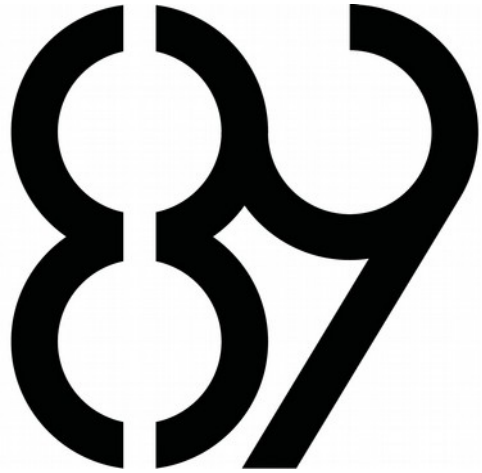
D A E M O N L A B

Programmation générale



VSNPRINTF / VSSCANF

Formatage de texte avancé
Lecture de texte avancée

- DaemonLab -
daemonlab@ecole-89.com

La fonction snprintf formate des données dans un espace mémoire délimité.
La fonction sscanf lit des données depuis une chaîne de caractère.

Vous devez choisir entre vsnprintf et vsscanf.

INDEX

Avant-propos :

01 – Détails administratifs
02 – Propreté de votre rendu
03 – Règlement quant à la rédaction du code C
04 – Construction de votre rendu
05 – Fonctions interdites
06 – vsnprintf
07 – vsscanf

01 – Détails administratifs

Votre travail doit être envoyé via l'interface de ramassage de l'Infosphère.

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

Vous devez respecter les règles des Tables de la Norme, sans exception.

04 – Construction 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) ./ et sous-dossiers seront compilés.
- Les fichiers seront compilés avec -W -Wall -Werror.

- 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 séparément.

05 – Fonctions interdites

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

- open
- close
- write
- read
- malloc
- free

Les macros va_start, va_arg et va_end ne sont pas des fonctions mais… des macros et sont donc **autorisées**.

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 – vnsprintf

Votre travail consistera à implémenter la fonction suivante :

```
int             e89_vsnprintf(char          *str,
                              size_t        size,
                              const char    *pattern,
                              va_list       ap);
```

Vous trouverez les informations dont vous avez besoin dans le manuel de snprintf.
Vous devrez implémenter au moins les formats %d, %c, %s, %p et %%.
Vous devrez implémenter les méthodes de bourrage espace et 0.

En utilisant votre fonction e89_vsnprintf, vous programmerez ensuite e89_snprintf et e89_printf.

Voici une étape intermédiaire pour vous aider :

```
bool            e89_catchar(char            *str,
                            size_t          *i,
                            size_t          max,
                            char            c);
```

La fonction catchar met à la position *i dans str le caractère c si *i est inférieur à max. Elle place le terminateur nul '\0' intelligemment de manière à ce que la fin soit toujours présente à l'endroit logique et sécuritaire.
Si le caractère c n'a pas été ajouté, elle renvoi alors faux sinon elle renvoi vrai.
Si un caractère a été ajouté, alors *i est augmenté de 1.

Cette fonction remplace aisément e89_putchar et permet d'ajouter à une chaîne de caractère existante les caractères que vous souhaitez écrire dans un buffer, par exemple, celui de e89_vnsprintf.

07 – vsscanf

Votre travail consistera à implémenter la fonction suivante :

```
int             e89_vsscanf(const char      *str,
                            const char      *pattern,
                            va_list         ap);
```

Vous trouverez les informations dont vous avez besoin dans le manuel de sscanf.
Vous devrez implémenter au moins les formats %d, %c, %s, %p, [, %n et %%.

Vous implémenterez ensuite, en utilisant e89_vsscanf les fonctions e89_sscanf et e89_scanf.