

# Systèmes d'exploitation

Cours no. 13

Jean-Sébastien Coron

Université du Luxembourg

December 5, 2009

- Révision des cours précédents.
  - Corrigé des TP 5 à 10.
- Communication inter-processus.

- Enoncé:

- Ecrire un script shell `taille` qui renvoie la taille d'un fichier en octets.
- En utilisant la commande `ls -l` et la commande `cut`.
- ```
$ ls -l monfich.c
-rw-r--r--  1 guest None 60 Oct 14 11:44
monfich.c
$ taille monfich.c
60
```

- La commande cut
  - Permet de sélectionner certaines parties des lignes d'un fichier ou de l'entrée standard (si aucun fichier n'est précisé).
  - `cut [-c] [-f] list [-n] [-d delim] [-s] [file]`
- Options:
  - `-c list`: spécifie les caractères à sélectionner.
    - `-c2-5` sélectionne les caractères 2 à 5 de chaque ligne.

- Options:
  - `-f list`: sélectionne pour chaque ligne les champs spécifiés, les champs étant délimités par un caractère délimiteur.
    - `-f1,5` sélectionne les champs 1 et 5.
  - `-d delim`: spécifie le caractère délimiteur.
  - `-s`: supprime les lignes sans caractère délimiteur.
- `list`:
  - Liste de nombres séparés par une virgule, avec `-` pour indiquer un intervalle.
  - `1,2,3,5` ou `1-3,5`

- Exemples:

- `$ echo "hello" | cut -c 2-4`  
ell

- `$ echo "he ll o wo" | cut -f2,3 -d' '`  
ll o

- Solution

```
#!/bin/bash
x='ls -l $1'
echo $x | cut -f5 -d' '
```

- La commande echo supprime les espaces en trop:

- `$ echo a b c`  
a b c

- Ecrire un script `existe` qui détermine si un fichier existe ou pas.
  - `$ existe toto`  
Le fichier `toto` existe
  - `$ existe tata`  
Le fichier `tata` n'existe pas

- Solution:

```
#!/bin/bash
if [ -s $1 ]
then
    echo "Le fichier $1 existe."
else
    echo "Le fichier $1 n'existe pas."
fi
```

- Ecrire un script shell `pidof` prenant en entrée le nom d'un programme et affichant la liste des numéros de processus correspondant à ce programme.
  - ```
$ pidof bash
1672
2888
$ pidof xterm
1025
2112
```

- Corrigé:

```
#!/bin/bash
```

```
ps -s | grep $1 | cut -c4-7
```

- `ps -s`

- Affiche la liste des processus s'exécutant sur la machine.

- `grep $1`

- Sélectionne les lignes correspondant au programme passé sur la ligne de commande.

- `cut -c4-7`

- Sélectionne les caractères 4 à 7.

- Enoncé:
  - Ecrire un programme `copie` qui copie un fichier dans un autre.
  - Le programme devra avoir le même fonctionnement que la commande UNIX `cp`:  
`copie fich1 fich2`  
copie le fichier `fich1` en `fich2`.

```
#include <stdio.h>
int main(char argc, char *argv[])
{
    FILE *f1,*f2;
    f1=fopen(argv[1],"r");
    f2=fopen(argv[2],"w");
    while(!(feof(f1)))
    {
        char c=fgetc(f1);
        if(!(feof(f1))) fputc(c,f2);
    }
    fclose(f1);  fclose(f2);
}
```

- Enoncé:
  - Ecrire en langage C un programme copie qui copie un fichier dans un autre, par blocks de 256 octets.
  - `$ copie fich1 fich2`

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    FILE *f1,*f2;
    f1=fopen(argv[1],"r");
    f2=fopen(argv[2],"w");
    char buf[256];
    int nread;
    do {
        nread=fread(buf,1,256,f1);
        fwrite(buf,1,nread,f2);
    } while (nread>0);
}
```

- Enonce:
  - Ecrire un programme qui crée un processus fils qui affiche à chaque seconde le nombre de secondes écoulées. Le processus père arrête le processus fils au bout de 10 secondes.
- Solution:

```
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
int main() {
    int i=0;
    int pidfils=fork();
```

```
if(pidfiles!=0)
{
    sleep(10);
    kill(pidfiles,SIGKILL);
} else
{
    while(1) {
        sleep(1); i++;
        printf("%d \n",i);
    }
}
}
```

- Pour faire communiquer des processus, on a besoin d'outils plus évolués que les signaux.
  - Les processus doivent pouvoir échanger de informations entre eux.
  - Un processus ne peut pas directement accéder à la mémoire d'un autre processus.
- Utilisation des fichiers.
  - Un processus peut écrire dans un fichier qui sera lu par un autre processus.
  - Inconvénients: lenteur, comment savoir qu'un processus a terminé d'écrire dans un fichier ?

- Sous UNIX, un tube permet de rediriger la sortie d'une commande vers l'entrée d'une autre:
  - `ls | sort -r`
- Faire communiquer deux processus en C:
  - Communication unidirectionnelle vers une commande:  
`popen()`
  - Communication bidirectionnelle entre processus: tubes FIFO:  
`mkfifo()`.

# La fonction `popen()`

- `FILE *popen(char *command, char *type)`
  - Ouvre un tube avec le processus lancé par la commande `command`.
  - "r" en lecture, "w" en écriture.
  - Retourne un pointeur sur un flux.
  - `void pclose(FILE *stream)` permet de fermer un tube.
- On utilise `fprintf()`, `fwrite()`, `fscanf()`, `fread()` pour communiquer avec le tube.

- Compter les lignes d'un fichier.

```
#include <stdio.h>
int main()
{
    char buf[256];
    FILE *f=popen("wc -l toto.c","r");
    fread(buf,1,256,f);
    pclose(f);
    printf("%s",buf); // 20 toto.c
    int n=atoi(buf);
    printf("%d\n",n); // 20
}
```