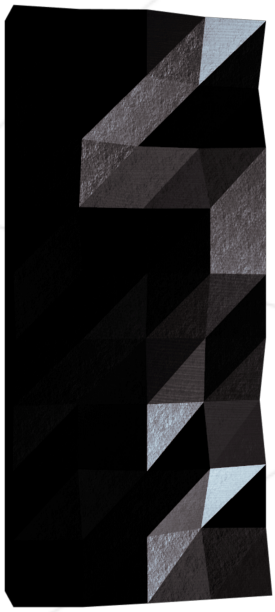

Minitalk

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

Version: 2

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 yearly production is about 30 tonnes.

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, librairies or functions that are forbidden on the main part of the project. Bonuses must be in a different file `_bonus.{c/h}`. 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.

- The executable files must be named `client` and `server`.
- You have to handle errors sensitively. In no way can your program quit unexpectedly (Segmentation fault, bus error, double free, etc).
- Your program cannot have memory leaks.
- You can use one global variable but it must be justified.
- On 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 III

Mandatory Part

- You must create a communication program in the form of a client and server.
- The server must be launched first, and after being launched it must display its PID.
- The client will take as parameters:
  - The server PID.
  - The string that should be sent.
- The client must communicate the string passed as a parameter to the server. Once the string has been received, the server must display it.
- Communication between your programs should ONLY be done using UNIX signals.
- The server must be able to display the string pretty quickly. By quickly we mean that if you think it is too long, then it is probably too long (hint: 1 second for 100 characters is COLOSSAL)
- Your server should be able to receive strings from several clients in a row, without needing to be restarted.
- You can only use the two signals `SIGUSR1` and `SIGUSR2`.

`Linux system do NOT queue signals when you already have pending signal of this type!  bonus time?`

Chapter IV

Bonus

- The server confirms every signal received by sending a signal to the client.
- support Unicode characters!

Chapter V

Submission and peer correction

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