FDI-DVC1 / my_printf

Modalités

Dépôt	https://rendu-git.etna-alternance.net/module-6391/activity-35756/group-770753
VM	$$VM$$
Correction	Moulinette
Durée	1 ou 2 semaines
Taille de groupe	Seul
Environnement	Debian
Flags de compilation	-Wall -Wextra -Werror

Objectifs

Notion	Description
Programmation en C	Établir des connaissances et acquérir des compétences permettant de développer en langage C

Interdits

- Toute triche, plagiat ou utilisation de fonctions interdites conduit à un -42
- Les séries de if/else pour gérer les modificateurs sont interdites ! Vous devez utiliser les pointeurs sur fonction, ou les switch/case
- Votre rendu ne doit contenir que les fichiers utiles à votre programme. Pas de fichiers temporaires ni de binaires
- Tout retour de fonction pouvant échouer (**`malloc`** , **`open`** , etc) doit être vérifié

Fonctions autorisées

- free(3)
- malloc(3)
- stdarg (man 3) :
  - va_start
  - va_arg
  - va_end
- write(2)

Consignes

Pour ce projet, nous vous demandons de re-implémenter **`printf`**, en vous basant sur la fonction originale.

Votre Makefile doit permettre de compiler une bibliothèque statique et une bibliothèque dynamique.

Pour cela, vous pouvez utiliser toutes les fonctions autorisées, ainsi que tout ce que vous avez codé vous-même.

Prototype :

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

Makefile

Les règles de votre Makefile doivent respecter les instructions suivantes :

- "my_printf_static" : compiler la bibliothèque statique
- "my_printf_dynamic" : compiler la bibliothèque dynamique
- "all" : compiler les deux bibliothèques. Cette règle devra être exécutée lors d'un "make" simple
- "clean" : effacer les fichiers objets (**`*.o`**)
- "fclean" : effacer les deux bibliothèques et les fichiers objets
- "re" : effacer les deux bibliothèques ainsi que les fichiers objets, et ensuite compiler les deux bibliothèques
- ".PHONY" : nous vous laissons vous renseigner sur son utilité et usage

Nom des bibliothèques :

- libmy_printf_ **`uname -m-uname -s`** .a

- libmy_printf_`uname -m-uname -s`.so

Pour créer une bibliothèque statique, voir les man de **ar** et **ranlib** et pour créer une bibliothèque dynamique, voir l'option **-shared** de gcc.

Si vous utilisez votre **libmy**, vous devrez inclure les **sources** de celle-ci dans vos **libmy_printf**.

Étape 1

Dans un premier temps, vous devrez gérer les modificateurs `%s`, `%c`, `%i` et `%d` :

```
int main(void)
{
my_printf("1 - une chaine\n");
my_printf("2 - %s\n", "une autre chaine");
my_printf("3 - %i\n", 42);
my_printf("4 - %s %d %s%c", "avec", 4, "parametres", '\n');
return (0);
}
```

```
$> ./a.out
1 - une chaine
2 - une autre chaine
3 - 42
4 - avec 4 parametres
$>
```

Étape 2

Vous devrez ensuite gérer `%o`, `%u`, `%x`, `%X` et `%%` :

```c
int main(void)
{
my_printf("1 - %o\n", 42); /* unsigned octal */
my_printf("2 - %u\n", (unsigned int)4200000000); /* unsigned decimal */
my_printf("3 - %x\n", 42); /* unsigned hexadecimal */
my_printf("4 - %X\n", 42); /* unsigned hexadecimal */
my_printf("5 - %d%%\n", 42);
return (0);
}
```

```
$> ./a.out
1 - 52
2 - 4200000000
3 - 2a
4 - 2A
5 - 42%
$>
```

Bonus

Vous pourrez aussi gérer les flags de formatage de printf ainsi que **%f**, **%F**, **%e**, **%E**, **%g**, **%G**, **%a**, **%A** :

```c
int main(void)
{
my_printf("1 - %.5f\n", 4 * cos(2.0)); /* float */
my_printf("2 - %e\n", 4 * cos(2.0)); /* double */
return (0);
}
```

```
$> ./a.out
1 - -1.66459
2 - -1.664587e+00
$>
```