minicharla

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

Versión 2

Contenido

Capítulo I

Prefacio

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

La mayoría de las plantas lo producen en pequeñas cantidades y actúa como un atractivo para muchos insectos depredadores. El cis-3-hexen-1-ol es un compuesto aromático muy importante que se utiliza en sabores de frutas y verduras y en perfumes.

La producción anual es de unas 30 toneladas.

Capitulo dos

Instrucciones comunes

- Su proyecto debe estar escrito de acuerdo con la Norma. Si tiene archivos/funciones de bonificación, se incluyen en la verificación de normas y recibirá un 0 si hay un error de norma en su interior.

- Sus funciones no deberían cerrarse inesperadamente (falla de segmentación, error de bus, doble libre, etc.) además de comportamientos indefinidos. Si esto sucede, su proyecto se considerará no funcional y recibirá un 0 durante la evaluación.

- Todo el espacio de memoria asignado al almacenamiento dinámico debe liberarse correctamente cuando sea necesario. Sin fugas será tolerado.

- Si el tema lo requiere, debe enviar un Makefile que compilará sus archivos fuente a la salida requerida con las banderas -Wall, -Wextra y -Werror, use cc, y su Makefile no debe volver a vincularse.

- Su Makefile debe contener al menos las reglas $(NOMBRE), all, clean, fclean y re.

- Para entregar bonificaciones a su proyecto, debe incluir una regla de bonificación a su Makefile, que agregará todos los encabezados, bibliotecas o funciones que están prohibidas en la parte principal del proyecto. Los bonos deben estar en un archivo diferente _bonus.{c/h}.
  La evaluación de la parte obligatoria y de bonificación se realiza por separado.

- Si su proyecto le permite usar su libft, debe copiar sus fuentes y su Makefile asociado en una carpeta libft con su Makefile asociado. El Makefile de su proyecto debe compilar la biblioteca utilizando su Makefile y luego compilar el proyecto.

- Lo alentamos a que cree programas de prueba para su proyecto, aunque este trabajo **no tendrá que enviarse y no será calificado.** Le dará la oportunidad de probar fácilmente su trabajo y el de sus compañeros. Encontrará estas pruebas especialmente útiles durante su defensa. De hecho, durante la defensa, eres libre de usar tus pruebas y/o las pruebas del compañero que estás evaluando.

- Envíe su trabajo a su repositorio git asignado. Solo se calificará el trabajo en el repositorio de git. Si se asigna Deep Thought para calificar su trabajo, se hará después de sus evaluaciones por pares. Si ocurre un error en cualquier sección de su trabajo durante la calificación de Deepthought, la evaluación se detendrá.

Capítulo III

Instrucciones del proyecto

- Asigne un nombre a sus archivos ejecutables cliente y servidor.
- Tiene que manejar los errores minuciosamente. De ninguna manera su programa debe salir de unex esperado (falla de segmentación, error de bus, doble libre, etc.).
- Su programa no debe tener **pérdidas de memoria.**
- Puede tener **una variable global por programa** (una para el cliente y otra para el servidor), pero tendrás que justificar su uso.
- Para completar la parte obligatoria, se le permite utilizar el siguiente funciones:
  - ÿ escribir
  - ÿ ft_printf y cualquier código equivalente USTED
  - ÿ señal
  - ÿ sigemptyset
  - ÿ sigaddset
  - ÿ sigaction
  - ÿ matar
  - ÿ getpid
  - ÿ malloc
  - ÿ gratis
  - ÿ pausa
  - ÿ dormir
  - ÿ dormido
  - ÿ salir

Capítulo IV

Parte Obligatoria

Debe crear un programa de comunicación en forma de **cliente** y **servidor.**

- El servidor debe iniciarse primero. Después de su lanzamiento, tiene que imprimir su PID.
- El cliente toma dos parámetros:
  - ÿ El PID del servidor.
  - ÿ La cadena a enviar.
- El cliente debe enviar la cadena pasada como parámetro al servidor.
  Una vez recibida la cadena, el servidor debe imprimirla.
- El servidor tiene que mostrar la cadena bastante rápido. Rápidamente significa que si piensas lleva demasiado tiempo, entonces probablemente sea demasiado tiempo.

¡ 1 segundo para mostrar 100 caracteres es **demasiado!**

- Su servidor debería poder recibir cadenas de varios clientes seguidos sin necesidad de reiniciar.
- La comunicación entre su cliente y su servidor debe hacerse **solo** usando señales UNIX.
- Solo puede utilizar estas dos señales: SIGUSR1 y SIGUSR2.

¡El sistema Linux NO pone en cola las señales cuando ya tiene señales pendientes de este tipo! ¿Tiempo extra?

Capítulo V

parte de bonificación

Lista de bonos:

- El servidor reconoce cada mensaje recibido devolviendo una señal al cliente.
- ¡Compatibilidad con caracteres Unicode!

La parte extra sólo se valorará si la parte obligatoria es PERFECTA. Perfecto significa que la parte obligatoria se ha hecho íntegramente y funciona sin fallas. Si no ha superado TODOS los requisitos obligatorios, su parte de bonificación no se evaluará en absoluto.

Capítulo VI

Presentación y evaluación por pares

Entregue su tarea en su repositorio de Git como de costumbre. Solo el trabajo dentro de su repositorio será evaluado durante la defensa. No dude en verificar dos veces los nombres de sus archivos para asegurarse de que sean correctos.