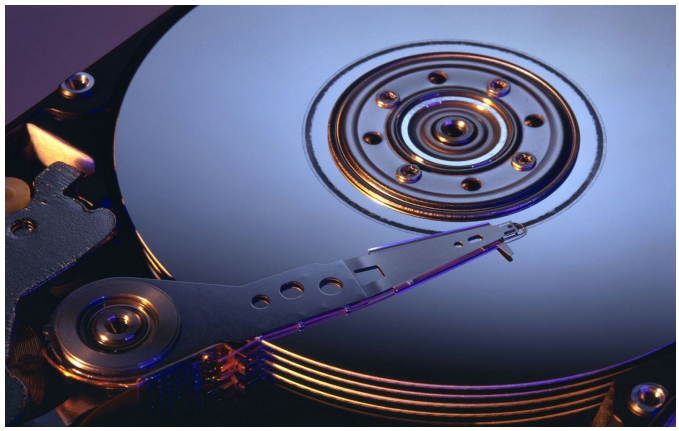
CS Games 2016

Operating Systems Competition

Participants	**2**
Workstations	**1**
Value	**5%**
Duration	**3 hours**

Hard-Drive



There are rumors, that in the past, pre-cataclysm studies were frquently conducted under the Dome. It was even once, elevated to the rank of craft. We called these craftsmen: **archaeologists**.

Long ago, a team of such men went on an expedition outside of the Dome. When they returned, they brought back a strange artefact, along with papers, damaged by the weight of time.

After a long and tedious translation effort, they found out that a great majority of the papers were uninportant: some periodics relating well-known facts, pre-alpha era advertisments, lewd illustrations, which were confiscated by the council, for authorized eyes only...

However, amongst these was one seemingly important piece of paper. One which described how a long-lost piece of machinery used to work: the Hard-Drive. After careful examination, they found out the artefact they brought back with them was one of these so-called Hard Drives. Ever since, work on the recuperation of their data had begun.

As of now, our scientists managed to extract data from these artefacts, and left us blobs containing said data. Along with them was a document, incrementally updated by generations of scientists on how the data is organised.

The only work remaining now, is to write a program to decipher and extract the data from these blobs...

Problem description

You must study the specification of the CFS (Coalition File System), and must develop a tool to extract the data from the blobs to a file hierarchy usable by the machines of the Dome.

Each blob you will be given is a dump from a hard-drive containing a single partition.
This specification will be given to you with this description.

Since the blobs[1] recovered from the Hard Drives are sometimes notoriously big (some say it's over 9000!), the program you are about to write must be extremely efficient.

[1] BLOB: Binary Large Object, a binary file.

Technical description

Because of this, you must program it using either C or C++.
The standard libraries of both languages are the only ones tolerated for this challenge.

We advise you to read, understand and know perfectly the file manipulation APIs from the C standard: `fopen`, `fread`, `fclose`, `ftell` and `fseek`, as they will be your best friends for this.

Program manual

Your program must work according to the following guidelines:

- ❏ The naming scheme is imposed for convenience during testing. You have to call your main entry point **CFSDecode.c/CFSDecode.cpp** (a penalty of 5% will be applied if you do not follow this convention)
- ❏ It must be used by taking the path to a blob as argument
- ❏ The output hierarchy must be a folder with the partition's name

Every blob we received were extracted from Hard Drives with a 512 bytes sector size. However, we have not yet finished extraction of data from all of them, and some remaining might be different.
As such, if your program could take a *-s* argument followed by another sector size, this would be greatly appreciated…

For instance, to change the sector size to 256, we'd type:

```
./CFSDecode -s 256 blob
```

You must send us the whole source code of your program.
We will proceed to compiling and thoroughly testing it on our servers on the blobs in our possession.

- ❏ The program must compile using either GCC or Clang.
- ❏ We enourage you to provide a Makefile with your code.

Evaluation

To evaluate your work, we will try them on a collection of blobs we have in our archives.
A subset of these blobs will be given to you for proper testing during developement.

We will take into account some key features of your program:

- Functionnality: a non-functioning program is of little use, therefore, its functionality is primordial - 80%
- Runtime efficiency: as we mentioned earlier, some blobs are big. Because of this, your program must efficiently work on large inputs - 15%
- Code quality - 5%, for each submission, we will evaluate the quality of your code on these criteria:
  - Naming: each of your functions. classes, variables muse have a cohesive and clear naming scheme for better maintenance of the winning team's program
  - Factorization: under the Dome, we consider plagiarism as theft. And we do not like thiefs… Copy-pasting is theft, factorise your code.
  - Spacing: make your code clearer by adding whitespaces and line feeds appropriately.
  - Documentation: good code is nicely documented
  - **WTFPM**: the final evaluation criteria, the most important one, is and always will be the What-The-Fuck-Per-Minute metric