CHAPITRE

TP 1

Unix et premiers programmes

Acclimatation à Unix : Rappels

Se connecter. Dans un premier temps, vous devez (re)prendre possession de votre compte sur la machine Turing par le biais de l'Environnement Numérique de Travail (ENT).

Commande mkdir, gestion de fichiers. On vous demande ensuite de créer un répertoire nommé TP1, de vérifier les droits d'accès à ce répertoire, puis de vous familiariser avec les commandes *cd* et *mv*, qui vous permettent de vous déplacer dans l'arborescence Unix.

Par la suite : à l'aide de l'éditeur de texte **nedit** ou **textedit**, ouvrez un fichier, puis familiarisez-vous avec les opérations d'édition : écriture, sauvegarde, configuration de l'éditeur. (Remarque : Il peut y avoir d'autres éditeurs de texte sur le système avec lesquels vous serez plus confortable pour travailler : comment obtenir des informations sur les éditeurs avec *man* ? ..) Configurez cet éditeur pour la mise en page du langage C, notre langage de programmation.

Lorsque vous saurez sauvegarder un fichier, étudiez les commandes Unix vues au premier semestre : faites-en une copie, voire renommez cette copie (avec *mv*). Vous est-il possible de copier un fichier dans le compte d'un autre utilisateur ?

Exercice : utiliser *grep* ou une combinaison de commandes *cat* et *grep* pour faire une recherche dans le fichier sauvegardé ci-haut (lequel ne sera pas vide).

Faites la même chose cette fois en recherchant un mot (il peut s'agir du nom d'une commande par exemple) dans le manuel Unix, avec *man*. Que donne la suite de commandes suivante ? (suggestion : lancer les commandes individuellement d'abord.) Pour y voir plus clair, regardez la définition de la commande *less* en tapant *man less*.

man tcsh | grep -i login | less

Son environnement de travail. La commande Unix *set* permet de voir le nom de plusieurs variables définies et initialisées par le système. Lancez-la et observez le nom et l'initialisation de ces variables, par exemple la variable SHELL : vous devriez voir ceci

SHELL=/bin/bash

Si cette variable SHELL est définie, vous pouvez également obtenir sa valeur par la commande *echo $SHELL*. C'est le shell (par défaut) qui gère votre session. Si votre shell est *tcsh*, alors la commande *setenv* est appropriée. La commande Unix *chsh* permet de définir son shell au moment de la connection. Il est toujours possible de changer de shell en évoquant à la ligne de commande le nom du shell souhaité, par exemple *tcsh*. Passez en shell tcsh, puis vérifiez vos variables systèmes avec *setenv*. Qu'observez-vous ? On revient au shell de connection en tapant *exit*.

Navigation internet. A l'aide d'un navigateur internet (mozilla ou netscape), retrouvez le site internet du cours. Il se trouve à l'url

http ://astro.u-strasbg.fr/scyon/STUSM/

Notez que certaines notes de cours seront mises à jour tout au long du semestre. En cliquant sur un lien d'extension 'pdf', vous pourrez visualiser le contenu du fichier.

Numéro 1 : un exemple, allo.c

Téléchargez le fichier allo.c du sous-répertoire Codes de l'url indiqué ci-haut. Sauvegardez ce texte puis revenez à la ligne de commande. Êtes-vous capable de lancer ce programme, en tapant hallo.c puis retour ? Pourquoi selon vous ? L'étape de la **compilation** permet de convertir le fichier de texte ascii en langage machine (binaire) plus près du noyau de calcul et de logique de l'ordinateur. Pour cela, il faut lancer une commande Unix de compilation C en évoquant le logiciel gcc

```
gcc  allo.c
```

qui convertit le texte du fichier allo.c en un exécutable nommé A.OUT. Lancez ce programme. Pouvez-vous modifier le texte affiché en éditant allo.c ?

Avec votre éditeur de texte créer un fichier main.c et tapez vos premières instructions en C :

```
1 + 3 ;
```

une simple instruction arithmtique. Pour exécuter cette instruction il est impératif d'identifier le début du programme et donc d'y insérer le mot-clé main :

```
main() {
1 + 3 ;
}
```

Tapez ensuite une commande de compilation telle que

```
gcc -o code.x  main.c
```

depuis la ligne de commandes pour créer un exécutable nommé code.x. Notez bien la présence de l'option (-o). Compilez votre code puis vérifiez que le fichier code.x possède bien le droit d'exécution. Lancez le programme : Mêmes questions qu'avant quant au résultat.

On reprend le code source main.c et maintenant nous y insérons le deuxième mot-clé du langage C : return, comme ceci :

```
main() {
return 1 + 3 ;
}
```

Sauvegardez puis recompilez et exécutez ce nouveau code. Le résultat est maintenant affecté au shell de votre session. Vérifier cela en lisant la valeur du shell à la ligne de commandes (variable ' ?'). Il est bien sûr plus élégant et utile de rédiger un programme permettant de sauvergarder des résultats intermédiaires. Pour cela il faut déclarer des variables. Par exemple appelons x une variable de valeur inconnue. L'instruction C

```
int x ;
```

permet de *créer* la variable x ; elle ne contient toutefois aucune valeur. Pour cela il faut affecter le résultat de l'opération arithmétique à la variable :

```
x = 1 + 3;
```

En insérant ces deux nouvelles instructions dans notre programme de départ nous aurons

```
main() {
int x ;
x = 1 + 3 ;
return x; }
```

Cette fois la varaible x admet une valeur numérique *après* l'opération d'affectation et c'est cette valeur qui est transmise au shell.
Exercez-vous avec d'autres valeurs pour bien assimiler le fonctionnement du programme. Est-ce que des valeurs négatives sont possibles ? Même question pour la multiplication et la division.

Numéro 2 : opérations arithmétiques

En vous inspirant du programme main.c, rédigez un court programme pour évaluer les expressions mathématiques suivantes

```
z = x + y / 2 * 3 ; j = i / 2 ; j = i / 2.0 ; z = i / 2 ;
z = i / 2.0 ;
```

suivant les déclarations

int i = 1, j ; int x = 2,y = 2 ; float z ;

Notez le nouveau mot-clé *float* qui permet d'identifier une variable acceptant des valeurs réelles. Quels résultats obtenez-vous dans chaque cas ? Il suffit de substituez pour la variable recherchée le nom de la variable de l'instruction *return* pour affectez le shell d'une nouvelle valeur. Étudiez ensuite l'instruction suivante :

```
z = ( y += 2 , (x+y)/2*3 ) ;
```

Quelle peut-être le résultat de cette instruction ?

Numéro 3 : printf

Reprenez le programme main.c déj vu au numéro 2, puis insérez la commande suivante :

```
printf( "Resultat = %i\n", x);
```

avant l'instruction return. En recompilant le programme que se passe-t-il ? Il faudra insérer une instruction spéciale pour exécuter l'instruction contenant le printf. Cf. le programme hallo.c du numéro 1.
Lorsque vous aurez vu comment utiliser printf, modifier votre programme du numéro 2 pour faire afficher les résultats en une seule exécution du programme. Votre programme final sera sauvegardé dans un fichier qu'on appellera Math.c.