



D A E M O N L A B

TR

« Translate »

- DaemonLab -

Le programme tr permet d'effectuer des changements de caractères dans un flux de donnée. Il peut également retirer des caractères.

Ce document est strictement personnel et ne doit en aucun cas être diffusé.



INDEX

- 01 – Avant-propos
- 02 – Fonctions autorisées
- 03 – Méthode de construction

- 04 – Projet



01 – Avant-propos

Votre travail doit être rendu via le dossier `~/projets/tr/` dans votre espace personnel.

Si vous faites erreur et que le dossier que vous utilisez pour votre rendu est différent, vous ne serez pas évalué faute d'avoir pu trouver votre travail.

Ce travail est à effectuer seul. Vous pouvez bien sûr 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.

Votre rendu doit respecter **strictement** l'ensemble des règles suivantes :

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

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

Votre programme doit respecter les Tables de la Norme dans leur intégralité. Vous êtes invité à les observer depuis **l'Infosphère**. Elles sont disponibles comme ressource de cette activité.



02 – Fonctions autorisées

La bibliothèque logicielle venant avec le C est vaste et disponible. La LibLapin, que vous utilisez dans vos projets multimédia, est également vaste... Cependant nous avons fait le choix de vous interdire leurs utilisation intégrales, afin de vous amener progressivement à reprogrammer vous même ses fonctionnalités les plus utiles.

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.

Vous n'avez le droit d'utiliser aucune fonction issue de la LibC ou de la LibLapin à l'exception de celles que nous vous autoriserons explicitement.

Pouvoir utiliser une fonction ne signifie pas nécessairement que celle-ci soit utile à votre cas.

Pour cette activité, issu de la LibC, vous n'avez le droit qu'à la liste suivante :

- | | |
|---------|----------|
| - open | - write |
| - close | - alloca |
| - read | - atexit |
| - srand | - rand |
| - cos | - sin |
| - atan2 | - sqrt |

Remarquez bien l'absence de **malloc** et de **free**. En effet ils sont **interdits** ! Issu de la LibLapin, vous avez le droit aux fonctions suivantes :

- | | | |
|-------------------------|------------------------|------------------------------|
| - bunny_start | - bunny_set_*_function | - bunny_open_configuration |
| - bunny_stop | - bunny_set_*_response | - bunny_delete_configuration |
| - bunny_malloc | - bunny_loop | - bunny_configuration_getf |
| - bunny_free | - bunny_create_effect | - bunny_configuration_setf |
| - bunny_new_pixelarray | - bunny_compute_effect | - bunny_configuration_* |
| - bunny_delete_clipable | - bunny_sound_play | - bunny_set_memory_check |
| - bunny_blit | - bunny_sound_stop | - bunny_release |
| - bunny_display | - bunny_delete_sound | - bunny_usleep |

La suite sur la page d'après.



Pour utiliser **bunny_malloc**, vous pouvez soit programmer directement avec, soit en utilisant le modèle de projet qui vous a été transmis, mettre **1** dans la variable de Makefile **BMALLOC**, qui transformera **malloc** en **bunny_malloc**.

Vous appellerez **bunny_set_memory_check** au début de votre fonction **main** de sorte à provoquer une vérification de vos allocations à la fermeture du programme.

bunny_malloc, par défaut, limitera votre consommation de RAM à 20Mo.

**Pour information :
Une image en 1920*1080 fait environ 8Mo.**

Une musique en 44kHz de 1 minute en stéréo fait environ 10Mo.

Vous devrez donc disposer d'une discipline de fer avec vos allocations... et probablement trouver des compromis.

L'utilisation de **bunny_malloc** parfois **cachera** des erreurs dans votre programme, et parfois en **révélera** : son principe d'allocation était différent de **malloc**, il sera parfois plus « fort » ou plus « faible ». Ne vous mentez pas à vous en disant « l'utilisation de **bunny_malloc** fait planter mon programme », ce n'est pas **bunny_malloc**, c'est *vous*.

N'hésitez pas à l'activer, à le désactiver (Et lorsqu'il est désactivé, à utiliser **valgrind**). Et n'oubliez pas que désormais, votre demande de RAM a de véritables chances d'échouer. Car exploiter aussi peu de RAM, cela va très vite...

Dans votre rendu, il ne devra pas y avoir la moindre trace de **malloc**.



03 – Méthode de construction

Il peut vous être demandé d'écrire des programmes ou des fonctions.

Dans le cas des programmes, il vous sera toujours demandé de fournir un **dossier** pour l'exercice le requérant. Un **Makefile** vous sera également demandé. Le **nom du programme** de sortie vous sera précisé à chaque fois. Un Makefile incorrect, un mauvais nom de programme, et votre correction n'aura pas lieu...

Dans le cadre des fonctions, il vous a demandé de fournir le fichier dans votre **dossier de bibliothèque personnelle**, de sorte à ce que vous puissiez utiliser toutes les fonctions que vous avez déjà réalisé jusqu'ici. Pour rappel, le dossier de votre bibliothèque doit être placé à la racine de votre espace personnel et s'appeler **libstd/**.

N'oubliez pas d'entretenir avec soin votre dossier **libstd/** de sorte à ce qu'il soit toujours propre, respecte la norme et soit en état de compiler... sans quoi elle fera obstacle à la correction.

Votre compilation devra toujours comporter les options **-W**, **-Wall** et **-Werror**.

Dans le cadre de la programmation multimédia, le système de correction établira toujours la variable d'environnement **BMALLOC** à 1. Si vous utilisez le modèle de projet, cela provoquera l'utilisation de **bunny_malloc** dans votre bibliothèque personnelle comme dans votre projet rendu.



06 – Projet

Le logiciel **tr** pour « translate » est un logiciel permettant d'effectuer des remplacements de caractère dans un flux et de supprimer des caractères, entre autres.

Écrivez le programme dont l'utilisation est la suivante :

```
$> ./tr -h
Usage is :
    ./tr SET1 SET2
    ./tr -d [-c]? SET1
```

Le programme **tr** dispose de deux fonctionnements. Le premier nécessite deux paramètres : SET1 et SET2. Il s'agit de deux chaînes de caractères de même taille. L'objectif de ce mode de fonctionnement est de remplacer chaque caractère dans SET1 par un caractère dans SET2 situé à la même position.

Le second mode de fonctionnement est celui avec l'option « -d ». Le programme se contentera d'effacer tous les caractères situés dans SET1. Si l'option « -c » est ajouté, alors le programme enlèvera tous les caractères non situés dans SET1.

Quelques exemples d'utilisations :

```
$> echo "Salut !" | ./tr "al" "ki"
Skiut !
$> echo "Beaaaark" | ./tr -d "a"
Berk
$> echo "Beaaaark" | ./tr -d -c "a"
aaaa
```

Les appels à read avec une taille ridicule comme 1, 2... tout ce qui vous éviterait de traiter une taille quelconque, ne vous permettra pas d'acquiescer toutes les médailles accessibles de ce projet.