D A E M O N L A B

TREE

Programmez la commande tree.
Faites pousser un bel arbre.

- DaemonLab -

INDEX

01 – Avant-propos

Votre travail doit être rendu via le dossier **~/colle/tree/** 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 a effectuer en binômes aléatoires. Vous pouvez communiquer avec votre binôme mais pas avec les autres équipes. Votre équipe doit être l'auteur de votre travail. Utiliser le code d'une autre équipe, 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
- close
- read
- opendir
- readdir
- closedir
- write
- stat
- malloc
- alloca
- free
- chdir

03 – Méthode de construction

Il va vous être demandé de rendre 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…

Des fonctions peuvent être demandés explicitement, mais dans le cadre de cette activité, cela ne signifie rien d'autre qu'une obligation de la fournir pleinement fonctionnelle : elle sera simplement testée séparément du reste, bien qu'il vous soit demandé de la fournir en même temps que le reste.

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.

L'objectif de cette organisation est de permettre une correction sans pour autant vous empêcher de profiter de votre travail.

06 – Projet

Votre version du logiciel tree affichera le contenu du dossier courant. Les dossiers situés dans le dossier courant seront parcouru **récursivement**. Le contenu de ces sous-dossiers sera précédé par une **indentation de deux espaces.** Les dossiers seront suivi du caractère '/'

Voici un exemple d'utilisation :

```
$> ls
abc def/ ghi jkl/
$> ls def/
mno pqr
$> ls jkl/
stu vwx/ yzA
$> ls jkl/vwx/
BCD EFG
$> tree
abc
def/
  mno
  pqr
ghi
jkl/
  stu
  vwx/
    BCD
    EFG
  yzA
```

Sur la page suivante, vous trouverez un exemple de code exploitant les fonctions dont vous avez besoin pour réaliser ce travail.

Vous aurez bien sur besoin de lire les manuels des fonctions **opendir**, **readdir** et **closedir** pour savoir tous ce que vous avez besoin de savoir.

Petit exemple de parcours du dossier courant :

```c
#include <dirent.h>

int main(void)
{
     DIR *dir ;
     struct dirent *browse;

     if ((dir = opendir("./")) == NULL)
          return (EXIT_FAILURE);
     while ((browse = readdir(dir)) != NULL)
     {
          puts(browse->d_name);
     }
     closedir(dir);
}
```