INF203 – Travaux pratiques, séance 3

Structures de contrôle en shell

Pour cette fois, vous n'aurez pas à installer quoi que ce soit. Créer simplement un répertoire TP3 dans votre ~/INF203/ .

1 Chaud ou froid

Cet exercice vise à vous faire écrire un script permettant de jouer à « Chaud ou Froid » en console. Le concept est le suivant : le script choisit aléatoirement un nombre compris entre 1 et 100 (ou n'importe quel nombre entier positif donné en argument), et l'utilisateur doit le trouver. Donc un jeu des plus passionnants s'il en est.

Pour ce faire, lorsque l'utilisateur entre un nombre, le script indique si le nombre donné est plus grand ou plus petit que le résultat attendu. Quand le nombre entré est le bon, le script termine en indiquant la bonne valeur.

Pour commencer, il faut choisir un nombre à trouver. Pour cela, vous avez à disposition la variable **$RANDOM** qui choisit un nombre entier positif compris entre 0 et 32767 (entier signé sur 16bits). Premier problème, *a priori*, 32767 est un peu plus grand que 100.

[**a**] Comment faire pour s'assurer que le nombre généré soit bien compris entre 1 et 100 ? Et entre 1 et n'importe quel entier positif (plus petit que 32768) donné en premier argument ?

Le script doit boucler jusqu'à ce que l'utilisateur trouve la bonne valeur. Pour ce faire, il lit un entier, vérifie que celui-ci est la valeur attendu, et si ce n'est pas le cas, boucle à nouveau.

[**b**] Quelle boucle pouvez-vous utiliser pour accomplir ceci ? Avec quelle condition ? Quelle instruction permet de lire une chaîne de caractères tapée au clavier ? Si vous ne vous en souvenez pas, allez observer les scripts du TP2.

[**c**] Écrivez un premier script **hotorcold.sh** (chaud ou froid, en anglais, si vous aviez un doute) qui demande en boucle un nombre à l'utilisateur jusqu'à ce que la bonne valeur soit donnée. Pour vous permettre de tester votre script, avant de boucler, faites afficher le nombre attendu dans la console. Mettez ce premier code dans votre compte-rendu.

Vous avez maintenant un script capable de demander à l'utilisateur un nombre jusqu'à que la valeur déterminée au début soit trouvée. Cela dit, il n'est pas exactement simple de trouver ce nombre sans indications ! Il va vous falloir ajouter un moyen d'afficher à l'utilisateur si le nombre entré est plus grand ou plus petit que celui attendu.

[**d**] Quelle structure de contrôle pouvez-vous utiliser pour cela ? Avec quelle(s) condition(s) ?

[**e**] Finalisez le script pour y intégrer cet affichage de « Plus grand » ou « Plus petit ». Ajoutez un message à la fin pour indiquer la victoire de l'utilisateur. Mettez le script final dans votre compte-rendu.

[**e** *bis*] En utilisant les redirections d'entrées, vous pouvez « tricher » afin de gagner immédiatement sans avoir à entrer aucune valeur durant l'exécution du script. Comment ? Expliquez la méthode de « triche » dans votre compte-rendu.

Indice : pensez à la redirection des entrées, < .

2 Le jeu du « morpion » (aussi appelé Tic-Tac-Toe)

Ici, vous allez écrire un script visant à permettre à deux joueurs de jouer (sur le même terminal) au jeu du « morpion ». Si vous en ignorez les règles, prenez quelques minutes pour aller les consulter en ligne.

On utilisera ici un affichage en « ACSII-Art », en exploitant les différents caractères ASCII pour afficher une grille. L'affichage sera sous cette forme :

```
|X| | |
|O| | |
| | | |
```

Un «X» (la lettre « x » en majuscule) indiquera que le joueur 1 a joué sur cette case, un «O» (la lettre « o » majuscule) indiquera que le joueur 2 a joué sur cette case.

Pour pouvoir faire ce script, vous aurez besoin d'utiliser un tableau (en Python, il s'agirait d'une liste avec une taille fixe). Ce tableau va être utilisé pour contenir le plateau de jeu. La case 0 du tableau correspond à la case en haut à gauche, la case un sera la case en haut au milieu du plateau,…

Voici une figure résumant la correspondance entre les index du tableau et le plateau :

```
|0|1|2|
|3|4|5|
|6|7|8|
```

Votre tableau doit donc faire 9 cases, chacune contenant un caractère, qui sera soit « » (juste un espace), soit «X», soit «O».

Pour créer un tableau de 9 cases remplit de « », utilisez l'instruction suivante :

tableau=(" " " " " " " " " " " " " " " " " ")
(9 paires de ")

Vous pourrez ensuite accéder aux cases du tableau de deux façons :

-${tableau[0]} pour accéder au **contenu** de la case 0 (c'est l'équivalent de $var pour une variable classique)

-tableau[0]="X" pour affecter à la case 0 du tableau la chaîne de caractères «X» (l'équivalent de var="X" pour une variable classique)

Attention, pour pouvoir utiliser correctement les tableaux, la première ligne de votre script DOIT être #!/bin/bash (et pas #!/bin/sh).

[**f**] Créez un script **tictactoe.sh.** Ajoutez-y d'abord la création d'un tableau de 9 cases rempli de « » puis les instructions permettant d'afficher le contenu du tableau sous la forme du plateau de jeu. Vous pouvez utiliser la commande **echo** (3 fois) pour ce faire. Par exemple, un tableau de la forme

tableau=(" " " " " " " " " " " " " " " " " ")

s’affichera :

et un tableau de la forme

tableau=("X" " " " " " " "O" " " " " " " "X")

s’affichera :

X		
	O	
		X

[**f** *bis*] Afin de vous faciliter la vie par a suite, mettez cet affichage dans une fonction affiche_plateau.

Vous pouvez désormais afficher le contenu de votre plateau à l’utilisateur. Il vous faut maintenant être capable de lire un coup fait par un joueur et de l’enregistrer dans votre tableau représentant le plateau. Pour le moment, nous considérerons qu’il n’y a qu’un seul joueur, le premier, qui joue. Le joueur devra entrer un nombre entier compris entre 0 et 8, supposé correct, et ce nombre correspondra au numéro de la case dans laquelle le joueur place sa marque (dans ce cas, un «X»).

[**g**] Ajoutez la possibilité de **lire** un coup donné au clavier, modifiez le tableau représentant le plateau en conséquence, puis affichez le nouvel état du plateau.

Pour l’instant, vous ne pouvez jouer qu’un seul coup. Ce qui est problématique pour un jeu à 2 joueurs… Déjà que le jeu du « morpion » à 2 n’est pas exactement le summum ludique, alors y jouer seul...

[**h**] Quelle(s) serai(ent) la (ou les) condition(s) d’arrêt du jeu ? Ajoutez cette(ces) condition(s) dans votre boucle de jeu. Vous pouvez dans un premier temps considérer que le but du jeu est uniquement de remplir la première ligne du plateau.

Le jeu peut désormais se terminer lorsqu'un joueur remplit une des conditions de victoire. Problème, vous n'avez toujours qu'un seul joueur... Il va falloir ajouter la possibilité d'un deuxième joueur, qui mettra des «O» dans le plateau.

Vous pouvez pour cela simplement compter le numéro de chaque coup. Si un coup est à un nombre impair (premier coup, troisième coup, cinquième coup,…) alors le joueur en cours est le premier, si le coup est à un nombre pair (deuxième coup, quatrième coup, sixième coup,...), alors le joueur jouant ce coup est le second.

[**i**] Comment ajouter cette différenciation dans votre script ? De quelle(s) structure(s) de contrôle avez vous besoin ? De quelle(s) nouvelle(s) variable(s) ?

Vous avez maintenant la base d'un jeu de « morpion ». Il y a encore beaucoup de fonctionnalités que vous pouvez ajouter à votre script, comme par exemple un affichage du joueur victorieux, un affichage plus clair de quel joueur doit jouer, l'utilisation de noms pour les joueurs, la vérification de la correction des coups,…

Il y a aussi de nombreuses améliorations possibles au script lui-même, comme par exemple l'utilisation de fonctions pour spécifier vos conditions de victoires.

[**j**] Dans votre compte-rendu, mettez le code complet de votre script du Tic-Tac-Toe, avec un court texte expliquant son utilisation, comme si vous écriviez un manuel pour un futur utilisateur.