uint8_t                  filename[31];
  uint8_t                  unused[6];
};
```

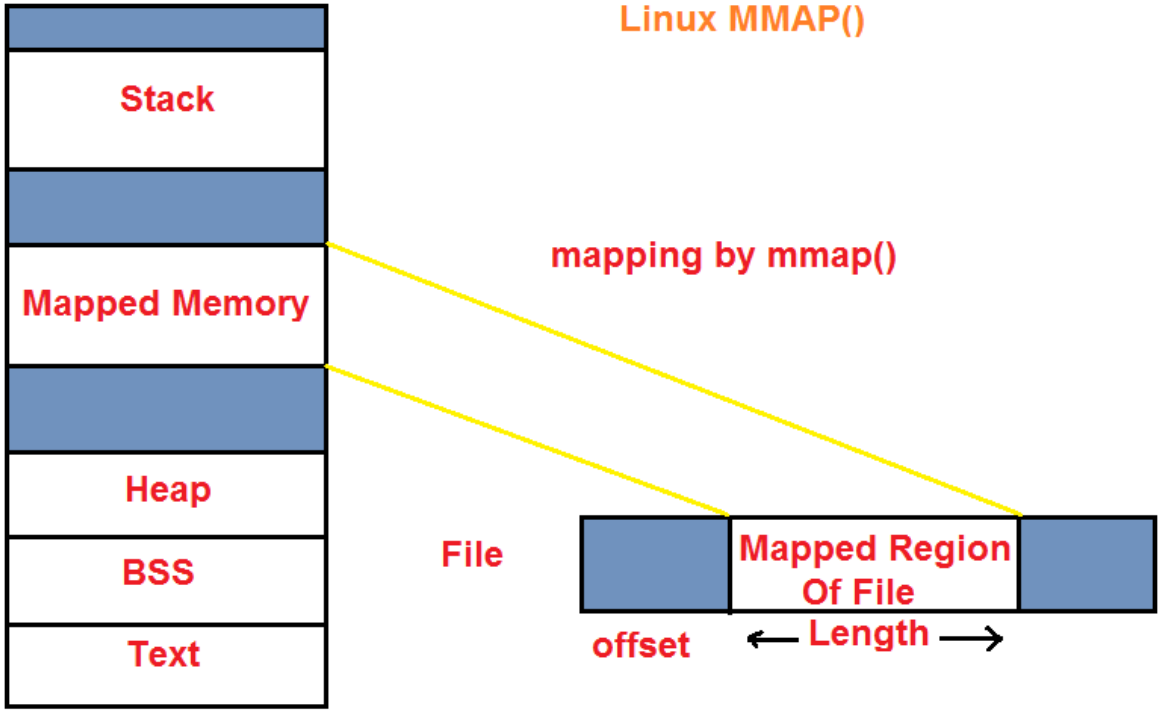
“__attribute__((__packed__))” is important and needed, otherwise, compiler optimizes for byte alignment

Hints on programming

mmap:

```
void *mmap(void *addr, size_t length, int prot,
int flags, int fd, off_t offset);
int munmap(void *addr, size_t length);
```

http://man7.org/linux/man-pages/man2/mmap.2.html



http://www.tutorialsdaddy.com/courses/linux-device-driver/lessons/mmap/

getting file descriptor

int open(const char *path, int oflags);

i.e., int fd = open("test.img", O_RONLY | O_WRONLY);

Value	Meaning
O_RDONLY	Open the file so that it is read only.
O_WRONLY	Open the file so that it is write only.
O_RDWR	Open the file so that it can be read from and written to.
O_APPEND	Append new information to the end of the file.
O_TRUNC	Initially clear all data from the file.
O_CREAT	If the file does not exist, create it. If the O_CREAT option is used, then you must include the third parameter.
O_EXCL	Combined with the O_CREAT option, it ensures that the caller *must* create the file. If the file already exists, the call will fail.

how to know the file size

fstat () int fstat (int fd, struct stat *buf);

```
struct stat {
        dev_t     st_dev;     /* ID of device containing file */
        ino_t     st_ino;     /* inode number */
        mode_t    st_mode;    /* protection */
        nlink_t   st_nlink;   /* number of hard links */
        uid_t     st_uid;     /* user ID of owner */
        gid_t     st_gid;     /* group ID of owner */
        dev_t     st_rdev;    /* device ID (if special file) */
        off_t     st_size;    /* total size, in bytes */
        blksize_t st_blksize; /* blocksize for file system I/O */
        blkcnt_t  st_blocks;  /* number of 512B blocks allocated */
        time_t    st_atime;   /* time of last access */
        time_t    st_mtime;   /* time of last modification */
        time_t    st_ctime;   /* time of last status change */
    };
```

Byte Ordering

strings are Endian-Independent

- **htonl/htons/ntohl/ntohs ()**
  - –#include <arpa/inet.h>
  - –uint32_t htonl(uint32_t hostlong);
    - The htonl() function converts the unsigned integer **hostlong** from host byte order to network byte order.
  - –uint16_t htons(uint16_t hostshort);
    - The htons() function converts the unsigned short integer **hostshort** from host byte order to network byte order.
  - –uint32_t ntohl(uint32_t netlong);
    - The ntohl() function converts the unsigned integer netlong from network byte order to host byte order.
  - – uint16_t ntohs(uint16_t netshort);
    - The ntohs() function converts the unsigned short integer netshort from network byte order to host byte order.

PART-3

1. Search for file name from directories (sub-directories)
2. Obtain file size (or #blocks) and starting block from the entry
3. Refer to FAT for finding the next block (may not be continuous)

PART-4

1. Get file attribute (size)
2. Find available blocks (FAT)
3. Update FAT and also dir entry
4. Copy (memcpy) file to the corresponding blocks.
   Need to refer to FAT when finding the next block.

you may use part-3 to copy it out again to check whether it was copied in successfully. Or use xxd.

xxd results

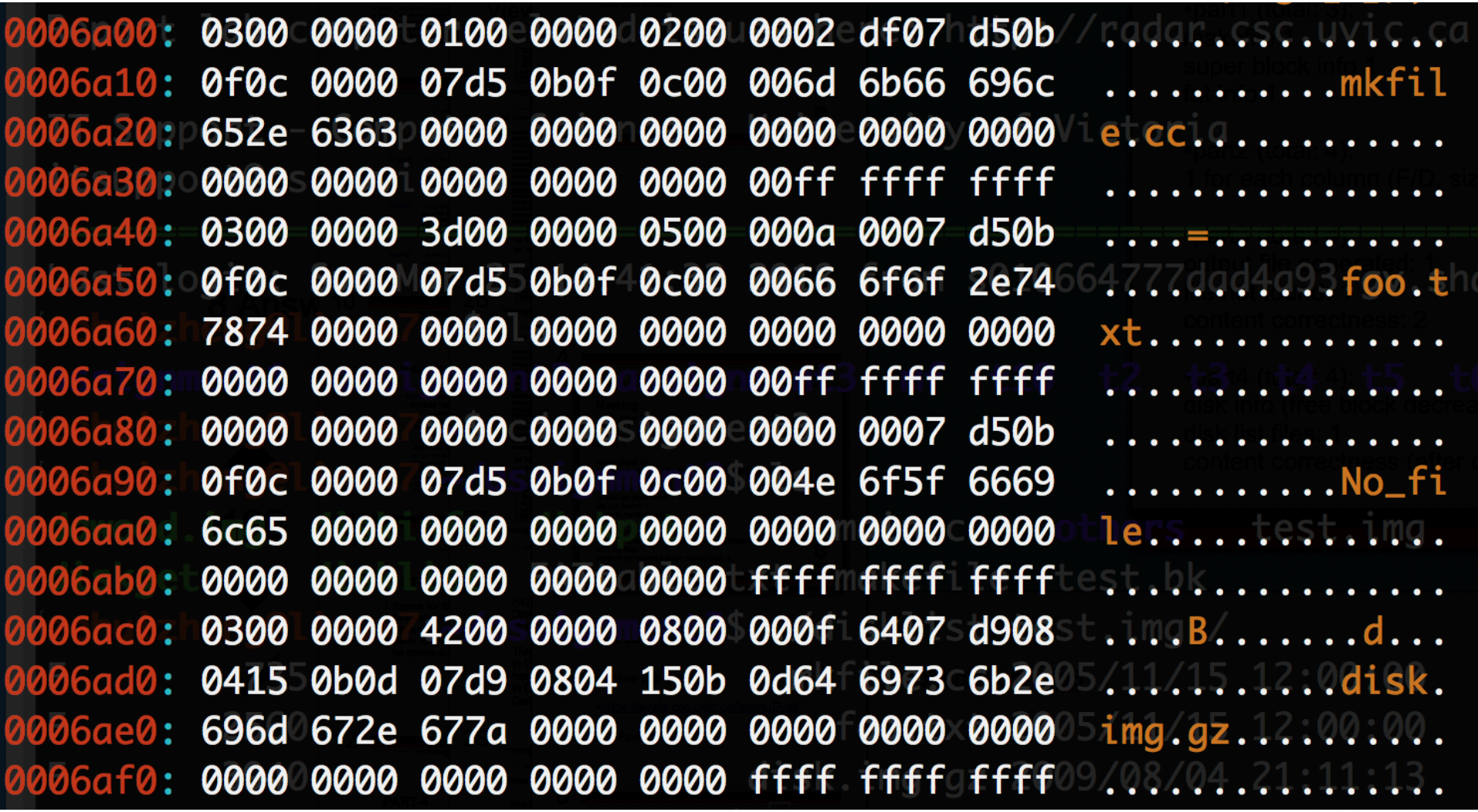
xxd test.img >> test.txt

```
// Super block
struct __attribute__((__packed__)) superblock_t {
  uint8_t   fs_id [8];
  uint16_t block_size;
  uint32_t file_system_block_count;
  uint32_t fat_start_block;
  uint32_t fat_block_count;
  uint32_t root_dir_start_block;
  uint32_t root_dir_block_count;
};
```

```
0000000: 4353 4333 3630 4653 0200 0000 1900 0000  CSC360FS........
0000010: 0002 0000 0032 0000 0035 0000 0008 0000  .....2...5......
0000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

FAT starts from block 2 and has 0x32 blocks
ROOT starts from block 0x35 and has 8 blocks


0000400: 0000 0001 0000 0034 0000 0001 0000 0001 ....4........
0000410: 0000 0001 0000 0001 0000 0001 0000 0001 ....
0000420: 0000 0001 0000 0001 0000 0001 0000 0001 ....
0000430: 0000 0001 0000 0001 0000 0001 0000 0001 ....
0000440: 0000 0001 0000 0001 0000 0001 0000 0001 ....
0000450: 0000 0001 0000 0001 0000 0001 0000 0001 ....
0000460: 0000 0001 0000 0001 0000 0001 0000 0001 ....
0000470: 0000 0001 0000 0001 0000 0001 0000 0001 ....
0000480: 0000 0001 0000 0043 0000 0001 0000 0001 ....C........
0000490: 0000 0001 0000 0001 0000 0001 0000 0001 ....
00004a0: 0000 0001 0000 0001 0000 0001 0000 0001 ....
00004b0: 0000 0001 0000 0001 0000 0001 0000 0001 ....
00004c0: 0000 0001 0000 0001 0000 0001 0000 0001 ....
00004d0: ffff ffff 0000 0036 0000 0037 0000 0038 ....6...7...8
00004e0: 0000 0039 0000 003a 0000 003b 0000 003c ...9...:...;...<
00004f0: ffff ffff 0000 003e 0000 003f 0000 0040 ....>...?...@
0000500: 0000 0041 ffff ffff 0000 0044 0000 0045 ...A.......D...E
0000510: 0000 0046 0000 0047 0000 0048 0000 0049 ...F...G...H...I
0000520: 0000 004a 0000 004b 0000 004c ffff ffff ...J...K...L....
0000530: 0000 004d 0000 004e 0000 004f ffff ffff ...M...N...O....
0000540: 0000 0000 0000 0000 0000 0000 0000 0000 ....


0006a00: 0300 0000 0100 0000 0200 0002 df07 d50b ..... uvic.ca
0006a10: 0f0c 0000 07d5 0b0f 0c00 006d 6b66 696c .....mkfil
0006a20: 652e 6363 0000 0000 0000 0000 0000 0000 e.cc.....
0006a30: 0000 0000 0000 0000 0000 00ff ffff ffff .....
0006a40: 0300 0000 3d00 0000 0500 000a 0007 d50b ....=.....
0006a50: 0f0c 0000 07d5 0b0f 0c00 0066 6f6f 2e74 .....foo.t
0006a60: 7874 0000 0000 0000 0000 0000 0000 0000 xt.....
0006a70: 0000 0000 0000 0000 0000 00ff ffff ffff .....
0006a80: 0000 0000 0000 0000 0000 0000 0007 d50b .....
0006a90: 0f0c 0000 07d5 0b0f 0c00 004e 6f5f 6669 .....No_fi
0006aa0: 6c65 0000 0000 0000 0000 0000 0000 0000 le.....test.img.
0006ab0: 0000 0000 0000 0000 0000 ffff ffff ffff test.bk.....
0006ac0: 0300 0000 4200 0000 0800 000f 6407 d908 ....B.....d...
0006ad0: 0415 0b0d 07d9 0804 150b 0d64 6973 6b2e .....disk.
0006ae0: 696d 672e 677a 0000 0000 0000 0000 0000 img.gz.....
0006af0: 0000 0000 0000 0000 0000 ffff ffff ffff .....

Marking

- part1 (total: 3):
  makefile 1
  super block info 1
  fat info 1

- part2 (total: 3):
  succeed one file: 2 (four columns each 0.5)
  succeed for all other files: 1

- part3 (total: 3):
  output file generated: 1
  file not found: 1
  content correctness: 1

- part4 (total: 3):
  disk info (free block decreased while allocated blocks increased): 1
  disk list files: 1
  content correctness (after copied out, use md5sum): 1

- part5 (total: 3):
  case 1: **1**; case 2: **1**; case 3: **0.5**; case 4: **0.5**

University of Victoria

What do you think about tutorials

Online Lab Evaluation

https://evals.csc.uvic.ca

By December 6

University of Victoria