
# Panneau défilant

## Peu d'espace, un long message, que faire ?

- DaemonLab -
daemonlab@ecole-89.com

*Cet exercice à réaliser en équipe consiste à afficher sur un cours espace une chaîne de caractère défilante.*

Nom de code : !pac1we1
Clôture du ramassage : 17/11/2019 23:59

Médailles accessibles :



# 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 première 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 règle de compilation particulière.
- Les fichiers seront compilés séparément.

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
- usleep

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.



# 06 – text_scrolling

Écrivez la fonction suivante :

```
void e89_text_scrolling(const char      *str,
                        int             size,
                        int             stop);
```

Cette fonction effectue **stop** action. Une action consiste à afficher du texte suivi d'un **retour chariot '\r'.** Le texte affiché est celui situé dans **str** et celui-ci est affiché maximum de **size** caractère.

La façon d'afficher ce texte est particulière : au départ, la ligne affichée est entièrement blanche. La chaîne de caractère commence à défiler depuis la droite de la ligne et se déplace caractère par caractère jusqu'à ce que son dernier caractère est disparu sur la gauche.

Lorsque le texte a fait un tour entier, cela recommence. On ne s'arrête que si **stop** actions ont été effectuées.

La ligne affiché doit toujours contenir **size** caractère. Utilisez des espaces comme bourrage.

Ci-dessous, un exemple avec « abc » comme **str**, 5 comme **size** et 10 comme **stop**.
Pour des raisons de lisibilité, le bourrage est effectué avec un tiret« - » et un saut de ligne est réalisé à la place d'un retour chariot.

```
$> ./a.out
-----
----a
---ab
--abc
-abc-
abc--
bc---
c----
-----
----a
$>
```



# 07 – text_bouncing

Écrivez la fonction suivante :

```
void e89_text_boucing(const char        *str,
                      int               stop);
```

Cette fonction effectue **stop** action. Une action consiste à afficher du texte après avoir effectué un mouvement Le texte affiché est celui situé dans **str.** La zone d'affichage fera deux fois la taille de str et le texte se déplacera de gauche à droite puis de droite à gauche. A chaque fois, c'est le contact d'un des bords avec une des extremités des lettres qui effectue le rebond.