Project description:

May it be a file, stdin, or even later a network connection, you will always need a way to read content line by line. It is time to start working on this function, which will be essential for your future projects.

Topics

1. Introduction
2. Dealing with files
3. Understanding static
4. Divide and conquering

Introduction

In case this is your first spark session then a warm welcome!
You might ask yourself: "What is a spark session? What is the point?"

A spark session simply serves as a way of achieving that initial 'spark' for solving on how to complete a certain project. It's goal is not a step-by-step guide / tutorial on how to complete the project.

They main goal is to teach you self sufficiency by showing and giving you tasks that encourage peer-to-peer interactions as well as being able to find the answer on your own.

To achieve this, a spark session composes of bite sized objectives that should help you give an easy footing into solving certain problems. Additionally, spark sessions are a hand-on approach with peer-to-peer in mind, so you will form in groups of 2-3 to work together and solve the following problems.

During the spark-session the moderator's purpose is to clear up confusion and work with the students in answering some but not all questions. They are much like C.A.T's in a way that they can help you with some technical stuff but not with actually solving the problems.

NOTE: When given an exercise each person should do their own work, however work together to solve problems!

Handling files

Before we start tackling our root problem, we first need to make sure we understand what were dealing with.

Handling files is one of the most common tasks in any program, you will and want to use it to store any sort of data for caching stuff or for reading say a configuration file. So, being able to understand how to use, access, read and write to files is crucial step!

- With your peers find the answer to the follwing:
  - Look up the following functions:
    - `open` , `close` , `read` , `write` .
    - What do they do ? How to they behave
    - What even is a *syscall* ?
  - In regards to `open` , identify the different flags and what they do.
  - What is a `fd` ? How they work under the hood ?
    - What happens when you pass an `fd` to `close` ?
    - Is it possible to lose a `fd` ?

Now that we are a bit familiar with how to work with `fd` 's, lets put that knowledge to use!

- Implement the following:
  - Write the function `ft_readfile` that will read the contents of an entire file.
    - Make sure to handle errors from the *syscalls* you use!
  - It only has a single parameter, `int fd` .
  - On success it returns the contents of the file as an *allocated* string.
  - On failure it simply returns `NULL` .

Understanding static

Now that we got files out of our way lets go and focus on static next. In `get_next_line` you are aksed to use the static keyword, now on its own it might not sound like much. But the keyword itself literally does just that what its called.

At first the static keyword might seem a bit archaic but you will quite quickly find it to be very straight forward.
So time to figure it all out, work with you peers to complete the following.

- The task is simple:
  - Use the internet to find the definition of a static variable
  - Identify its unique characteristics depending on how it is used.

Now that you hopefully understand the keyword…

- Implement the following:

    - Write a function that declares a `static int`, its initial value will be 0.
        - Afterwards in increments this value by 1.
        - Prints out the value to `stdout` using `write`.
    - Execute this function 7 times.
    - Take note of the behaviour of this function.

Divide and conquering

It might seem difficult to imagine how someone could implement a complex projects.
"Where do I even start with this?" is the very first common question one asks themselves when trying do a project.

For instance how would one create a graphical program that displays a 3D rotating cube? At first you have no idea where to even look for information. However if you split up your tasks / problems into smaller and smaller pieces it becomes much more easier to see what it takes to complete the bigger picture.

This section does not ask for much but it will be a bit of brain storming with your peers to figure out what it takes to complete `get_next_line`.

So to put this into practice, with your peers:

- Work out how you would split up the main problem of implementing `get_next_line`
    - Which functions do you need ?
    - How can we use static to our advantage ?
    - …