



Étape : Licence Sciences Technologies (semestre 2)
Épreuve de : Utilisation des systèmes informatiques
Date : 18 mai 2005
Heure : 08h30
Documents manuscrits et livres autorisés
Épreuve de Monsieur Delmas

UE : INF103
Durée : 1h30

Répondre dans les cadres directement sur le sujet ou cocher lorsque nécessaire la ou les bonnes réponses –
ATTENTION : cocher une mauvaise réponse retire un point
Indiquer votre identificateur d’anonymat :

★ Peut-on faire fonctionner des programmes sur un ordinateur en l’absence de système d’exploitation ?

Oui ☐ Non ☐

★ Décrire brièvement le rôle d’un système d’exploitation

[Empty box for describing the role of an operating system]

★ Un processus n’est créé qu’à l’initiative d’un autre processus.

Oui et il n’y a aucune exception ☐ Non ☐
Oui, mais il y a une et une seule exception ☐

★ Supposons que vous exécutiez un programme contenant une instruction du type “boucle infinie” (e.g., for(;;) ou while(1)) ce processus va t’il monopoliser de manière continue et indéfiniment le processeur (en d’autres termes, les autres processus sont-ils bloqués ?)

Oui ☐ Non ☐

★ Justifier votre réponse précédente

[Empty box for justifying the previous answer]

★ Rappelez brièvement le principe de fonctionnement d’un “appel système”

[Empty box for recalling the principle of system call operation]

★ Si on insère des morceaux de code du noyaux écrits en langage machine dans une application, expliquer pourquoi et comment cette application ne pourra pas se substituer au noyau.

★ Après l'exécution des commandes ci-dessous, quel est la valeur de retour de la commande pwd ?

```
ls -ald /usr/local/bin
drwxr-xr-x  2 root root 4096 avr 14 17:51 /usr/local/bin
$ cd /usr/local/bin ; pwd
```

- 1 /usr/local/bin
- 0 bash : /usr/local/bin Permission non accordée

★ Comment faire afficher par le shell la valeur de retour de la dernière commande exécutée ?

- echo \$\$ echo ?
- echo # echo \$!
- echo \$* echo \$?

★ Quel est le système de fichiers physique natif des systèmes Unix ?

- NTFS FAT32
- ext2 iso9660

★ En supposant que l'un de vos processus Emacs ayant le pid n°13457 soit bloqué. Par quelle commande devez vous commencer pour tuer ce processus ?

- kill -15 13457 kill -9 13457

★ Justifier votre réponse précédente :

★ Sous les systèmes du types Unix, quel est le login d'usage du super utilisateur ?

- admin root
- superuser obiwan kenobi

★ Cochez les affirmations vraies : le super utilisateur peut,

- lire le contenu de tous les fichiers utilisateurs traverser tous les répertoires
- changer le mot de passe des utilisateurs connaître le mot de passe des utilisateurs
- changer les droits des fichiers des utilisateurs retrouver un fichier utilisateur effacé par mégarde
- changer les liens de parenté entre processus changer un compte utilisateur en super utilisateur

★ Expliquer ce qui différencie un processus en avant-plan d'un processus en arrière-plan

* Comment peut-on passer un processus en avant-plan dans l'état STOP ?

- Control-z Escape
Control-c Control-d

* Le script-shell mkdir-cd.sh a pour but de créer un nouveau répertoire et de placer l'utilisateur directement dans ce répertoire

```
$ cat mkdir-cd.sh
#!/bin/bash

mkdir $1
cd $1
$
```

Donner un exemple d'utilisation de ce script mkdir-cd.sh sous votre shell afin de créer et de vous placer directement dans le répertoire ~/foo (on supposera que ce répertoire n'existait pas).

* Écrire une autre solution permettant à l'utilisateur d'utiliser une commande mkdir-cd mais qui n'utilise pas de script-shell.

* Quel est l'ordre de création des processus lorsque l'on tape une commande du type :

```
$ processus-1 | processus-2 | processus-3
```

- 1, 2 et 3 sont créés en même temps 3 puis 2 puis 1
1 puis 2 puis 3 Cela peut être n'importe quel ordre

* Que produira la suite de commandes suivante :

```
$ pwd
/usr/local/bin
$ firefox &
$ date
lun avr 25 18:54:18 CEST 2005
$ xclock&
$ echo $?
0
$ ps
3882 pts/3    00:00:00 bash
27650 pts/3  00:00:23 firefox
27651 pts/3  00:00:00 xclock
27652 pts/3  00:00:00 ps
$ echo $!
```

- * En supposant que vous appartenez au groupe Unix etu et que vous ayez le login martin ; que produira la suite de commandes suivante :

```
$ pwd
/tmp
$ ls -ald /var/spool/mail
drwxrwx--- 2 root mail 4096 mai 6 2004 /var/spool/mail
$ cd /var/spool/mail 2>/dev/null && pwd ; echo SUCCES || pwd ; echo ECHEC
```

```
$ cd /var/spool/mail 2>/dev/null && (pwd ; echo SUCCES) || (pwd ; echo ECHEC)
```

- * Que produit l'exécution des commandes suivantes

```
$ MONSHELL=bash
$ sh
$ echo "/$(pwd)$MONSHELL/"
```

```
$ Control-d
$
$ ( export FOO=bar ); export BAR=foo
$ sh
$ export NAVIGATEUR=firefox
$ echo /$MONSHELL/$FOO/'$BAR'/"$NAVIGATEUR/"
```

```
$ Control-d
$ echo mon navigateur est $NAVIGATEUR
```

- * Que produit l'exécution des commandes suivantes

```
~$ cat mystere.sh
#!/bin/bash

COMMAND=$1
shift
ARGUMENTS=$*
```

\$COMMAND \$ARGUMENTS

```
~$ echo $DISPLAY
renoir:0.0
~$ chmod 555 ./execute.sh
~$ ./execute.sh echo $DISPLAY
```

```
~$ ./execute.sh echo '$DISPLAY'
```

- ★ Changer le script-shell `execute.sh` afin que
 - \$ `./execute.sh echo '$DISPLAY'`
 - produise le même résultat que
 - \$ `./execute.sh echo $DISPLAY`

- ★ Que produit l'exécution des commandes suivantes

```
$ echo "Rien ne sert de courir ; il faut partir à point." | tr -cs [:alpha:] '\n'
```

- ★ Écrire un script-shell `occurrences.sh` qui se comporte comme un filtre et qui affiche sur la sortie standard la fréquence d'apparition (classé par ordre croissant) de chaque mot provenant du flux d'entrée standard. Votre script-shell ne fera pas de différences entre majuscules et minuscules. Par exemple :

```
$ cat verlain.txt
Il pleure dans mon coeur
Comme il pleut sur la ville;
Quelle est cette langueur
Qui pénètre mon coeur?
$ ./occurrences.sh < verlain.txt
 2 mon
 2 il
 2 coeur
 1 ville
 1 sur
 1 qui
 1 quelle
 1 pleut
 1 pleure
 1 pénètre
 1 langueur
```

1 la
1 est
1 dans
1 comme
1 cette

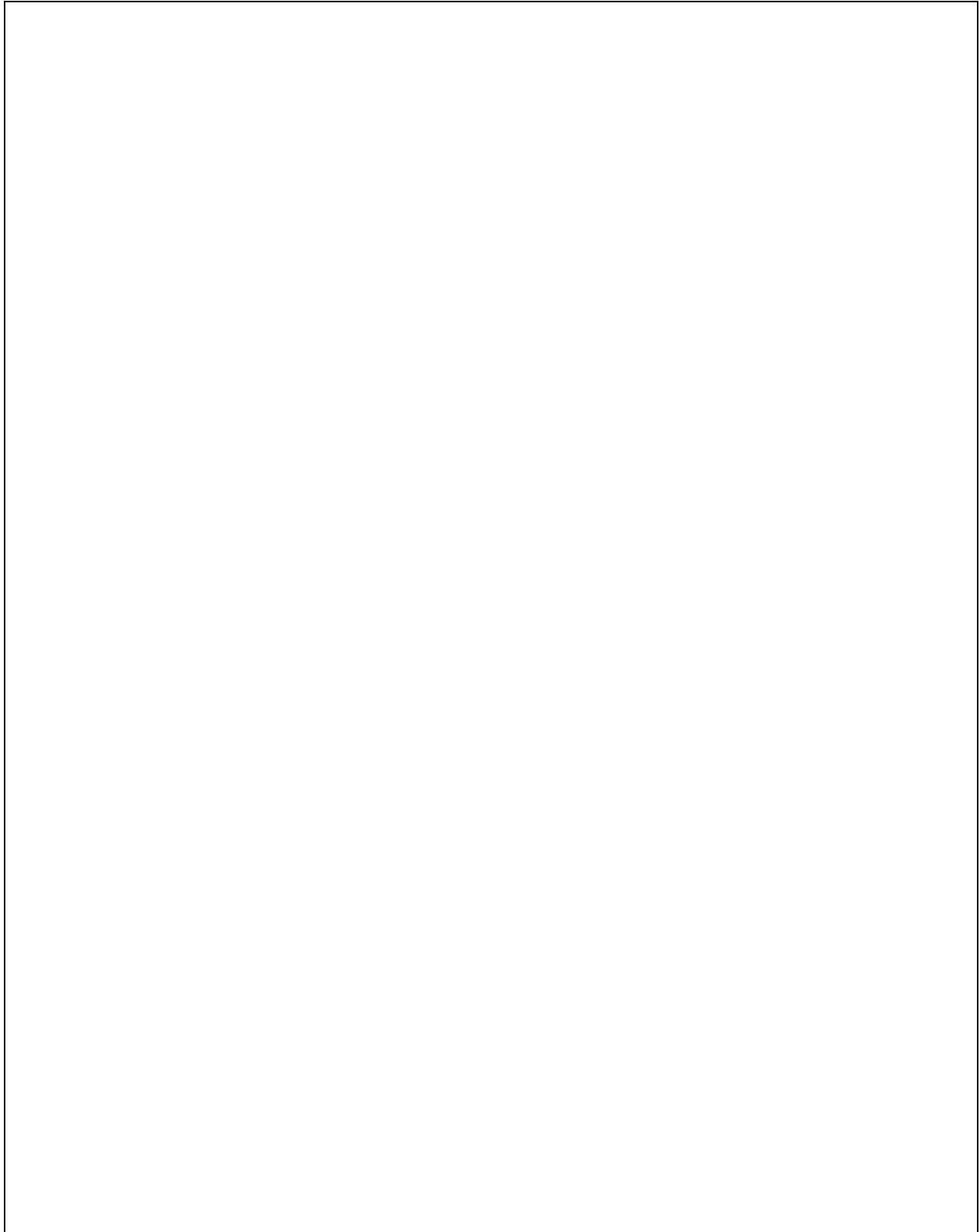
\$



- ★ Écrire un script-shell `nettoyer.sh` qui supprime à partir du répertoire courant tous les fichiers dont le nom se termine par un caractère '~' (même si le nom du fichier commence par un caractère '.') ou dont le nom débute et se termine par un caractère '#'; ces fichiers sont fréquemment des fichiers temporaires. La suppression de ces fichiers devra s'effectuer récursivement dans toute la sous-arborescence du répertoire courant et devra fonctionner même à l'intérieur des sous-répertoire dont le nom commence par '.'. Outre l'ensemble des commandes internes au shell `bash`, les seules commandes externes utilisables sont `dirname`, `basename` et bien entendu `rm`. Votre script-shell **NE DEVRA PAS** utiliser les commandes `ls`, `find`, `grep` ou `egrep`.



- ★ Modifier votre script-shell `nettoyer.sh`, afin qu'il prenne en paramètre un entier ≥ 0 , permettant de limiter la descente récursive de votre script-shell. Par exemple, `nettoyer.sh 0` supprimera les fichiers temporaires du répertoire courant mais n'ira pas dans les sous-répertoire. La commande `nettoyer.sh 2` supprimera les fichiers temporaire uniquement dans le répertoire courant, ses sous-répertoires et ses sous sous-répertoires, mais ne dépassera pas le deuxième niveau de sous-répertoires. Pour ce faire, vous pourrez utiliser la commande `expr`. Pensez également à placer un `usage` dans votre script. Si vous ne passez pas un entier en paramètre à votre script-shell, alors il n'y aura pas de limitation quant à la profondeur de la descente récursive.



FIN.

Annexes – Extraits de Pages du manuel

A **uniq**

UNIQ(1) FSF UNIQ(1)

NAME

uniq - remove duplicate lines from a sorted file

SYNOPSIS

uniq [OPTION]... [INPUT [OUTPUT]]

DESCRIPTION

Discard all but one of successive identical lines from INPUT (or standard input), writing to OUTPUT (or standard output).

-c, --count
prefix lines by the number of occurrences

-d, --repeated
only print duplicate lines

-D, --all-repeated
print all duplicate lines

-f, --skip-fields=N
avoid comparing the first N fields

-i, --ignore-case
ignore differences in case when comparing

-s, --skip-chars=N
avoid comparing the first N characters

-u, --unique
only print unique lines

-w, --check-chars=N
compare no more than N characters in lines

-N same as -f N

+N same as -s N (obsolescent; will be withdrawn)

--help display this help and exit

--version
output version information and exit

A field is a run of whitespace, then non-whitespace characters. Fields are skipped before chars.

[...]

B **tr**

TR(1) FSF TR(1)

NAME

tr - translate or delete characters

SYNOPSIS

tr [OPTION]... SET1 [SET2]

DESCRIPTION

Translate, squeeze, and/or delete characters from standard input, writing to standard output.

-c, --complement
first complement SET1

-d, --delete
delete characters in SET1, do not translate

-s, --squeeze-repeats
replace sequence of characters with one

-t, --truncate-set1
first truncate SET1 to length of SET2

--help display this help and exit

--version
output version information and exit

SETs are specified as strings of characters. Most represent themselves. Interpreted sequences are:

\NNN character with octal value NNN (1 to 3 octal digits)

\\ backslash

\a audible BEL

\b backspace

\f form feed

\n new line

\r return

\t horizontal tab

\v vertical tab

CHAR1-CHAR2

all characters from CHAR1 to CHAR2 in ascending order

[CHAR*]

in SET2, copies of CHAR until length of SET1

[CHAR*REPEAT]

REPEAT copies of CHAR, REPEAT octal if starting with 0

[:alnum:]

all letters and digits

[:alpha:]

all letters

[:blank:]

all horizontal whitespace

[:cntrl:]

all control characters

[`:digit:`]
all digits

[`:graph:`]
all printable characters, not including space

[`:lower:`]
all lower case letters

[`:print:`]
all printable characters, including space

[`:punct:`]
all punctuation characters

[`:space:`]
all horizontal or vertical whitespace

[`:upper:`]
all upper case letters

[`:xdigit:`]
all hexadecimal digits

[`=CHAR=`]
all characters which are equivalent to CHAR

Translation occurs if `-d` is not given and both SET1 and SET2 appear. `-t` may be used only when translating. SET2 is extended to length of SET1 by repeating its last character as necessary. Excess characters of SET2 are ignored. Only [`:lower:`] and [`:upper:`] are guaranteed to expand in ascending order; used in SET2 while translating, they may only be used in pairs to specify case conversion. `-s` uses SET1 if not translating nor deleting; else squeezing uses SET2 and occurs after translation or deletion.

[...]