Piscine - C - Tek1

Sujet Jour 06

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 avoir de `main()` dans aucun fichier de votre repertoire 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 empèche 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`

- Presque toutes ces fonctions existent dans la librairie "string". Pour avoir une explication complète sur le fonctionnement d'une des fonctions, il suffit de faire `man`.

Indices `Pour my_strcpy : man strcpy`

- Il est évident, qu'aucune des fonctions suivantes ne doit contenir des fonctions de la librairie "string".
- Rendu :
  `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06`

Exercice 0

- Le répertoire `Jour_06` doit contenir un fichier nommé `auteur` contenant votre login suivi d'un `'\n'`.
- Exemple :

```
foo_b@moulinette> cat -e auteur
foo_b$
```

Exercice 1 - my_strcpy

- Écrire une fonction qui copie une chaîne dans une autre. La chaîne de destination aura déjà la mémoire suffisante pour copier la chaîne source.
- Elle devra être prototypée de la façon suivante :

```
1   char *my_strcpy(char *dest, char *src);
```

- Elle doit renvoyer `dest`.
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_01/my_strcpy.c`

Indices `Indice : man strcpy`

Exercice 2 - my_strncpy

- Écrire une fonction qui copie `n` caractères d'une chaîne dans une autre. La chaîne de destination aura déjà la mémoire suffisante pour contenir `n` caractères. Ajouter `'\0' si n > à la longueur de la chaîne`. Ne pas ajouter `'\0' si n < à la longueur de la chaîne` (car `dest` n'est pas censé pouvoir contenir plus de `n` octets).
- Elle devra être prototypée de la façon suivante :

```
1   char *my_strncpy(char *dest, char *src, int n);
```

- Elle doit renvoyer `dest`.
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_02/my_strncpy.c`

Indices `Indice : man strncpy`

Exercice 3 - my_revstr

- Écrire une fonction qui inverse une chaîne de caractères.
- Elle devra être prototypée de la manière suivante :

```
1   char *my_revstr(char *str);
```

- Elle doit renvoyer `str`.
- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_03/my_revstr.c`

Exercice 4 - my_strstr

- Reproduire le fonctionnement de la fonction `strstr`.
- Elle devra être prototypée de la façon suivante :

```
1   char *my_strstr(char *str, char *to_find);
```

- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_04/my_strstr.c`

Indices `Jetez un oeil aux fonctions my_strcmp et my_strncmp`

Exercice 5 - my_strcmp

- Reproduire le fonctionnement de la fonction `strcmp`.
- Elle devra être prototypée de la façon suivante :

```
1   int my_strcmp(char *s1, char *s2);
```

- Elle devra renvoyer des valeurs identiques à son homologue strcmp(3)
- Rendu :
  `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_05/my_strcmp.c`

Indices `man 3 strcmp`

Exercice 6 - my_strncmp

- Reproduire le fonctionnement de la fonction `strncmp`.
- Elle devra être prototypée de la façon suivante :

```
1   int my_strncmp(char *s1, char *s2, int n);
```

- Elle devra renvoyer des valeurs identiques à son homologue strncmp(3)
- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_06/my_strncmp.c`

Exercice 7 - my_strupcase

- Écrire une fonction qui met en majuscule chaque lettre de chaque mot.
- Elle devra être prototypée de la façon suivante :

```
1   char *my_strupcase(char *str);
```

- Elle devra renvoyer `str`.
- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_07/my_strupcase.c`

Exercice 8 - my_strlowcase

- Écrire une fonction qui met en minuscule chaque lettre de chaque mot.
- Elle devra être prototypée de la façon suivante :

```
1   char *my_strlowcase(char *str);
```

- Elle devra renvoyer `str`.
- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_08/my_strlowcase.c`

Exercice 9 - my_strcapitalize

- Écrire une fonction qui met en majuscule la première lettre de chaque mot et le reste du mot en minuscule.
- Elle devra être prototypée de la façon suivante :

```
1   char *my_strcapitalize(char *str);
```

- Elle devra renvoyer `str`
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_08/my_strcapitalize.c`

Indices `"salut, comment ca va ? 42mots quarante-deux ; cinquante+et+un"` donne `"Salut, Comment Ca Va ? 42mots Quarante-Deux ; Cinquante+Et+Un"`

Exercice 10 - my_str_isalpha

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des caractères alphabétiques et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

```
1   int  my_str_isalpha(char *str);
```

- Elle devra renvoyer 1 si `str` est une chaîne vide.
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_10/my_str_isalpha.c`

Exercice 11 - my_str_isnum

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des chiffres et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

```
1   int my_str_isnum(char *str);
```

- Elle devra renvoyer 1 si `str` est une chaîne vide.
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_11/my_str_isnum.c`

Exercice 12 - my_str_islower

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des caractères alphabétiques en minuscule et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

```
1   int my_str_islower(char *str);
```

- Elle devra renvoyer 1 si `str` est une chaîne vide.
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_12/my_str_islower.c`

Exercice 13 - my_str_isupper

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des caractères alphabétiques en majuscule et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

```
1   int my_str_isupper(char *str);
```

- Elle devra renvoyer 1 si `str` est une chaîne vide.
- Rendu : `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_13/my_str_isupper.c`

Exercice 14 - my_str_isprintable

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des caractères affichables et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

```
1   int my_str_isprintable(char *str);
```

- Elle devra renvoyer 1 si `str` est une chaîne vide.
- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_14/my_str_isprintable`

Indices `man isprint`

Exercice 15 - my_putnbr_base

- Écrire une fonction qui affiche un nombre à l'écran, dans une base donnée.
- Ce nombre est fourni sous la forme d'un `int`, et la base sous la forme d'une `chaîne de caractères`
- La base contient tous les symboles utilisables pour afficher le nombre :
  - `0123456789` est la base décimale couramment utilisée pour représenter nos nombres
  - `01` est une base binaire
  - `0123456789ABCDEF` une base hexadecimale
- La fonction devra renvoyer le nombre passé en paramètre
- La fonction doit gérer les nombres negatifs.
- Elle devra être prototypée de la façon suivante :

```
1   int my_putnbr_base(int nbr, char *base);
```

- Rendu :
  `/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_15/my_putnbr_base.c`

Exercice 16 - my_getnbr_base

- Écrire une fonction qui renvoie un nombre.
- Ce nombre est connu sous la forme d'une `chaîne de caractères`.
- La chaîne de caractères exprime le nombre dans une base particulière, passée en second paramètre.
- La fonction doit gérer les nombres négatifs.
- La fonction doit gérer plusieurs signes + ou - à la suite.
- Si un paramètre contient une erreur la fonction renvoie 0.
  - str est une chaîne vide
  - la base est vide
  - str contient des caractères qui ne sont pas dans la base
  - la base contient 2 fois le même caractère
  - ...
- Elle devra être prototypée de la façon suivante :

```
1   int my_getnbr_base(char *str, char *base);
```

- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_16/my_getnbr_base.c`

Exercice 17 - my_showstr

- Écrire une fonction qui affiche une chaîne de caractères à l'écran. Si cette chaîne contient des caractères non-imprimables, ils devront être affichés sous forme hexadecimale (en minuscules) en les précédant d'un "backslash".
- Elle devra être prototypée de la façon suivante :

```
1   int my_showstr(char *str);
```

- La fonction renvoie toujours 0.
- Rendu :
`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_17/my_showstr.c`

Indices `my_showstr("Coucou\nca va ?") affiche "Coucou\0aca va ?"`

Exercice 18 - my_showmem

- Écrire une fonction qui affiche une zone mémoire à l'écran.
- L'affichage de la zone mémoire est séparée en trois colonnes :
  - L'adresse en hexadécimal du premier caractère de la ligne.
  - Le contenu en hexadécimal.
  - Le contenu en caractères imprimables.
- Si un caractère est non imprimable il sera remplacé par un point.
- Chaque ligne doit gérer 16 caractères.

```
?>./my_showmem
00000000: 5361 6c75 7420 6c65 7320 616d 696e 6368  Salut les aminch
00000010: 6573 2063 2765 7374 2063 6f6f 6c20 7368  es c'est cool sh
00000020: 6f77 206d 656d 206f 6e20 6661 6974 2064  ow mem on fait d
00000030: 6520 7472 7563 2074 6572 7269 626c 6500  e truc terrible.
00000040: 2e00 0102 0304 0506 0708 090e 0f1b 7f    ..............
?>./my_showmem | cat -te
00000000: 5361 6c75 7420 6c65 7320 616d 696e 6368  Salut les aminch$
00000010: 6573 2063 2765 7374 2063 6f6f 6c20 7368  es c'est cool sh$
00000020: 6f77 206d 656d 206f 6e20 6661 6974 2064  ow mem on fait d$
00000030: 6520 7472 7563 2074 6572 7269 626c 6500  e truc terrible.$
00000040: 2e00 0102 0304 0506 0708 090e 0f1b 7f    ..............$
?>
```

- Cette fonction renvoie toujours 0.
- Elle devra être prototypée de la façon suivante :

```
1   int my_showmem(char *str, int size);
```

- Rendu :

`/afs/epitech.net/users/group/login/rendu/piscine/Jour_06/ex_18/my_showmem.c`

Indices Attention à bien aligner même s'il manque des caractères