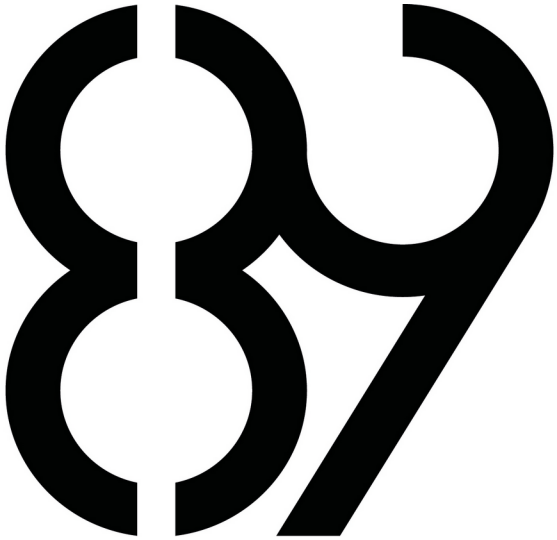

Journée 08

Manipulation de fichiers

- DaemonLab -
daemonlab@ecole-89.com

Dans cette journée, vous allez apprendre à manipuler des fichiers à l'aide des interfaces POSIX. Jetez un œil à vos fonctions autorisées.

Nom de code : !pac1j08
Clôture du ramassage : 22/12/2019 23:59

Médailles accessibles :

Définition des médailles à venir

INDEX

Avant-propos :

Exercices principaux :

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 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 séparement.

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

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 – cat

Mettre dans le fichier : cat.c

Écrivez le programme **e89_cat**.

Le programme **e89_cat** prend 0 paramètres ou plus. Tous ces paramètres sont des fichiers. Il affiche ces fichiers en l'état et quitte. Si aucun paramètre n'est donné, **e89_cat** lit sur l'entrée standard tant que celle-ci est ouverte.

Voici quelques exemples de tests qui seront effectués sur votre programme :

```
$> man malloc > manmalloc
$> man gcc > mangcc
$> ./e89_cat manmalloc mangcc
…
$> ./e89_cat
Je tape des caractères
Je tape des caractères
^D
$> cat /dev/null | ./e89_cat
```

Votre programme, en somme, doit réagir comme la commande **cat**.

Un test de performance sera effectué sur votre programme.
Celui-ci doit être le plus rapide possible !

07 – Manipulations de fichiers

Fichier : file.h et file.c

Écrivez les fonctions suivantes, ainsi que les tests associés :

```
ssize_t get_file_length(const char *file);

bool load_file(
      const char *file,
      void **data,
      size_t *len
);

bool save_file(
      const char *file,
      const void *data,
      size_t len
);
```

La fonction get_file_length renvoi la longueur en octet du fichier passé en paramètre. En cas de problème, un code erreur est renvoyé et errno est établi.

La fonction load_file renvoi vrai ou faux si elle se passe bien ou mal. Si tous se passe bien, un pointeur vers la donnée chargée depuis le fichier file sera stocké dans *data. La taille de la donnée chargée en octet sera stockée dans *len. En cas d'erreur, *data et *len seront *laissés en l'état*. N'oubliez pas errno.

La fonction save_file renvoi vrai ou faux si elle se passe bien ou mal. Si tous se passe bien, le fichier file sera crée ou modifié avec les données situés dans data sur len octets. Toute information qui se serait préalablement trouvé dans file sera effacé. En cas d'erreur, tentez de restaurez le plus possible la situation préalable. N'oubliez pas errno.

08 – Paramètres de la ligne de commande

Fichier : get_option.c

Écrivez la fonction suivante ainsi que son test :

```
char *get_option(
        int argc,
        char **argv,
        const char *opt
);
```

La fonction get_option récupère les paramètres du main ainsi qu'un paramètre décrivant le paramètre à récupérer.

Par exemple « o » cherchera à vérifier la présence de l'option « -o » dans la ligne de commande et renverra l'option elle-même (la chaîne « -o »).

Si le nom du paramètre est suivi du caractère ':', alors get_option renverra la valeur associée à l'option. Par exemple dans « -a file », elle renverra « file ».

Les valeurs de retour de get_option n'ont pas a être alloué. Seuls des pointeurs issus de argv sont renvoyés.

Si l'option demandée n'est pas trouvée, alors NULL est renvoyé.

09 – match

Fichier : match.c

Écrivez la fonction suivante ainsi que son test :

```
bool match(const char *pattern, const char *str);
```

La fonction **match** vérifie que **str** est compatible avec **pattern**.

La chaîne de caractère **pattern** peut contenir des symboles spéciaux, les symboles **?** et *****.

Le point d'interrogation signifie « n'importe quel caractère ».

L'étoile signifie n'importe quelle quantité de n'importe quel caractère (Même aucun)

Voici quelques exemples :

- Le **pattern** « c??le » sera compatible avec « colle » et « coule ».
- Le **pattern** « *.c » sera compatible avec « match.c », « puts.c » etc.
- Le **pattern** « e89_*.c » sera compatible avec « e89_putchar.c », « e89_strlen.c », etc.