Embedded Operating Systems

Agenda

- File IO System Calls
  - write()
  - lseek()
- Disk allocation mechanisms
- Free space management mechanisms
- Ext2/3 File System
- Journaling

File IO syscalls

write() syscall

- Make your notes

lseek() syscall

- Change the file position in open file table entry (struct file --> f_pos).
- And returns new file position (from the beginning of the file).
- Examples:
  - lseek(fd, 0, SEEK_SET);
    - filp->f_pos = 0;
  - lseek(fd, offset, SEEK_SET);
    - filp->f_pos = offset;
  - lseek(fd, 0, SEEK_END);
    - filp->f_pos = size; // file size (from the inode)
  - lseek(fd, offset, SEEK_END);
    - filp->f_pos = size + offset; // note that offset will be negative
  - lseek(fd, offset, SEEK_CUR);
    - filp->f_pos = filp->f_pos + offset;
- MCQ: Which of the following is equivalent to C library function rewind(fp)? A. lseek(fd, 0, SEEK_SET); B. lseek(fd, 0, SEEK_END); C. lseek(fd, 0, SEEK_CUR); D. None of these
- MCQ: Which of the following is equivalent to C library function ftell(fp)? A. lseek(fd, 0, SEEK_SET); B. lseek(fd, 0, SEEK_END); C. lseek(fd, 0, SEEK_CUR); D. None of these
- MCQ: How to find size of the file?

FileSystem Operations

- Directory Operations
  - Create directory -- mkdir()
  - Delete directory -- rmdir()
  - List directory -- opendir(), readdir(), closedir()
  - Change directory -- chdir()

- File Operations
  - Rename -- link() + unlink()
  - Delete -- unlink()
  - Links -- link(), symlink()
  - Get Info -- stat()
  - User Mask -- umask()
  - Change permissions -- chmod()
  - Change owner -- chown()

umask

- User file permission mask stored in fs_struct (associated with task_struct) of each process.
- File permission = file mode & ~umask, where file mode is arg3 of open.
- Umask of current process is changed using "umask" command, that calls umask() syscall.
- Umask of parent process is inherited to the child process.

File Allocation Mechanisms

- The file data blocks are allocated on the file system (data blocks region) on the disk.
- The data blocks can be allocated in various ways (depending on file system).
  - Contiguous allocation
  - Linked list allocation
  - Indexed allocation
- The information about allocated blocks is maintained into inode (FCB).

Contiguous Allocation

- Number of blocks required for the file are allocated contiguously.
- inode of the file contains starting block address and number of data blocks.
- Faster sequential and random access.
- Number of blocks required for the file may not be available contiguously. This is called as "External Fragmentation".
- To solve this problem, data blocks of the files can be shifted/moved so that max contiguous free space will be available. This is called as "defragmentation".
- File may not grow if blocks after the file are already allocated to some other file.

Linked Allocation

- Each data block of the file contains data/contents and address of next data block of that file.
- inode contains address of starting and ending data block.
- No external fragmentation, faster sequential access.
- Slower random access.
- e.g. FAT

Indexed Allocation

- A special data block contain addresses of data blocks of the file. This block is called as "index block".
- The address of index block is stored in the inode of the file.

- No extern Indexed Allocation
- A special data block contain addresses of data blocks of the file. This block is called as "index block".
- The address of index block is stored in the inode of the file.
- No external fragmentation, faster random and sequential access.
- File cannot grow beyond certain limit.

Free Space Management Mechanisms

- The free data blocks information is maintained in the super-block.
- Super block use some data structures to maintain this information.
    - Bit vector
    - Linked list
    - Grouping
    - Counting

File Systems

- File System is way of organizing files (data and metadata) on the disk.
- File systems differ in metadata (attributes), supported data block sizes, file system layout (boot block + super block + inode list + data blocks), disk allocation mechanisms, free space management mechanisms, etc.

UFS

- UNIX File System
- Modified indexed allocation
- Maximum file size = 16 GB (with 4 KB data block size)
- Inode has array of 13 members
    - 10 --> Direct data blocks
    - 11 --> Single indirect data blocks
    - 12 --> Double indirect data blocks
    - 13 --> Triple indirect data blocks

FAT

- Maximum file size = 4 GB.
- FAT -- File Allocation Table
    - A special data structure in the file system on the disk.
    - Array implementation of a linked list.
    - Each element in the FAT table, keep address of next block.
    - This table is used for disk allocation as well as for free space mgmt.
- Data blocks stores only data.
- FAT directory entries stores all info about file (no separate inode struct).
    - file name
    - size
    - attributes (read-only, hidden,)
    - start block
- Boot record (1 sector = 512 bytes) -- information about file system

    - label
    - FAT table information
    - etc.
- Do not support Journaling.

NTFS

- Windows platform
- Much sophisticated than FAT
- Use B-tree for disk allocation
- Max file size > 100 GB
- Also use Journaling mechanism
- Refer: wiki

Ext2

- Similar to UFS

Ext3

- Ext3 = Ext2 + Journaling