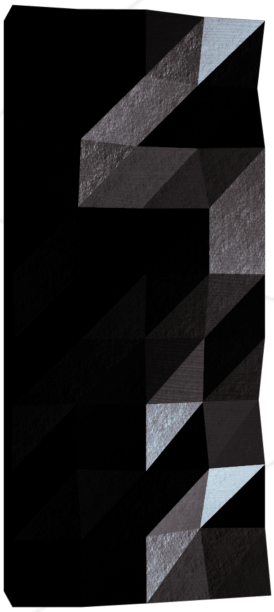

Projet du module Algorithmie I

get_next_line

Staff staff@42.fr

Résumé: Ce projet a pour but de vous faire coder une fonction qui renvoit une ligne terminée par un retour à la ligne lue depuis un descripteur de fichier.

Table des matières

Chapitre I

Préambule

James Deen, de son vrai nom Bryan Matthew Sevilla, né le 7 février 1986, est un acteur de films pornographiques américain. Egalement réalisateur, il entre très tôt dans l'industrie pornographique en 2004, alors âgé de 18 ans.

James Deen naît à Los Angeles et grandit à Pasadena. De culture juive, il raconte : "Je m'identifie avant tout à la culture du judaïsme. Je ne saurais l'expliquer, mais aussitôt que j'apprends la judéité de quelqu'un, je me sens lié à lui".

James confie avoir vu du porno pour la première fois à la maternelle, et que c'est dès ce moment que l'idée d'en faire lui est venue. Devenu adolescent, c'est en écoutant Jenna Jameson lors d'une émission de radio (Loveline que cette dernière animait) que le jeune James demanda à la star du cinéma pour adulte comment devenir une star lui-même. Celle-ci lui répondit simplement "en se masturbant devant vingt de ses meilleurs amis". Écoutant ces conseils, le jeune homme s'exécute et s'exhibe dans les soirées lycéennes. Il prend alors le nom de James Deen, qui comme le célèbre homophone, s'affiche l'air sombre en fumant une cigarette vétu de son blouson en cuir.

Fort de ces premières expériences, James décide alors de rencontrer toutes les jeunes femmes ayant un lien dans l'industrie du porno qu'il lui est possible de contacter ; il est présenté à Pamela Peaks, qui lui propose divers noms de scène, dont Clint Cullingus. Deen, qui séduit beaucoup de producteurs de films X, devra cependant faire ses preuves étant donné son très jeune âge - 18 ans - et son inexpérience à l'écran. En une seule année de carrière, James tourna dans 600 films, ce qui est un record.

En 2010, Deen est apparu dans près de 1000 films pornographiques et attribue son succès à son apparence différente des autres acteurs bodybuildés, bardés de tatouages et exhibant leur sexe surdimensionnés.

James Deen est à l'affiche en 2013 de The Canyons de Paul Schrader. Il n'a pas vu de différence majeure entre la manière dont il a tourné à Hollywood et le cinéma porno, "le sexe en moins".

FIGURE I.1 – James Deen

Chapitre II

Sujet

II.1 Partie obligatoire

- Ecrivez une fonction qui retourne une ligne lue dans un descripteur de fichier.
- On appelle “ligne” une suite de caractères terminée par un ’\n’ (code ascii 0x0a) ou bien par End Of File (EOF).

https://latedev.wordpress.com/2012/12/04/all-about-eof/ : cet article peut se montrer pertinent pour qui sait lire.

- Votre fonction aura le prototype suivant :

```
int     get_next_line(int const fd, char ** line);
```

- Le premier paramètre est le descripteur de fichier depuis lequel lire.
- Le second paramètre est l’adresse d’un pointeur sur caractère qui servira à stocker la ligne lue sur le descripteur de fichier.
- la valeur de retour peut être 1, 0 ou -1 selon qu’une ligne a été lue, que la lecture est terminée ou bien qu’une erreur est survenue respectivement.
- Votre fonction get_next_line doit renvoyer son resultat sans le ’\n’.
- Un appel en boucle à votre fonction get_next_line permettra donc de lire le texte disponible sur un descripteur de fichier une ligne à la fois jusqu’à la fin du texte.
- Assurez-vous que votre fonction se comporte bien lorsqu’elle lit depuis un fichier, depuis l’entrée standard, depuis une redirection, etc.

II.2 Rendu

- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi d'un '\n' :

```
$>cat -e auteur
xlogin$
$>
```

- Vous ne devez rendre que deux fichiers : `get_next_line.c` et `get_next_line.h`
- Il ne doit y avoir aucune fonction `main` dans votre rendu.
- Ne rendez pas de `Makefile`.
- Seul le contenu présent sur votre dépot sera évalué en soutenance.

II.3 Considérations techniques

- Votre fichier `get_next_line.h` doit au moins contenir le prototype de la fonction `get_next_line` et une macro permettant de choisir le nombre de caractères maximum lus à chaque appel de `read`. Cette valeur sera modifiée en soutenance pour évaluer la robustesse de votre rendu. Par exemple :

```
#define BUFF_SIZE 32
```

Est ce que votre code fonctionne toujours si BUFF_SIZE vaut 9999 ? Et si BUFF_SIZE vaut 1 ? Et 10000000 ? Savez-vous pourquoi ?

- Interdiction d'utiliser des variables globales pour sauvegarder les caractères qui ont été lus mais non renvoyés par votre fonction. Les variables statiques sont autorisées.

Savoir ce qu'est une variable statique est un bon début : https://en.wikipedia.org/wiki/Static_variable

- Si vous êtes malin et que vous utilisez votre biliothèque `libft` pour votre `get_next_line`, vous devez en copier les sources et le `Makefile` associé dans un dossier nommé `libft` qui devra être à la racine de votre dépot de rendu. Lors de la soutenance, votre correcteur compilera votre rendu de la manière suivante (le fichier `main.c` sera celui de votre correcteur) :

```
$>make -C libft/ fclean
$>make -C libft/
$>gcc -Wall -Wextra -Werror -I libft/includes/ -c get_next_line.c
$>gcc -Wall -Wextra -Werror -I libft/includes/ -c main.c
$>gcc -o test_gnl get_next_line.o main.o -L libft/ -lft
```

- Si au contraire vous n'êtes pas malin et que vous n'utilisez pas votre bibliothèque `libft` pour votre `get_next_line`, votre correcteur compilera votre rendu de la manière suivante lors de la soutenance (le fichier `main.c` sera celui de votre correcteur) :

```
$>gcc -Wall -Wextra -Werror -c get_next_line.c
$>gcc -Wall -Wextra -Werror -c main.c
$>gcc -o test_gnl get_next_line.o main.o
```

II.4 Fonctions autorisées

- `read`
- `malloc`
- `free`

II.5 Bonus

Le projet `get_next_line` est simple et laisse peu de latitudes pour ajouter des bonus, mais je suis certain que vous avez beaucoup d'imagination. Si vous avez réussi parfaitement la partie obligatoire, cette section propose quelques pistes pour aller plus loin. Attention, aucun bonus ne sera comptabilisé si la partie obligatoire n'obtient pas au moins 18 sur 20.

- Réussir `get_next_line` avec une seule variable statique.
- Pouvoir gérer plusieurs descripteurs de fichiers avec votre `get_next_line`. Par exemple, si les descripteurs de fichier 3, 4 et 5 sont accessibles en lecture, alors on peut appeler `get_next_line` une fois sur 3, une fois sur 4, à nouveau une fois sur 3, puis une fois sur 5, etc, sans perdre le fil de la lecture sur chacun des descripteurs.

Chapitre III

Consignes

- Ce projet ne sera corrigé que par des humains.
- Votre projet doit être à la Norme. La Norminette sera utilisée pour vérifier la Norme qui s'applique donc dans son ensemble.
- Vous devez gérer les erreurs de façon sensible. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, bus error, double free, etc).
- Toute mémoire allouée sur le tas doit être libérée proprement quand nécessaire.
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi d'un '\n' :

```
$>cat -e auteur
xlogin$
$>
```

- Si vous choisissez de rendre ce projet en utilisant votre bibliothèque `libft`, il vous est formellement interdit d'y ajouter des fonctions spécifiques à votre rendu de `get_next_line` pour contourner les limitations de la Norme. celà sera considéré comme triche lors de la soutenance. Votre `get_next_line` doit tenir en 5 fonctions de 25 lignes maximum. Le respect de cette consigne sera méticuleusement vérifié en soutenance. Inutile de venir au bocal demander si telle ou telle fonction est acceptable car vous voulez l'ajouter à votre bibliothèque. Demandez-vous plutot si votre fonction brise cette consigne ou non et servez-vous de cette chose disgracieuse située au dessus de vos épaules. Si vous respectez cette règle vous êtes bien entendu encouragés à étendre votre bibliothèque avec des fonctions génériques dont vous avez découvert l'utilité au cours de ce projet.

Chapitre IV

Notation

La notation de `get_next_line` s'effectue en deux temps :

- En premier lieu, votre partie obligatoire sera testée. Elle sera notée sur 20 points.
- Ensuite, vos bonus seront évalués. Ils seront notés sur 5 points.
- Les bonus ne seront évalués que si votre partie obligatoire a obtenu la note de 18 sur 20.

Bon courage à tous !