TP1 : Algorithmes numériques
Résolution de systèmes

4 octobre 2019

1 Quelques éléments de Linux

Ce document liste un ensemble de commandes utiles sous UNIX.

1.1 Systèmes de fichiers

Créer un fichier	**cat** > *nom_fichier*
Créer un répertoire	**mkdir** *nom_repertoire*
Visionner le contenu d'un fichier	**cat** *nom_fichier*
Lister le contenu du répertoire courant	**ls**
Lister le contenu d'un autre répertoire	**ls** *nom_repertoire*
Déplacer un fichier ou un répertoire	**mv** nom_source_nom_destination
Copier un fichier dans le même répertoire	**cp** *nom_fichier_source nom_fichier_destination*
Copier un fichier dans un autre répertoire	**cp** *nom_fichier_source nom_repertoire*
Supprimer un fichier	**rm** *nom_fichier*
Supprimer un répertoire (vide)	**rmdir** *nom_repertoire*
Supprimer un répertoire et son contenu	**rm -r** *nom_repertoire*
Changer de répertoire	**cd** *nom_repertoire*
Changer de répertoire parent	**cd** ..
Changer de répertoire personnel	**cd**
Afficher le répertoire courant	**pwd**

Il est possible d'afficher les résultats d'une commande dans un fichier, en la redirigeant grâce à l'opérateur >.
nom_commande > *nom_fichier* : remplace le contenu du fichier par le résultat de la commande.
Pour ajouter le résultat d'une commande en fin d'un fichier, il suffit de doubler l'opérateur.

Pour exécuter une commande en tâche de fond, il faut terminer la commande par &.
Pour accéder à la documentation d'une commande, taper : **man** *nom_commande*.

2 Quelques éléments de C

2.1 Composition d'un programme

Un programme s'écrit dans un fichier dont l'extension est **.c**
Un tel fichier comprend :

- l'inclusion de bibliothèques de fonctions Par exemple, la bibliothèque standard d'entrée/sortie : #**include** <**stdio.h**>
- l'écriture de sous-programmes éventuels.
- l'écriture du programme principal : le **main**(point d'entrée du programme) qui contient :
  - la déclaration des variables
  - les débuts et fins de blocs (entre les caractéres { et })
  - les instructions terminées par des points-virgules.

Exemple :

```
#include <stdio.h>
#define PI 3.14
main()
{

        int A,B,Diff ;
        printf("entrez la valeur de A") ;
        scanf("%d",&A) ;
        printf("entrez la valeur de B") ;
        scanf("%d",&B) ;
        While (A != B)
        {
        Diff = A - B ;
        if (A<B) A = Diff ;
        else B = Diff ;
        }
        printf("Le PGCD est %d  n",A) ;
}
```

2.2 Commentaires

Toute séquence de caractéres comprises entre /* et */ permet de commenter un programme source et sera ignorée par la compilateur.

2.3 Variables et valeurs

Les valeurs sont contenues dans des variables.
Les variables sont des identificateurs constités d'une suite de caractéres chiffres et/ou lettres qui commence par une lettre (_ est considéré comme une lettre et les symboles a et A sont différents) et ne correspond pas à un mot réservé.

Il existe des mots réservés dans le langage C : **auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, void, volatile, while**.

Pour déclarer une variable, il faut choisir un nom et décider son type, c'est à dire son format.

2.3.1 Types

char	caractère (codé comme un entier entre 0 et 255)
int	entier
float	nombre flottant (avec décimales)
double	nombre flottant avec double précision

On peut alors déclarer une variable comme suit :
type nom_variable = valeur ;
La variable va être créée et va contenir (pour l'instant) la valeur renseignée. Le symbole ; termine la déclaration de la variable.

2.3.2 Affectation

Cette instruction permet de ranger une valeur dans une variable. Le symbole d'affectation est le signe = (égal simple).
La partie gauche du symbole reçoit le résultat de l'évaluation de la partie droite.

A = 8	La zone mémoire de nom A reçoit la valeur 8
A = A+10	Le programme ajoute 10 au contenu de la variable A et range le résultat dans la variable A en écrasant l'ancienne valeur.

2.3.3 Tableau

Les variables simples ne contiennent qu'une valeur.
Pour conserver un ensemble de valeurs de même type, on utilise un tableau.
Pour conserver les notes de 26 étudiants, on peut par exemple déclarer le tableau suivant :
float notes[26] ;
La note du premier étudiant sera contenue dans notes[0] et celle du dernier dans notes[25] , en considérant que chaque étudiant est caractérisé par un numéro.
On peut déclarer des tableaux à plusieurs dimensions.

2.3.4 Structures de contrôles

Le programmeur dispose d'un ensemble de structures de contrôles pour l'écriture d'un programme.
Ces structures permettent d'effectuer des tests et de répéter une séquence d'instructions en fonctions de critéres prédéfinis.

2.3.5 Conditionnelles

Pour effectuer un test, on utilise la structure suivante

if(*test*)
{
instructions_si ;
}

else
{
instructions_sinon ;
}
Les différents opérateurs de tests sont : == (égalité), != (différence), <, <=, >, >=
On peut effectuer plusieurs tests avec les opérateurs logiques : && (ET logique), || (OU logique)
.

2.3.6 Répétitives

Elles permettent la répétition d'une instruction ou d'un bloc d'instructions. Elles seront choisies en fonction du nombre de répétitions à effectuer.
Dans le cas d'un nombre de répétitions connu, on utilise une boucle **for** :
Par exemple, pour effectuer $S = \sum_{i=1}^{9} i$ on pourra utiliser le code suivant
int S = 0 ;
for (int i = 1 ; i <= 9 ; i++)
{
S = S + i ;
}
Dans le cas où l'on ne connaît pas le nombre de répétitions, on utilise une boucle while :
while(*condition d'arrêt*)
{
instructions ;
}

2.3.7 Chaînes de caractères

Une chaîne de caractères est un tableau contenant des caractères. Ce tableau a la particularité de se terminer par le caractère spécial \0 qui représente la fin de la chaîne.

char *chaine*[50]	Déclaration d'une chaîne de 50 caractères
strlen *chaine*	Retourne la longueur de la chaîne (sans compter le caractère \0)
strcpy(*chaine,"hello"*)	Copie le texte "hello" dans la variable chaine
strcat(*chaine,"world"*)	Concatène le texte "world" en fin de la variable chaine
strcmp(*chaine1,chaine2*)	Compare les deux chaines (retourne 0 si identique)

2.3.8 Entrées / Sorties

Le langage C offre la possibilité de réaliser des affichages formatés de texte et de contenu de variables. Cette notion de format est également exploitée lors de la lecture de valeurs au clavier. Le caractère \n permet un saut de ligne.
Pour utiliser ces fonctions, il faut inclure la bibliothèque stdio.h par l'instruction suivante que l'on écrit en début de programme :

#include < *stdio.h* >

2.3.9 Ecriture

Cette fonction, contenue dans la bibliothèque standard stdio.h, attend comme premier paramètre la chaîne de caractère à imprimer avec éventuellement la description des formats d'affichage des variables à afficher dans cette chaîne. Les paramétres suivants correspondent, dans l'ordre, à chaque variable dont on souhaite afficher la valeur.
Exemple :
printf("Les valeurs des variables X et Y sont : %d et %d \n",X,Y) ;
Les deux %d seront respectivement remplacer à l'écran par la valeur de X et de Y.

2.3.10 Lecture

Cette fonction, également contenue dans la bibliothèque standard stdio.h, attend comme premier paramètre la chaîne décrivant les formats de lecture des variables à lire au clavier. Les paramètres suivants sont, dans l'ordre, l'adresse des variables dont on souhaite lire la valeur.
Exemple :
scanf("%d %d \n",&X, &Y) ;

Le & permet d'accéder à l'adresse des variables.

2.3.11 Les formats

Les fonctions printf et scanf utilisent les mêmes formats.
Entier : %d
Réel : %f
Caractère : %c
Chaîne de caractères : %s

2.4 Lecture / Ecriture dans un fichier

2.4.1 Ouverture d'un fichier

La déclaration d'une variable associée à un fichier se fait par la commande :
FILE * fp ;
L'ouverture du fichier s'effectue ensuite par l'instruction :
fp = fopen ("nom_fichier", "w") ;
où le deuxième paramètre peut être :

"r"	Ouvre un fichier existant pour la lecture
"w"	Crée et ouvre un fichier pour l'écriture. Tout fichier existant est remplacé.
"a"	Ouvre un fichier en ajout. L'ajout se fait à la fin d'un fichier existant.
"r+"	Ouvre un fichier existant pour la lecture et l'écriture.
"w+"	Crée et ouvre un fichier existant pour la lecture et l'écriture. Tout fichier existant est remplacé.
"a+"	Ouvre un fichier pour la lecture et l'ajout de données à la fin d'un fichier existant.

Un fichier est créé s'il n'en existe pas.

2.4.2 Lecture / Ecriture

Ces fonctions sont réalisées par les fonctions **fscanf** et **fprintf** correspondant aux fonctions scanf et printf.
On rajoute un premier paramétre indiquant sur quel fichier on travaille.
Exemple d'écriture dans un fichier :
fprintf(fp, "La variable A vaut %d", A) ;

2.4.3 Fermeture d'un fichier

La fermeture du fichier se fait par l'instruction : **fclose(fp) ;**

2.5 Compilation

La compilation d'un programme (*nom_fichier.c*) en un exécutable (*nom_executable*) se fait par la commande UNIX
gcc nom_fichier.c -o nom_executable
Il produit un exécutable que l'on peut exécuter en tapant *nom_executable*
L'option **-lm** permet d'obtenir certaines fonctions mathématiques prédéfinies.