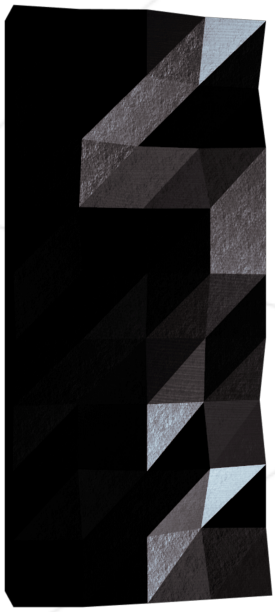

Minitalk

Summary:
The purpose of this project is to code a small data exchange program using UNIX signals.

Version: 3.01

Contents

Chapter I

Foreword

The cis-3-Hexen-1-ol, also known as (Z)-3-hexen-1-ol and leaf alcohol, is a colorless oily liquid with an intense grassy-green odor of freshly cut green grass and leaves.

It is produced in small amounts by most plants, and it acts as an attractant to many predatory insects. Cis-3-Hexen-1-ol is a very important aroma compound that is used in fruit and vegetable flavors and in perfumes.

The annual production is approximately 30 metric tons.

Chapter II

Common Instructions

- Your project must be written in C.
- Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check and you will receive a `0` if there is a norm error inside.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a `0` during the evaluation.
- All heap allocated memory space must be properly freed when necessary. No leaks will be tolerated.
- If the subject requires it, you must submit a `Makefile` which will compile your source files to the required output with the flags `-Wall`, `-Wextra` and `-Werror`, use cc, and your Makefile must not relink.
- Your `Makefile` must at least contain the rules `$(NAME)`, `all`, `clean`, `fclean` and `re`.
- To turn in bonuses to your project, you must include a rule `bonus` to your Makefile, which will add all the various headers, libraries or functions that are forbidden on the main part of the project. Bonuses must be in a different file `_bonus.{c/h}` if the subject does not specify anything else. Mandatory and bonus part evaluation is done separately.
- If your project allows you to use your `libft`, you must copy its sources and its associated `Makefile` in a `libft` folder with its associated Makefile. Your project's `Makefile` must compile the library by using its `Makefile`, then compile the project.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done

after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter III

Project instructions

- Name your executable files `client` and `server`.
- You must submit a `Makefile` that compiles your source files. It must not perform unnecessary relinking.
- You are allowed to use your `libft`.
- You have to handle errors thoroughly. Under no circumstances should your program quit unexpectedly (segmentation fault, bus error, double free, and so forth).
- Your program must be free of **memory leaks**.
- You are allowed to use **one global variable per program** (one for the client and one for the server), but its usage must be justified.
- To complete the mandatory part, you are allowed to use the following functions:
  - `write`
  - `ft_printf` and any equivalent YOU coded
  - `signal`
  - `sigemptyset`
  - `sigaddset`
  - `sigaction`
  - `kill`
  - `getpid`
  - `malloc`
  - `free`
  - `pause`
  - `sleep`
  - `usleep`
  - `exit`

Chapter IV

Mandatory Part

You must create a communication program in the form of a **client** and a **server**.

- The server must be started first. Upon launch, it must print its `PID`.
- The client takes two parameters:
  - The server `PID`.
  - The string to send.
- The client must send the specified string to the server.
  Once received, the server must print it.
- The server must display the string without delay. If it seems slow, it is likely too slow.

`If displaying 100 characters takes 1 second, the program is too slow!`

- Your server should be able to receive strings from several clients in a row without needing to restart.
- Communication between the client and server must `exclusively` use UNIX signals.
- You can only use these two signals: `SIGUSR1` and `SIGUSR2`.

`Linux system does NOT queue signals when you already have pending signals of this type! Bonus time?`

Chapter V

Bonus part

Bonus list:

- The server must acknowledge each received message by sending a signal to the client.
- Unicode characters support!

The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been fully completed and functions without any errors. If you have not met ALL the mandatory requirements, the bonus part will not be evaluated.

Chapter VI

Submission and peer-evaluation

Submit your assignment in your `Git` repository as usual. Only the content within your repository will be assessed during the defense. Don't hesitate to double-check the names of your files to ensure they are correct.

file.bfe:VAAe8ElCrUAbXivzOueiIpv/u/ia9PL50+HI+8/bgPKLESHlp
tPLpuOPW9zWV/LwDVaOqCRCGu6Gopk1XOi6Kn7t