DAEMONLAB

b64

ZGVjb2RlLW1vaSBzaSB0dSBwZXV4AA==

- DaemonLab -
pedagogie@ecole-89.com

J'espère que vous aimez les opérations bit à bit.

Table des matières

Détails administratifs

Votre travail doit être **envoyé par mail**, à l'adresse `infosphere@ecole-89.com` avec l'objet : « `[Prog][Rendu] b64` ».

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'utiliser la commande suivante :

```
$ tar cvfz prénom.nom.tar.gz fichier1 ficher2
fichier3
```

Le nom de l'archive étant à compléter avec votre nom et prénom, et `fichier1`, `fichier2`, `fichier3` par les fichiers ou dossiers que vous souhaitez y mettre. Vous pouvez vérifier le contenu de votre archive à l'aide de la commande `tar -t mon_archive.tar.gz`.

Ce travail est à 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 reçues 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.

Rendu

Votre rendu est sous la forme d'un projet, vous devez rendre un répertoire structuré comme tel. Avec dossiers `src/` et `include/` ainsi qu'un `makefile`.

Votre programme et vos éventuelles bibliothèques doivent être compilées avec la règle `all` du `makefile`.

Si vous utilisez des bibliothèques personnelles, veillez à les rendre avec votre projet. L'inclusion de fichiers déjà compilés est interdite. (Aucun `.o`, `.a`, `.so` ou exécutable)

Le **nom de votre binaire doit être** : `b64`.

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.

Médailles accessibles :

Rendu propre
Votre rendu respecte les règles de propretés imposées.

Règlement quant à la rédaction du code C

Votre programme doit respecter la **Table des Normes**.

Fonctions autorisées

Les fonctions autorisées sont les suivantes :
- `write`,
- `read`,
- `open`,
- `close`,
- `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.

Au cours de la ruée, si vous estimez qu'une fonction interdite est nécessaire, vous pouvez demander à ce qu'elle soit autorisée, *en vous justifiant*.

Synopsis

Vous devez faire un programme qui encode et décode des buffers et des fichiers, peu importe ce qu'ils contiennent, en utilisant la base 64.

La base 64 code 3 octets en entrée sur 4 caractères de la base en sortie.

Exemple d'une conversion de trois caractères ASCII.

Text content	M								a								n							
ASCII	77								97								110							
Bit pattern	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Index	19								22								5							
Base64-encoded	T								W								F							
	u																							

Vous devez donc manipuler les bits sous-jacents aux octets que l'on vous demande de traiter, et de les convertir 3 par 3 en base 64.

Commandes

Votre rendu prend la forme d'un programme qui s'utilise en ligne de commande avec l'un des deux verbes suivants, ainsi qu'un fichier source `src` à analyser et un fichier destination `dest` à produire.

Il est possible de remplacer `src` par un `-`, auquel cas la lecture dois se faire depuis l'entrée standard. Il est possible d'omettre `dest` ou de le remplacer par un `-`, pour que l'écriture se fasse sur la sortie standard. Il est aussi possible d'omettre `src` si `dest` l'est aussi et que l'on veut que la lecture se faire sur l'entrée standard. Exemples :

```
./b64 encode src dest

./b64 decode src dest
```

Erreurs

En cas d'erreur, le programme doit afficher sa nature et sortir avec le code 89.
Si le fichier source n'existe pas, par exemple, votre programme dois afficher :

```
$ ./b64 encode tata
tata: No such file or directory
```

Aide

Vous devez aussi implémenter une commande d'aide que l'on peut invoquer en faisant `./b64 help`.

Elle doit indiquer comment l'on doit se servir du programme, en anglais. C'est-à-dire, quelles sont ses commandes et quels paramètres prennent-elles.