



printf

Parce que putnbr et putstr ne sont pas assez

Résumé: Ce projet est clair et efficace. Vous devez recoder printf. Vous aurez dès lors la possibilité de le réutiliser dans vos futurs projets. Ce projet porte principalement sur les arguments à taille variable.

Table des matières

I	Introduction	2
II	Règles communes	3
III	Partie obligatoire	4
IV	Bonus part	5

Chapitre I

Introduction

La versatilité de `printf` en C représente un bon exercice de programmation pour nous. Ce projet est d'une difficulté modérée. Il vous permettra d'utiliser les `kwargs` en C La clé de la réussite pour `ft_printf` est un code bien structuré et extensible.

Chapitre II

Règles communes

- Votre projet doit être codé à la Norme. Si vous avez des fichiers ou fonctions bonus, celles-ci seront incluses dans la vérification de la norme et vous aurez 0 au projet en cas de faute de norme.
- Vos fonctions ne doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.
- Toute mémoire allouée sur la heap doit être libérée lorsque c'est nécessaire. Aucun leak ne sera toléré.
- Si le projet le demande, vous devez rendre un Makefile qui compilera vos sources pour créer la sortie demandée, en utilisant les flags `-Wall`, `-Wextra` et `-Werror`. Votre Makefile ne doit pas relink.
- Si le projet demande un Makefile, votre Makefile doit au minimum contenir les règles `$(NAME)`, `all`, `clean`, `fclean` et `re`.
- Pour rendre des bonus, vous devez inclure une règle `bonus` à votre Makefile qui ajoutera les divers headers, librairies ou fonctions qui ne sont pas autorisées dans la partie principale du projet. Les bonus doivent être dans un fichier `_bonus.{c/h}`. L'évaluation de la partie obligatoire et de la partie bonus sont faites séparément.
- Si le projet autorise votre `libft`, vous devez copier ses sources et son Makefile associé dans un dossier `libft` contenu à la racine. Le Makefile de votre projet doit compiler la librairie à l'aide de son Makefile, puis compiler le projet.
- Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs.
- Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.

Chapitre III

Partie obligatoire

Function name	<code>ft_printf</code>
Prototype	<code>int ft_printf(const char *, ...);</code>
Fichiers de rendu	<code>*.c</code>
Paramètres	Votre référence est <code>man printf</code>
Valeur de retour	Votre référence est <code>man printf</code>
Fonctions externes autorisées	<code>malloc</code> , <code>free</code> , <code>write</code> , <code>va_start</code> , <code>va_arg</code> , <code>va_copy</code> , <code>va_end</code>
Description	Écrivez une fonction qui reproduit le comportement du vrai <code>printf</code>

- Vous devez recoder la fonction `printf` de la `libc`
- Vous n'avez pas à gérer de buffer, contrairement au vrai `printf`
- Vous devez gérer les conversions suivantes : `cspdiuxX%`
- Vous devez gérer n'importe quelle combinaison de flags `'-0.*'` et la taille de champ minimale
- Votre rendu sera comparé au vrai `printf`



`man 3 printf / man 3 stdarg`

Chapitre IV

Bonus part

- Si la partie obligatoire n'est pas parfaite, ne faites pas les bonus
- Vous n'avez pas d'obligation de faire tous les bonus
- Gérez une ou plusieurs des conversions suivantes : **nfge**
- Gérez un ou plusieurs des flags suivants : **l ll h hh**
- Gérez tous les flags suivants : **'# +**