Système UNIX et programmation

M. Davy MOUSSAVOU- Consultant Linux,
Certifié Linux International

Rappel des concepts fondamentaux du langage C

Objectifs du chapitre

Après l'étude ce chapitre, l'étudiant connait:

- Comment utiliser un pointeur en C;
- Comment faire appel une fonction en C;
- Comment utiliser une structure de donnée.

Notions de pointeur

Définition:

C’est une variable qui contient l’adresse d’une autre variable. Un pointeur a pour valeur l’adresse d’un objet en C d’un type donné(pointeur est typé).

Notions de pointeur

Définition:

C’est une variable qui contient l’adresse d’une autre variable. Un pointeur a pour valeur l’adresse d’un objet en C d’un type donné(pointeur est typé).

Concepts théoriques et déclaration(1/3)

Deux opérateurs sont introduits en C pour les pointeurs:

- L'opérateur **&** qui désigne l'***adresse*** d'une variable ;
- L'opérateur * donne la ***valeur*** de l'objet pointé.

Déclaration

int *p; // la variable **p** est un pointeur d'un entier
int a=4; // **a** est un entier qui vaut 4
p=&a ; // **p** pointe vers l'adresse de **a**
printf("%d", a); // affiche la valeur de ***a*** donc 4
printf("%p", &a); // affiche l'adresse de ***a***
printf("%d", p); // affiche la valeur du pointeur, donc une adresse
printf("%d", *p); // affiche la valeur de ***a*** c'est-à-dire 4

Concepts théoriques et déclaration(3/3)

A retenir

Pour la variable *a*:

- " ***a*** " *signifie que l'on veut la valeur de* ***a***
- " &***a*** " *signifie que l'on veut l'adresse où se trouve la variable* ***a***

Pour un pointeur ***p***:

- " ***p*** " *signifie que l'on veut la valeur de* ***p****(c'est une adresse)*
- " ***p** " *signifie que l'on veut la valeur de la variable qui se trouve à l'adresse contenue dans* ***p***

Passage de paramètre à une fonction(1/2)

En C le passage de paramètres à une fonction peut se faire de deux moyens:

- Par valeur: c'est-à-dire qu'une copie est effectuée sur la pile d'appel. Ainsi toutes les modifications de la variable effectuées dans la fonction seront perdues une fois de retour à l'appelant.

- Par adresse: la fonction change la valeur du paramètre

Passage de paramètre à une fonction(2/2)

```
#include <stdio.h>
 /* Ce code échange le contenu de deux variables */
 void echange(int * a, int *b)
 {      int tmp;
        tmp = *a;
        *a = *b;
        *b = tmp;
}
 int main(void) {
        int a = 5;
        int b = 2;
        printf("a = %d, b = %d.\n", a, b);
        /* On passe à 'echange' les adresses de a et b. */
       echange(&a, &b);
       printf("a = %d, b = %d.\n", a, b);
       echange(&a, &b);
       printf("a = %d, b = %d.\n", a, b);
 return 0;
 }
```

Notion sur les structures des données(1/4)

En C le programmeur peut créer ses propres types de variables. Il y a plusieurs types de variables personnalisées :

- Les structures: permet de traiter un ensemble d'informations hétérogènes comme un tout.
- Les énumérations: permet de définir un liste de valeur possible pour une variable.

Notion sur les structures des données(2/4)

Déclaration1

```
struct  Personne1 {

char  nom[40];

char prenom [100];

char adresse[100];

int  tel;

char  ville[100];
};
```

Déclaration2

```
typedef struct
Personne2  {

char  nom[40];

char prenom [100];

char adresse[100];

int  tel;

char  ville[100];
}Personne2;
```

Notion sur les structures des données(3/4)

Manipulation

```
int main(int argc, char *argv[])
{
 struct Personne1 etudiant;
printf("Quel est votre nom ? ");
scanf("%s", etudiant.nom);
printf("Votre prenom ? ");
scanf("%s", etudiant.prenom);
printf("Votre addresse? ");
scanf("%s", etudiant.addresse);
printf("Votre Téléphone? ");
scanf("%s", etudiant.tel);
printf("Votre ville ? ");
scanf("%s", etudiant.ville);
printf("Vous vous appelez %s %s votre adresse %s", etudiant.prenom,
etudiant.nom,etudiant.addresse,etudiant.tel,etudiant.ville);
 return 0;
}
```

Notion sur les structures des données(4/4)

Manipulation

```
int main(int argc, char *argv[])
{
 Personne2 etudiant;
printf("Quel est votre nom ? ");
scanf("%s", etudiant.nom);
printf("Votre prenom ? ");
scanf("%s", etudiant.prenom);
printf("Votre addresse? ");
scanf("%s", etudiant.addresse);
printf("Votre Téléphone? ");
scanf("%s", etudiant.tel);
printf("Votre ville ? ");
scanf("%s", etudiant.ville);
printf("Vous vous appelez %s %s votre adresse %s",
etudiant.prenom,
etudiant.nom,etudiant.addresse,etudiant.tel,etudiant.ville
);
return 0;
```

Définition d'un appel système (1/3)

A retenir

Un appel système est une fonction système qui permet au programmeur de communiquer directement avec le noyau du système d'exploitation.

Les commandes sont des instructions usuelles exécutées par le shell: **cd, ls, mount, chown, etc.** ces commandes sont elles-mêmes écrites en C, et se basent sur des appels système.

Définition d'un appel système (2/3)

Attention!

Il peut arriver qu'une commande et un appel système aient le même nom.
Ex: chown,chmod,mount...

Définition d'un appel système (3/3)

Où trouver de l'aide?

L'Unix est système qui s'explique par lui-même. La ***man*** est le compagnon du programmateur système.

En général, la section 2 du man permet d'accéder à la page de manuel des appels système.

Ex:

man 2 stat

Les appels systèmes relatifs au systèmes de fichiers

- **mount(2), umount(2), getfsstat(2)**
- **stat(2), lstat(2),fsstat(2)**
- **chmod(2), lchmod(2) et fchmod(2)**
- **chown(2), lchown(2) et fchown(2)**
- **utimes(2), lutimes(2) et futimes(2)**
- **link(2), unlink(2)**
- **rename(2)**
- **symlink(2), readlink(2)**
- **opendir(3), readdir(3) et closedir(3)**
- **chdir(2), fchdir(2)**

Afficher les informations sur un fichier

- Écrire un programme en C qui affiche les informations sur un fichier passé en paramètre.
- Vous devez utiliser l'appel système **stat(2)** ou **lstat(2)**

Afficher le contenu d'un répertoire

- Écrire un programme en C qui affiche le contenu d'un répertoire
- Vous devez utiliser les appels systèmes **opendir(2),readdir(2)** et **closedir(2)**