Master 2 CCI - 2014-2015
Introduction aux Systèmes et Réseaux
L. Gonnord & J.-P. Gelas
http://laure.gonnord.org/pro/teaching/systemeCCI.html

Examen 2014-2015 - 2h – Éléments de correction

Calculatrices, téléphones et autres dispositifs portables interdits. Tous documents papier autorisés.

Consignes Vous pouvez sauter des questions, mais pensez à bien numéroter vos réponses. On attend des réponses claires et concises, mais avec des explications !

1 Langage de commandes et scripts

Question 1:

Supposons que le contenu de `/usr/bin` soit le suivant :

```
add-apt-repository
addpart
addr2line
adhocfilelist
afm2afm
afm2pl
afm2tfm
aften
aircrack-ng
airdecap-ng
airdecloak-ng
airolib-ng
aj
aj5
ajbrowser
ajc
ajdoc
alacarte
aleph
allcm
```

Qu'affiche la commande suivante :

```
ls /usr/bin/a[f-i]r*
```

Solution: Cela imprime la liste (avec chemin) des fichiers et répertoires dont le nom commence par a, suivi de toute lettre "entre f et i", suivi de r. Si le nom est un répertoire cela affiche aussi le contenu. Ici on obtient :

```
/usr/bin/aircrack-ng        /usr/bin/airdecap-ng
/usr/bin/airdecloak-ng      /usr/bin/airolib-ng
```

Question 2:

Voici le contenu d'un fichier annuaire nommé `annu.txt` :

```
jeanne:martin:060606:82
martin:dupont:777:020
juliette:dupond:1515:17
```

(a) Qu'affiche la commande suivante : `cat annu.txt | cut -d: -f 2,3`

Solution: Elle affiche les colonnes 2 et 3 du fichier, les colonnes étant délimitées par le caractère ' :' :

```
martin:060606
dupont:777
dupond:1515
```

(b) Comment afficher les lignes d'information concernant tous les `dupont` ou `dupond` ?

Solution:

```
grep dupon[td] annu.txt
```

Question 3:

Vous tapez la commande `ls -l`. Vous obtenez à l'affichage les deux lignes suivantes. Pour chacune donnez le maximum d'informations.

```
-rwxr--r-- 1  moi ens   577     2007-07-17 22:28 tagpics.sh
drwx------ 13 moi ens  4096     2009-08-16 23:59 tmp
```

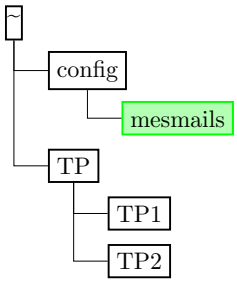
Solution:

— La première ligne indique l'existence d'un fichier, modifié pour la dernière fois le 17/07/2007 à 22h28, dont la taille est 577 octets. Ce fichier a pour propriétaire `moi`, du groupe `ens`. Le propriétaire peut éditer, lire, exécuter le fichier. Les autres personnes (groupe, reste des utilisateurs de la machine) peuvent lire le contenu (et le copier).

— La deuxième ligne indique l'existence d'un répertoire modifié pour la dernière fois le 16/8/2009 à 23h59, dont la taille est 4096 octets. Il contient 13 sous répertoires (dont `.` et `..`. Seul `moi` peut lire (ls), écrire (créer un nouveau fichier, renommer le répertoire), ou exécuter (`cd tmp`) ce répertoire.

Question 4:

Donner une suite de commandes qui permet de créer l'arborescence suivante dans votre répertoire personnel (répertoires encadrés, fichier en grisé). On créera le fichier à la ligne de commandes (sans faire appel à un éditeur) :



Solution: On commence par exemple par se mettre à la racine de son compte utilisateur :

```
cd ~
mkdir config TP
touch config/mesmails
mkdir TP/TP1 TP/TP2
```

2 Script

Question 1:

L'objet de cette partie est d'écrire un script nommé `toto.sh`. On écrira un unique script, et le numéro des questions sera écrit en commentaire. **Attention les différentes parties du script peuvent être dans un ordre différent des questions.**

Le script doit :

1. Stocker la date courante dans la variable 'now' (commande `date`)

2. Stocker le pid (l'identifiant du processus courant) de votre script dans la variable 'pid'. *On pourra utiliser le fait que le pid est stocké dans la variable* `$$`
3. Tester la présence du fichier `/tmp/exam14.sh.lock`. Si ce fichier existe, sortir avec un message d'erreur informatif. Dans le cas contraire, le créer avec la commande `touch`.
4. Dans un fichier de log nommé `toto.log`, écrivez la date, suivie du message du style "le script a bien démarré". Faites attention à ne pas écraser ledit fichier à chaque écriture. Ci dessous un exemple :

```
$ cat toto.log
Mon Nov 20 14:58:18 CEST 2014: ./toto.sh a bien démarré
Mon Nov 20 14:58:33 CEST 2014: ./toto.sh a bien démarré
Mon Nov 20 14:59:37 CEST 2014: ./toto.sh a bien démarré
```

5. Tester la présence du fichier de log (décrit ci-dessus). Si il existe, comptez le nombre de lignes n du fichier de log et affichez un sur la sortie standard le message "le script a déjà démarré <n> fois", ou le cas échéant "C'est la première fois que le script est exécuté".
6. Terminer en supprimant le fichier verrou `/tmp/exam14.sh.lock` et retourner 0.
7. À quoi peut donc servir le fichier verrou ?

Solution:

```
#!/bin/bash

now=$(date) #stocker la date
pid=$$      #stocker le pid

#questions 3 et 7 Ce fichier sert à garantir l'absence de deux exécutions
#concurrentes de ce script
lockfile=/tmp/exam14.sh.lock

if [ -e $lockfile ]
then
    echo "Fichier de lock présent!"
    exit 1
else
    touch $lockfile
fi

#questions 4 et 5

logfile=./toto.log

if [ -e $logfile ]
then
    nblines=$(cat $logfile| wc -l)
    echo "Le script a déjà démarré $nblines fois"
else
    echo "C'est la première fois que ce script est exécuté"
    touch $logfile
fi

echo "$date : $0 a bien démarré" >> $logfile

rm -f $lockfile
exit 0
```

3 Fichiers et processus

Question 1:

Vous venez de recevoir le petit code source ci-dessous par email de la part d'un inconnu qui vous veut du bien. Le message joint explique que vous devez le compiler et l'exécuter sur votre machine. Expliquez *en 3 lignes* pourquoi vous ne le ferez pas ou plus précisément expliquez ce qui risque d'arriver à votre machine si vous l'exécutez.

```c
#include <unistd.h>

int main(void)
{
  while(1) {
    fork();
  }
  return 0;
}
```

Solution: Le programme va créer toujours plus de processus, causant une surcharge en termes de ressources mémoires. Le système ne pourra rapidement plus s'exécuter correctement, causant un plantage de la machine.

Question 2:

Considérons le programme suivant (dans lequel on a enlevé les include) :

```c
int pid,fp,n,j,u;
char ch[4];

fp = open("fichier1.txt",O_RDONLY);
printf("Valeur fp à l'ouverture = %d\n",fp);
n = fork();

if (n == -1) {
  exit(-1);
}

if (n == 0) {
  for (j=0;j<3;j++){
    read(fp,ch,4);
    printf ("Hello 0 : %s\n",ch);
  }
} else {
  for (j=0;j<3;j++){
    read(fp,ch,4);
    printf ("Hello 1 : %s\n",ch);
  }
} // fin else
close(fp);
return 0;
}
```

ainsi qu'une exécution :

```
Valeur fp à l'ouverture = 3
Hello 0 : Bonj
Hello 0 : our
Hello 0 : les
Hello 1 : peti
Hello 1 : ts l
Hello 1 : oups
```

(a) Expliquer cette exécution. Que contenait le fichier `fichier.txt` au lancement du programme ?

Solution: Le fichier `fichier.txt` contenait du texte, commençant par "Bonjour les petits loups". Le programme ouvre ce fichier et stocke son file descriptor dans la variable `fd` (le numéro est 3 car les fd 0,1,2 sont réservés pour l'entrée standard, ...). Ensuite un processus fils est créé, causant une exécution de deux processus qui exécutent le même code :

- Le processus fils exécute la partie "then" du test ($n = 0$) : il lit 3 fois dans le fichier 4 caractères, et après chaque lecture écrit sur la sortie standard `Hello 0` et les 4 caractères qu'il vient de lire.
- Le processus père exécute la partie "else", et réalise des lectures dans le fichier, et des impressions sur la sortie standard, préfixés par `Hello 1`.

(b) Est-ce qu'une exécution qui n'écrirait pas les caractères du fichier dans l'ordre est possible ? Donner 3 traces d'exécutions possibles.

Solution: Il s'agit d'exécution concurrente, il n'y a aucun ordre imposé entre les instructions du père et celles du fils. En particulier, le fils peut faire une lecture, et le père lecture+impression, avant l'écriture du fils, causant une exécution qui imprimerait :

```
Hello 1 : our
Hello 0 : Bonj
...
```

(c) Rajouter ce qu'il faut pour obliger l'impression des `Hello 0` avant les `Hello 1`.

Solution: Il faut obliger le père à attendre la fin du fils, en ajoutant un appel à la fonction `wait` ligne 18, après le `else` : `wait(&n)`.

(d) Que se passe-t-il si le fichier `fichier.txt` n'existe pas ? Remédiez à ce problème.

Solution: Si le fichier n'existe pas, les lectures se font sur un fichier qui n'existe pas, et les impressions affichent la valeur initiale du tableau `ch`, c'est-à-dire des caractères quelconques car ce tableau n'a pas été initialisé correctement. On peut se prévenir de ce comportement en affichant un message d'erreur et en sortant (ajout à la ligne 7) :

```
if (fp<0) {
     fprintf(stderr,"fichier inexistant\n");
     exit(-1);
 }
```