Journée 02

Chaînes de caractères

- DaemonLab -
daemonlab@ecole-89.com

Durant cette journée, vous allez commencer à faire de la magie avec votre ordinateur. Nous abordons les chaînes de caractères et donc les manipulations manuelles de la mémoire.

Nom de code : !pac1j02
Clôture du ramassage : 24/11/2019 23:59

Médailles accessibles :

Définition des médailles à venir

INDEX

01 – Détails administratifs

Votre travail doit être envoyé via l'interface de ramassage du **TechnoCentre** :

http://technocentre.ecole89.com/ramassage

Le numéro de code présent sur ce sujet vous est propre : vous devrez le renseigner en rendant votre travail. En cas d'erreur, votre travail ne sera pas associée à l'activité et votre travail ne sera pas ramassé.

Pour cette activité, vous rendrez votre travail sous la forme d'une archive au format tar.gz. Cette archive devra contenir l'ensemble de votre travail tel que demandé dans la section 4.

Pour créer cette archive .tar.gz, il vous suffit d'utilliser la commande suivante :

```
$> tar cvfz mon_fichier.tar.gz fichier1 fichier2 fichier3
```

Le nom « mon_fichier.tar.gz » étant à remplacé par le nom que vous souhaitez donner votre fichier archive, et « fichier1 », « fichier2 », « fichier3 » par les fichiers ou dossiers que vous souhaitez mettre dans cette archive. Vous pouvez vérifier le contenu de votre archive à l'aide de la commande « **`tar -t mon_archive.tar.gz`** ».

Ce travail est a effectuer seul. Vous pouvez bien sur échanger avec vos camarades, néanmoins vous devez être l'auteur de votre travail. Utiliser le code d'un autre, c'est **tricher**. Et tricher annule **toutes** les médailles que vous avez reçu sur l'activité. La vérification de la triche est réalisée de la même manière que la correction : de manière **automatique**. Prenez garde si vous pensez pouvoir passer au travers.

Médailles accessibles :

Réussir à rendre :
Vous avez réussi à envoyer votre travail au système de correction.

02 – Propreté de votre rendu

Votre rendu, c'est à dire le contenu de l'archive ou du dépôt que vous entrez sur l'interface du TechnoCentre, doit respecter **strictement** l'ensemble des règles suivantes :

- Il ne doit contenir **aucun** fichier objet. (*.o)
- Il ne doit contenir **aucun** ichier tampon. (*~, #*#)
- Il ne doit pas contenir votre production finale (programme ou bibliothèque)

La présence d'un fichier interdit mettra immédiatement fin à votre évaluation.

Médailles accessibles :

Rendu propre
Votre rendu respecte les règles de propretés imposées.

03 – Règlement quant à la rédaction du code C

En général, le code source de votre programme doit impérativement respecter un ensemble de règles de mise en page définie par les Table de la Norme.

Pour cette activité, nous vous libérons de cette contrainte qui vous sera néanmoins très bientôt imposée pour l'ensemble de vos programmes écrits en C.

04 – Construction partielle de votre rendu

Le programme de correction va construire une sous-partie déterminée de votre rendu afin d'effectuer des tests dessus. En voici les paramètres :

- Les fichiers qui seront compilés sont ceux qui auront l'extension *.c.
- Seuls les fichiers dans le(s) dossier(s) ./ seront compilés.
- Les fichiers seront compilés sans règles de compilation particulière.

- Tous les fichiers seront compilés **ensemble**, cela signifie donc que chaque fonction doit être unique, et que vous pouvez utiliser les fonctions des autres exercices dans chaque exercice.

L'ensemble du code que vous rendez doit pouvoir être compilé.
En cas d'échec de la compilation, vous ne serez pas évalué.

Médailles accessibles :

Construction partielle
Les éléments requis de votre projet se construisent séparement.

05 – Fonctions interdites

Vous n'avez le droit à aucune autre fonction que celle précisée dans la liste ci-dessous :

- write

L'utilisation d'une fonction interdite est assimilée à de la triche.
La triche provoque l'arrêt de l'évaluation et la perte des médailles.

Voici le prototype de la fonction write, que vous trouverez dans unistd.h

```
#include <unistd.h>
ssize_t write(int fd, const char* data, size_t count);
```

La fonction write sert à écrire des données et fonctionne de la façon suivante :
- Son premier paramètre, **fd**, permet de choisir où est ce que l'on écrit. Si l'on envoi la valeur 1, cela signifie sur « la sortie standard du programme ».
- Son second paramètre, **data**, est l'adresse où se situe la donnée que l'on souhaite écrire.
- Son troisième paramètre, **count**, est la taille que l'on souhaite écrire. Généralement, c'est la taille de **data**. size_t est un type specifique de nombre entier.
- La fonction write renvoi -1 en cas d'erreur, sinon elle renvoi le nombre de bytes qu'elle à écrit. Pour plus d'information, vous pouvez taper « main 2 write » dans votre terminal.

Pour finir, write est un **appel système**. C'est un outils que vous apporte votre **système d'exploitation UNIX** et non une fonction du langage C. Les fonctions du langage C sont construites à l'aide de ce genre d'éléments. Ci-dessous, une implémentation de e89_putchar qui renvoi vrai si l'on a bien écrit le caractère demandé, afin de vous donner un exemple :

```
bool e89_putchar(char c)
{
	// Si write renvoit 1, alors 1 == 1 est vrai.
	return (write(1, &c, 1) == 1);
}
```

06 – strlen

Fichier à rendre : strlen.c

Ecrivez la fonction suivante, ainsi que son test :

```
size_t e89_strlen(const char *str);
```

Cette fonction prend en paramètre une chaîne de caractère dont elle va renvoyer la longueur calculée. Tout comme la véritable fonction strlen, vous n'avez pas à gérer le pointeur NULL : c'est à l'appelant de faire attention. Vous pouvez le faire si vous le souhaitez, mais la correction automatique ne le fera pas. Tant qu'à faire, utiliser une valeur de errno que vous jugez pertinente.

Ci-dessous, un main de test pour vous aider à vous lancer dans cette journée :

```
int main()
{
	if (e89_strlen("chaine de 12") == 12)
		e89_putchar('y') ;
	else
		e89_putchar('n') ;
	return (0);
}
```


07 – puts

Fichier à rendre : puts.c

Ecrivez la fonction suivante, ainsi qu'un test minimal vérifiant la valeur de retour :

```
int e89_puts(const char *str);
```

Cette fonction affiche la chaîne de caractère passée en paramètre suivi d'un saut de ligne. Si une erreur a lieu, la fonction renverra -1 et laissera les fonctions sous-jacentes définir errno.

08 – rputs

Fichier à rendre : rputs.c

Ecrivez la fonction suivante, ainsi qu'un test minimal vérifiant la valeur de retour :

```
int e89_rputs(const char *str);
```

Cette fonction affiche la chaîne de caractère passée en paramètre **à l'envers** suivi d'un saut de ligne. Si une erreur a lieu, la fonction renverra -1 et laissera les fonctions sous-jacentes définir errno.

09 – strcpy

Fichier à rendre : strcpy.c

Nous entrons dans une nouvelle phrase d'extension de votre part du travail !

Vous allez programmer la fonction **e89_strcpy**, la fonction **test_strcpy** qui testera **tc_strcpy** comme précédemment, mais je ne vous détaillerai plus les fonctions à faire désormais ! (Pas celle qui existent dans votre système, du moins).

De ce fait, pour savoir ce que fait la fonction **strcpy**, tapez **« man strcpy »** dans votre terminal (ou sur internet) et reproduisez-en le fonctionnement.

10 – strcmp

Fichier à rendre : strcmp.c

Implémentez les fonctions **e89_strcmp** et **test_strcmp**, basé sur la fonction **strcmp**.

11 – strchr

Fichier à rendre : strchr.c

Implémentez les fonctions **e89_strchr** et **test_strchr**, basé sur la fonction **strchr**.

12 – strrchr

Fichier à rendre : strrchr.c

Implémentez les fonctions **e89_strrchr** et **test_strrchr**, basé sur la fonction **strrchr**.

13 – memset

Fichier à rendre : memset.c

Implémentez les fonctions **e89_memset** et **test_memset**, basé sur la fonction **memset**.

14 – Aller plus loin

Implémenter les fonctions suivantes, ainsi que leurs tests :

- strnlen

- strncpy

- strlcpy

- strncmp

- strcasecmp

- strncasecmp

- strchrnul

15 – En cas de difficulté

Consultez le manuel de la fonction **bzero** et écrivez **e89_bzero** qui en est un clone ainsi que **test_bzero** qui teste la fonction **tc_bzero** qui sera apportée par le système de correction.