Alexandre Louvet - 2024-11-28

Fiche de révision Programmation Système

Cette fiche de révision n'est pas exhaustive, elle regroupe seulement le minimum à savoir pour comprendre le cours et les TPs et pour établir un point sur son niveau de connaissances.

Bash

⊙ *Qu'est-ce qu'un SE?*

Slide 1, Chapitre 3-7

⊙ *Qu'est-ce qu'un shell?*

Slide 1, Chapitre 16-17

⊙ *Comment sont représentées les permissions sous linux et comment les manipuler?*

rwx(user)rwx(groupe)rwx(others)

Commandes à connaître

```
mkdir #créé un dossier
pwd #écrit le chemin courant
cd #se déplace dans l'arborescence de fichiers
rm #supprime un fichier (-r pour supprmer un dossier)
mv #déplace un fichier
cp #copire un fichier (-r pour supprmer un dossier)
chmod #modifie les permissions d'un fichier

cat #concatene des fichiers et les écrit dans la sortie standart
grep #recherche une regex/chaîne de caractères dans une autre
wc #compte le nombre de mots/lignes/octets dans un fichier
cut #enlève des sections de chaque ligne d'un fichier
```

⊙ *Comment rediriger la sortie/l'entrée d'une commande?*

>, <, >>, <<

```
$>echo "Hello world!" > helloworld.txt
$>echo "Hello world!" >> helloworld.txt
$>cat helloworld.txt
Hello world!
Hello world!
```

⊙ *Comment emboîter des commandes?*

On utilise le symbole | pour diriger la sortie de la commande de gauche vers celle de droite:

```
$>ls -l | grep "hel"
.rw-r--r-- user date helloworld.txt
```

⊙ *Comment exécuter une commande en arrière-plan?*

En lançant la commande suivie de &.

⊙ *Comment passer des arguments à une commande et comment les utiliser dans un script bash?*

On les met à la suite de la commande séparés par des espaces. Dans le script `$n` fait référence au n-ième paramètre (`$0` étant le nom du script).

⊙ *Revoir la syntaxe de:*

`if.. then ..fi`, `case .. esac`, `for .. do .. done` Slide 34-35, Chapitre 1,

`while .. do .. done` Slide 36, Chapitre 1,

`function` Slide 38, Chapitre 1,

expressions arithmétiques `$(())$` et tests `$[ ]$` Slide 33, Chapitre 1

⊙ *Comment utiliser une commande `bash` dans un script?*

`$(cmd)`

Processus

⊙ *Qu'est-ce qu'un processus?*

Slide 2, Chapitre 3

⊙ *Comment on identifie un processus?*

Grâce à son PID. On peut lister les processus grâce aux commandes: `ps`, `pstree`.

En `C`, la commande `pid_t getpid()` renvoie le PID du processus.

⊙ *Comment créer un processus fils?*

`pid_t pid = fork()` en `C`

⊙ *Quel est l'utilité du recouvrement? Comment en faire un?*

Pour "remplacer" un programme i.e. lui attribuer les ressources du processus actuel pour s'executer à la place.

Slide 12, Chapitre 2

⊙ *Comment terminer l'exécution d'un processus?*

Commande `exit()` (à ne pas oublier pour libérer les ressources attribuées).

⊙ *Comment un processus fils obtient des ressources de calcul? Comment les libère-t-elles? Quelle commande pour assurer la terminaison de fils dans de bonnes conditions?*

Le processus fils partage les ressources attribuées au processus père. Il les libère en se terminant avec `exit()` et il faut `wait()` dans le père pour assurer que celui-ci libère bien les ressources attribuées au fils.

⊙ *Quel statut a un processus fils dont le père a fini son exécution*

Processus orphelin.

⊙ *Quel statut a un processus fils qui a fini son execution et attend la terminaison du processus père*

Processus zombie.

Ordonnancement

⊙ *Qu'est-ce que l'ordonnancement des tâches, pourquoi cela existe?*

C'est l'organisation des tâches et l'attribution des ressources de calcul. Cela sert à éviter que des processus n'aient pas de temps de calcul ou que d'autre en aient trop.

⊙ *Comment calculer le temps de rotation d'un processus?*

T complétion - T arrivée

⊙ *Comment calculer le temps d'attente d'un processus?*

T obtention (temps avant de commencer les calculs) - T arrivée

⊙ *Connaître les 4 algos d'ordonnancement détaillés dans le cours*

Slide 31-36, Chapitre 2

⊙ *Comment modifier la priorité d'un processus?*

Commande `nice` pour ajuster la priorité des processus.

Tubes

⊙ *Quelle est l'utilité d'un tube (pipe)?*

Le but est de permettre la communication entre des processus via un système de lecture/écriture dans un fichier.

⊙ *Comment créer un tube?*

```
int p[2]; // Un tube est un tableau de deux éléments, le premier pour la lecture, le second pour l'écriture
pipe(pipe); // Transforme le tableau pipe en pipe
```

⊙ *Comment lire dans un tube? Peut-on lire toutes les informations dans le tube dans un ordre arbitraire?*

On lit dans le tube avec la commande `read`:

```
char c;
int taille = sizeof(char);
read(p[0], &c, taille); // On lit dans le tube p une information ayant la taille d'un char (i.e. un char si on a bien écrit des caractères dans le tube) et on stocke cette information dans c (passé en référence)
```

On lit les informations contenues dans un tube dans l'ordre où elles ont été écrites (First In, First Out).

⊙ *Comment écrire dans un tube?*

On écrit dans un tube avec la commande `write`:

```
write(p[1], "a", 1); // On écrit 1 caractère de la chaîne "a" dans le tube p
```

⊙ *Comment créer un tube nommé*

On utilise la commande `mkfifo`:

```
mkfifo nom # créé le tube "nom" en bash
```

```
mkfifo("nom")
```

⊙ *Comment ouvrir/fermer un tube nommé?*

On utilise les commandes `open()` et `close()`:

```
int tube1 = open("tube1", "O_WRONLY"); // Ouvre le tube "tube1" en mode écriture
seule
int tube2 = open("tube2", "O_RDONLY");  // Ouvre le tube "tube1" en mode lecture
seule
close(tube1);
```

⊙ *Que se passe-t-il si un tube est lu par processus 1 et écrit par processus 2 et que processus 1 ferme le tube?*

Processus 2 ne pourra plus écrire dans le tube car il faut toujours qu'un tube ait un lecteur.

Système de fichiers

⊙ *Quels sont les différents types de fichiers sur linux?*

- fichiers réguliers (e.g. fichiers textes)
- répertoires
- liens symboliques
- tubes
- sockets
- fichiers spéciaux
  - en mode bloc
  - en mode caractère

Slide 3, Chapitre 21

⊙ *Qu'est-ce qu'un bloc logique dans le stockage?*

C'est la plus petite unité de stockage d'un système de fichiers.

⊙ *Qu'est-ce qu'un INODE?*

C'est la structure qui contient toutes les informations sur un fichier. En particulier des références aux adresses des blocs où sont enregistrés le fichier (directement ou indirectement).

⊙ *Que signifie qu'un bloc d'un fichier est référencé indirectement dans son inode?*

Cela signfie que l'inode contient une référence à l'adresse mémoire d'un bloc qui lui-même contient un pointeur pointant vers l'adresse mémoire du bloc.

⊙ *Que signifie qu'un système de fichiers est journalisé?*

Cela signifie que le système de fichier maintient un historique des changements sur les fichiers (uniquement des meta-données du fichier pour les systèmes meta-journalisés).