eXia Shell 2.0

Une nouvelle dimension du shell

PROJET SYSTEME & P.P.

Julio Santilario jsantilario@cesi.fr

1ère année cycle préparatoire 2015/2016

CONTENU

1. Introduction

France, 09/12/2015.

THOR, entreprise internationale spécialisée dans la défense militaire européenne, travaille depuis quelques années pour le Ministère de la Défense. Parmi les projets en technologie de pointe, on peut citer l'informatique embarquée des avions rafales ou la gestion informatisée du porte-avions Charles-de-Gaulle.

Suite aux tragiques événements du 7 janvier et du 13 novembre 2015 à Paris, l'Etat français a pris la décision de commander à THOR le développement d'un nouveau système informatique pour les services de renseignements basé sur un noyau Linux et en langage C.

Les Unités d'Enseignements « Programmation Procédurale » et « Systèmes Linux » réalisées à l'EXIA.CESI depuis le mois d'octobre, vous permettent de répondre à cet appel d'offre. Plusieurs équipes ont été sélectionnées dans toute la France pour y répondre et présenter le 18/12/2015 un prototype opérationnel du nouveau shell 2.0.

CESI exia
ÉCOLE D'INGÉNIEURS INFORMATIQUE

2. Historique du shell

Un interpréteur de commandes ou shell (CLI pour command-line interpréter en anglais) est un programme faisant partie des composants de base d'un système d'exploitation. Sa fonction est d'interpréter les commandes qu'un utilisateur tape au clavier dans l'interface en ligne de commande. Il s'agit d'une surcouche « logique » qui s'appuie sur le noyau de la machine.

Le rôle de l'interpréteur de commandes est de lire l'entrée (clavier, fichier, etc), de l'interpréter et si besoin est de lancer une commande. Ou alors d'envoyer une erreur… car les entrées doivent respecter une certaine syntaxe qui peut varier sensiblement d'un interpréteur à l'autre. Autrement dit la syntaxe du shell forme un langage de programmation de haut niveau (un langage de script directement interprété).

Les premiers systèmes capables d'interpréter des lignes de commandes sont apparus au début des années 1960, en même temps que le clavier informatique.

Sous UNIX, la ligne de commande a toujours été le moyen privilégié de communication avec l'ordinateur. Le Bourne shell (sh) est l'interpréteur originel de l'environnement UNIX. À son époque, sa grande originalité était l'utilisation de tubes (caractère « | »), qui permettent de connecter la sortie d'une commande à l'entrée d'une autre. On peut ainsi écrire des commandes complexes à partir de commandes simples.

L'invite est l'interface la plus simple à réaliser et conserve de nombreux avantages par rapport aux environnements graphiques :

- Précision et simplicité d'automatisation des tâches (mode batch) ;
- Contrôle à distance ;
- Uniformité ;
- Stabilité ;
- Faible consommation des ressources.

Le travail de tout interprète de commande peut se résumer à l'algorithme très simple suivant

```
TANT QUE l'utilisateur ne ferme pas la session
FAIRE
      # Émettre un signe d'invite (prompt)
      # Lire la ligne courante
      # Exécuter la commande indiquée sur cette ligne
FIN
```

Il y a de nombreux interpréteurs. Les plus connus sont bash (version améliorée du shell Bourne sous Unix), ksh (version améliorée du shell Korn sous Unix) et tcsh (version améliorée du shell C sous Unix). La commande help affiche la liste des commandes internes du shell. Par défaut, c'est le shell Bash qui est installé avec Linux. C'est aussi le plus puissant et le plus utilisé.

Mais ce ne sont pas les seuls. Voici la liste d'autres shells disponibles :

/bin/ash	/bin/sh
/bin/bash	/usr/bin/ksh
/bin/csh	/usr/bin/tcsh
/bin/tdash	/usr/bin/zsh

L'initialisation du shell bash, à son ouverture, se fait à l'aide de plusieurs scripts (.profile, /etc/profile, .bash_profile)

Sources :

https://fr.wikipedia.org/

http://www.linux-france.org/article/memo/node81.html

3. L'appel d'offre – SHELL 2.0

3.1.INTRODUCTION

Nous avons évoqué, dans le point 2, les avantages du shell en ligne de commandes : simplicité d'utilisation, efficacité, stabilité, ….

Tous les « accros » du système Unix puis Linux vous diront qu'il n'y a rien de mieux. Toutes les avantages, on voudrait toujours les garder. Néanmoins, il y a aussi des inconvénients. Un des inconvénients majeurs du shell est qu'il faut être initié : l'utilisateur doit apprendre et se rappeler d'un tas de commandes et de quelques paramètres pour pouvoir réaliser les tâches souhaitées et/ou obtenir les informations recherchées.

Vous pourriez ajouter au paragraphe précédent, votre propre expérience lors de la réalisation des premières manipulations sur Linux lors du module « Système Linux et Programmation Procédurale » à l'EXIA.CESI. Voici quelques commentaires entendus ici ou là sur ce système lorsqu'on le découvre :

- on répète toujours les mêmes commandes (cd, ls, …)
- il faut (presque toujours) taper à nouveau la commande.
- en plus il est « case sensitive » !
- la ligne de commande il va falloir s'y faire !
- il ne comprend que la commande et en plus elle est en anglais
- lorsqu'on affiche les résultats de la commande, il est souvent en noir/blanc.
- on doit apprendre à déchiffrer les résultats, serait-il possible d'être plus clair ?
- il est difficile à l'utiliser sans être initié

Le service de renseignement de l'Etat n'est pas seulement composé d'informaticiens. Ce sont des policiers, des magistrats, des agents de terrain qui utilisent le système pas en tant qu'experts, uniquement en tant qu'utilisateurs lambda. Eux, comme vous actuellement, sont obligés de se rappeler de quelques commandes du shell pour obtenir les informations souhaitées. Ceci n'est pas satisfaisant pour la plupart des agents et ils se plaignent de perdre beaucoup de temps. C'est aussi très problématique pour les nouvelles embauches, car le temps de formation est très long.

Les ingénieurs de THOR ont fait appel aux étudiants de l'EXIA.CESI pour leur demander de les aider à penser et à implémenter le SHELL 2.0. Puis pour leur permettre de laisser leur marque de fabrique, THOR propose de rajouter un « Easter Eggs » en définissant eux même la séquence d'activation.

https://www.digitalocean.com/community/tutorials/top-10-linux-easter-eggs

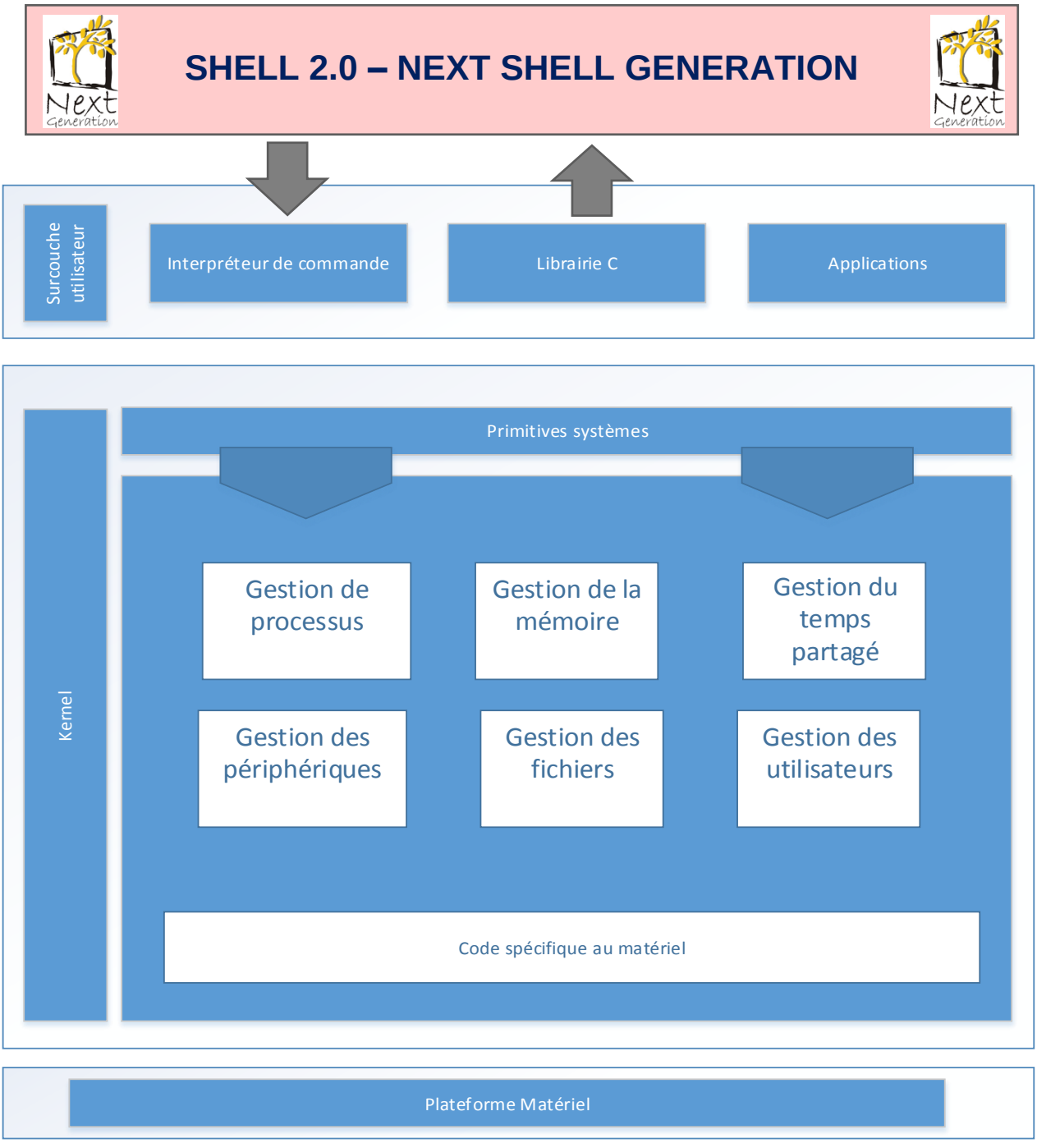
3.2. Specifications fonctionnelles

Le principe de fonctionnement du SHELL 2.0 est très simple : la machine doit s'adapter à l'homme et pas l'homme à la machine. Cette évolution est dans l'ère du temps des systèmes « intelligents » qui vont s'intégrer dans l'usine du futur. Le SHELL 2.0 fait partie d'un projet de plus grande envergure, qui permettra une modernisation des systèmes informatiques et une efficacité accrue de nos services de renseignements afin de prévenir, le plus possible, le risque d'attentats.

Voici les fonctionnalités souhaitées pour le SHELL 2.0 :

- Il doit être lancé lors de la connexion d'un utilisateur au système
- Le shell doit pouvoir être configurable (équivalent .profile). Par exemple, pouvoir configurer le prompt, le mode, la langue.
- 3 modes d'utilisation du shell doivent exister :
    - mode expert, commandes comme le shell classique
    - mode assistant pas à pas
    - mode langage naturel
- Le système doit être capable de comprendre les demandes des utilisateurs dans leur propre langue (et pas seulement des abréviations en anglais).
- Il doit inclure un « easter-eggs » : les tours de Hanoï.
- L'affichage de commandes devra être plus simple et l'interprétation du contenu plus intuitif.
- Le shell tient à jour un fichier log qui trace toutes les commandes lancées par le shell (même celles qui sont erronées).

Voici, ci-dessous, un schéma d'architecture pour mieux visualiser la tâche qui vous est demandée :



3.3. Specifications techniques

Lors du démarrage du shell 2.0 (à la connexion de l'utilisateur ou lors du lancement du shell), le shell doit lire dans un fichier les paramètres suivants

- Choix du mode : langage naturel, assistant, expert
- Choix de la langue de travail : anglais, français, allemand, espagnol, …
- Choix du prompt de l'invite de commande.

Voici la description des autres aspects techniques :

- Affichage d'un écran de bienvenue au shell avec quelques instructions si nécessaire.
- Initialisation de l'historique de session : soit vide, soit en rechargeant les informations apprises lors de sessions précédentes.
- Affichage de l'historique avec différents tris possibles en fonction de critères.
- Paramétrage du nombre de disques du « Easter-Eggs »
- Affichage et enregistrement du temps passé pour résoudre le « easter-eggs » et le score réalisé.
- Enregistrement du fichier de log avec toutes les informations de commandes lancées.

Vous trouverez plus de détails dans les points suivants.

1. Structures de données

Vous devez réfléchir aux structures de données (struct, tableaux, listes, listes chaînées, files, pile, …) qui seront nécessaires pour les différentes fonctionnalités du shell 2.0.

Le choix de la structure de données est le plus important et doit être fait au début du projet de façon très rigoureuse. De cela dépendront tous les algorithmes que vous allez implémenter par la suite.

Vous trouverez une grille en annexe qui vous aidera à formaliser les structures des données ainsi que les prototypes des fonctions.

Voici quelques conseils :

- Pensez à des tableaux à deux dimensions (statiques ou dynamiques)
- Pensez à regrouper des informations sur un même « objet » à travers de « struct ».
- L'utilisation des pointeurs peut sans doute vous simplifier l'implémentation de certains algorithmes et vous faciliter la manipulation de ces structures de données complexes.
- Choix entre les piles et les files en fonction des algorithmes à coder.

2. Les 3 modes du shell

- **Mode expert** : utilisé par les (super) spécialistes qui connaissent par cœur toutes les commandes du système. Toutes les commandes d'un système Linux doivent pouvoir être exécutées, en faisant un appel au shell sh (par exemple) depuis votre shell.

- **Mode assistant** ; mode à base de menus et sous-menus qui permettent à l'utilisateur de choisir l'action à réaliser avec les options de la commande. Par exemple, un premier niveau de menu pourrait présenter une liste de commandes. En fonction de l'action choisie, le shell présentera des options propres à cette commande. Ce mode doit pouvoir fonctionner avec plusieurs langues.

  Ce mode aura un premier niveau de menu qui regroupe des familles de commandes :

```
exiaShell20> LISTE DE FAMILLES DE COMMANDES
1 – Gestion de processus
2 – Gestion de programmes
3 – Gestion des utilisateurs
4 – Gestion de périphériques
5 – Gestion des fichiers/répertoires

Choix (1 -5) ?
```

  Voici un exemple en choisissant la Gestion des fichiers et répertoires :

```
exiaShell20> Qu'est-ce que vous voulez faire maintenant ?
1 – Lister un répertoire
2 – Aller dans un répertoire
3 – Créer un répertoire
4 – Supprimer un répertoire
5 – Ouvrir un fichier avec nano
…

Choix (1 -10) ?
```

  J'ai décidé de lister un répertoire. Un sous-menu va me demander des informations supplémentaires :

```
exiaShell20>  1 – LISTER UN REPERTOIRE
Qu'est-ce que vous voulez faire maintenant ?
1 – Nom du  répertoire à lister
2 – Afficher tout le contenu
3 – Afficher que les répertoires
4 – Afficher que les fichiers
5 – Afficher les fichiers qui m'appartient uniquement
6 – Afficher les fichiers exécutables
7 – Afficher en ordre alphanumérique croissant
…
10 – Exécuter
Choix (1 -10) ?
```

ATTENTION : ce n'est qu'un exemple. C'est à vous de définir plus en détails la liste d'actions et les sous-menus par action.

Lors de l'exécution de l'action, vous devrez faire un affichage plus visuel et intuitif en formatant l'affichage. Pour cela, vous utiliserez des couleurs (voir sous-section 5 ci-dessous)

Par exemple si j'affiche tout le contenu d'un répertoire, vous pouvez obtenir une sortie qui pourrait ressembler à celle-ci :

```
exiaShell20>  1 – LISTER UN REPERTOIRE
exia
cesi
fichier.c
exe
```

En rouge, « exia et cesi » car ce sont des répertoires ; « fichier.c » en vert car c'est fichier lambda puis « exe » en bleu car c'est un exécutable.

ATTENTION : ce n'est qu'un exemple. C'est à vous de définir les différents types de fichiers à colorier et leur couleur.

Dans ce mode, vous devez présenter au moins une quinzaine d'actions possibles réparties dans les 5 catégories et leurs sous-menus correspondants. Il peut exister un ou plusieurs sous-niveaux, si cela vous convient mieux dans votre implémentation. **Au moins une commande par catégorie doit être un affichage d'information (**par ex., affichage des utilisateurs, des fichiers, des processus, ...).

Comme ce mode peut fonctionner avec plusieurs langues, il n'est pas envisageable d'inclure tous les menus et sous-menus dans le code. Vous devez penser à une solution avec des fichiers externes (voir section suivante), en fonction de la langue paramétrée par l'utilisateur, sont chargés les bons libellés dans les structures en mémoire.

Au moins un algorithme de tri et de recherche doit être implémenté pour une des actions présentées.

- **Mode langage naturel** : l'utilisateur écrit « la demande » en langage naturel. Le shell doit analyser la phrase et exécuter la commande adéquate ou afficher un message d'erreur s'il n'a pas compris « la commande » à exécuter. Ce mode doit pouvoir être paramétrable et accepter au moins deux langues parmi les langues suivantes (anglais, français espagnol, allemand, italien, …).

Voici un exemple.

```
exiaShell20> Qu'est-ce que vous voulez faire maintenant ?
_
```

Le shell attend que vous lui donniez une action à faire :

```
exiaShell20> Qu'est-ce que vous voulez faire maintenant ?
J'aimerais lister le répertoire /bin
```

Un autre utilisateur aurait pu saisir une autre formulation :

```
exiaShell20> Qu'est-ce que vous voulez faire maintenant ?
voir le contenu du répertoire /bin en ordre croissant
```

ou

```
exiaShell20> Qu'est-ce que vous voulez faire maintenant ?
lister /bin
```

Dans n'importe que cas, le shell doit réaliser la même action et vous afficher le contenu du répertoire /bin. (vous pouvez réutiliser le même formatage que pour le mode assistant)

Vu la grande diversité d'utilisateurs, le mode naturel doit pouvoir comprendre plusieurs langues (pas en même temps) selon le paramétrage initial de la session de l'utilisateur.

Pour que ce mode puisse fonctionner, vous devez avoir « un dictionnaire ». Ce dictionnaire doit être externalisé (car multi-langue exigé) et lors du démarrage du shell, chargée en mémoire dans les structures que vous allez prévoir à cet effet. Vous définirez votre propre syntaxe que vous rajouterez dans le « dictionnaire ». Vous pouvez choisir de faire un fichier par langue ou un seul fichier pour toutes les langues. Votre choix doit être argumenté.

Dans ce mode, vous devez présenter aussi une dizaine d'actions possibles qui sont interprétables en langage naturel.

Certain nombre des fonctions peuvent être partagées entre les différents modes (tri, recherche, affichages, exécution de l'action, etc …)

3. Easter-eggs

Le module « **Easter-Eggs** » sera basé sur les tours de Hanoï. Il consiste à déplacer des disques de diamètres différents empilés dans une « tour départ » à une « tour d'arrivée » en passant par une « tour intermédiaire », et ceci en un minimum de coups, tout en respectant les deux règles suivantes :

- on ne peut déplacer plus d'un disque à la fois ;
- on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.

Pour faire apparaître « l'Easter-Eggs », vous devez définir la séquence d'activation. Puis il doit proposer avant de commencer, le niveau de difficulté choisi, pouvant choisir le nombre de disques entre 3 et 10.

Voici la configuration initiale lors du démarrage avec 4 disques :

```
   1|1                    |                    |
  22|22                   |                    |
 333|333                  |                    |
4444|4444                 |                    |

    T1                   T2                   T3
Mouvement ?
```

On peut dire que la tour départ est la tour 1, la tour intermédiaire est la tour 2 et la tour d'arrivée est la tour 3. Donc, pour déplacer le disque qui se trouvent en haut de la pile, il suffit de de saisir la consigne :

12

qui déplacera le disque de la tour 1 vers la tour 2 comme dans l'image suivante :

```
    |                    |                    |
  22|22                  |                    |
 333|333                 |                    |
4444|4444               1|1                   |

    T1                   T2                   T3
Mouvement ?
```

Ce module doit vérifier que le déplacement respecte les règles ainsi que la fin du « jeu » quand tous les disques se retrouvent dans le bon ordre dans la tour d'arrivée.

```
    |                    |                   1|1
    |                    |                  22|22
    |                    |                 333|333
    |                    |                4444|4444

    T1                   T2                   T3

SIMULATION TERMINEE :
Nombre de mouvements : 30
Temps : 10:56
```

Ces informations seront stockées également dans le fichier de score (voir section ci-dessous).

4. Fichiers

Comme vous avez pu constater dans la section précédente, plusieurs fichiers seront nécessaires pour l'implémentation de ce projet :

- **pour le paramétrage général du shell** : choix de la langue, choix du mode
- **pour le multi-langue** : des fichiers de menus pour le mode assistant et des fichiers dictionnaires pour le mode langage naturel
- **pour l'historique de commandes** : quel que soit le mode utilisé, toute action exécutée par le shell, doit être enregistrée dans un fichier qu'on appellera « l'historique »
- **le log de la session** : fichier des traces de tout ce qui se passe dans la session (minimum , jour et heure de lancement de l'action avec ses paramètres)
- **fichier des scores du Easter-Eggs** : date et heure, nom utilisateur, difficulté, temps passé pour résoudre le « jeu », nombre de mouvements.
  La gestion du temps se fera avec la fonction « time() ». Avec cette fonction vous allez pouvoir calculer le temps passé pour résoudre le jeu en calculant le temps au début et à la fin.

5. Affichage en couleurs

Nous souhaitons que l'affichage soit très intuitif et que les résultats affichés soient facilement interprétables par l'utilisateur. Vous devez utiliser les couleurs pour :

- mieux présenter les résultats de commandes (répertoires, exécutables, en fonction des droits des fichiers, etc …)
- pour le module « Easter-Eggs », une couleur par diamètre du disque ou pour les tours.

Cet article sur les affichages en couleur sur Linux vous sera utile pour cela.

http://www.linuxforums.org/forum/programming-scripting/88-color-console.html

"Text color output is not defined in ANSI C/C++. Instead the creators of the language left that to be operating system dependent. In Linux, to change text color you must issue what are known as terminal commands. To do this you just change your output statement to contain a terminal command."

```c
#include <stdio.h>

#define ANSI_COLOR_RED     "\x1b[31m"
#define ANSI_COLOR_GREEN   "\x1b[32m"
#define ANSI_COLOR_YELLOW  "\x1b[33m"
#define ANSI_COLOR_BLUE    "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN    "\x1b[36m"
#define ANSI_COLOR_RESET   "\x1b[0m"

int main (int argc, char const *argv[]) {

  printf(ANSI_COLOR_RED     "This text is RED!"     ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_GREEN   "This text is GREEN!"   ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_YELLOW  "This text is YELLOW!"  ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_BLUE    "This text is BLUE!"    ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_MAGENTA "This text is MAGENTA!" ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_CYAN    "This text is CYAN!"    ANSI_COLOR_RESET "\n");

  return 0;
}
```

All the colors that I have found are:

\033[22;30m - black
\033[22;31m - red
\033[22;32m - green
\033[22;33m - brown
\033[22;34m - blue
\033[22;35m - magenta
\033[22;36m - cyan
\033[22;37m - gray
\033[01;30m - dark gray
\033[01;31m - light red
\033[01;32m - light green
\033[01;33m - yellow
\033[01;34m - light blue
\033[01;35m - light magenta
\033[01;36m - light cyan
\033[01;37m - white

6. Utilisation de primitives systemes

Les primitives/appel système et fonctions suivantes, étudiés pendant le module système, vous seront indispensables pour ce projet :

PRIMITIVES/APPELS SYSTEME	FONCTIONS C
• fork • wait • exec (et ses variantes) • getpid • open • close • read • write • exit	• opendir • readdir • printf • scanf • strcpy • strcmp • free • malloc • strlen

Les actions lancées par le shell 2.0 doivent être exécutés dans un processus fils.

Voici une liste d'autres fonctions et primitives qui vous seront, sans doute, très utiles :

PRIMITIVES/APPELS SYSTEME	FONCTIONS C
• access • stat • chdir • time • kill • pipe	• memset • fopen • fclose • fprintf • fscanf • perror • sleep • « tests sur des caractères » : isalpha, isupper, … • « conversion de types » : atoi, atof, atol • system – appelle le shell et le passe la commande, par exemple `system("clear")`

Vous pouvez trouver toutes les primitives et fonctions dans les manuels de Linux. Voici un lien qui fait une compilation assez synthétique :

http://www.enseignement.polytechnique.fr/informatique/INF583/anx.pdf

4. Modalités d'évaluation

L'évaluation du projet se réalisera sur la base de critères suivants :

- **Evaluation individuelle :**

  Il s'agit d'évaluer votre présentation, votre travail, votre implication et vos résultats. Une ou deux questions vous seront posées pendant la soutenance pour évaluer votre savoir-faire.

  L'évaluation individuelle est obligatoire. Cela signifie que l'évaluation ne donnera pas une note ABCD, mais une appréciation « Acquis », « Non acquis ». Comme cette note est obligatoire, vous devez avoir « Acquis » pour pouvoir faire la moyenne des notes de l'U.E. et prétendre valider cette U.E.

  Une note d'évaluation par les pairs peut vous être demandée par le jury (s'il a des doutes sur votre travail).

- **Soutenance, Supports et Documentation du groupe** :

	A
La soutenance (oral) [Comportement professionnel]	La soutenance est parfaitement préparée. La présentation est dynamique. Le passage de parole est fluide et pertinent.
Supports (Forme) [Comportement professionnel]	Le support de type Power Point est agréable et construit de manière pertinente et soignée. Le rapport est agréable à lire et aéré. Tous les documents sont soignés. Pas de faute d'orthographe.
Documentation (Fond) [Savoir]	Tous les documents sont construits de manière pertinente. Toutes les informations nécessaires apparaissent dans le rapport. Aucune erreur n'apparait dans les informations du rapport.

- **Efficacité & Fonctionnement du groupe** : vous allez obtenir des points en fonction des fonctionnalités implémentées. Voici la grille.

	FONCTIONNALITE	NOMBRE DE POINTS MAX
SHELL 2.0	Paramétrage initiale : démarrage sur le shell, prompt, etc …	5
	Mode expert – lancement de commandes « classiques »	5
	Mode Assistant pas à pas – menus, sous-menus, exécution des opérations demandées	15
	Mode langage naturel – interprète un petit nombre des commandes en langage naturel (environ une dizaine)	15
	Au moins deux langues sont implémentées Utilisation des fichiers externes (dictionnaire, menus, …)	10
	Affichage formaté du résultat des commandes : ls, ps, etc …	10
	Historique des commandes existant et exploitable (lecture/écriture)	5
	Easter-Eggs : paramétrage du défis, lancement et affichage du jeu, possibilité de pouvoir jouer	15
	Easter-Eggs : vérifier fin du jeu et mouvements licites, avertir si le jeu est terminé, affichage score/temps & enregistrement sur fichier	15
	Fichiers de logs – écriture et lecture correctes de ce fichier	5
	TOTAL POINTS REALISES	**100**

- **Choix techniques implémentés du projet** :

ITEM	A
STRUCTURES DES DONNES (coeff 3)	Il sait implémenter les piles et les files avec des structures dynamiques (tableaux et listes chaînées) Sait utiliser les pointeurs Les algorithmes utilisés sont fonctionnels sur ces structures
LECTURE/ECRITURE DES FICHIERS (coeff 3)	La lecture des fichiers permet le chargement correcte des données en mémoire L'écriture dans un fichier se réalise correctement Utilisation de fichiers de langue externalisés (il n'a pas codé plusieurs langues dans le code)
FONCTIONNALITES DE BASE (coeff 2)	Utilisation des printf pour l'affichage des menus Utilisation correcte de la fonction scanf ou équivalent pour lire les données saisies par l'utilisateur à la console Utilisation correcte des boucles et conditions
UTILISATION DE FONCTIONS &PRIMITIVES (coeff 2)	Le code n'est pas redondant Le code est découpé en fonctions Les noms des fonctions et des paramètres sont assez parlants. Les fonctions sont utilisées au moins une fois Utilisation d'au moins 10 primitives système.
CODE COMMENTE (coeff 1)	Les commentaires représentent plus de 10% du code.
DECOUPAGE EN FICHIERS (coeff 1)	Le programme est découpé en 3 fichiers .c ou plus. Le contenu de fichiers est bien identifié (affichage, actions, etc …) Les structures de données et les prototypes des fonctions se trouvent dans le .h

5. Livrables techniques attendus

Les livrables sont les suivants :

- Le code réalisé (dans les fichiers .c et .h).
- Les fichiers de paramétrage et de « langues ».

NOTA : le code devrait être organisé de la façon suivante :

- Fichiers pour les affichages (.c et .h) : contiennent toutes les fonctions/procédures nécessaires pour les affichages à l'écran, les saisies des informations, la validation de ces informations d'entrée et la gestion des messages d'erreurs.
- Fichiers pour les actions (.c et .h) : contiennent les structures de données et les fonctions/procédures nécessaires pour les calculs et opérations
- Fichiers principaux contenant la fonction main et les appels aux fonctionnalités demandées. (Il peut en avoir un exécutable ou plusieurs, en fonction de comment avez-vous découpez votre projet).

6. Informations diverses

1. Contraintes techniques

Tous les développements doivent être réalisés sur un ***environnement Linux***. Lors de la soutenance vous devrez montrer tous les développements et les démonstrations sur Linux.

ATTENTION : aucun projet ne sera recevable s'il est développé sur un autre environnement que Linux. La note de groupe sera automatiquement « **D** » et la note individuelle obligatoire sera « **Non acquis** ».

Vous êtes libre d'utiliser ou pas un IDE sur Linux. Vous pouvez utiliser la VM du module ou vous pouvez utiliser celle qui vous conviennent le mieux (à condition de ne pas perdre 3 jours à installer une version qui fonctionne !!).

Vous êtes également invités à installer les « Addons Invité » de VirtualBox qui vous permettront d'utiliser les répertoires partagés entre la VM et la machine hôte.

2. Conseils

Concentrez-vous sur la programmation en C, les structures de données, le découpage du code et les affichages de chacune des demandes et tout se passera très bien. **Concentrez-vous sur toutes les fonctionnalités demandées** pour ne pas en oublier une importante.

Les spécifications de ce projet vous permettent d'assigner des fonctionnalités distinctes à chaque personne du groupe, si vous le souhaitez. **Cela permettra aux jurys de mieux évaluer votre contribution individuelle**. Néanmoins, mettez-vous d'accord sur les structures à utiliser au début du projet car plusieurs personnes du groupe pourraient être emmenées à les utiliser.

> **ASTUCE :** par exemple, n'attendez pas à avoir le module d'écriture d'un fichier au point pour faire le module de lecture. Il suffit de se mettre d'accord sur le format et le contenu du fichier avant de commencer.

Vous devez montrer, lors de la soutenance, vos connaissances sur les structures d'allocation dynamique (malloc, free) et une bonne maîtrise de structures de données utilisées (piles, files … avec tableaux ou avec des listes chaînées).

Si vous implémentez des algorithmes de tri et de recherche, vous pouvez, dans un premier temps utiliser des algos simples (bulle, insertion, sélection) et dans un deuxième temps, s'il vous reste du temps, implémenter un algorithme plus efficace (merge, quicksort). De même pour la recherche (séquentielle puis dichotomique), mais, attention, pour que la recherche dichotomique soit efficace, la liste doit être préalablement triée.

Vous pouvez utiliser n'importe quelle structure de données présentée dans le module ou n'importe quelle autre que vous connaissez si vous êtes à l'aise avec la programmation en C. Dans tous les cas, réfléchissez bien à la plus adaptée à chaque problématique, mais dans tous les cas, sachez argumenter le pourquoi de votre choix.

3. Organisation du projet

Le projet se déroule en groupes de **3 personnes** (maximum 4) du 10 au 18 décembre 2015

- Le vendredi 11 décembre, en fin de journée, chaque groupe doit rendre le document décrivant l'analyse du problème et le choix des fonctions et variables (cf. document "Feuille d'avancement"). Pour cette phase vous serez assistés par un étudiant de A5 qui sera votre référent pour le projet.

- Les documents attendus (**ci-dessus**) sont à rendre le jeudi 17 décembre à 12h00 au plus tard en version électronique par mail et en version papier au tuteur responsable du projet.
    - Un dossier technique comprenant un rappel du projet, la répartition finale des tâches des membres du groupe, la description de la réalisation effective du projet et l'utilisation de l'interface et un bilan de groupe et individuel. Un planning « simple » sera inclus dans ce rapport.
    - Le code source (commenté) du projet (en .zip). Inclure les fichiers de langues et/ou paramétrages.
- L'après-midi du 17 décembre sera consacré à la préparation de la soutenance avec la réalisation d'un support (ppt, prezi, autre …)
- Dans la journée du 18 décembre, chaque groupe devra présenter et défendre son travail au cours d'une soutenance orale de 20 minutes avec séance de questions-réponses d'une même durée. Chaque membre du groupe devra prendre la parole de manière équitable.

PS : ceci est bien un Post-scriptum

Si vous travaillez bien pendant la semaine, vous pourrez profiter pendant les vacances et aller voir le volet 7 de la saga Star War, car, comme dit Yoda :