

PDS

① Cour du 11/09

POSIX : normalisation du syst de ~~fichier~~
de l'interface

Un cpu est composé de deux modes :

- noyau pr OS et système
- utilisateur pr app / Bibliot / Prog

P DS :

• protocole d'accès au périph, etc

{ - Gérer les ress • Organisat°, arborescence ...

- Abstraire → { • représentat° concrète des données

- Sécuriser • Virtualisat°
 { Fichier ordinaire d'objets
 répertoire : liste d'entité

valable pr tt syst.

• contrôle d'accès

read exec

RWX → proprio

writ° → Groupe

→ public

3 Petites

par groupe

50	25	12	6	3	32	1
400	50	25	13	7	4	

PDS

① car du 18/09

les assert sont très utile pour connaître
l'erreur qui est survenue, à quelle ligne etc.

Lister le contenu d'un répertoire

- ouvrir
- Boucle de lecture (Position courante)
- fermer

obtenir la cible d'un lien symbolique

```
while (de = readdir(rep)) != NULL {  
    printf ("%d %s", de->d_ino, de-> );  
    fflush(strdout);  
    res = stat (de->s_d_name, &st);  
    assert (res != -1);  
    if (S_ISLINK (st.st_mode)) {  
        res = readlink (de->name, cible, TAILLE-1);  
        assert (res != -1);  
        cible [res] = '\0';  
        printf ("-%s\n", cible);  
    }  
}
```

-D XOPEN_SOURCE → versionnée de POSIX

Système de Fichier (FS) :

Slogan: "tout est fichiers"

Obj:

- Persistence
- Communicat° (inter-CPU)
- Structurat° En grouponne

Deux composants:

- Accès du périphérique
- Organisat° des données sur le périph.

Arborescence ↗

Chemin:
- Absolu : depuis la racine /
- relatif : pas absolu

Dans tout répertoire (liste d'entrée):

Deux entrée :
! . ! → répertoire courant
! . ! → _____ parent

Chemin: → (numéro de périphérique,
numéro D'i - noeud)
identifiant unique

→ {Données
Contenu
Propriété}

Deux octets particuliers: 'x' 'y'
↓

racine et séparateur

entre un répertoire et une entrée

Lors d'on maploc on vérifie qu'il a fonctionné
avec un assert.

I-nœud → caractérise le type de chemin

Système Type de noeud opération	Fichier ordinaire	Répertoire
ouvrir	open / creat	opendir
Lire	read	readdir
écrire	write	creat / mkdir / unlink, rmdir
fermer	close	closedir

ouverture d'un chemin :

chemin → N° périphérique
N° d'i-Noeud → i-noeud

~~unlink~~

→
Pourquoi n'y a plus de chemin physique, le
garbage collector détruit l'objet

Type de noeud OP	Pien symbolique
ouvrir	Symlink (créer)
Lire	
écrire	
fermer	

} → ReadLink

Descripteur de fichier : Ps -> Pv → Tmp/Pog

identifiant à rappeler

Dans les appels système suivants



Exo cat.c: regrouper les include nécessaires

Define ~~taille~~ TAILLE = 16

```
int main (int argc, char *argv[3]) {  
    int fd_fect, afficher;  
    char Tampon[TAILLE];  
    int Pos;  
  
    assert (argc > 1);  
    fd_fect = open (argv[1], O_RDONLY);  
    assert (fd_fect != -1);  
  
    while ((Pos = read (fd_fect, Tampon, TAILLE)) > 0) {  
        afficher = sprintf (Tampon, Pos);  
        assert (afficher == Pos);  
    }  
    assert (Pos != -1);  
    close (fd_fect);  
    return 0;  
}
```

à la place:

```
for (int i=0; i< Pos; i++) {  
    printf ("%c", Tampon[i]);  
}
```

Où: (en mieux):

```
afficher = write (STDOUT_FILENO, Tampon, Pos);  
assert (afficher == Pos);
```

PDS

corr ② du 25/03
corr ① du 9/03

ut l'isateur root : c'est le plus haut utilisateur

Dans l'fd : qui est le fd d'un proc.

- 0 ↗
 - 1 ↗
 - 2 ↗
- sont des liens symboliques et font référence aux différentes sorties (stat, err).

Bash ≠ shell



↳ exécute le code

GUI → nommé aussi pbs
lien machine - User

Plusieurs proc partagent les mêmes liens symboliques
cela est la première notion de synchronisation

On réalise un code qui exécute : ls -al /tmp

execp

int

main (argv , argc) {

assert(0)

↓

One sat
pas d'exec

← execp (" ls ", "ls", "-al", "/tmp", (char *)NULL)

return 0;

?

system (read one command shell)