TP. Système d’exploitation 02
Révision

BELKHIR Maria

Programmation C sous Linux

Quelles sont les étapes par lesquelles passe un problème en information jusqu'à sa résolution (du problème a l'exécutable) on explique les étapes

- **Problème ➔ algorithme ➔ code.c ➔ code objet ➔ exécutable**

La syntaxe de la commande gcc ? **gcc -o code code.c**

Que représentent les arguments **`argc`** et **`argv`** de la fonction **`main`** ?

- Argc : variable de type entier, elle contient le nombre d'arguments que l'utilsateur doit faire entrer lors de l'execution
- Argv : variable de type tableau de pointeur, elle pointe vers les @ des variables utilisées dans le programme. Le argv[0] : pointe vers le nom du programme lui-même.

Manipulation de fichiers sous Linux - Partie 1

access() : Vérifier les permissions d'utilisateur réel à un fichier

```
#include <unistd.h>

int access(const char *pathname, int flag);
```

Flag	Rôle
F_OK	Vérifie l'existence d'un fichier
R_OK	Vérifie le droit d'accès en lecture
W_OK	Vérifie le droit d'accès en écriture
X_OK	Vérifie le droit d'accès en exécution

Manipulation de fichiers sous Linux - Partie 1

open() : est utilisé pour ouvrir un fichier ou créer un nouveau fichier s'il n'existe pas encore.

```
#include <fcntl.h>

int open(const char *pathname, int flags, /* mode_t mode */ );
```

Flag	Rôle
O_RDONLY	Ouvrir le fichier en lecture seulement
O_RDWR	Ouvrir le fichier en lecture/écriture
O_WRONLY	Ouvrir le fichier en écriture seulement
O_APPEND	Vérifie le droit d'accès en exécution
O_CREAT	Créer le fichier s'il n'existe pas
O_TRUNC	Ouvrir le fichier et supprimer son contenu
O_EXCL	Utilisé avec O_CREAT. Si le fichier existe, O_CREAT sera ignoré.

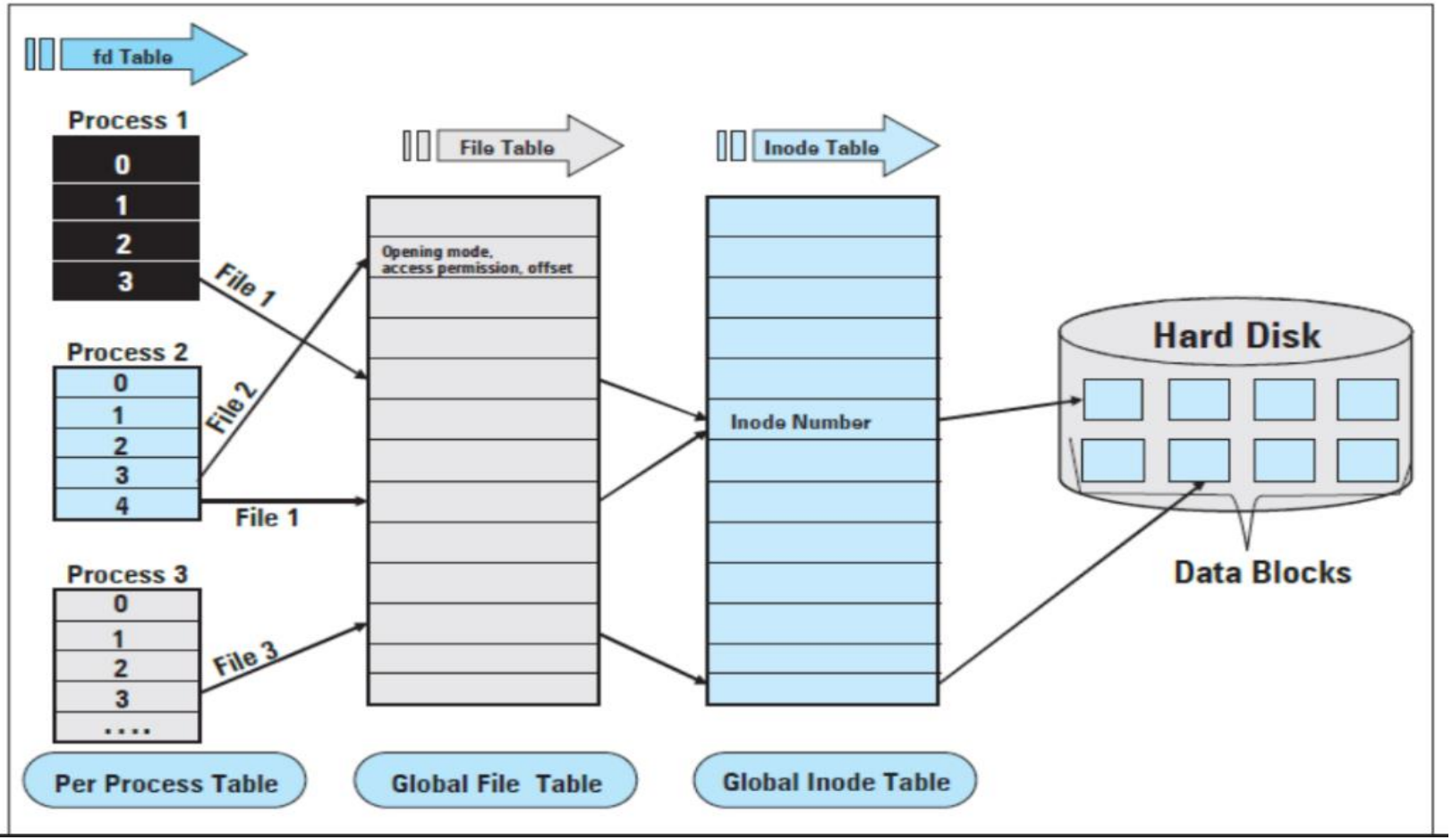
Manipulation de fichiers sous Linux - Partie 1

t_mode mode: Constante qui définit les permissions d'accès.

Define	Value
S_IRUSR	0400 / Droit de lecture pour utilisateur
S_IWUSR	0200 / Droit d'écriture pour utilisateur
S_IXUSR	0100 / Droit d'exécution pour utilisateur
S_IRGRP	0040 / Droit de lecture pour le groupe
S_IWGRP	0020 / Droit d'écriture pour le groupe
S_IXGRP	0010 / Droit d'exécution pour le groupe
S_IROTH	0004 / Droit de lecture pour les autres
S_IWOTH	0002 / Droit d'écriture pour les autres
S_IXOTH	0001 / Droit d'exécution pour les autres

Manipulation de fichiers sous Linux - Partie 1

Descripteur de fichier : est un entier qui identifie de manière unique un fichier ouvert ou un autre type de flux de données.
0 :stdin/ 1 : stdout /2 : std_err.



Manipulation de fichiers sous Linux - Partie 1

```
#include <unistd.h>

ssize_t read(int fd, void buf[count], size_t count);
```

```
#include <unistd.h>

ssize_t write(int fd, const void buf[.count], size_t count);
```

fd: descripteur de fichier qu'on veut lire/écrire.

buf[count]: structure de donnée dans laquelle on récupère/enregistrer l'information

count: le nombre d'octets qu'on veut lire/écrire à la fois.

Ils retournent -1 en cas d'erreur. Quand read arrive à la fin du fichier elle retourne 0.

Manipulation de fichiers sous Linux - Partie 2

Stat() : est utilisé pour obtenir des informations sur un fichier, comme ses attributs (par exemple, sa taille, son type, ses permissions) et d'autres métadonnées.

```
#include <sys/stat.h>
#include <sys/types.h>
int stat(const char *pathname, struct stat *StatusBuf);
```

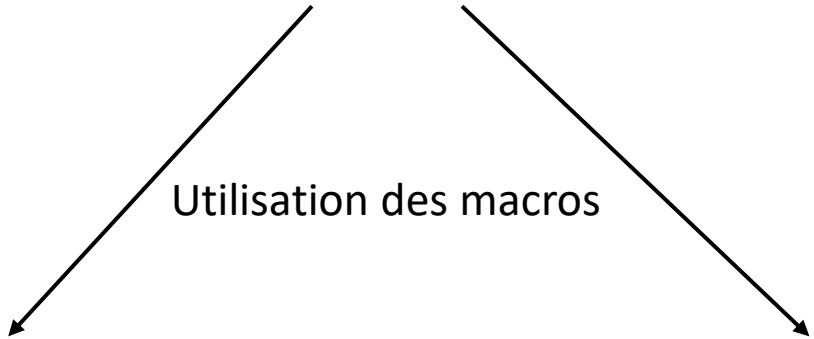
- Stat() retourne une structure de donnée de type « struct stat ». Cette dernière est utilisée pour stocker les informations récupérées par stat().
- Elle retourne -1 en cas d'erreur.

Manipulation de fichiers sous Linux - Partie 2

```
struct stat {
   dev_t     st_dev;          /* Identifiant du périphérique contenant le fichier */
   ino_t     st_ino;          /* Numéro d'inode */
   mode_t    st_mode;     /* Mode du fichier (permissions et type) */
   nlink_t   st_nli           /* Nombre de liens */
   uid_t     st_uid;          /* Identifiant de l'utilisateur propriétaire */
   gid_t     st_gid;          /* Identifiant du groupe propriétaire */
   dev_t     st_rdev;       /* Identifiant du périphérique (s'il s'agit d'un périphérique spécial) */
   off_t     st_size;         /* Taille en octets */
   blksize_t st_blksize;  /* Taille de bloc optimale pour le système de fichiers */
   blkcnt_t  st_blocks;  /* Nombre de blocs alloués */
   time_t    st_atime;   /* Temps du dernier accès */
   time_t    st_mtime;  /* Temps de la dernière modification */
   time_t    st_ctime;    /* Temps du dernier changement */
};
```

Manipulation de fichiers sous Linux - Partie 2

```
mode_t    st_mode;     /* Mode du fichier (permissions et type) */
```



Si (S_ISREG(StatusBuffer.st_mode) = 0)
Alors c'est un fichier régulier

Si (S_ISDIR(StatusBuffer.st_mode) = 0)
Alors c'est un répertoire

Les macros du langage de programmation C sont un outil puissant qui permet aux développeurs de définir des morceaux de code, des constantes de type fonction réutilisables.

Manipulation de fichiers sous Linux - Partie 2

```
uid_t    st_uid;        /* Identifiant de l'utilisateur propriétaire */
gid_t    st_gid;        /* Identifiant du groupe propriétaire */
```

```
#include <pwd.h>
...
uid_t id = 0;
struct passwd *pwd;


pwd = getpwuid(StatusBuffer.st_uid);
```

Manipulation de fichiers sous Linux - Partie 2

opendir(), readdir et closedir sont des fonctions de la librairie C qui permettent d'ouvrir, lire et fermer un repertoire, respectivement.

```
#include <dirent.h>
#include <sys/types.h>

DIR *opendir(const char *dirname);
struct dirent *readdir(DIR *dirp);
int closedir(DIR *dirp);
```

- Opendir() retourne un pointeur vers une structure de donnée de type DIR.
- Readdir() lit le contenu du flux et retourne une pointeur vers une autre structure de donnée de type struct dirent.

Manipulation de fichiers sous Linux - Partie 2

La structure dirent contient les informations suivantes:

```
struct dirent {
ino_t          d_ino;       /* Numéro d'inode de l'entrée de répertoire */
off_t          d_off;       /* Décalage à l'entrée suivante de readdir() */
unsigned short d_reclen;    /* Longueur de cet enregistrement */
unsigned char  d_type;      /* Type de fichier (DT_REG, DT_DIR, etc.) */
char           d_name[256]; /* Nom de l'entrée de répertoire */
};
```

Manipulation de fichiers sous Linux - Partie 2

- Getopt(): est une fonction de la librairie C qui parcourt la ligne de commande, elle permet d'analyser les options à caractère court ou long d'une manière flexible.
- Getopt() retourne le caractère se trouvrant juste après le caractère « - » dans la ligne de commande et retourne NULL à la fin de la ligne de commande.
- La valeur de l'index de l'argument se trouvant juste après l'option sera enregistrer dans la variable ***optind*** et le contenu de l'argument lui-même sera enregistrer dans la variable ***optarg***

```
#include <unistd.h>

    int getopt(int argc, char *argv[], const char *optstring);
```

Manipulation de fichiers sous Linux - Partie 2

```
$ gcc –o exemple exemple.c
$./exemple –f Documment
```

```
#include <unistd.h>

int main(int argc, char** argv)
{
        ……
    while ((int opt = getopt(argc, argv, “f")) != -1) {
        switch (opt) {
        case ‘f':
            ……
            break;
}
```

optind = 2
optarg = “ Documents “