TD n°1 – Introduction à Unix et au C

Le but de ce TD est de prendre en main l'environnement Unix et d'avoir un premier contact avec le langage C, en analysant et en programmant quelques programmes simples.

L'environnement Unix

Outre le fait qu'il s'agit d'un système d'exploitation répandu, complet, puissant, multitâche et stable, le choix de l'environnement Unix pour la programmation en C se justifie du fait que ce langage a été développé en particulier pour réécrire le système Unix.

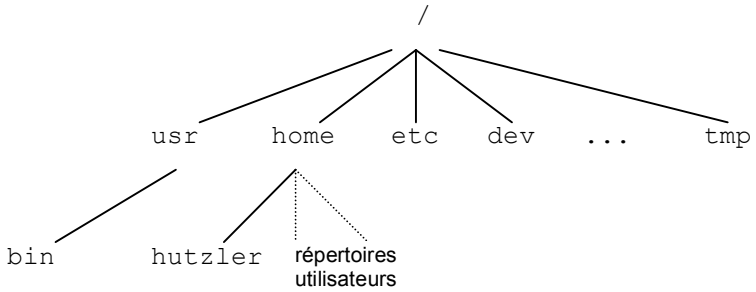
La connexion

Sous Unix, les utilisateurs sont référencés par un nom d'utilisateur (le **login**) et un mot de passe (le **password**). Ceci permet d'établir des permissions différentes suivant les utilisateurs, notamment concernant l'accès aux fichiers (voir l'instruction **ls** ci-après).

Le système de gestion de fichiers

Comme la plupart des systèmes d'exploitation, Unix gère les ***fichiers*** en les regroupant et les organisant en une hiérarchie de ***répertoires*** et de sous-répertoires. La racine du système de gestion de fichier (SGF), c'est-à-dire le répertoire englobant tous les autres, est désigné par le symbole **/**. C'est ce même symbole qui permet de séparer le nom d'un répertoire du nom d'un sous-répertoire. Il existe deux autres noms de répertoires réservés :

. désigne le répertoire courant ;
.. désigne le répertoire parent du répertoire courant, celui qui le contient ;



Ainsi le répertoire `bin` contenu dans le répertoire `usr` lui-même contenu dans le répertoire racine `/` sera désigné sous le nom `/usr/bin`. Ce nom est dit ***absolu*** car il indique le ***chemin*** permettant d'y accéder à partir de la racine du SGF, c'est-à-dire la suite de répertoires à parcourir pour aller du répertoire racine au répertoire cible. Le nom du répertoire ou du fichier est dit ***relatif*** lorsque le chemin pour accéder à ce répertoire ou fichier est donné à partir du répertoire courant. Si l'on suppose que l'on se trouve dans le répertoire `hutzler`, par exemple, le même répertoire `bin` aura pour nom relatif `../../usr/bin` (il faut remonter de deux niveaux dans l'arborescence de fichiers avant de pouvoir redescendre dans la bonne branche).

Les principales commandes qui permettent la manipulation des répertoires et des fichiers sont les suivantes :

pwd : affiche le chemin permettant d'accéder au répertoire courant. Dans le répertoire `hutzler`, le résultat de cette commande sera : `/home/hutzler`.

mkdir <nom_de_répertoire> : crée un répertoire dont le nom est donné en argument ;
ex :

```
mkdir TD1_C       # crée un répertoire appelé TD1_C
```

cd <nom_de_répertoire> : se déplace dans le répertoire spécifié ;
ex :

```
cd TD1_C          #  se déplace dans le répertoire TD1_C
```

ls [<nom_de_répertoire>] : affiche le contenu du répertoire courant (ou celui spécifié en argument) ;

utilisation des métacaractères ? et * :
? représente n'importe quel caractère
* représente n'importe quelle suite de caractères

ex :

```
ls *.c      # renvoie tous les fichiers dont l'extension est .c
ls prog?.c  # renvoie tous les fichiers commençant par prog, se
            # prolongeant par n'importe quel caractère et
            # finissant par l'extension .c
```

droits d'accès :
l'option -l permet d'afficher les droits d'accès aux différents fichiers. Le système Unix distingue trois types d'opérations (la **lecture**, l'**écriture** ou l'**exécution**) et trois types d'utilisateurs (celui qui possède le fichier, les utilisateurs qui appartiennent au même groupe, tous les autres).

ex:

```
-rwxr--r--   1 hutzler     users        75 Jan 27 22:40 exo1.c
```

rm <nom_de_fichier> : supprime le fichier spécifié
rm –r <nom_de_répertoire> : supprime le réperoire spécifié

mv <nom_de_fichier> <nom_de_répertoire> : déplace le fichier (ou le répertoire) dont on a donné le nom dans le répertore indiqué
ex :

```
mv ex1.c TDC      # déplace le fichier ex1.c dans le répertoire TDC
```

cp <nom_de_fichier> <nom_de_répertoire> : copie le fichier dont on a donné le nom dans le répertore indiqué
ex :

```
cp ex1.c TDC      # crée une copie du fichier ex1.c dans le
                  # répertoire TDC
```

Introduction au langage C

Exercice 1

- Créez un répertoire TDC
- Déplacez-vous dans le répertoire TDC, créez un répertoire TD1, et déplacez-vous dans ce répertoire
- Lancez l'éditeur xemacs
- Tapez le programme suivant et enregistrez-le dans le fichier ex1.c

```
1     #include <stdio.h>
2
3     main()
4     {
5          printf("ceci est mon premier programme\n");
6     }
```

- Pour compiler le programme, tapez la commande `cc ex1.c -o ex1`. La commande a pour effet de transformer le fichier source en fichier exécutable par la machine.
- Tapez la commande `ex1` pour exécuter le programme.

Exercice 2

- Tapez, compilez et exécutez le programme suivant :

```
1     #include <stdio.h>
2
3     main()
4     {
5          int c;
6
7          printf("longueur du côté : ");
8          scanf("%d", &c);
9          printf("aire du carré : %d", c*c);
10    }
```

- Que se passe-t-il si l'on supprime la ligne 5?
- Que se passe-t-il si l'on supprime le symbole `&` à la ligne 8?
- Que se passe-t-il si l'on supprime le symbole `;` à la fin de la ligne 7?
- Modifiez le programme pour calculer l'aire du cube de côté `c`
- Modifiez le programme pour calculer l'aire d'un rectangle dont on demande la longueur et la largeur à l'utilisateur

Types primitifs

`char`	codé sur un seul octet, pouvant contenir un caractère du jeu de caractères de la machine utilisée
`int`	un nombre entier, qui reflète typiquement la taille naturelle des entiers sur la machine utilisée
`float`	un nombre en virgule flottante en simple précision
`double`	un nombre en virgule flottante en double précision

On peut appliquer des qualificatifs supplémentaires à ces types de base:

- **`short`** et **`long`** peuvent s'appliquer au type entier (donnant ainsi `short int`, `long int`): les entiers de type `short` sont souvent codés sur 2 octets, ceux de type `long` sont souvent codés sur 4 octets, ceux de type `int` peuvent être codés sur 2 ou 4 octets en fonction de la machine.
- **`long`** peut également s'appliquer au type double. Comme pour les types entiers, le nombre d'octets utilisé pour coder les types `float`, `double` et `long double` dépend de la machine utilisée.
- **`signed`** et **`unsigned`** s'appliquent au type `char` et à tous les types entiers (par exemple, si le type char est codé sur un octet, les variables de type `signed char` pourront prendre les valeurs de -128 à +127 tandis que les variables de type `unsigned char` pourront prendre les valeurs de 0 à +255).

Format d'affichage

`%d`	affiche un entier décimal
`%6d`	affiche un entier décimal sur une largeur minimum de 6 caractères
`%f`	affiche un flottant
`%6f`	affiche un flottant sur une largeur minimum de 6 caractères
`%.2f`	affiche un flottant avec 2 chiffres après le point décimal
`%6.2f`	affiche un flottant sur au moins 6 caractères, avec 2 chiffres après le point décimal

Séquences d'échappement

`\n` fin de ligne `\t` tabulation

Exercice 3

- Tapez et exécutez le programme suivant :

```
1     #include <stdio.h>
2
3     #define PI 3.14159
4
5     /* calcule l'aire du cercle */
6     float aire(int r)
7     {
8          return PI*r*r;
9     }
10
11    main()
12    {
13         int r;
14
15         printf("rayon du cercle : ");
16         scanf("%d", &r);
17         printf("aire du cercle : %f", aire(r));
18    }
```

- Ecrivez la fonction qui renvoie la circonférence du cercle dont on donne le rayon
- Que se passe-t-il si l'on rajoute un symbole ; à la fin de la ligne 3?

Exercice 4

- Tapez et exécutez le programme suivant :

```
1     #include <stdio.h>
2
3     main()
4     {
5          int fahr, celsius;
6
7          fahr = 0;
8          while(fahr <= 300) {
9                  celsius = 5 * (fahr - 32) / 9;
10                 printf("%d\t%d\n", fahr, celsius);
11                 fahr = fahr + 20;
12         }
13    }
```

- Modifiez le programme de manière à paramétrer les valeurs de départ, de fin et d'incrément
- Que se passe-t-il si l'on enlève les parenthèses à la ligne 9?
- Que se passe-t-il si l'on enlève le symbole { à la ligne 8 et le symbole } à la ligne 12

Exercice 5

- Tapez et exécutez le programme suivant :

```
1     #include <stdio.h>
2
3     int puiss(int base, int n)
4     {
5          int i, p;
6
7          p = 1;
8          for (i = 1; i<= n; i++)
9                  p = p * base;
10         return p;
11    }
12
13    main()
14    {
15         int i;
16
17         for (i = 0; i < 10; i++)
18                 printf("%d %d %d\n", i, puiss(2, i), puiss(-3, i));
19    }
```

- Réécrivez la fonction `puiss` en utilisant une boucle `while` à la place d'une boucle `for`
- Que se passe-t-il si l'on remplace la ligne 5 par la ligne `int I, P;` ?

Exercice 6

- Tapez et exécutez le programme suivant :

```
1     #include <stdio.h>
2
3     int min(int c)
4     {
5          if (c >= 'A' && c <= 'Z')
6                  return c + 'a' - 'A';
7          else
8                  return c;
9     }
10
11    main()
12    {
13         char c = ' ';
14
15         printf("Entrez un caractère : ");
16         scanf("%c", &c);
17         printf("%d\n", min(c));
18    }
```

- Ecrivez une fonction `int isdigit(int c)` qui renvoie la valeur 1 si le caractère c correspond à un chiffre, la valeur 0 sinon.

Table des codes ASCII en hexadécimal

code	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	NP	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	"	#	$	%	&	´	(	)	*	+	,	-	.	/
0x30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Table des codes ASCII en décimal

code	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
10	LF	VT	NP	CR	SO	SI	DLE	DC1	DC2	DC3
20	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
30	RS	US	SP	!	"	#	$	%	&	´
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	DEL		