

CSE 120, Answers to HW3

Chapter 4

Q 23

This is the same type of problem as in the Midterm. The final answers are below:

FIFO: 6 page faults

LRU: 7 page faults.

Q 29

- (a) Page 2 will be replaced: it is in the lowest group $R = 0$, $M = 0$.
- (b) Page 3 will be replaced: it is the oldest page, loaded at time 110.
- (c) Page 1 will be replaced: it is the least recently used, referenced at time 265.
- (d) Page 2 will be replaced: it is the oldest page with $R = 0$.

Q37

Fragmentation is when the physical memory contains unusable holes after a series of allocation and de-allocation of chunks of memory.

With internal fragmentation these holes are due to allocation/de-allocation of memory within a single address space and occurs with paging systems. These are due to processes not filling up pages given to them entirely.

With external fragmentation on the other hand the holes are in between memory allocated to segments, each segment corresponding to a logical address space. These therefore occur in systems using pure segmentation.

Chapter 6

Q 4

Usually in file systems that have the notion of records, files are read or written in record units. Whenever such a system has variable record lengths it then becomes important to have that information available to the OS.

Q 7

Having a single **rename** operation allows a system to make the operation atomic if it so chooses. Consider a system with multiple users, where one is currently updating a file A, and another is trying to rename it. If this is done in two operations the first user's update may be lost, while a single **rename** operation could be made to allow the operation only when the first user is finished with the file (another option is to allow the single renaming to complete, i.e. just make the change in the inode as long as only the inode number and the disk location of the file is required for the first user to write it back when finished).

Q13

First let's see how long it takes to transfer an entire file from disk to memory or vice versa:

$$t = 8\text{KB}/(8 \cdot 10^3)\text{KB/s} = 10^{-3} \text{ s} = 1 \text{ msec.}$$

The total time to relocate a file is therefore:

$$2 \cdot (4 + 5 + t) = 20 \text{ msec.}$$

To compact half of the 16 GB disk, we are relocating about $8 \cdot 10^6 \text{KB}/8\text{KB}$ files, i.e. 10^6 files, so this would take about $10^6(10) \text{ msec} = 10000 \text{ s}$ or close to 3 hours.

Note:

Notice we used powers of 10 here for simplification, e.g. 1MB is really 2^{10}KB , not 1000KB.

Q20

Advantage of hard link:

Opening a hard linked file requires just as many disk accesses as any other file, while a symbolic link requires additional disk operations due to the indirections involved.

Advantage of symbolic link:

The symbolic link can reference a file on a different file system (the link can be followed over a network). A lesser advantage discussed in the book is below:

Since the original file's i-node is not shared while using a symbolic link, the OS book-keeping is more manageable when the original file is deleted.

Q22

- (a) 1111 1111 1111 0000
- (b) 1000 0001 1111 0000
- (c) 1111 1111 1111 1100
- (d) 1111 1110 0000 1100

Q36

First let's find out how many addresses can be held on a single block on disk:

Let this be $c = 2^{10}/4 = 2^8$ addresses.

The i-node can directly reference 10 blocks of the file, and the c blocks through the single indirect entry, c^2 blocks through the double indirect entry, and last c^3 blocks of the file can be referenced through the triple indirect entry.

The max file size is therefore:

$$(10 + c + c^2 + c^3)\text{KB.}$$

Q37

Every single directory on the path has to be opened and searched to find the inode number of the next item on the path. The inode has to then been fetched from disk to find the address of the corresponding file, and then the file has to be read from disk (for directories this requires a single disk operation since only one block is fetched).

The total number of disk operations is therefore:

$$1+2 + 2 + 2 + 2 + 1 = 10.$$

The actual detailed steps are below:

Retrieve disk block for /

Retrieve inode for /usr

Retrieve disk block for /usr

Retrieve inode for /usr/ast

Retrieve disk block for /usr/ast

Retrieve inode for /usr/ast/courses

Retrieve disk block for /usr/ast/courses

Retrieve inode for /usr/ast/courses/os

Retrieve disk block for /usr/ast/courses/os

Retrieve inode for /usr/ast/courses/os/handout.t **Done !**