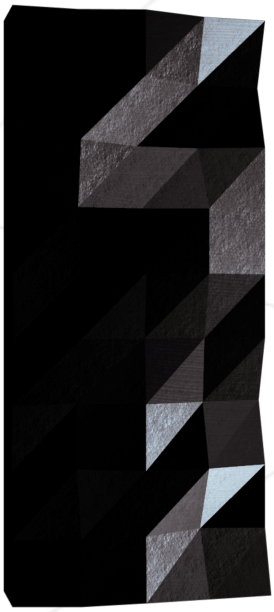

21sh

Le Retour De La Vengeance

Staff pedago pedago@42.fr

Résumé: Vous allez repartir de votre minishell et le rendre plus robuste pour vous approcher petit à petit d'un vrai shell fonctionnel. Vous allez içi ajouter divers features, telles que la gestion du multi-commande, des différentes redirections, ainsi que l'édition de ligne qui vous permettra d'utiliser les flèches du clavier par exemple.

Table des matières

Chapitre I

Préambule

I.1 Le codeur, le samouraï des temps modernes !

Faites du Kendo ! Vous apprendrez au moins la discipline.

I.2 Petite recette de Homard Bleu Breton

I.2.1 Le respect du pêcheur !

Ou un autre comme celui du Canada qui est moins cher. Le problème reste toujours le prix. En effet, le homard bleu coute un bras. Il est donc très important d'être particulièrement attentif à votre préparation.

Commencez par acheter un homard. Disons qu'il y a des saisons pour le homard. Il se pêche non pas au large, mais au contraire très proche des côtes du finistère (Pen ar bed en breton dans le texte). En particulier dans le raz de sein, a savoir entre la pointe du raz et l'ile de sein ! C'est un endroit que vous pouvez facilement trouver. Si vous voyez où est New York, alors c'est juste avant !

La description du lieu tient en quelques mots. En effet, les vents tournent violement au niveau du Pôle Nord, et creusent des vagues. Les depressions s'y mettent aussi et du coup, une énorme houle (tres grande vague avec une frequence extremement basse) se crée. Ces ondes arrivent sur les côtes du l'île de sein par le nord et "s'écrasent" sur l'île. L'onde augmente sa fréquence, et pour garder sa puissance la hauteur du signal augmente aussi.

Du coup d'énormes vagues se forment au nord-est de l'île. Une partie de la vague esquive l'obstacle en accélérant et en formant des gigantesques creux pour compenser les vagues. Ces creux s'écrasent eux-même sur la pointe du raz, et si vous ajoutez le vent, vous devez imaginer une scène absolument incroyables de plusieurs mètres de vagues et de houles se déchainant sur la côte.

Aussi étrange que cela puisse paraître, c'est dans cet envrionnement que s'épanouit notre futur diner. Le problème c'est d'aller le chercher la où il vit avec le Bar (entre autres) qui adore chasser dans ces eaux.

Du coup la liste des personnes sur le monument aux mort s'agrandit d'années en années avec quelques personnages célèbres dont le défunt ancien PDG de Michelin.

Aussi soyez respectueux du homard, certes, mais surtout du pêcheur qui est allé le pécher (ce qui explique son prix).

I.2.2 La recette elle-même

Ce n'est pas non plus hyper compliqué. Vous avez un homard vivant ? Bien !

- Prenez un énorme couteaux sans dents qui coupe à mort. Mettez votre homard sur une planche, les pattes vers le haut. Prenez ses pinces, mettez les au-dessus de sa tête, et découpez le en commençant par la tête (histoire d'abréger les souffrances).
- Prenez une grande poêle à frire au fond de laquelle vous mettrez du beurre (demi-sel n'hesitez pas sur la dose !) à fondre.
- Mettez à roussir le premier demi homard (ou les deux demi en une fois) coté chair.
- Deux options s'offrent à vous : soit vous déglacez avec du cognac (vous balancez un fond de cognac et vous faites flamber), soit vous ne faites pas. Notez qu'avec deux moités, vous pouvez imaginer faire les deux ! (dans ce cas faites pas les cons, commencez par la version non cognac).
- Vous avez bien entendu mis votre four en pré-chauffage (genre 180), mettez donc au four 15 minutes. ça marche aussi au BBQ.
- Dégustez sans tarder !

I.2.3 La double recette

`Ce serait idiot de ne pas faire celle là surtout au prix de votre homard`

A l'issue du repas, il est hérétique de jetter les coquilles à la poubelle. Sortez une énorme casserole, et mettez du beurre au fond. Faites revenir toutes les coquilles. Quand elles sont bien rousses, balancez un peu d'eau froide pour gratter le fond de votre casserole et permettre aux sucs de se decoller du fond (en gros avec une cuillere en bois, vous grattez le fond). Quand tout est décollé, ajoutez 3 litres (ou plus) d'eau, que vous portez à ébullition et faites réduire d'un tiers à frémissement. L'idée c'est que dès lors vous allez récupérer un fumé de poisson (de homard) de folie.

I.2.4 Les pates au fumé de homard

Allez acheter des pâtes. Des "de Cecco" (je tiens à la marque c'est important ce sont les seules pates qui valent le coup). Faites les cuire et en fin de cuisson, après avoir jeté

l'eau, remettez les sur le feux (au fond de la même casserole) mais en y ajoutant un peu de votre fumé précédement réservé.

I.2.5 "Et on boit quoi avec tout ça ?"

Attendez, vous venez de vous en mettre pour 80Euros de homard, vous allez pas nous mettre un pif à deux balles si ? Il n'y a pas tant que ça de solutions. La meilleure c'est de partir sur un Corton Charlemagne Grand Cru. C'est la blague a 70Euros ! En effet, le CC (Corton Charlemagne, dont je vous ferai un exposé lors du prochain sujet) a la puissance nécessaire pour supporter celle du homard. Il faut être franc, la colline de Corton c'est compliqué. Tout n'est pas Grand Cru sur cette colline. Je vous conseille plus les Corton qui sont sur Aloxe. Partons donc sur un Michel Voarick. C'est un peu rustique comme vin, mais c'est marrant de boire un Corton qui porte un nom breton sur du homard !

I.2.6 La conclusion

Au final, votre homard va vous couter dans les 150 Euro pour 4 (je vous fait grâce du paquet de pâtes). Et bien l'informatique c'est pareil. Derrière ce plat, se cache le travail de la mer, celui du pêcheur, celui du cuisinier et un millénaire de tradition viticole. Ce prix représente tout ça. Mais du point de vue du consommateur, ça reste un crustacé mort dans une assiette ! (je vous fais grâce de la théorie de Tom Reagan que je vous invite à lire cepedant). Souvenez-vous bien que lorsque vous irez vendre vos projets, le client dira que c'est trop cher sans penser aux dangers rencontrés par le Samouraï qui a bêtement tenté de pécher un homard au Katana, plutôt que d'utiliser un casier !

Chapitre II

Introduction

Grâce au projet `Minishell`, vous avez pu découvrir une partie de l'envers du décor d'un shell, tel que celui que vous utilisez tous les jours. En particulier, la création et la synchronisation de processus avec les fonctions `fork` et `wait`.

Le projet `21sh` vous propose d'aller plus loin en ajoutant, entre autres, la communication inter-processus à l'aide des pipes.

Vous pourrez découvrir, ou redécouvrir si vous avez fait le projet `ft_select`, les `termcaps`. Cette bibliothèque vous permet d'ajouter une édition de ligne à votre shell. Vous allez donc pouvoir corriger une typo dans une commande sans devoir tout retaper ou bien répéter la commande précédente grâce à un historique ! Bien entendu, il va falloir coder ces comportements. La bonne nouvelle c'est que `termcaps` vous en donne les moyens, la mauvaise c'est qu'elle ne le fera pas à votre place !

Chapitre III

Objectifs

La programmation `Unix`, c'est très bien. Les 3 projets de Shell de lécole vous permettent de découvrir une bonne partie de l'API de votre système et ça ne peut vous faire que du bien.

Cependant, les projets de Shell sont avant tout des interprètes de commandes et vous proposent une initiation à une partie extrêmement importante de l'informatique : la compilation. Les interprètes sont des programmes qui lisent et exécutent d'autres programmes, à la différence des compilateurs qui sont des programmes qui lisent et traduisent d'autres programmes vers d'autres langages. Mais interprètes et compilateurs ont plus de points communs que de différences : qu'il s'agisse d'exécuter ou de traduire un programme, il faut déjà comprendre le programme en question et savoir détecter et rejeter les programmes malformés.

Vous le savez peut-être déjà, mais l'ensemble des commandes qu'on peut envoyer à un shell forment un langage. Ce langage a des règles lexicales, syntaxiques et sémantiques que votre `21sh` va devoir respecter en effectuant un certain nombre d'étapes précises et largement documentées sur le net. Par exemple à la section "2.10 Shell Grammar" de ce document.

La clef d'un `21sh`, puis d'un `42sh` réussis sont une organisation claire et maitrisée de votre code. Vous pouvez essayer de vous convaincre qu'il n'y a qu'un simple `split` á appliquer sur les espaces de la ligne de commande pour que le tour soit joué. Pour vous éviter une perte de temps, considérez que cette solution équivaut á un aller simple vers la catastrophe.

Je vous laisse ici quelques mots clefs qu'il serait sage de comprendre : "analyse lexicale", "lexer", "analyse syntaxique", "parser", "analyse sémantique", "interprète" (ou "interpreter" en anglais), et bien sur "arbre de syntaxe abstraite" (ou "abstract syntax tree", "AST" en anglais).

Chapitre IV

Consignes

- Ce projet ne sera corrigé que par des humains. Vous êtes donc libres d'organiser et nommer vos fichiers comme vous le désirez, en respectant néanmoins les contraintes listées ici.
- L'exécutable doit s'appeller `21sh`.
- Vous devez rendre un `Makefile`. Ce `Makefile` doit compiler le projet, et doit contenir les règles habituelles. Il ne doit recompiler et relinker le programme qu'en cas de nécessité.
- Si vous êtes malin et que vous utilisez votre biliothèque `libft` pour votre `minishell`, vous devez en copier les sources et le `Makefile` associé dans un dossier nommé `libft` qui devra être à la racine de votre dépôt de rendu. Votre `Makefile` devra compiler la bibliothèque, en appelant son `Makefile`, puis compiler votre projet.
- Votre projet doit être en `C` et à la Norme. La norminette fait foi.
- Vous devez gérer les erreurs de façon raisonnée. En aucun cas votre programme ne doit quitter de façon inattendue (segmentation fault, bus error, double free, etc...).
- Votre programme ne doit pas avoir de fuites mémoire.
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi d'un '\n' :

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

- Dans le cadre de votre partie obligatoire, vous avez le droit d'utiliser les fonctions suivantes :
  - `malloc`, `free`
  - `access`
  - `open`, `close`, `read`, `write`
  - `opendir`, `readdir`, `closedir`
  - `getcwd`, `chdir`
  - `stat`, `lstat`, `fstat`
  - `fork`, `execve`
  - `wait`, `waitpid`, `wait3`, `wait4`
  - `signal`, `kill`
  - `exit`
  - `pipe`
  - `dup`, `dup2`
  - `isatty`, `ttyname`, `ttyslot`
  - `ioctl`
  - `getenv`
  - `tcsetattr`, `tcgetattr`
  - `tgetent`
  - `tgetflag`
  - `tgetnum`
  - `tgetstr`
  - `tgoto`
  - `tputs`
- Vous avez l'autorisation d'utiliser d'autres fonctions dans le cadre de vos bonus, à condition que leur utilisation soit dûment justifiée lors de votre évaluation.
- Vous pouvez poser vos questions sur le forum, sur slack, ...

Chapitre V

Partie obligatoire

Pour commencer, toutes les fonctionnalités attendues pour le `minishell` sont implicitement obligatoires pour le `21sh`. De plus, vous devez ajouter les fonctionnalités suivantes :

- une édition de la ligne de commande à l'aide de la bibliothèque `termcaps`. Voir le détail du tavail attendu ci-après.
- le séparateur de commandes "`;`"
- les pipes "`|`"
- les quatre redirections "`<`", "`>`", "`<<`" et "`>>`"
- les aggrégations de descripteurs de fichiers, par exemple pour fermer la sortie d'erreur :

```
$> ls
riri
$> rm riri; cat riri 2>&-

```

Vous un exemple représentatif des commandes que votre `21sh` doit pouvoir exécuter correctement :

```
$> mkdir test ; cd test ; ls -a ; ls | cat | wc -c > fifi ; cat fifi
.   ..
      5
$>
```

Concernant l'édition de ligne, vous devez au moins gérer les fonctionnalités suivantes. Les touches à utiliser sont des indications, vous êtes libres d'en utiliser d'autres à condition que l'utilisation de votre shell reste logique et intuitive. Votre correcteur décidera de ce qu'il entend par logique et intuitif, faites donc attention à ne pas vous laisser emporter par votre créativité.

- Editer la ligne à l'endroit où se trouve le curseur.
- Déplacer le curseur vers la gauche et vers la droite pour pouvoir éditer la ligne à un endroit précis. Les nouveaux caractères doivent bien entendu s'insérer entre les caractères existants de la même manière que dans un shell ordinaire.

- Utiliser les flèches du haut et du bas pour naviguer dans l'historique des commandes que l'on pourra alors éditer si le coeur nous en dit (la ligne, pas l'historique, hein).
- Couper, copier et/ou coller tout ou partie d'une ligne avec la séquence de touches qui vous plaira.
- Se déplacer par "mot" vers la gauche et vers la droite avec `ctrl+LEFT` et `ctrl+RIGHT` ou toute autre combinaison de touche raisonnable.
- Aller directement au début et à la fin d'une ligne avec `home` et `end`.
- Ecrire ET éditer une commande sur plusieurs lignes. Dans ce cas, on apprecierait que `ctrl+UP` et `ctrl+DOWN` permettent de passer d'une ligne à l'autre de la commande en restant sur la même colonne ou la colonne la plus appropriée sinon.
- Si une partie parenthésée de la commande n'est pas refermée avant l'appui sur la touche `return`, le shell revient à la ligne et attend la fin de la commande. Par partie parenthésée, on entend une partie de la commande entre quotes, doubles quotes, back quotes, parenthèses, crochets, accolades, etc.
- Implémenter la fonctionnalité des combinaisons de touches `ctrl+D` et `ctrl+C` dans l'édition de la ligne (sachant que le `ctrl+C` pour arrêter un programme en cours, c'est bien aussi).

Chapitre VI

Partie bonus

Les bonus ne seront évalués que si votre partie obligatoire est PARFAITE. Par PARFAITE, on entend bien évidemment qu'elle est entièrement réalisée, et qu'il n'est pas possible de mettre son comportement en défaut, même en cas d'erreur aussi vicieuse soit-elle, de mauvaise utilisation, etc. Concrètement, cela signifie que si votre partie obligatoire n'obtient pas TOUS les points à la notation, vos bonus seront intégralement IGNORÉS.

Pas mal de features seront au menu du 42sh. Voici néanmoins une liste de bonus que vous pouvez implémenter dès maintenant :

- Rechercher dans l'historique avec `ctrl+R`
- Implémentation d'une table de hash pour les binaires
- La complétion simple ou avancée avec `tab`.
- Mode bindings `Emacs` et/ou `Vim` activable et désactivable à loisir.
- Coloration syntaxique du shell activable et désactivable à loisir.
- D'autres bonus que vous jugez utiles.

Chapitre VII

Rendu et peer-évaluation

Rendez votre travail sur votre dépôt `GiT` comme d'habitude. Seul le travail présent sur votre dépot sera évalué.

Bon courage à tous et n'oubliez pas votre fichier auteur !