Piscine - C - Tek1

Sujet Jour 09

Responsable Astek astek_resp@epitech.eu

Table des matières

Consignes

- Le sujet peut changer jusqu'à une heure avant le rendu.
- Vos exercices doivent être à la norme.
- Vous ne devez pas avoir de `main()` dans aucun fichier de votre répertoire de rendu.
- Pour chaque repertoire de chaque exercice nous allons compiler vos fichiers avec la commande `cc -c *.c`, ce qui va génerer tous les fichiers `.o` que nous allons ensuite linker un par un en y ajoutant notre `main.c` et notre `my_putchar.c` :

```
$> cd ex\_01
$> cc -c *.c
$> cc *.o ~moulinette/main\_ex\_01.o ~moulinette/my\_putchar.o -o ex01
$> ./ex01
[...]
```

- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.
  Si un seul de vos fichiers empêche la compilation avec `*.c`, la moulinette ne pourra pas vous corriger et vous aurez `0`. Vous avez donc tout intéret à effacer vos rendus d'exercices ne fonctionnant pas.
- Vous n'avez le droit qu'à la fonction `my_putchar` pour faire les exercices qui suivent. Cette fonction sera fournie, donc :
  - vous ne devez pas avoir lors du rendu de fichier `my_putchar.c`
  - la fonction `my_putchar` ne doit être mise dans aucun des fichiers rendus
- Pensez à en discuter sur le forum piscine !
- Travaillez en local !
  C'est-à-dire que pour chaque exercice vous devez le compiler sur votre compte linux puis, une fois qu'il fonctionne, le copier sur votre compte AFS.
  Ceci dans le simple but de ne pas surcharger les serveurs car vous êtes nombreux.

  

  Indices `Faites-vous un script shell pour copier vos fichiers sur l'AFS`

- Rendu :
  `/afs/epitech.net/users/group/login/rendu/piscine/Jour_09`
- Nous compilons avec votre lib et vos includes. Il doit donc y avoir les droits adéquats :
  - En lecture et exécution sur les répertoires
    `/afs/epitech.net/users/group/login/rendu/lib/my` et
    `/afs/epitech.net/users/group/login/rendu/include`
  - En lecture sur `/afs/epitech.net/users/group/login/rendu/lib/my/libmy.a` et `/afs/epitech.net/users/group/login/rendu/include/my.h`
- Donc vérifiez tous vos droits avant de venir nous demander pourquoi vous avez 0.
- De plus, le programme de correction utilise votre lib. Si votre `my_str_to_wordtab` ou `my_show_wordtab` ne marche pas, n'essayez même pas de faire les exercices 4 et 5.

Exercice 01 : my_macroABS.h

- Écrire une macro `ABS` qui remplace un paramètre par sa valeur absolue :

```
1   #define ABS(Value)
```

- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_09/ex_01/my_macroABS.h`

Exercice 02 : my.h

- Écrire votre fichier `my.h`
- Il contient tous les prototypages de vos fonctions de votre `libmy.a`.
- Rendu :
  `/afs/epitech.net/users/group/login/rendu/include/my.h`

Exercice 03 : my_param_to_tab

- Écrire une fonction qui stocke les paramètres du programme dans un tableau de structures et qui renvoie l'adresse de la première case du tableau.
- Tous les éléments du tableau devront être traités, y compris `av[0]`.
- Elle devra être prototypée de la façon suivante :

```
1   struct s_stock_par *my_param_to_tab(int ac, char **av);
```

- Le tableau de structures devra être alloué, et la dernière case contiendra 0 dans son élément `str` pour signaler la fin.
- La structure est définie comme suit :

```
1   struct s_stock_par
2   {
3     int size_param;
4     char *str;
5     char *copy;
6     char **tab;
7   };
```

  - `size_param` étant la longueur du paramètre
  - `str` étant l'adresse du paramètre
  - `copy` étant la copie du paramètre
  - `tab` étant le tableau retourné par `my_str_to_wordtab`
- Vous ne devez pas rendre la structure `struct s_stock_par`, la moulinette utilisera la sienne, ainsi que le typedef :

```
1   typedef struct s_stock_par t_stock_par;
```

- Nous testons votre fonction avec `my_show_wordtab`. Prenez les mesures nécessaires pour que ça marche (on ne compile pas de `my_show_wordtab.c`)
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_09/ex_03/my_param_to_tab.c`

Exercice 04 : my_show_tab

- Écrire une fonction qui affiche le contenu d'un tableau créé par la fonction précédente.
- Elle devra être prototypée de la façon suivante :

```
1   int my_show_tab(struct s_stock_par *par);
```

- Vous ne devez pas rendre la structure `struct s_stock_par`, la moulinette utilisera la sienne, ainsi que le typedef :

```
1   typedef struct s_stock_par t_stock_par;
```

- Pour chaque case, on affiche (un élément par ligne) :
  - le paramètre
  - la taille
  - chaque mot (un par ligne)
- Nous testons votre fonction avec `my_show_wordtab`. Prenez les mesures nécessaires pour que ça marche (on ne compile pas de `my_show_wordtab.c`)
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_09/ex_04/my_show_tab.c`

Exercice 05 : is_big_endian

- Écrire une fonction qui renvoie `1` si la machine est en big endian, et `0` autrement
- Jetez un coup d'oeil au cours pour la notion de big endian et de little endian.
- Elle devra être prototypée de la façon suivante :

```
1   int is_big_endian(void);
```

- Pensez à la tester sur la machine maya (ssh maya.epitech.net).
  - Sur maya gcc se trouve là : `/usr/sfw/bin/gcc`
  - Si maya ne fonctionne pas demandez à vos responsables de piscine sur quelle machine vous connecter.
- Rendu :
  `/afs/epitech.net/users/group/login/rendu/piscine/Jour_09/ex_05/is_big_endian.c`