



T.P. 1

Introduction UNIX

Initiation au langage C

DEA SyAM 2000/2001

Par
Philippe Lucidarme : lucidarm@lirmm.fr

Introduction UNIX

L'histoire des systèmes UNIX

Le système Unix a été mis au point par Ken Thompson dans les laboratoires Bell dans le New Jersey aux Etats-Unis. Le but de Ken Thompson était de mettre au point un système interactif simple pour faire tourner un jeu qu'il avait créé (space travel, une simulation du système solaire).

La première version de ce système a vu le jour en 1969, il s'inspirait des principaux systèmes d'exploitation de l'époque (Multics, Tenex), et était destiné à une utilisation mono-utilisateur, d'où son nom (Unix= Multix uni-utilisateur à priori).

Peu de temps après, D.Ritchie a rejoint l'équipe de K.Thompson afin de mettre au point, en 1971, une version d'UNIX permettant la multiprogrammation.

Parallèlement, D.Ritchie participe grandement à la définition du langage C (puisque'il est considéré comme un de ses créateurs avec B.W.Kernighan), ainsi l'ensemble du système a été entièrement réécrit en langage C en 1973.

En 1975, à partir de la version 6 du système, Unix va enfin être commercialisé.

Lorsque le système passe à la version 7, l'évolution s'accompagne de nombreuses modifications notables telles que:

- la suppression du bridage lié à la taille des fichiers
- une meilleure portabilité du langage
- l'ajout de plusieurs utilitaires

1983 marque l'apparition de UNIX system V, un système Unix commercialisé par AT&T. De son côté l'Université de Californie met au point une variante du système destinée aux systèmes VAX nommée UNIX BSD. Les deux systèmes se sont longtemps fait la guerre et c'est le system V qui en est sorti vainqueur.

De nos jours les systèmes Unix restent très présents dans les milieux professionnels et de l'éducation grâce à sa grande stabilité et son utilisation en réseau.

Le système UNIX

Le système Unix est constitué d'un système d'exploitation (le noyau), d'un interpréteur de commandes (shell) et de nombreux utilitaires (assembleur, compilateurs pour de nombreux langages, traitements de texte, messagerie électronique, ...)

Tableau des principales commandes UNIX

Commande Unix	Description	équivalent DOS
ls	liste le contenu d'un répertoire	dir
cd	change de répertoire	cd
cd ..	répertoire parent	cd
mkdir	crée un nouveau répertoire	md
rmdir	supprime un répertoire	deltree
cp	copie de fichier	copy, xcopy
mv	déplacement de fichier	move
rm	supprime le fichier	del
passwd	change le mot de passe de l'utilisateur	
cat	affiche le contenu du fichier	type
more	affiche le contenu du fichier avec des pauses	type more
man apropos	aide sur la commande demandée	help
lpr	imprime le fichier demandé	print
chmod	change l'attribut d'un fichier <i>chmod XXX fichier</i> XXX= Utilisateur Groupe Autres ou X représente un entier 1<X<7 Lecture=1, Ecriture=2, Execution=4 X=Lecture+Ecriture+Execution	
chfn	change les informations personnelles vues avec finger	
chsh	change le shell : <i>chsh user emplacement_du_shell</i>	
finger	liste des utilisateurs en ligne	
traceroute	trace le chemin entre la machine locale et la machine visée	
ftp [machine] [port] get put quit	transfert de fichier entre la machine locale et la machine cible récupère un fichier envoie un fichier quitte la session FTP	
telnet [machine]	effectue un <u>telnet</u>	
talk	permet de parler à un utilisateur connecté talk user	
mesg	autorise ou non la commande talk <i>mesg n</i> : Empeche la reception de messages talk <i>mesg y</i> : Permet la reception de messages talk	
bye	Déconnexion	

Première approche

Ouvrez une session X-Win32.
Déplacez vous dans le répertoire DEA2000.
Et créez un répertoire portant votre nom.
Dans ce répertoire exécuter la commande :

```
echo Bonjour ! > test.txt
```

Affichez le contenu du fichier test.txt

Exécutez la commande :

```
echo Au revoir ! >> test.txt
```

Affichez le contenu du fichier test.txt

Hello World !

Nous allons maintenant créer notre premier programme. Pour être original, celui-ci va afficher le texte : « hello world ! ».

Emacs est un éditeur de texte (comme edit sous dos). C'est grâce à celui ci que nous allons créer notre premier programme. Lancez Emacs, puis saisissez le programme suivant :

```
#include <stdio.h>
```

```
void main()  
{  
printf("Hello World");  
}
```

Sauvegardez le, nous allons maintenant compiler ce programme à l'aide de la fonction suivante :

```
g++ nomsourc.c -o nomexec
```

Si la compilation s'est passée correctement, un nouveau fichier est apparu dans le répertoire courant : nomexec. Exécuter ce programme en tapant son nom.

Deuxième programme

Vous allez maintenant créer de la même façon un programme qui affiche successivement votre nom, votre prénom, votre date de naissance et votre âge en passant une ligne entre chaque information. \n placé dans une chaîne de caractère remplace le retour chariot.

Exécutez ce programme.

Quittez Emacs, effacez tous les fichiers et dossiers que vous avez créé avant de vous déconnecter.

Initiation au langage C

Pour plus de commodité, nous allons maintenant utiliser l'interface Microsoft Developer Studio. Vous le trouverez dans le menu Démarrer, Programmes, Microsoft Visual C++.

Saisissez le programme suivant :

```
#include <stdio.h>

void main ()
{
    printf("Hello World !");
}
```

La création du fichier exécutable se fait en 2 temps :

1 : La compilation (Ctrl-F7)

2 : La création des liens entre les différents fichiers du projet : Build (F7)

Pour exécuter directement un programme : F5

Exécutez le programme.

Si vous n'avez pas le temps de lire le contenu de la fenêtre, vous pouvez utiliser l'instruction `getch ()` qui attend que l'utilisateur tape une touche au clavier pour continuer. Cette instruction appartient à la bibliothèque `<conio.h>`. De manière générale, pour avoir de l'aide sur une fonction, placez le curseur sur cette instruction et tapez F1.

Les types de données

Voici un tableau récapitulatif des différents types de données :

Type de donnée	Signification	Printf Scanf	Taille (en octets)	Plage de valeurs acceptées
char	Caractère	%c	1	-128 à 127
unsigned char	Caractère non signé	%c	1	0 à 255
short int	Entier court	%d	2	-32768 à 32767
unsigned short int	Entier court non signé	%d	2	0 à 65535
int	Entier	%d	2	-32768 à 32767
unsigned int	Entier non signé	%d	2	0 à 65535
long int	Entier long	%d	4	-2 147 483 648 à 2 147 483 647
unsigned long int	Entier long non signé	%d	2	0 à 4 294 967 295
float	flottant (réel)	%f	4	$3.4 \cdot 10^{-38}$ à $3.4 \cdot 10^{38}$
double	flottant double	%f	8	$1.7 \cdot 10^{-308}$ à $1.7 \cdot 10^{308}$
long double	flottant double long	%f	10	$3.4 \cdot 10^{-4932}$ à $3.4 \cdot 10^{4932}$

Pour pouvoir utiliser une variable, il faut d'abord la déclarer. En effet, chaque variable est stockée dans un endroit de la mémoire; le fait de la déclarer réserve cette emplacement mémoire.

Par exemple : la réservation d'un chiffre entier se fait de la façon suivante :

```
int nom_var ;           // Permet de réserver deux octets en mémoire pour la variable nom_var.  
  
int nom_var = 5 ;       // Permet de la déclarer et de l'initialiser en même temps.
```

Ecrire un programme qui permet de saisir deux chiffres réels, puis affiche le produit de ces deux chiffres.

Ce petit programme permet de saisir deux nombres entiers et d'en afficher la somme.

```
// Déclare les variables a et b  
int a ,b ;  
  
// Saisie les deux entiers  
scanf (" %d %d ",&a,&b);  
  
// Affiche la somme  
printf ("La somme est : %d ", a+b );
```

Les structures conditionnelles

On appelle structure conditionnelle les instructions qui permettent de tester si une condition est vraie ou non. Il existe deux instructions permettant de réaliser ce test :

L'instruction if	L'instruction switch
<pre>if (condition réalisée) { liste d'instructions Vrai } else { autre série d'instructions Faux }</pre>	<pre>switch (<i>Variable</i>) { case <i>Valeur1</i>: Liste d'instructions 1 break; case <i>Valeur1</i>: Liste d'instructions 2 break; case <i>Valeurs_n</i>: Liste d'instructions n break; default: Liste d'instructions Default break; }</pre>

L'instruction if permet de tester si une condition est vraie ou fausse. Si elle est vraie, c'est la liste d'instructions dans les premières accolades qui s'exécute, sinon, ce sont les instructions après le else.

L'instruction switch permet de traiter un nombre de cas important en une seule fois : si Variable est égal à Valeur_n c'est la liste d'instructions n qui s'exécute. Si Variable n'est égal à aucun des cas, c'est Liste d'instructions Default qui s'exécute.

Opérateur	Dénomination
== A ne pas confondre avec le signe d'affectation (=)!!	Opérateur d'égalité
<	Opérateur d'infériorité stricte
<=	Opérateur d'infériorité
>	Opérateur de supériorité stricte
>=	Opérateur de supériorité
!=	Opérateur de différence

Utilisation du if : Réalisez un programme qui permet à l'utilisateur de saisir une température. Si cette température est supérieure à 15°, le programme affiche : « Nous sommes à Montpellier ! », sinon il affiche « Nous sommes à Lille ! »

Utilisation du switch : Réalisez un second programme qui permet à l'utilisateur de saisir une lettre, puis l'ordinateur affiche un mot commençant par cette lettre (On se limitera aux lettres de a à d)

Les boucles

Les boucles sont des structures qui permettent d'exécuter plusieurs fois la même série d'instructions jusqu'à ce qu'une condition ne soit plus réalisée. Il existe deux instructions permettant de réaliser des boucles :

Boucle for	Boucle while
for (compteur; condition ; incrémentation) { liste d'instructions }	while (condition réalisée) { liste d'instructions }

Utilisation du for : Réaliser un programme qui compte jusqu'à cent. Modifier ce programme afin de ne lui faire compter que les chiffres pairs (modification de l'incrément).

Utilisation du while : Réaliser un programme qui demande à l'utilisateur de saisir des chiffres entiers jusqu'à ce que ce chiffre soit 5.

Le juste chiffre

Pour terminer, réalisez un programme qui tire un nombre au hasard (fonction rand). L'utilisateur doit trouver ce nombre. A chaque proposition l'ordinateur lui dit si la solution est supérieure ou inférieure. Initialisez le générateur aléatoire avec :

```
srand( (unsigned)time( NULL ) );      ( Inclure time.h )
```

