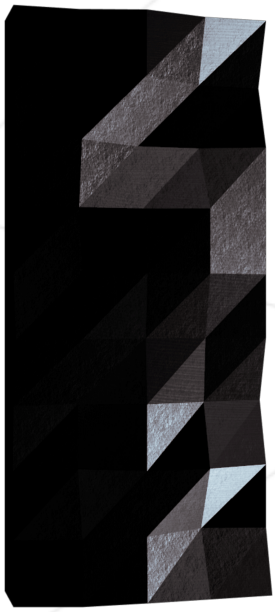

21sh

The vengence's return

Staff pedago pedago@42.fr

Summary: You'll have to start from your minishell and make it stronger to get little by littler closer to a real functionnal shell. You'll add couple of features such as multi-commande management, redirections as well as line edition that will allow you to use arrows for example.

Contents

Chapter I

Foreword

I.1 How to Make Perfect Cheesecake

Serves 8 to 10

I.2 What You Need

I.2.1 Ingredients

For the cheesecake:

- 2 pounds cream cheese
- 1 cup sugar
- 1 tablespoon cornstarch, or 2 tablespoons all-purpose flour (optional)
- 1/8 teaspoon salt
- 1/2 cup sour cream
- 2 teaspoons lemon juice (optional)
- 1 teaspoon vanilla extract
- 3 large eggs
- 1 large egg yolk

For the crust:

- 12 whole graham cracker rectangles (6 ounces)
- 5 tablespoons butter, plus extra to grease the pan

I.2.2 Equipment

- 9-inch or 10-inch springform pan
- Aluminum foil
- Food processor
- Stand mixer or handheld mixer
- Measuring cups and spoons
- Spatula
- Roasting pan or other dish big enough to hold the springform pan

I.2.3 Instructions

- Preheat the oven and warm the cream cheese: Preheat the oven to 350°F with a rack in the lower-middle position. Take the blocks of cream cheese out of their boxes and let them warm on the counter while you prepare the crust, about 30 minutes.
- Rub the pan with butter: Use your fingers to rub a small pat of butter all over the bottom and sides of the pan.
- Wrap the pan in foil: Cut two large pieces of foil and lay them on your work surface in a cross. Set the springform pan in the middle and fold the edges of the foil up around the sides of the pan. The foil gives you extra protection against water getting into the pan during the water bath step.
- Prepare the crust: Crush the graham crackers in a food processor (or in a bag using a rolling pin) until they form fine crumbs — you should have 1 1/2 to 2 cups. Melt 5 tablespoons of butter in the microwave or on the stovetop and mix this into the graham cracker crumbs. The mixture should look like wet sand and hold together in a clump when you press it in your fist. If not, add extra tablespoons of water (one a time) until the mixture holds together. Transfer it into the springform pan and use the bottom of a glass to press it evenly into the bottom. (For step-by-step instructions of this step, see How to Make a Graham Cracker Crust.)
- Bake the crust: Place the crust in the oven (be careful not to tear the foil). Bake for 8 to 10 minutes until the crust is fragrant and just starting to brown around the edges. Let the crust cool on a cooling rack while you prepare the filling.
- Mix the cream cheese, sugar, cornstarch, and salt: Combine the warmed cream cheese, sugar, cornstarch, and salt in the bowl of a stand mixer fitted with a paddle attachment (or use a handheld mixer). Mix on medium-low speed until the mixture is creamy, like thick frosting, and no lumps of cream cheese remain. Scrape down the beater and the sides of the bowl with a spatula.

- Mix in the sour cream, lemon juice, and vanilla: Add the sour cream, lemon juice, and vanilla to the bowl and beat on medium-low speed until combined and creamy. Scrape down the beater and sides of the bowl with a spatula.

- Mix in the eggs and yolk one at a time: With the mixer on medium-low speed, beat in the eggs and the yolk one at a time. Wait until the previous egg is just barely mixed into the batter before adding the next one. At first, the mixture will look clumpy and broken, but it will come together as the eggs are worked in.

- Stir a few times by hand: Scrape down the beater and sides of the bowl with a spatula. Stir the whole batter a few times by hand, being sure to scrape the bottom of the bowl, to make sure everything is incorporated. The finished batter should be thick, creamy, and silky. Don't worry if you see a few specks of un-mixed cream cheese here and there; they will melt into the batter during baking and won't affect the finished cheesecake.

- Pour the batter over the cooled crust: Check to make sure the crust and the sides of the pan are cool — if they're cool enough to comfortably touch, you can go on. Pour the batter over the cooled crust and spread it into an even layer against the sides of the pan.

- Transfer the pan to the water bath: Transfer the pan to a roasting pan or other baking dish big enough to hold it. Bring a few cups of water to a boil and pour the water into the roasting pan, being careful not to splash any water onto the cheesecake. Fill the pan to about an inch, or just below the lowest edge of foil.

- Bake the cheesecake: Bake the cheesecake at 350°F for 50 to 60 minutes. Cakes baked in a 10-inch pan will usually cook in 50 to 55 minutes; cakes in a 9-inch pan will cook in 55 to 60 minutes. The cheesecake is done when the outer two to three inches look slightly puffed and set, but the inner circle still jiggles (like Jell-o) when you gently shake the pan. Some spots of toasted golden color are fine, but if you see any cracks starting to form, move on to the next step right away.

- Cool the cheesecake in the oven: Turn off the oven and crack the door open. Let the cheesecake cool slowly for one hour.

- Run a knife around the edge of the cake and cool the cake completely: After an hour, remove the cheesecake from the oven and from the water bath, unwrap the foil, and transfer it to a cooling rack. Run a thin-bladed knife around the edge of the cake to make sure it's not sticking to the sides (which can cause cracks as it cools). Let the cheesecake cool completely on the rack.

- Chill the cheesecake for four hours in the refrigerator: Chill the cheesecake, uncovered, for at least four hours or up to three days in the refrigerator. This step is crucial for letting the cheesecake set and achieving perfect cheesecake texture — don't rush it.

- Top the cheesecake and serve: Take the cheesecake out of the fridge about 30 minutes before you plan to serve. Unmold the cake and top the cheesecake just before serving. You can serve the cake right from the bottom of the springform pan, or use a large off-set spatula to gently unstick the crust from the pan and transfer it to a serving platter. Leftovers will keep, uncovered and refrigerated, for several days.

I.2.4 Ideas for Topping Your Cheesecake

- Spread the top with a thin layer of sour cream or whipped cream
- Pour soft chocolate ganache over the top of the cheesecake
- Add chopped fresh fruit, either all on its own or tossed with a fruit syrup
- Warm some peanut butter with a little cream to form a sauce and pour this over the cheesecake

Chapter II

Introduction

Thanks to the `Minishell` project, you discovered a part of what is behind the scene of a shell, such as the one you use everyday. And more specifically the process' synchronisation creation with functions like `fork` and `wait`.

The `21sh` project will make you go further by adding, amoungst other things, inter-process communication using pipes.

You'll discover, or rediscover if you worked on the `ft_select` project, `termcaps`. This library will allow you to add to your shell a line edition feature. You'll then be able to edit a typo made on your command without having to retype it completely as well as repeat a previous command using history. Of course you'll have to code those features. Good news is `termcaps` will help you do it, bad news is it'll not do it for you!

Chapter III

Objectives

`Unix` programmation is great, The school's 3 shell projects allow you to discover a big part of the system's API and it can only be good for you.

However, the shell projects are commands interpreter above all and initiate you to a very important part of IT: compilation. Interpreter are programs that read and execute other programs, unlike compilers which translate other programs into other langages. Interpreters and compilers have more in common than they have differences though: whether it comes to executing or translating a program, first one need to understand the program itself, and be able to detect and reject malformed programs.

You probably know this already, but the set of commands we can send to a shell form a langage. This langage has lexical, syntactic and semantics rules, that will have to be respected by your 21sh by going through a set of very precise steps well documented on the internet. For example the "2.10 Shell Grammar" section of this document.

The key to a successful `21sh`, and `42sh` later on is a clear and well managed code organisation. Be asured that a simple space based `split` on your command line will not do the trick here. To avoid losing time, consider this solution as a one way ticket to disaster.

Here are couple of key words that i suggest you to properly understand: "lexical analysis", "lexer", "syntactic analysis", "parser", "semantics analysis","interpreter", and of course "abstract syntax tree" (or "AST").

Chapter IV

General Instructions

- This project will be evaluated by humans only. You're allowed to organise and name your files as you see fit, but you must follow the following rules.
- The executable file must be named `21sh`.
- Your `Makefile` must compile the project and must contain the usual rules. It must recompile and re-link the program only if necessary.
- If you are clever, you will use your library for your `21sh`. Submit also your folder `libft` including its own `Makefile` at the root of your repository. Your `Makefile` will have to compile the library, and then compile your project.
- Your project must be written in accordance with the Norm. Only norminette is authoritative.
- You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).
- Your program cannot have memory leaks.
- You'll have to submit a file called `author` containing your username followed by a '\n' at the root of your repository.

```
$>cat -e author
xlogin$
```

- Within the mandatory part, you are allowed to use only the following libc functions:
  - `malloc`, `free`
  - `access`
  - `open`, `close`, `read`, `write`
  - `opendir`, `readdir`, `closedir`
  - `getcwd`, `chdir`
  - `stat`, `lstat`, `fstat`
  - `fork`, `execve`
  - `wait`, `waitpid`, `wait3`, `wait4`
  - `signal`, `kill`
  - `exit`
  - `pipe`
  - `dup`, `dup2`
  - `isatty`, `ttyname`, `ttyslot`
  - `ioctl`
  - `getenv`
  - `tcsetattr`, `tcgetattr`
  - `tgetent`
  - `tgetflag`
  - `tgetnum`
  - `tgetstr`
  - `tgoto`
  - `tputs`
- You are allowed to use other functions or other librairies to complete the bonus part as long as their use is justified during your defense. Be smart!
- You can ask your questions on the forum, on slack...

Chapter V

Mandatory part

To begin with, every `minishell` features are implicitly part of the `21sh` mandatory part. Furthermore you'll have to add the following new features:

- A line edition feature using the `termcaps` library. Check the following description below.
- The `ctrl+D` et `ctrl+C` keys combination features for line edition and process execution.
- The "`;`" command line separator
- Pipes "`|`"
- The 4 following redirections "`<`", "`>`", "`<<`" et "`>>`"
- File descriptor aggregation, for example to close the standard error output:

```
$> ls
riri
$> rm riri; cat riri 2>&-
```

Here is a representative example of commands your `21sh` must be able to execute correctly:

```
$> mkdir test ; cd test ; ls -a ; ls | cat | wc -c > fifi ; cat fifi
.   ..
      5
$>
```

Regarding the line edition, you must at least manage the following features. The keys to be used are used as examples, you're free to use other ones as long as your shell remains logical and intuitive. The person evaluating you will decide what's logical and intuitive, so be careful not to get carried away with creativity.

- Edit the line where the cursor is located.

- Move the cursor left and right to be able to edit the line at a specific location. Obviously new characters have to be inserted between the existing ones similarly to a classic shell.

- Use up and down arrows to navigate through the command history which we will then be able to edit if we feel like it (the line, not the history)

- Cut, copy, and/or paste all or part of a line using the key sequence you prefer.

- Move directly by word towards the left or the right using `ctrl+LEFT` and `ctrl+RIGHT` or any other reasonable combination of keys.

- Go directly to the beginning or the end of a line by pressing `home` and `end`.

- Write AND edit a command over a few lines. In that case, we would love that `ctrl+UP` and `ctrl+DOWN` allow to go from one line to another in the command while remaining in the same column or otherwise the most appropriate column.

- Completely manage quotes and double quotes, even on several lines (expansions excluded).

Chapter VI

Bonus part

We will look at your bonuses if and only if your mandatory part is EXCELLENT. This means that your must complete the mandatory part, beginning to end, and your error management must be flawless, even in cases of twisted or bad usage. If that's not the case, your bonuses will be totally IGNORED.

There are quite a few features that will appear in 42sh. Here is however a list of bonuses that you can implement immediately:

- Search through history using `ctrl+R`
- Implement a hash table for binary files
- Simple or advanced completion using `tab`.
- `Emacs` and/or `Vim` binding mode freely activable or deactivable.
- Syntactic shell coloration freely activable or deactivable.
- Any additional bonus that you will feel useful.

Chapter VII

Submission and peer evaluation

Submit your work on your `GiT` repository as usual. Only the work on your repository will be graded.

Good luck to all and don't forget your author file!