Minit lk

Resumen: El propósito de este proyecto es cre r un pequeño progr m de interc mbio de d tos utiliz ndo señ les UNIX.

Versión: 2

Índice general

Capítulo I

vance

El cis-3-Hexen-1-ol, t mbién conocido como (Z)-3-hexen-1-ol y lcohol de hoj s, es un líquido ceitoso incoloro con un olor intenso hierb y hoj s verdes recién cort d s.

Se produce en pequeñ s c ntid des por l m yorí de l s pl nt s y ctú como un tr yente p r muchos insectos depred dores. El cis-3-hexen-1-ol es un compuesto de rom muy import nte que se utiliz en s bores de frut s y verdur s y en perfumes.

L producción nu l es de lrededor de 30 tonel d s.

Capítulo II

Instrucciones generales

- Tu proyecto deberá est r escrito en C.
- Tu proyecto debe est r escrito siguiendo l Norm . Si tienes rchivos o funciones dicion les, est s están incluid s en l verific ción de l Norm y tendrás un `0` si h y lgún error de norm en cu lquier de ellos.
- Tus funciones no deben termin r de form inesper d (segf ult, bus error, double free, etc) ni tener comport mientos indefinidos. Si esto p s tu proyecto será consider do no funcion l y recibirás un `0` dur nte l ev lu ción.
- Tod l memori sign d en el he p deberá liber rse decu d mente cu ndo se neces rio. No se permitirán le ks de memori .
- Si el enunci do lo requiere, deberás entreg r un `Makefile` que compil rá tus rchivos fuente l output requerido con l s fl gs `-Wall`, `-Werror` y `-Wextra` y por supuesto tu `Makefile` no debe h cer relink.
- Tu `Makefile` debe contener l menos l s norm s `$(N ME)`, `all`, `clean`, `fclean` y `re`.
- P r entreg r los bonus de tu proyecto deberás incluir un regl `bonus` en tu `Makefile`, en l que ñ dirás todos los he ders, librerí s o funciones que estén prohibid s en l p rte princip l del proyecto. Los bonus deben est r en rchivos distintos `_bonus.{c/h}`. L p rte oblig tori y los bonus se ev lú n por sep r do.
- Si tu proyecto permite el uso de l `libft`, deberás copi r su fuente y sus `Makefile` soci dos en un directorio `libft` con su correspondiente `Makefile`. El `Makefile` de tu proyecto debe compil r primero l librerí utiliz ndo su `Makefile`, y después compil r el proyecto.
- Te recomend mos cre r progr m s de prueb p r tu proyecto, unque este tr b jo **no será entregado ni evaluado**. Te d rá l oportunid d de verific r que tu progr m funcion correct mente dur nte tu ev lu ción y l de otros comp ñeros. Y sí, tienes permitido utiliz r est s prueb s dur nte tu ev lu ción o l de otros comp ñeros.
- Entreg tu tr b jo en tu repositorio `Git` sign do. Solo el tr b jo de tu repositorio `Git` será ev lu do. Si Deepthought ev lú tu tr b jo, lo h rá después de tus com-

p ñeros. Si se encuentr un error dur nte l ev lu ción de Deepthought, est h brá termin do.

Capítulo III

Instrucciones específicas

- Los ejecut bles deben ll m rse `client` y `server`.
- Deberás proporcion r un rchivo `Makefile`, que compil rá los rchivos fuente de tu progr m . No deberá h cer relink.
- Puedes us r tu `libft`.
- Debes gestion r los errores con cuid do. B jo ningun circunst nci tu progr m puede termin r inesper d mente (segf ult, bus error, double free, etc).
- Tu progr m no puede tener **leaks de memoria**.
- Puedes utiliz r **una variable global por programa** (un p r el cliente y otr p r el servidor), pero tendrás que justific r su uso.
- En l p rte oblig tori se te permite utiliz r l s siguientes funciones:

```
write
ft_printf y cualquier equivalente que TÚ hayas programado
signal
sigemptyset
sigaddset
sigaction
kill
getpid
malloc
free
pause
sleep
usleep
exit
```

Capítulo IV

Parte obligatoria

- Debes cre r un progr m de comunic ción en l form de un **cliente** y un **servidor**.
- El servidor debe l nz rse primero, tr s l nz rse debe mostr r su `PID`.
- El cliente tom rá como p rámetros:

  El `PID` del servidor.

  L string que deberí m nd rse.

- El cliente debe comunic r l string p s d como p rámetro l servidor. Un vez l string se h y recibido, el servidor debe mostr rl .
- El servidor debe ser c p z de mostr r l string suficientemente rápido. Por rápido queremos decir que si piens s que es está t rd ndo mucho, prob blemente es que está t rd ndo dem si do.

`1 segundo para mostrar 100 caracteres es ¡mu hiiiiisimo!`

- Tu servidor debe poder recibir strings de distintos clientes consecutiv mente, sin necesid d de reinici r.
- L comunic ción entre tu cliente y tu servidor debe h cerse **SOLO** utiliz ndo señ les UNIX.
- Solo puedes utiliz r est s dos señ les: `SIGUSR1` y `SIGUSR2`.

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

Capítulo V

Extras

List de bonus:

- El servidor confirm c d señ l recibid m nd ndo un señ l l cliente.
- Soport los c r cteres Unicode.

La parte extra solo será evaluada si la parte obligatoria está PERFECT . 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.