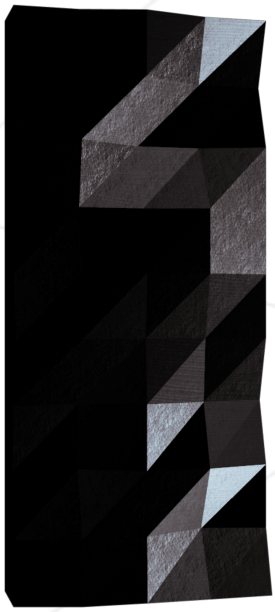

# ft_printf

## Parce que putstr et putnbr, ça va bien 2 minutes

Staff pedago pedago@42.fr

*Résumé: Vous en avez assez de faire vos affichages en alternant ft_putstr et ft_putnbr ? Vous n'avez pas le droit d'utiliser printf ? Recodez le votre ! Ce sera l'occasion de découvrir une feature du C - les fonctions variadiques - et de vous entrainer à la gestion fine des options d'affichage. Vous aurez ensuite le droit d'utiliser votre ft_printf dans tous vos projets ultérieurs.*

# Table des matières

# Chapitre I

# Préambule

Voici la liste des posters de motivation que l'on peut trouver dans le bureau de Barney Stinson au fil des saisons de How I Met Your Mother :

- Awesomeness : "When I get sad, I stop being sad and be Awesome Instead. True Story. Barney Stinson"
- Conformity : "It's the one who is different that gets left out in the cold."
- Courage : "True greatness comes when you're tested. Theodore Roosevelt"
- Challenge : "We either find a way or we make one"
- Opportunity : "You will always miss 100% of the shots you don't take."
- Teamwork : "Coming together is the beginning. Keeping together is progress. Working together is success. Henry Ford."
- Teamwork : "The chain is only as strong as the weakest link"
- Perseverance : "Continuous effort is the key to unlocking your potential. Sir Winston Churchill"
- Perfection : "It is not good enough to win, everybody else should lose"
- Strength : "What the mind can conceive, it can achieve"

# Chapitre II

# Introduction

Quelque soit le langage de programmation considéré, la fonction `printf` (ou ses équivalents) est toujours une fonction extrêmement pratique. La raison principale est bien évidemment le confort de formatage, et le support de types hétérogènes en nombre variable. Certaines variantes proposent même de pouvoir écrire la chaine de caractères résultat sur un file descriptor ou un stream particulier, ou bien même carrément de récuperer cette chaine sans l'imprimer. Bref, c'est une fonction incontournable que nous vous proposons de recoder aujourd'hui pour l'ajouter à votre `libft` et en profiter dans tous vos projets suivants, comme `ft_ls` par exemple...

# Chapitre III

# Objectifs

La versatilité de la fonction `printf` en `C` représente pour nous un excellent exercice de programmation. D'une difficulté modérée, ce projet va vous permettre de découvrir les fonctions variadiques en `C` dans un contexte particulièrement adapté, et de voir un excellent exemple d'un "dispatcher" rudimentaire implémenté en `C` à l'aide d'un tableau de pointeurs sur fonctions.

La clef d'un `ft_printf` réussi est un code bien structuré et facilement extensible, car plus le temps passera, plus vous serez tentés de continuer à étendre votre `ft_printf`, vous facilitant ainsi la vie sur les prochains projets. Prenez donc le temps de coder proprement en gardant en tête que vous devrez relire votre code d'ici quelques semaines ou quelques mois pour étendre ses fonctionnalités selon votre besoin. Ce serait dommage de ne pas le faire parce que vous ne pouvez pas vous relire, non ?

# Chapitre IV

# Consignes générales

- La fonction doit s'appeler `ft_printf`.
- Votre projet doit être à la Norme.
- Vous devez gérer les erreurs de façon sensible. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, etc...)
- Vous devez fournir un Makefile qui compilera une libftprintf.a. Cette lib sera linké à notre main de test pour vous donner votre résultat.
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi d'un '\n' :

```
$>cat -e auteur
xlogin$
```

- Vous avez le droit d'utiliser les fonctions suivantes :
  - `write`
  - `malloc`
  - `free`
  - `exit`
  - les fonctions du man 3 `stdarg`
- Vous pouvez poser vos questions sur le forum.

# Chapitre V

# Partie obligatoire

- Vous devez recoder la fonction `printf` de la `libc`
- Votre fonction s'appelera `ft_printf` et sera prototypée de la même façon que `printf`
- Vous ne ferez pas la gestion de buffer présente dans la fonction `printf` de la `libc`.
- Vous devez gérer les conversions suivantes : `csp`
- Vous devez gérer les conversions suivantes : `diouxX` avec les flags `hh`, `h`, `l`, `ll`
- Vous devez gérer la conversion suivante : `f` avec les flags `l` et `L`
- Vous devez gérer le `%%`
- Vous devez gérer les flags `#0-+` et espace
- Vous devez gérer la taille minimum du champ
- Vous devez gérer la précision

`man 3 printf / man 3 stdarg`

# Chapitre VI

# Partie bonus

Voici quelques idées de bonus intéressants à réaliser, voire même utiles. Vous pouvez évidemment ajouter des bonus de votre invention, qui seront évalués à la discrétion de vos correcteurs.

- Gestion de conversions plus délicates : `e`, `g`. Attention votre flag `L` doit être étendu à ces deux conversions pour que le bonus soit comptabilisé !
- Gestion de flags plus delicats : `*`, `$`, `'`
- Gestion de flags non existants : `%b` pour afficher en binaire, `%r` pour afficher une chaine avec des caractères non imprimables, `%k` pour afficher une date à un format ISO quelconque, etc.
- Gestion de modifieurs pour gérer les couleurs, les fd ou des choses fun comme ça :)

```
printf("Le fichier{cyan}%s{eoc} contient : {red}%s{eoc}", filename, str);
```

# Chapitre VII

# Rendu et peer-évaluation

Rendez-votre travail sur votre dépot `GiT` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance.