ft_printf

Parce que ft_putnbr() et ft_putstr(), c'est pas assez

Résumé:
Ce projet est assez direct. Vous devez recoder printf().
Vous apprendrez principalement à vous servir d'un nombre d'arguments variable.

Version: 9

Table des matières

Chapitre I

Introduction

La versatilité de `printf()` en `C` représente un bon exercice de programmation. Ce projet est d'une difficulté modérée.

Il vous permettra de découvrir les **fonctions variadiques** en `C`.

La clé de la réussite pour `ft_printf` est un code bien structuré et extensible.

`Une fois ce projet validé, vous pourrez ajouter votre ft_printf() à votre libft afin de l'utiliser dans vos prochains projets C du cursus.`

Chapitre II

Règles communes

- Votre projet doit être écrit en C.
- Votre projet doit être codé à la Norme. Si vous avez des fichiers ou fonctions bonus, celles-ci seront inclues dans la vérification de la norme et vous aurez 0 au projet en cas de faute de norme.
- Vos fonctions de 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 libéré 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 différent : `_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

Nom du programme	`libftprintf.a`
Fichiers de rendu	`Makefile, *.h, */*.h, *.c, */*.c`
Makefile	`NAME, all, clean, fclean, re`
Fonctions externes autorisées	`malloc, free, write, va_start, va_arg, va_copy, va_end`
Libft autorisée	`Oui`
Description	`Une bibliothèque qui contient ft_printf(), une fonction imitant la fonction printf() originale`

Vous devez implémenter la fonction `printf()` de la `libc`.

Le prototype de `ft_printf()` devra être :

```
int     ft_printf(const char *, ...);
```

Voici quelques impératifs à respecter :

- Contrairement à la fonction `printf()` originale, vous ne devez pas gérer de *buffer*.
- Vous devez gérer les conversions suivantes : `cspdiuxX%`
- Votre rendu sera comparé à la fonction `printf()` originale.
- Vous devez utiliser la commande `ar` pour créer votre bibliothèque.
  L'utilisation de la commande `libtool` est interdite.
- Votre `libftprintf.a` doit être créé à la racine de votre dépôt.

Vous devez implémenter les conversions suivantes :

- `%c` Affiche un seul caractère.
- `%s` Affiche une chaîne de caractères (telle que définie par la convention `C`).
- `%p` L'argument de pointeur `void *` doit être affiché en hexadécimal.
- `%d` Affiche un nombre décimal (base 10).
- `%i` Affiche un entier en base 10.
- `%u` Affiche un nombre décimal non signé (base 10).
- `%x` Affiche un nombre en hexadécimal (base 16) avec des lettres minuscules.
- `%X` Affiche un nombre en hexadécimal (base 16) avec des lettres majuscules.
- `%%` Affiche un signe pourcentage.

Chapitre IV

Partie bonus

Vous n'avez pas l'obligation de faire tous les bonus.

Liste de bonus :

- Gérez toute combinaison des drapeaux suivants : '`-0.`' ainsi que la largeur minimale du champ avec toutes les conversions.
- Gérez tous les drapeaux suivants : '`# +`' (Oui, l'espace est un drapeau valide)

Si vous envisagez de faire des bonus, il vaut mieux réfléchir à leur implémentation dès le début afin d'éviter une approche "naïve".

Les bonus ne seront évalués que si la partie obligatoire est PARFAITE. Par parfaite, nous entendons complète et sans aucun dysfonctionnement. Si vous n'avez pas réussi TOUS les points de la partie obligatoire, votre partie bonus ne sera pas prise en compte.

Chapitre V

Rendu et peer-evaluation

Rendez votre travail sur votre dépot `Git` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

Une fois ce projet validé, vous pourrez ajouter votre `ft_printf()` à votre `libft` afin de l'utiliser dans vos prochains projets `C` du cursus.