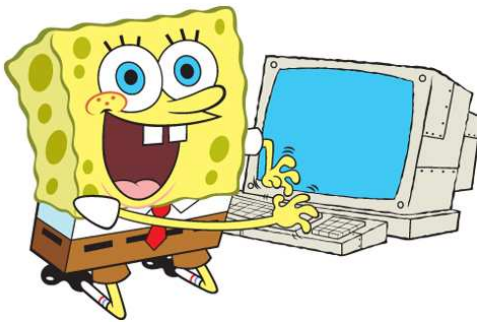
CS 366
Assignment #3
D is for Database

Due 2/22/17, in class



Problem Statement

The sponge has taken over Facebook and now needs a program to maintain his employee database.

There are two types of employees: salaried and hourly. All employees have an integer employee_id (eid), a no more than 20 character first name, and a no more than 30 character last name. In addition, salaried employees have an integer salary (dollars per year), while hourly employees have an integer wage and an integer number of hours worked.

Goal

The goals of this assignment are 1) to build a database that exploits random file access, and 2) provide an opportunity to use C's struct and union constructs.

Analysis (What is the client's problem)

[Software Engineers first construct an analysis (what is it) and then a design (how will it be done).]

The sponge needs the database to support the following operations:

`-n` create a new data base,
`-c` create an employee (given an eid, first name, last name, and (salary or (wage and hours worked))),
`-d` display an employee (given their eid), and
`-u` update an employee's salary or hours worked (given their eid and the new salary or hours worked).

Which operation and all the other values should be provided to the program as command-line arguments (e.g., `mydb -d 42`).

Design Notes (How will this problem be solved)

For a small database it is possible to read the entire database into memory and then work with it. For larger databases this is not feasible and thus database records are read into memory only when needed. Fast data structures for doing this include the B-tree, which is an on-disk variant of a hash table. For this assignment we will use a simpler data structure: an on-disk array. To make this work each record written to the file must be the same size. This allows the use of a variant of the basic array access equation: address of element = base address of array + index × element size. In our case the base address of the array will be the start of the file making the offset of the record at index i, $i \times$ record length.

What to hand in

(1) A well-formatted 2-up printout of your source code. You **must** use `a2ps` **after** removing all the tabs from your code.

(2) A GitHub repository that includes your analysis, design, build plan, test plan, and source code.
- Include a `Readme.md` file in your repository.
- Yes, `git log` will tell me if you Hail Mary it.
- It is bad (for your grade) if you commit derivable files (e.g., your `.o` files).
- I expect to pull your code, run make, and then run my test script.

Assignment Notes

(1) Assume that all user input is valid (e.g., no salaries of "`lots`").

(2) Assume that reading the entire database into memory is infeasible.

(3) (Re)Read the two restrictions found on Page 912.

(4) The repository includes some support code that you may use and a `Makefile` that you **must** use!

(5) clone https://classroom.github.com/assignment-invitations/fc84e6d7632c34a79b0a356e2bd88323