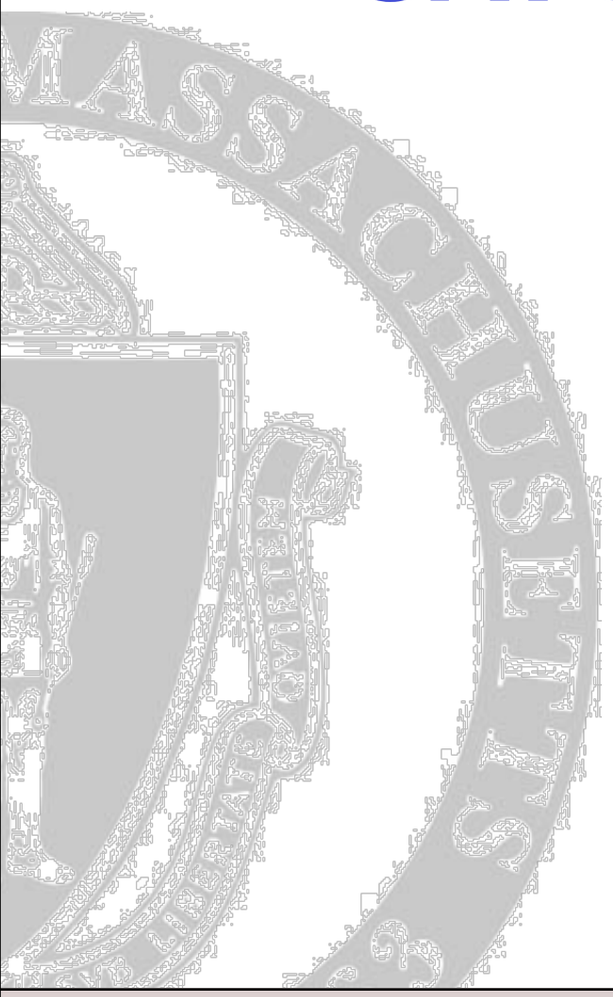
CMPSCI 377: Operating Systems



Lab 3

Goal

- Write a simple UNIX-like file system
  - disk size: ***128***KB.
  - block size: ***1***KB (***128*** blocks, ***1*** super block + ***127*** ordinary blocks).
  - one root directory, no subdirectories.
  - maximum of ***16*** files.
  - maximum size of a file: ***8*** blocks.
  - each file has a unique name (less than ***8*** characters)

Super Block

- free block list
  - ***128*** bytes, each byte corresponds to a block.
  - *i*-th byte ***0***: the *i*-th block is *free*.
  - *i*-th byte *not* ***0***: the *i*-th block is *in use*.

- 16 index nodes

```
char name[8]; //file name
int size;     // file size (in number of blocks)
int blockPointers[8]; // direct block pointers
int used;             // 0 => inode is free; 1 => in use
```

  - size for each node is 56 bytes

Basic Requirements

- create(char name[8], int size)
  - size is the number of blocks, it should not be larger than 8
- delete(char name[8])
- read(char name[8], int blockNum, char buf[1024])
- write(char name[8], int blockNum, char buf[1024])
- ls(void)

Implementation (Simulate the "Disk")

- use a file to simulate
  - For Java, use byte arrays (**byte**[]).
  - For C, use char arrays (char *).
  - You can refer to the programs provided in ***Getting Started*** part of the lab page.

Implementation (*Blocks Allocation Scheme*)

- Contiguous Allocation
  - OS allocates a contiguous chunk of free blocks when it creates a file.
  - Some *Pros* and *Cons*
    - easy to implement
    - random access is very quick
    - easily to cause fragmentation
    - hard to shrink and grow

Implementation (*Blocks Allocation Scheme*)

- Linked Files
  - In the file descriptor, keep a pointer to the first sector/block.
  - In each block, keep a pointer to the next block.
  - You can also use this scheme. Of course, you can revise the structure suggested in the lab page if it is necessary.

- Some *Pros* and *Cons*
  - no fragmentation issues
  - easy to shrink and grow
  - random access is very slow

Implementation (*Blocks Allocation Scheme*)

- Indexed files
  - OS keeps an array of block pointers for each file.
  - The user or OS must declare the maximum length of the file when it is created.
  - OS allocates an array to hold the pointers to all the blocks when it creates the file, but allocates the blocks only on demand.
  - OS fills in the pointers as it allocates blocks.

- The idea is suggested to implement your simple file system.

Implementation (*Blocks Allocation Scheme*)

- Indexed files
  - OS keeps an array of block pointers for each file.
  - The user or OS must declare the *maximum length* (a cons) of the file when it is created.
  - OS allocates an array to hold the pointers to all the blocks when it creates the file, but allocates the blocks only on demand.
  - OS fills in the pointers as it allocates blocks.

- The idea is suggested to implement your simple file system.

Extra Credit Question

- The fragmentation is most likely happen when you also use a *Contiguous Allocation* scheme.

- The only restriction is that your defragmenter should not use a memory buffer of more than 17 KB.

- I think if you finish this part, you could get grade more 100 for the lab. I will check this with Doctor Shenoy.

Remaining Quizzes

- *Page Replacement Algorithms* used in Demand Paging like *LRU*
  - This is one of the counted algorithms which helps make today's computer system better.
  - Maybe a bit hard to grasp, so it is suggested to review the contents after class.
- Replacement Policies for *Multiprogramming*. Why is different policy necessary?
- Topics in the Remaining Lectures.

Have a nice weekend