



Piscine - C - Tek1

Sujet Jour 07

Responsable Astek astek_resp@epitech.eu



Table des matières

Consignes	2
00 - Exam machine	3
01 - libmy.a	4
01-1 - my_strcat	5
01-2 - my_strncat	6
01-3 - my_strlcat	7
01 - 3 (suite) - man strlcat	8
02 - my__aff__params	9
03 - my__rev__params	10
04 - my__sort__params	11



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`, en y ajoutant notre `main.c` et votre lib :

```
$> cd ex_01
$> cc *.c ~moulinette/main_ex_01.o -L/u/all/login/rendu/lib -o ex01 -lmy
$> ./ex01
[...]
```

- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement spécifiés 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 intérêt à effacer vos rendus d'exercices ne fonctionnant pas.



pour tous les exercices autres que celui de la lib, vos fichiers ne doivent pas contenir les fonctions demandées dans la lib.

- 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_07`



00 - Exam machine

- Regarder les traces de l'examen de samedi qui se trouvent dans `/afs/epitech.net/users/group/login/epreuve/` et comprendre votre note. après chaque examen il doit y avoir des traces dans votre dossier epreuve. Pensez à les regarder pour tirer profit de vos erreurs.



01 - libmy.a

- Créer votre bibliothèque my, elle se trouvera dans `/afs/epitech.net/users/group/login/rendu/lib/my/` et s'appellera `libmy.a`.
- Cette bibliothèque doit contenir toutes les fonctions suivantes :

```
1  void my_putchar(char c);
2  int my_isneg(int nb);
3  void my_put_nbr(int nb);
4  void my_swap(int *a, int *b);
5  void my_putstr(char *str);
6  int my_strlen(char *str);
7  int my_getnbr(char *str);
8  void my_sort_int_tab(int *tab, int size);
9  int my_power_rec(int nb, int power);
10 int my_square_root(int nb);
11 int my_is_prime(int nombre);
12 int my_find_prime_sup(int nb);
13 char *my_strcpy(char *dest, char *src);
14 char *my_strncpy(char *dest, char *src, int nb);
15 char *my_revstr(char *str);
16 char *my_strstr(char *str, char *to_find);
17 int my_strcmp(char *s1, char *s2);
18 int my_strncmp(char *s1, char *s2, int nb);
19 char *my_strupcase(char *str);
20 char *my_strlowercase(char *str);
21 char *my_strcapitalize(char *str);
22 int my_str_isalpha(char *str);
23 int my_str_isnum(char *str);
24 int my_str_islower(char *str);
25 int my_str_isupper(char *str);
26 int my_str_isprintable(char *str);
27 int my_showstr(char *str);
28 int my_showmem(char *str, int size);
29 char *my_strcat(char *dest, char *src);
30 char *my_strncat(char *dest, char *src, int nb);
31 int my_strlcat(char *dest, char *src, int size);
```

- Votre bibliothèque `libmy.a` devra impérativement être à sa place car elle sera utilisée pour compiler tous vos autres programmes (vous ne devez plus mettre dans votre rendu le code source des fonctions qui sont dans votre lib).
- Les droits habituels doivent s'y appliquer :
 - 710 pour le dossier `rendu`
 - 710 pour le dossier `lib`
 - 750 pour le dossier `my`
 - 640 pour le fichier `libmy.a`
- Rendu :
`/afs/epitech.net/users/group/login/rendu/lib/my/libmy.a`



Pour ce jour vous devez en plus coder les 3 fonctions des exercices suivants à inclure dans votre lib my



01-1 - my_strcat

- Écrire une fonction qui copie une chaîne de caractères à la suite d'une autre (voyez le man système associé).
- Elle devra être prototypée de la façon suivante :

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

- Rendu :
/afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_01/my_strcat.c



Indices man strcat



01-2 - my_strncat

- Écrire une fonction qui copie **n** caractères d'une chaîne de caractères à la suite d'une autre.
- Elle devra être prototypée de la façon suivante :

```
1 char *my_strncat(char *dest, char *src, int nb);
```

- Rendu :
/afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_01/my_strncat.c



Indices man strncat



01-3 - my_strlcat

- Écrire une fonction qui copie une chaîne de caractères à la suite d'une autre sur une longueur maximale de 1.
- Elle devra être prototypée de la façon suivante :

```
1 int my_strlcat(char *dest, char *src, int size);
```

- Rendu :
/afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_01/my_strlcat.c



Indices le man de strlcat se trouve sur la page suivante



Indices pour tester la vraie fonction pensez à vous ssh sur un freebsd

?>ssh freebsd.epitech.net



01 - 3 (suite) - man strlcat

NAME

strncpy, strlcat – size-bounded string copying and concatenation

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
size_t strncpy(char *dst, const char *src, size_t size);
size_t strlcat(char *dst, const char *src, size_t size);
```

DESCRIPTION

The **strncpy()** and **strlcat()** functions copy and concatenate strings respectively.

They are designed to be safer, more consistent, and less error prone replacements for **strncpy(3)** and **strncat(3)**.

Unlike those functions, **strncpy()** and **strlcat()** take the full size of the buffer (not just the length) and guarantee

to NUL-terminate the result (as long as size is larger than 0 or, in the case of **strlcat()**, as long as there is at least one byte free in dst).

Note that you should include a byte for the NUL in size. Also note that **strncpy()** and **strlcat()** only operate on true "C" strings.

This means that for **strncpy()** src must be NUL-terminated and for **strlcat()** both src and dst must be NUL-terminated.

The **strncpy()** function copies up to size - 1 characters from the NUL-terminated string src to dst, NUL-terminating the result.

The **strlcat()** function appends the NUL-terminated string src to the end of dst.

It will append at most size - **strlen(dst)** - 1 bytes, NUL-terminating the result.

RETURN VALUES

The **strncpy()** and **strlcat()** functions return the total length of the string they tried to create.

For **strncpy()** that means the length of src. For **strlcat()** that means the initial length of dst plus the length of src.

While this may seem somewhat confusing it was done to make truncation detection simple.

Note however, that if **strlcat()** traverses size characters without finding a NUL, the length of the string

is considered to be size and the destination string will not be NUL-terminated (since there was no space for the NUL).

This keeps **strlcat()** from running off the end of a string.

In practice this should not happen (as it means that either size is incorrect or that dst is not a proper "C" string).

The check exists to prevent potential security problems in incorrect code.

EXAMPLES

The following code fragment illustrates the simple case :

```
1 char *s, *p, buf[BUFSIZ];
2 ...
3 (void)strncpy(buf, s, sizeof(buf));
4 (void)strlcat(buf, p, sizeof(buf));
```

To detect truncation, perhaps while building a pathname, something like the following might be used :

```
1 char *dir, *file, pname[MAXPATHLEN];
2 ...
3 if (strncpy(pname, dir, sizeof(pname)) >= sizeof(pname))
4     goto too long;
5 if (strlcat(pname, file, sizeof(pname)) >= sizeof(pname))
6     goto too long;
```

Since we know how many characters we copied the first time, we can speed things up a bit by using a copy instead of an append :

```
1 char *dir, *file, pname[MAXPATHLEN];
2 size_t n;
3 ...
4 n = strncpy(pname, dir, sizeof(pname));
5 if (n >= sizeof(pname))
6     goto too long;
7 if (strncpy(pname + n, file, sizeof(pname) - n) >= sizeof(pname) - n)
8     goto too long;
```

However, one may question the validity of such optimizations, as they defeat the whole purpose of **strncpy()** and **strlcat()**.

As a matter of fact, the first version of this manual page got it wrong.

SEE ALSO

snprintf(3), **strncat(3)**, **strncpy(3)**

HISTORY

The **strncpy()** and **strlcat()** functions first appeared in OpenBSD 2.4, and made their appearance in FreeBSD 3.3.



02 - my__aff__params

- Écrire un programme qui affiche les arguments reçus en ligne de commande.
- Il s'agit ici d'un programme, vous devrez donc mettre une fonction `main` dans un `.c` du répertoire de rendu.
- La moulinette va tester votre programme de la manière suivante :

```
$> cd /afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_02/my__aff__params/  
$> cc my__aff__params.c -L/afs/epitech.net/users/group/login/rendu/lib/my -lmy  
$> ./a.out
```

- Vous devez afficher tous les arguments, y compris `argv[0]`.
- Tous les arguments devront être sur une ligne séparée.
- Dossier de rendu :
/afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_02/my__aff__params
- Exemple :

```
$> ./a.out test "Ceci est un test " retest | cat -e  
./a.out$  
test$  
Ceci est un test $  
retest$  
$>
```



03 - my_rev_params

- Écrire un programme qui affiche les arguments reçus en ligne de commande dans l'ordre inverse.
- La moulinette va tester votre programme de la manière suivante :

```
$> cd /afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_03/my_rev_params/  
$> cc my_rev_params.c -L/afs/epitech.net/users/group/login/rendu/lib/my -lmy  
$> ./a.out
```

- Vous devez afficher tous les arguments, y compris argv[0].
- Tous les arguments devront être sur une ligne séparée.
- Dossier de rendu :
/afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_03/my_rev_params
- Exemple :

```
$> ./a.out test "Ceci est un test " retest | cat -e  
retest$  
Ceci est un test $  
test$  
./a.out$  
$>
```



04 - my_sort_params

- Écrire un programme qui affiche les arguments reçus en ligne de commande triés par ordre ascii.
- La moulinette va tester votre programme de la manière suivante :

```
$> cd /afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_04/my_sort_params/  
$> cc my_sort_params.c -L/afs/epitech.net/users/group/login/rendu/lib/my -lmy  
$> ./a.out
```

- Vous devez afficher tous les arguments, y compris argv[0].
- Tous les arguments devront être sur une ligne séparée.
- Dossier de rendu :
/afs/epitech.net/users/group/login/rendu/piscine/Jour_07/ex_04/my_sort_params
- Exemple :

```
$> ./a.out test "Ceci est un test " retest | cat -e  
./a.out$  
Ceci est un test $  
retest$  
test$  
$>
```