

TP de Synthèse – Ensea in the Shell

Séances 1 et 2 (8 h)

C. BARÈS, N. PAPAZOGLOU

Objectifs : Réaliser un micro shell, affichant les codes de sortie et les temps d'exécution des programmes lancés.

Conseils généraux :

- Le TP se fait en binôme.
- Vous devez créer un projet (pour 2) sur github et y synchroniser vos fichiers, vous devez fournir le lien github à votre encadrant avant la fin de la première séance, votre dépôt github doit être public,
- Vous êtes fortement encouragés à écrire **un** fichier par question (en copiant le fichier de la question précédente), ou de faire un commit à chaque fin de question,
- Utilisez des commentaires pertinents (pas de `:i++; //incrément de i`);
- De même, le découpage de votre programme en fonctions correctement nommées doit améliorer la lisibilité de votre code,
- Votre code doit être en anglais (noms des variables et fonctions, commentaires).
- Nommez vos constantes, n'utilisez pas de nombres « magiques »,
- N'utilisez pas le `printf`, il ne se marie pas bien avec les `read` et les `write`,
- Pour manipuler les chaînes de caractères, utiliser l'entête `string.h`, utilisez toujours les fonctions commençant par `strn...`,
- L'utilisation de la fonction `system` est interdite,
- Vous devez appeler votre professeur à chaque checkpoint (reperé par le symbole \triangle) pour qu'il valide votre avancement. S'il n'est pas disponible à ce moment précis, continuez et appelez le plus tard.
- Vous avez jusqu'à 24h après la fin du TP pour rendre votre rapport qui sera sous la forme `readme` dans votre github,

Créez un micro shell, que vous appellerez `enseash`, qui doit servir à lancer des commandes et à afficher des informations sur leur exécution.

Les fonctionnalités suivantes sont demandées, à réaliser dans l'ordre :

1. Affichage d'un message d'accueil, suivi d'un prompt simple. Par exemple :


```
$ ./enseash
Bienvenue dans le Shell ENSEA.
Pour quitter, tapez 'exit'.
enseash %
```

2. Exécution de la commande saisie et retour au prompt (REPL : read-eval-print loop) :
 - a) Lecture de la commande saisie,
 - b) Exécution d'une commande simple (sans argument),
 - c) Retour au prompt `enseash %` et attente de la commande suivante.

```
enseash % fortune
Today is what happened to yesterday.
enseash %
Sun Dec 13 13:19:40 CET 2020
enseash %
```

3. Gestion de la sortie du shell avec la commande “exit” ou un <ctrl>+d;


```
enseash % exit
Bye bye...
$
```

4.  Affichage du code de retour (ou du signal) de la commande précédente dans le prompt :

```
enseash % un_programme
enseash [exit:0] % un_autre_programme
enseash [sign:9] %
```

5. Mesure du temps d’exécution de la commande en utilisant l’appel clock_gettime :

```
enseash % un_programme
enseash [exit:0|10ms] % un_autre_programme
enseash [sign:9|5ms] %
```

6.  Exécution d’une commande complexe (avec arguments);

```
enseash % hostname -i
10.10.2.245
enseash % fortune -s osfortune
"However, complexity is not always the enemy."
-- Larry Wall (Open Sources, 1999 O'Reilly and Associates)
enseash %
```

7.  Gestion des redirections vers stdin et stdout avec ‘<’ et ‘>’;

```
enseash % ls > filelist.txt
enseash [exit:0|1ms] % wc -l < filelist.txt
44
enseash [exit:0|4ms] %
```

8. Gestion de la redirection de type pipe avec ‘|’.

```
enseash % ls | wc -l
44
enseash [exit:0|5ms] %
```

9. Renvoi du prompt immédiat avec ‘&’ (exécution de programmes en arrière plan).

- a) Définir une structure de données pour la gestion des processus en arrière plan,
- b) Utilisation d’un wait non-bloquant pour les processus en arrière plan,
- c) Gestion de l’affichage des informations pour les programmes en arrière plan
- d) Correction de la mesure des temps d’exécution (appel à wait4).

```
enseash % sleep 10 &
[1] 3656
enseash [1&] %
[1]+  Ended: sleep 10 &
enseash [exit: 0|10s] %
```