ANNÉE 2005-2006 SEPTEMBRE 2006

Licence Sciences Technologies

Étape : Licence Sciences Technologies (semestre 2) **UE :** INF103
Épreuve de : Utilisation des systèmes informatiques **Durée :** 1h30
Date : 6 septembre 2006
Heure : 08h30
Documents manuscrits et livres autorisés
Épreuve de Monsieur Delmas

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

⋆ Cochez les systèmes d'exploitation qui sont du type Unix.

Linux ..... ☐ Windows NT ..... ☐
MacOS X ..... ☐ Sun/Solaris ..... ☐
NetBSD ..... ☐ FreeBSD ..... ☐

⋆ 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

⋆ Rappelez brièvement le principe de fonctionnement d'un "appel système"

⋆ 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.

⋆ Tous les processus se voient attribué un PID unique par le noyau, il n'y a aucune exception :

Vrai ..... ☐ Faux ..... ☐

⋆ Quel est le PID de l'ordonnanceur ?

1 ..... ☐ Cela peut être n'importe quel PID ..... ☐
Aucun ..... ☐ -1 ..... ☐

⋆ Comment faire afficher par le *shell* bash la valeur de retour d'une commande ?

echo $$ ..... ☐ echo ? ..... ☐
echo # ..... ☐ echo $! ..... ☐
echo $* ..... ☐ echo $? ..... ☐

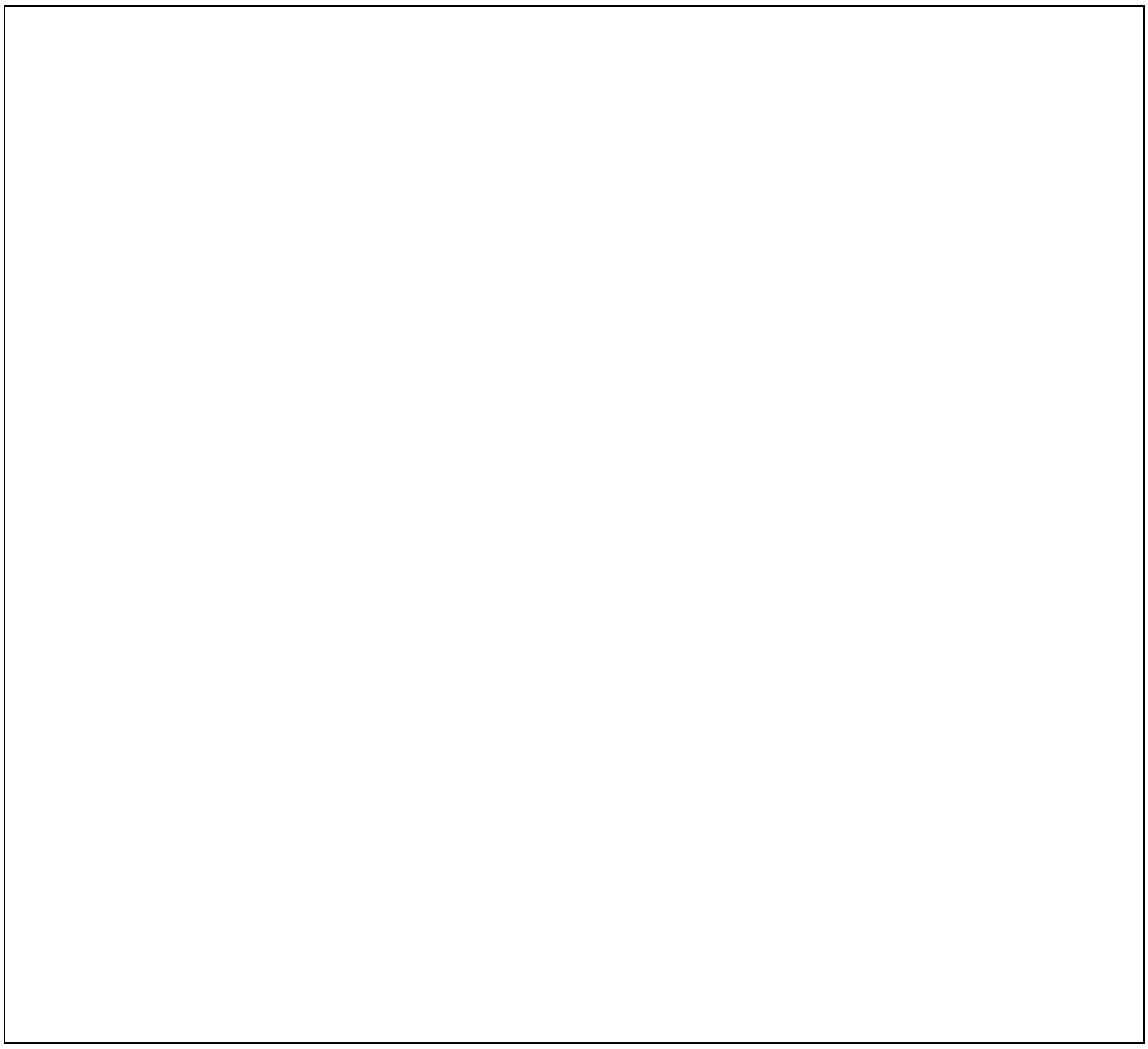
⋆ Que produit l'exécution des commandes suivantes

```
$ ls /bin
ssh ntp ynu yp yn.conf yn.rpmnew ynu.conf ynu.canz ynu.rpnew zsh
$ R='/bin/[y-z]??.*[f-w]'
$ T=R
$ echo $R
```

```
$ echo "$T"
```

⋆ Dessiner les liens de parentés à partir de votre shell ***bash*** des processus engendré par l'exécution des commandes ci-après. Vous dessinerez également dans les « patatoïdes » représentant vos processus leur table des descripteurs de fichiers (sauf pour le processus bash) et indiquerez sur quoi pointe les descripteurs 0,1 et 2.

```
$ date
mer jan 11 15:59:11 CET 2006
$ ls | more
tmp
genetique-cognitif
usi
$ (ps aux | tail -n 1000 | less)
```



⋆ Le script-shell boule-de-gomme.sh prend 2 paramètres, un entier suivi d'un nom de fichier.

```
$ cat boule-de-gomme.sh
#!/bin/bash

if [[ ( $(wc -l < $2) -lt $1 ) || ( $1 -le 0 ) ]]
then
 cat $2
else
 cat $2  | head -$(expr $1 - 1)
 cat $2  | tail -$(expr $(wc -l < $2) - $1)
fi
$
```

En supposant que le fichier verlaine.txt contienne

```
$ cat verlaine.txt
Il pleure dans mon Coeur
Comme il pleut sur la ville;
Quelle est cette langueur
Qui pénètre mon coeur ?
$
```

Que produira la commande
`$ ./boule-de-gomme.sh 10 verlaine.txt; ./boule-de-gomme.sh 3 verlaine.txt`

⋆ En supposant que l'un de vos processus Emacs ayant le pid n°12452 soit bloqué. Par quelle commande devez vous commencer pour tuer ce processus ?

kill -15 12452 ..... ☐ kill -9 12452 ..... ☐

⋆ Justifier votre réponse précédente :

⋆ 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 ..... ☐
Cela peut être n'importe quel ordre ..... ☐ 1 puis 2 puis 3 ..... ☐

⋆ Que produit l'exécution des commandes suivantes

```
$ REPONSE_A_LA_QUESTION=42
$ sh
$ echo /$REPONSE_A_LA_QUESTION/
```

```
$ Ctrl-d
$
$ ( export TMP=/tmp ); export BAR=foo
$ bash
$ export TREIZE=XIII
$ echo /$REPONSE_A_LA_QUESTION/$TMP/$BAR/$TREIZE/
```

```
$ Ctrl-d
$ echo \*$TREIZE\*
```

⋆ 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-répertoires dont le nom commence par un caractère '.'.

⋆ 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-répertoires. La commande nettoyer.sh 2 supprimera les fichiers temporaires uniquement dans le répertoire courant, ses sous-répertoires et ses sous sous-répertoires, mais ne dépassera pas le deuxième niveau de sous-répertoire. 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 dirname

```
NAME
       dirname - strip non-directory suffix from file name

SYNOPSIS
       dirname NAME
       dirname OPTION

DESCRIPTION
       Print  NAME  with  its trailing /component removed; if NAME contains no /~s, output ~.~
       (meaning the current directory).

       --help display this help and exit

       --version
              output version information and exit

EXAMPLES
       dirname /usr/bin/sort
              Output "/usr/bin".

       dirname stdio.h
              Output ".".

AUTHOR
       Written by David MacKenzie and Jim Meyering.

REPORTING BUGS
       Report bugs to <bug-coreutils@gnu.org>.

COPYRIGHT
       Copyright © 2006 Free Software Foundation, Inc.
       This is free software.  You may redistribute copies of it under the terms  of  the  GNU
       General  Public  License <http://www.gnu.org/licenses/gpl.html>.  There is NO WARRANTY,
       to the extent permitted by law.

SEE ALSO
       The full documentation for dirname is maintained as a Texinfo manual.  If the info  and
       dirname programs are properly installed at your site, the command

              info dirname

       should give you access to the complete manual.

dirname 5.96                        June 2006
DIRNAME(1)
```

B basename

```
BASENAME(1)                      User Commands                      BASENAME(1)

NAME
       basename - strip directory and suffix from filenames

SYNOPSIS
       basename NAME [SUFFIX]
       basename OPTION

DESCRIPTION
       Print  NAME with any leading directory components removed.  If specified, also remove a
       trailing SUFFIX.
```

```
       --help display this help and exit

       --version
              output version information and exit

EXAMPLES
       basename /usr/bin/sort
              Output "sort".

       basename include/stdio.h .h
              Output "stdio".

AUTHOR
       Written by FIXME unknown.

REPORTING BUGS
       Report bugs to <bug-coreutils@gnu.org>.

COPYRIGHT
       Copyright © 2006 Free Software Foundation, Inc.
       This is free software.  You may redistribute copies of it under the terms  of  the  GNU
       General  Public  License <http://www.gnu.org/licenses/gpl.html>.  There is NO WARRANTY,
       to the extent permitted by law.

SEE ALSO
       The full documentation for basename is maintained as a Texinfo manual.  If the info and
       basename programs are properly installed at your site, the command

              info basename

       should give you access to the complete manual.

basename 5.96                      June 2006                       BASENAME(1)
```

C echo

```
NAME
       echo - display a line of text

SYNOPSIS
       echo [OPTION]... [STRING]...

DESCRIPTION
       Echo the STRING(s) to standard output.

       -n     do not output the trailing newline

       -e     enable interpretation of backslash escapes

       -E     disable interpretation of backslash escapes (default)

       --help display this help and exit

       --version
              output version information and exit

       If -e is in effect, the following sequences are recognized:

       \0NNN  the character whose ASCII code is NNN (octal)

       \\     backslash

       \a     alert (BEL)
```

```
       \b     backspace

       \c     suppress trailing newline

       \f     form feed

       \n     new line

       \r     carriage return

       \t     horizontal tab

       \v     vertical tab

       NOTE: your shell may have its own version of echo, which usually supersedes the version
       described here.  Please refer to your  shell~s  documentation  for  details  about  the
       options it supports.
[...]
```

D expr

```
NAME
       expr - evaluate expressions

SYNOPSIS
       expr EXPRESSION
       expr OPTION

DESCRIPTION
       --help display this help and exit

       --version
              output version information and exit

       Print  the  value  of  EXPRESSION  to  standard  output.   A blank line below separates
       increasing precedence groups.  EXPRESSION may be:

       ARG1 | ARG2
              ARG1 if it is neither null nor 0, otherwise ARG2

       ARG1 & ARG2
              ARG1 if neither argument is null or 0, otherwise 0

       ARG1 < ARG2
              ARG1 is less than ARG2

       ARG1 <= ARG2
              ARG1 is less than or equal to ARG2

       ARG1 = ARG2
              ARG1 is equal to ARG2

       ARG1 != ARG2
              ARG1 is unequal to ARG2

       ARG1 >= ARG2
              ARG1 is greater than or equal to ARG2

       ARG1 > ARG2
              ARG1 is greater than ARG2

       ARG1 + ARG2
              arithmetic sum of ARG1 and ARG2
```

```
       ARG1 - ARG2
              arithmetic difference of ARG1 and ARG2

       ARG1 * ARG2
              arithmetic product of ARG1 and ARG2

       ARG1 / ARG2
              arithmetic quotient of ARG1 divided by ARG2

       ARG1 % ARG2
              arithmetic remainder of ARG1 divided by ARG2

       STRING : REGEXP
              anchored pattern match of REGEXP in STRING

       match STRING REGEXP
              same as STRING : REGEXP

       substr STRING POS LENGTH
              substring of STRING, POS counted from 1

       index STRING CHARS
              index in STRING where any CHARS is found, or 0

       length STRING
              length of STRING

       + TOKEN
              interpret TOKEN as a string, even if it is a

              keyword like ~match~ or an operator like ~/~

       ( EXPRESSION )
              value of EXPRESSION

       Beware that many operators need to be escaped or quoted for  shells.   Comparisons  are
       arithmetic  if both ARGs are numbers, else lexicographical.  Pattern matches return the
       string matched between \( and \) or null; if \( and \) are not used,  they  return  the
       number of characters matched or 0.

       Exit  status  is 0 if EXPRESSION is neither null nor 0, 1 if EXPRESSION is null or 0, 2
       if EXPRESSION is syntactically invalid, and 3 if an error occurred.

AUTHOR
       Written by Mike Parker.
[...]
```