Programmation Système - Rapport TP

Remarque :
J'ai :

```
// !! Pour masquer l'erreur de sbrk !!
#pragma GCC diagnostic ignored "-Wdeprecated-declarations"
```

dans mon mini_lib.h pour ne pas avoir les erreurs lié au sbrk !
Je l'ai commenté dans mon code pour le rapport final.

⬇️ SUITE DU RAPPORT ⬇️

Exercice 5 :

L'initialisation du buffer avec des '\0' assure que la mémoire allouée est propre, c'est-à-dire qu'elle est remplie de zéros, et donc prête à être utilisée. C'est très important lorsqu'on travaille avec des chaînes de caractères, des tableaux ou d'autres structures où la valeur initiale de chaque élément est significative.

Exercice 6 :

La fonction free() **libère d'une certaine façon la mémoire**, mais **elle ne la supprime pas directement**. Elle indique au système que la mémoire n'est plus utilisée, mais le contenu reste inchangé jusqu'à ce qu'elle soit réallouée.

Exercice 12 :

Je propose deux solutions à tester :

➡ Test n°1 :

Je procède à l'allocation de trois emplacements mémoire. Ensuite, je libère l'emplacement du milieu (p2) et teste si je peux réallouer cet espace mémoire avec une taille égale (p4) ou inférieure (p5).

```
maxx@Air-de-Maxime mon_tp % ./mon_exe
p1 : 0x10024c020
p2 : 0x10024c0a5
p3 : 0x10024c12b
p4 : 0x10024c0a5 (doit être p2)
p5 : 0x10024c0a5 (doit être p2 & p4)
```

➡ Test n°2 :

Je réserve trois emplacements mémoire (p1, p2, p3) puis les libère. Ensuite, je réalloue des espaces mémoire (p11, p22, p33) pour vérifier s'ils reprennent les emplacements libérés précédemment. Enfin, je libère ces nouveaux emplacements.

```
p1 : 0x102468020
p2 : 0x1024680a5
p3 : 0x10246812b
p1, p2 et p3 sont free !
p11 : 0x102468020 (= p1)
p22 : 0x1024680a5 (= p2)
p33 : 0x10246812b (= p3)
```

Exercice 17 :

```c
// Exercice 17 : Test du mini_printf
char *str_1 = "Bonjour, ça va ? \n";
printf("1) Avec '\\n' : \n");
mini_printf(str_1);

char *str_2 = "Bonjour, ça va ?";
printf("2) SANS '\\n' \n");
mini_printf(str_2);
```

```
maxx@MacBook-Air-de-Maxime mon_tp % ./mon_exe
1) Avec '\n' :
Bonjour, ça va ?
2) SANS '\n' :
```

Le problème est que "str_2" ne s'affiche pas !

Exercice 18 :

Dans le mini_exit() on ajoute une fonction qui vide et affiche ce qu'il reste dans le buffer, maintenant, on obtient :

```
maxx@MacBook-Air-de-Maxime mon_tp % ./mon_exe
1) Avec '\n' :
Bonjour, ça va ?
2) SANS '\n' :
Bonjour, ça va ?%
```

Exercice 22 :

Les problèmes sont les suivants :
- pour mini_strcpy :
Si la chaîne source (s) est plus grande que la chaîne destination (d), cela peut entraîner un dépassement de tampon.
Il faut également vérifier que la source et la destination ne soit pas NULL !

- pour mini_strcmp :
La fonction pourrait potentiellement provoquer un dépassement de capacité en cas de différence significative entre les valeurs ASCII de *s1 et *s2
Il faut également vérifier que s1 et s2 ne soit pas NULL !

Exemple d'exécution du mini_shell :

```
maxx@MacBook-Air-de-Maxime mon_tp % ./mon_exe
######################################## FORMULAIRE D'AIDE POUR LES APPELS DE FONCIONS DU MINI_SHELL ########################################
La fonction : 'mini_echo' s'appelle de la manière suivante :
----> mini_echo chaine1 chaine2 ... chaineN
La fonction : 'mini_touch' s'appelle de la manière suivante :
----> mini_touch name_file
La fonction : 'mini_cat' s'appelle de la manière suivante :
----> mini_cat name_file
La fonction : 'mini_tail' s'appelle de la manière suivante :
----> mini_tail N_dernière_ligne name_file
La fonction : 'mini_head' s'appelle de la manière suivante :
----> mini_head N_première_ligne name_file
La fonction : 'mini_clean' s'appelle de la manière suivante :
----> mini_clean name_file
La fonction : 'mini_grep' s'appelle de la manière suivante :
----> mini_grep mot_cherché name_file
La fonction : 'mini_wc' s'appelle de la manière suivante :
----> mini_wc name_file
La fonction : 'mini_clear' s'appelle de la manière suivante :
----> mini_clear

Pour QUITTER le mini_shell : exit

--> Si vous avez besoin d'aider pour une fonction <--
----> mini_help name_fonction
##############################################################################################################################################



mini_shell > mini_echo salut je suis le texte !

>>> salut je suis le texte !

######################################## Fin du processus fils  --> Processus père en attente de commande ..... ########################################

mini_shell > mini_head 2 ./test_txt/test.txt

>>> 1Salut      je suis Maxime          BEGOUD
          2Salut     je     suis       Maxime     BEGOUD

######################################## Fin du processus fils  --> Processus père en attente de commande ..... ########################################

mini_shell > mini_cat ./test_txt/test.txt

>>> 1Salut      je suis Maxime          BEGOUD
          2Salut     je     suis       Maxime     BEGOUD
3Salut      je       suis         Maxime      BEGOUUD
     4Salut      je suis Maxime          BEGOUD
5Salut      je suis         Maxime        BEGOUUD

######################################## Fin du processus fils  --> Processus père en attente de commande ..... ########################################

mini_shell >
```

nombre de caractères copiés

3. ***int*** *mini_strcmp(**char*** *s1, **char*** *s2)* : compare les chaînes s1 et s2, retourne 0 si identique

Exercice 22 Quels sont les problèmes de ces fonctions (notamment en terme de sécurité)? Proposer une correction de ces fonctions.

fonction recopiera les éléments dans *buffer_write* et mettra a jour l'index *ind_write*. L'écriture, via un *write*, ne sera déclenchée que lorsque le *buffer_write* sera plein.

buffer_write sera alloué au premier appel de *mini_fwrite*, d'une taille *IOBUFFER_SIZE* et l'index *ind_write* sera positionné à 0. Cet index sera incrémenté à chaque écriture.

Cette fonction renvoie -1 en cas d'erreur ou le nombre de caractères écrits.

Exercice 32 Ajouter le code à votre *main* pour tester cette fonction.

demander si ok **Exercice 33** Ajouter une fonction ***int*** *mini_fflush(MYFILE* file)* qui va forcer l'écriture (via un *write*) des données non-écrites présentes dans *buffer_write*. Cette fonction renvoie -1 en cas d'erreur ou le nombre de caractère écrit.

demander si ok **Exercice 34** Que se passe-t'il si le programme se termine alors que le buffer d'écriture n'était pas plein ? Ajouter le code permettant de corriger ce problème dans *mini_exit* (Pensez a ajouter une liste des fichiers ouverts pour pouvoir tous les flusher)

demander si ok **Exercice 35** Ajouter la fonction ***int*** *fclose(MYFILE* file)* qui ferme le fichier, i.e, le flush, le supprime de la liste, et utilise l'appel système *close* pour le fermer. Cette fonction retourne -1 en cas d'erreur.

Exercice 36 Ajouter la fonction ***int*** *mini_fgetc(MYFILE* file)* qui renvoie un caractère lu, -1 en cas d'erreur.

Exercice 37 Ajouter la fonction ***int*** *mini_fputc(MYFILE* file, **char** c)* qui écrit un caractère, -1 en cas d'erreur.

demander si ok **Exercice 38** Ajouter le code à votre *main* pour tester cette fonction.

2 Commandes Système

En utilisant vos fonctions, implémenter les programmes suivants :

marche mais segf **Exercice 39** *mini_touch file* qui crée un fichier vide *file* si il n'existe pas.

marche mais caractère null **Exercice 40** Implémenter la commande *mini_cp src dest* qui réalise une copie du fichier *src* dans *dest*. Proposer une méthode pour benchmarker votre programme (et le comparer au résultat du TD précédent)

- *mini_echo chaine* qui affiche à l'écran la chaîne passée en paramètre (qui peut contenir des espaces !)

taille buffer à revoir
- *mini_cat* : prend en paramètre un fichier ; affiche son contenu
- *mini_head -n N* : prend en paramètre un fichier ; affiche les N premières lignes
- *mini_tail -n N* : prend en paramètre un fichier ; affiche les N dernières lignes
- *mini_clean* : prend en paramètre un fichier ; crée un fichier vide si il n'existe pas, ou le remet à zéro si il existe
- *mini_grep* : prend en paramètre un mot et un nom de fichier, affiche toutes les lignes contenant ce mot
- *wc* : prend en paramètre un nom de fichier, affiche le nombre de **mots** du fichier

Nous implémenterons d'autres fonctions lors du prochain TP

3 SHELL

En utilisant vos fonctions, implémenter les programmes suivants :

un "mini_clear" à été exécuté entre temps