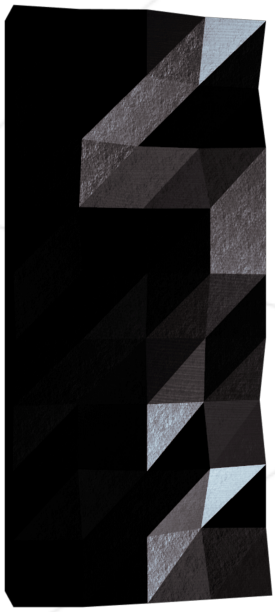

printf

Va siendo hora de reemplazar putnbr y putstr

Resumen: Este proyecto va bastante al grano. Deberás programar la función printf. Esperamos que puedas utilizarla en tus futuros proyectos sin miedo a ser considerado un tramposo. Aprenderás principalmente a utilizar funciones variádicas.

Índice general

Capítulo I

Introducción

La versalidad de la función `printf` en `C` hace de este un buen reto de programación. Se puede calificar como un proyecto de dificultad moderada. Te permitirá descubrir las funciones variádicas en `C`.
Un consejo: la clave para un buen `ft_printf` es un código bien estructurado y extensible.

Capítulo II

Instrucciones generales

- 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 dentro.
- Tus funciones no deben terminar de forma inesperada (segfault, bus error, double free, etc) ni teener comportamientos indefinidos. Si esto pasa tu proyecto será considerado no funcional y recibirás un `0` durante la evaluación.
- Toda la memoria alocada en heap deberá liberarse adecuadamente cuando sea necesario. No se permitirán leaks de memoria.
- Si el subject lo requiere, deberás entregar un `Makefile` que compilará tus archivos fuente al output requerido con las flags `-Wall`, `-Werror` y `-Wextra`, 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 a tu repositorio `Git` asignado. Solo el trabajo de tu repositorio `Git` será evaluado. Si Deepthought evalúa tu trabajo, lo hará después de tus compañeros. Si se encuentra un error durante la evaluación de Deepthought, la evalaución terminará.

Capítulo III

Parte obligatoria

Nombre de programa	`libftprintf.a`
Archivos a entregar	`*.c, */*.c, *.h, */*.h, Makefile`
Makefile	`all, clean, fclean, re, bonus`
Funciones autorizadas	`malloc, free, write, va_start, va_arg, va_copy, va_end`
Se permite usar libft	`sí`
Descripción	`Escribe una librería que contenga la función ft_printf, que imite el printf real`

- El prototipo del ft_printf deberá ser `int ft_printf(const char *, ...);`
- Debes programar la función `printf` de `libc`.
- No tiene por qué gestionar el buffer como lo hace el `printf` real.
- Deberá implementar las siguientes conversiones: `cspdiuxX%`.
- Su funcionamiento se contrastará con el printf original.
- Debes utilizar el comando `ar` para crear tu librería, el uso del comando `libtool` se prohibe.

Una pequeña y simple descripción de las conversiones que se te piden:

- %c para imprimir un solo carácter.
- %s para imprimir una string.
- %p el puntero void * dado como argumento se imprimirá en hexadecimal.
- %d para imprimir un número decimal (de base 10).
- %i para imprimir un entero en base 10.
- %u para imprimir un número decimal (de base 10) sin signo.
- %x para imprimir un número hexadecimal (de base 16), en minúscula.
- %X para imprimir un número hexadecimal (de base 16), en mayúscula.
- %% para imprimir el signo del porcentaje.

`para una referencia completa: man 3 printf / man 3 stdarg`

Capítulo IV

Parte bonus

- Si la parte obligatoria no está perfecta, no pienses en los bonus.
- No tienes por qué hacer todos los bonus.
- Gestiona todas las combinaciones de las siguientes flags para todas las conversiones: "`-0.`" y el flag de longitud mínima (minimum field width).
- Gestiona todas las combinaciones de las siguientes flags para todas las conversiones: "`# +`" (sí, una es un espacio).

`Si tienes intención de hacer los bonus, deberías pensar en cómo hacerlos desde el principio para evitar enfocar el proyecto incorrectamente.`