

# Minitalk

Resumen: El propósito de este proyecto es crear un pequeño programa de intercambio de datos utilizando señales UNIX.

Versión: 2

# Índice general

I.	Avance	2
II.	Instrucciones generales	3
III.	Instrucciones específicas	5
IV.	Parte obligatoria	6
V.	Extras	7
VI.	Entrega y evaluación de pares	8

# Capítulo I

#### Avance

El cis-3-Hexen-1-ol, también conocido como (Z)-3-hexen-1-ol y alcohol de hojas, es un líquido aceitoso incoloro con un olor intenso a hierba y hojas verdes recién cortadas.

Se produce en pequeñas cantidades por la mayoría de las plantas y actúa como un atrayente para muchos insectos depredadores. El cis-3-hexen-1-ol es un compuesto de aroma muy importante que se utiliza en sabores de frutas y verduras y en perfumes.

La producción anual es de alrededor de 30 toneladas.

#### Capítulo II

#### Instrucciones generales

- Tu proyecto deberá estar escrito en C.
- Tu proyecto debe estar escrito siguiendo la Norma. Si tienes archivos o funciones adicionales, estas están incluidas en la verificación de la Norma y tendrás un 0 si hay algún error de norma en cualquiera de ellos.
- Tus funciones no deben terminar de forma inesperada (segfault, bus error, double free, etc) ni tener comportamientos indefinidos. Si esto pasa tu proyecto será considerado no funcional y recibirás un 0 durante la evaluación.
- Toda la memoria asignada en el heap deberá liberarse adecuadamente cuando sea necesario. No se permitirán leaks de memoria.
- Si el enunciado lo requiere, deberás entregar un Makefile que compilará tus archivos fuente al output requerido con las flags -Wall, -Werror y -Wextra y por supuesto tu Makefile no debe hacer relink.
- Tu Makefile debe contener al menos las normas \$(NAME), all, clean, fclean y re.
- Para entregar los bonus de tu proyecto deberás incluir una regla bonus en tu Makefile, en la que añadirás todos los headers, librerías o funciones que estén prohibidas en la parte principal del proyecto. Los bonus deben estar en archivos distintos \_bonus.{c/h}. La parte obligatoria y los bonus se evalúan por separado.
- Si tu proyecto permite el uso de la libft, deberás copiar su fuente y sus Makefile asociados en un directorio libft con su correspondiente Makefile. El Makefile de tu proyecto debe compilar primero la librería utilizando su Makefile, y después compilar el proyecto.
- Te recomendamos crear programas de prueba para tu proyecto, aunque este trabajo no será entregado ni evaluado. Te dará la oportunidad de verificar que tu programa funciona correctamente durante tu evaluación y la de otros compañeros. Y sí, tienes permitido utilizar estas pruebas durante tu evaluación o la de otros compañeros.
- Entrega tu trabajo en tu repositorio Git asignado. Solo el trabajo de tu repositorio Git será evaluado. Si Deepthought evalúa tu trabajo, lo hará después de tus com-

~ ~		* 1 D
pañeros. Si se en terminado.	ncuentra un error durante la evalu	uación de Deepthought, esta habrá

#### Capítulo III

#### Instrucciones específicas

- Los ejecutables deben llamarse client y server.
- Deberás proporcionar un archivo Makefile, que compilará los archivos fuente de tu programa. No deberá hacer relink.
- Puedes usar tu libft.
- Debes gestionar los errores con cuidado. Bajo ninguna circunstancia tu programa puede terminar inesperadamente (segfault, bus error, double free, etc).
- Tu programa no puede tener leaks de memoria.
- Puedes utilizar una variable global por programa (una para el cliente y otra para el servidor), pero tendrás que justificar su uso.
- En la parte obligatoria se te permite utilizar las siguientes funciones:
  - o write
  - o ft\_printf y cualquier equivalente que TÚ hayas programado
  - o signal
  - sigemptyset
  - o sigaddset
  - o sigaction
  - o kill
  - o getpid
  - o malloc
  - o free
  - o pause
  - o sleep
  - o usleep
  - $\circ$  exit

### Capítulo IV

#### Parte obligatoria

- Debes crear un programa de comunicación en la forma de un cliente y un servidor.
- El servidor debe lanzarse primero, tras lanzarse debe mostrar su PID.
- El cliente tomará como parámetros:
  - El PID del servidor.
  - o La string que debería mandarse.
- El cliente debe comunicar la string pasada como parámetro al servidor. Una vez la string se haya recibido, el servidor debe mostrarla.
- El servidor debe ser capaz de mostrar la string suficientemente rápido. Por rápido queremos decir que si piensas que es está tardando mucho, probablemente es que está tardando demasiado.



1 segundo para mostrar 100 caracteres es ¡muchíiiiisimo!

- Tu servidor debe poder recibir strings de distintos clientes consecutivamente, sin necesidad de reiniciar.
- La comunicación entre tu cliente y tu servidor debe hacerse **SOLO** utilizando señales UNIX.
- Solo puedes utilizar estas dos señales: SIGUSR1 y SIGUSR2.



Linux no pone en cola señales cuando ya tienes señales pendientes de este tipo. ¿Hora de hacer bonus?

# Capítulo V

#### Extras

#### Lista de bonus:

- El servidor confirma cada señal recibida mandando una señal al cliente.
- Soporta los caracteres Unicode.



La parte extra solo será evaluada si la parte obligatoria está PERFECTA. Perfecta significa que la parte ogligaria se ha completado entera y funciona sin ningún error. Si no has alcanzado todos los requisitos de la parte obligaroria, tu bonus no será evaluada de ninguna manera.

# Capítulo VI

# Entrega y evaluación de pares

Como es habitual, entrega tu trabajo en el repositorio Git. Solo el trabajo en tu repositorio será evaluado. Comprueba dos veces que los nombres de tus archivos sean los correctos.