Programming Assignment 3

Programming Assignment 3 consists of a single part, unlike the last two. This program does what the name implies, simulates a disk (Well, a Hard Drive). The simulation was written in C and compiles under the GNU89 Standard.

Program Information

The program only takes one argument, and that is a text file that is used to simulate files being either added, modified, or removed from the file system. It should be noted that files are not actually created, only simulated. The virtual disk has 10 MB (1024 * 1024 * 10 bytes) with certain sectors of size 512 bytes. As a result, there are 20480 total sectors to work with.

How it works

The program has 3 different structures. A disk struct, a sector struct, and a file struct. These structs are all related. The disk struct contains a vector of structs, which are all of the structs available in the disk. Also, in the disk struct, there is a vector that contains information about the files. The file struct contains a pointer to the first sector in the sector vector that stores that file. Each sector has "prev" and "next" pointers which point to the next sector in memory that has information about a file. They also have a bool that tells if they are allocated or not, and a pointer to the file struct. This allows constant time access to the beginning of the file from any sector since we can simply go to the file struct, and then go to the beginning of the file from there. The disk struct keeps track of sectors allocated and the total number of sectors altogether.

When allocating, it will start from the first sector, and look for any sectors which are open. Once it finds a sector that is open, it will store 512 bytes in there. Then it will look for the next open sector and store 512 more bytes in it. It will do this until all of the data in the file is stored in memory. So a file with 3300 bytes will take up 7 sectors. It should be noted that the last sector doesn't have to be completely filled with 512 bytes.

When modifying the file, it will try to store information into the final sector if it has space left. Afterwards, if it needs more sectors, it will allocate more sectors. If it needs less sectors, it will free them from memory.

When removing, it will go to the file struct and get access to the very first sector that has information on the file. It will then traverse through all sectors via the "next" pointer and release that sector back to the disk as free space.

Compilation

This application comes with a makefile which can be used to either compile or remove the program.

To compile disksim:

```
make
```

To remove disksim:

```
make clean
```

Notes

The program defines the disk attributes in "#define" tags. You can change this to make more space available or to make sectors larger. Also, you can enable DEBUG mode by modifying the defines. I decided not to include these as command line arguments since they are not meant to be changed by the end-user.

Synopsis

```
./disksim file
```

Command: ./disksim file_operations.txt

Output:

```
Allocated Sectors     : 40
Total Sectors         : 20480
Disk Utilisation Ratio: 0.00195312%
Number of Files       : 7
Free Space            : 9.98047 MB (10465280 bytes)
```

Comments

- I asked the professor about the number of sectors and he said that it doesn’t matter. The reason is because 10,000,000 / 512 is 19531.25, which isn’t possible. The actual definition of a Megabyte, to a computer, is 1048576 bytes. Thus, when multiplied by 10, holds the sector total of 20480 instead of 20000. The professor said this is okay.