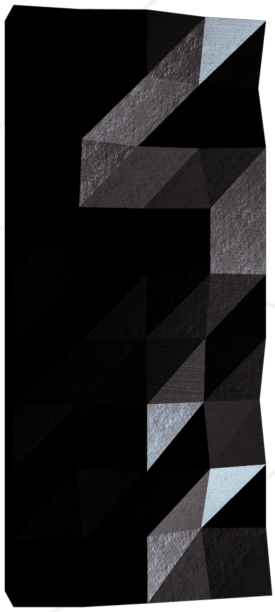

printf

Porque putnbr y putstr no son suficientes

Resumen: Este proyecto es claro y eficaz. Tiene que volver a programar printf. A partir de ahí, tendrá la posibilidad de reutilizarlo en sus proyectos futuros. Este proyecto trata principalmente de los argumentos de tamaño variable.

Índice general

Capítulo I

Introducción

La versatilidad de `printf` en `C` nos permite hacer un buen ejercicio de programación. La dificultad de este proyecto es moderada. Le va a permitir utilizar los kwargs en `C`. La clave del éxito para `ft_printf` reside en un código bien estructurado y extensible.

Capítulo II

Reglas comunes

- Su proyecto debe estar programado respetando la Norma. Si tiene archivos o funciones extras, entrarán dentro de la verificación de la norma y, como haya algún error de norma, tendrá un 0 en el proyecto.
- Sus funciones no pueden pararse de forma inesperada (segmentation fault, bus error, double free, etc.) salvo en el caso de un comportamiento indefinido. Si esto ocurre, se considerará que su proyecto no es funcional y tendrá un 0 en el proyecto.
- Cualquier memoria reservada en el montón (heap) tendrá que ser liberada cuando sea necesario. No se tolerará ninguna fuga de memoria.
- Si el proyecto lo requiere, tendrá que entregar un Makefile que compilará sus códigos fuente para crear la salida solicitada, utilizando los flags `-Wall`, `-Wextra` y `-Werror`. Su Makefile no debe hacer relink.
- Si el proyecto requiere un Makefile, su Makefile debe incluir al menos las reglas `$(NAME)`, `all`, `clean`, `fclean` y `re`.
- Para entregar los extras, debe incluir en su Makefile una regla `bonus` que añadirá los headers, bibliotecas o funciones que no estén permitidos en la parte principal del proyecto. Los extras deben estar dentro de un archivo `_bonus.{c/h}`. Las evaluaciones de la parte obligatoria y de la parte extra se hacen por separado.
- Si el proyecto autoriza su `libft`, debe copiar sus códigos fuente y y su Makefile asociado en un directorio libft, dentro de la raíz. El Makefile de su proyecto debe compilar la biblioteca con la ayuda de su Makefile y después compilar el proyecto.
- Le recomendamos que cree programas de prueba para su proyecto, aunque ese trabajo **no será ni entregado ni evaluado**. Esto le dará la oportunidad de probar fácilmente su trabajo al igual que el de sus compañeros.
- Deberá entregar su trabajo en el git que se le ha asignado. Solo se evaluará el trabajo que se suba al git. Si Deepthought debe corregir su trabajo, lo hará al final de las evaluaciones por sus pares. Si surge un error durante la evaluación Deepthought, esta última se parará.

Capítulo III

Parte obligatoria

Nombre de la función	`ft_printf`
Prototipo	`int ft_printf(const char *, ...);`
Ficheros a entregar	`*.c`
Parámetros	`Su referencia será man printf`
Valor de retorno	`Su referencia será man printf`
Funciones externas autorizadas	`malloc, free, write, va_start, va_arg, va_copy, va_end`
Descripción	`Escriba una función que reproduzca el comportamiento del verdadero printf`

- Tiene que volver a programar la función `printf` de la `libc`
- No necesita gestionar el buffer, a diferencia del verdadero `printf`
- Tendrá que gestionar las siguientes conversiones: `cspdiuxX%`
- Tendrá que gestionar cualquier combinación de flags '`-0.*`' y el tamaño mínimo del campo
- Se comparará su entrega con el verdadero printf

`man 3 printf / man 3 stdarg`

Capítulo IV

Parte extra

- Si la parte obligatoria no está perfecta, no haga la parte extra
- No está obligado a hacer la parte extra
- Gestione una o varias de las siguientes conversiones: `nfge`
- Gestione uno o varios de los siguientes flags: `l ll h hh`
- Gestione todos los flags siguientes: `'# +`