Licence d'informatique
Module de C/Unix

TD : Programmes en shell, commandes `sed` et consœurs

Exercice 1 (Commande `ip-number`)
La commande `ypcat hosts` affiche la liste de toutes les machines connues de la machine sur laquelle elle est exécutée. En plus des noms des machines est aussi donné le numéro IP (Internet Protocol) associée à chaque machine, et un eventuel commentaire. Le format du résultat retourné par cette commande est le suivant :

```
134.206.10.127  coors.lifl.fr coors
134.206.10.28   luxe.lifl.fr luxe
134.206.10.126  labatt.lifl.fr labatt
134.206.10.123  templier.lifl.fr templier pc326 # PC 3em etage
```

Écrire un shell-script `ip-number` qui retourne le numéro IP de la machine sur laquelle est exécutée ce shell-script. Si un paramètre est donné, la commande affichera le numéro IP de la machine dont le nom est passé en paramètre. La commande `hostname` affiche le nom de la machine sur laquelle elle est exécutée.

Exercice 2 (Commande `qui`)
La commande `who` affiche à l'écran la liste des connexions des utilisateurs de la machine. Pour chaque connexion sont donnés le nom de login, la console du login, la date de login, et le display éventuel si la connexion se fait à travers un serveur X ou la machine distante si la connexion est faite par réseau. Le format d'affichage est le suivant :

```
levaire  tty1     Nov  5 14:23
levaire  ttyp0    Nov  5 14:24 (:0.0)
levaire  ttyp2    Nov  5 14:34 (:0.0)
cagniot  ttyp3    Nov  5 15:23 (fana)
```

Écrire une commande `qui` qui affiche la liste des logins des utilisateurs connectés à la machine. Chaque login ne doit apparaître qu'une seule fois.

Exercice 3 (Ma commande `cut`)
L'utilisation de la commande `cut -d`*`char`* `-f`*`list`* est difficile pour identifier des champs séparés par des espaces si plusieurs espaces peuvent séparer deux champs. On propose de développer une commande `mcut` équivalent cette commande qui contourne cette difficulté.

Question 3.1 (Commande `replace`) Écrire un shell-script `replace` qui lit son entrée standard et la renvoie sur sa sortie standard en supprimant toute suite d'un caractère passé en paramètre par une seule occurence de ce caractère. Le caractère est donné par l'argument `-d` immédiatement suivi de la valeur du caractère (attention aux caractères séparateurs du shell).

Question 3.2 (Commande `mcut`) À partir de la question précédente, écrire un shell-script `mcut` qui considère toute suite de séparateurs comme un unique séparateur et a évidemment le même comportement que `cut`. On supposera que `mcut` accepte l'option `-d` comme premier argument et l'option `-f` comme second argument.

Id : td2shell.tex,v 1.5 2001/11/05 10 :03 :21 marquet Exp

Exercice 4 (Commande `maxsize`)
Écrire un shell-script `maxsize` qui renvoie l'entrée du répertoire courant, ou d'un répertoire passé en paramètre, ayant la plus grande taille (que l'on affichera). On prendra pour taille la valeur retournée par la commande `ls -l`, dont le format d'affichage est le suivant :

```
% ls -l .
total 4
drwxr-xr-x   2 duchmol  licence      1024 Nov   6 17:27 ./
drwxrwxrwt   7 duchmol  licence      3072 Nov   6 17:27 ../
-rwxr-xr-x   1 duchmol  licence      163  Nov  18 17:17 mcut
```

Les champs sont ici séparés par des suites de blancs. On veillera à supprimer la première ligne du résulat de `ls -l` sur un répertoire qui contient `total`.

Exercice 5 (Arithmérique sur les durées)
Écrire les fonctions Bourne Shell `addtime` et `difftime` qui acceptent deux paramètres de date au format Jour+Heure :Minute et qui affichent respectivement sur leur sortie standard la somme et la différence de ces deux dates suivant le même format. Le champ Jour est optionnel. Par exemple,

```
% addtime 1+12:34 13:25
2+1:59
```

La commande `ps`

La commande `ps` liste les processus (programmes en cours d'exécution) lancés par l'utilisateur. Les processus sont organisés sous forme d'un arbre. Un processus a toujours un processus père unique. Seul le processus de numéro 1 (`init`) n'a pas de père.

Différentes options permettent d'afficher les informations suivantes :
- `UID` et `USER` : l'identificateur et le nom de l'utilisateur ;
- `PID` : le numéro du processus ;
- `PPID` : le numéro du processus père (celui qui a lancé le processus) ;
- `%MEM` et `%CPU` les pourcentages d'utilisation du processeur et de la mémoire par le processus ;
- `SIZE` : la taille en kilo-octets du processus.

Parmi les nombreuses options de `ps`, les combinaisons suivantes seront utiles :
- `ps ux` affiche, entre autres, le nom de l'utilisateur, la taille mémoire utilisée par chacun des processus pour l'utilisateur courant ;
- `ps aux` affiche sous le même format que `ps ux` les informations pour l'ensemble des processus des utilisateurs connectés à la machine ;
- `ps uxl` affiche aussi pour chacun des processus de l'utilisateur le numéro de son processus père ;
- `ps auxl` affiche sous le même format que `ps ux` les informations pour l'ensemble des processus des utilisateurs connectés à la machine.

Par défaut, le nom de la commande (champ `COMMAND`) est affiché sur un nombre réduit de caractères. L'option `w` affiche ce champ sur un plus grand nombre de caractères. Des exemples sont donnés aux figures 1 à 3.

Exercice 6 (Commande `topmemoire`)
Question 6.1 (Commande `summemoire`) Écrire un shell-script `summemoire` qui calcule la somme en kilo-octets de la mémoire utilisée par tous les processus de l'utilisateur dont le nom est passé en paramètre ; par défaut l'utilisateur courant.

```
% (set dix="123456789 " ; echo $dix$dix$dix$dix$dix$dix$dix$dix)
123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
% ps ux
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
marquet    376  0.0  0.0  1468     0   1 SW   09:12   0:00 (login)
marquet    384  0.0  0.0  1628    28   1 S    09:13   0:00 (tcsh)
marquet    564  0.0  0.0  1136    28   1 S    09:16   0:00 (startx)
marquet    565  0.0  0.2  1820    88   1 S    09:16   0:00 (xinit)
marquet    569  0.0  0.1  1132    36   1 S    09:16   0:00 (startkde)
marquet    571  0.1  4.6  5616  1420   1 S    09:16   0:02 kfm
marquet    573  0.0  2.1  4104   660   1 S    09:16   0:00 kbgndwm
marquet    574  0.0  2.0  4084   628   1 S    09:16   0:00 krootwm
marquet    575  0.1  4.6  4888  1444   1 S    09:16   0:03 kpanel
marquet    576  0.0  4.7  4192  1452   1 S    09:16   0:00 knotes
marquet    577  0.1  4.9  3756  1524   1 S    09:16   0:03 kwm
marquet    578  0.0  0.1  1336    52   1 S    09:16   0:00 (tcsh)
marquet    581  0.0  0.1  1336    44   1 S    09:16   0:00 (tcsh)
marquet    582  0.0  0.1  1336    32   1 S    09:16   0:00 (tcsh)
marquet    583  0.0  0.1  1336    44   1 S    09:16   0:00 (tcsh)
marquet    584  0.0  0.1  1336    32   1 S    09:16   0:00 (tcsh)
marquet    663  0.0  4.7  4576  1452   1 S    09:16   0:01 kvt -restore kvtrc.34
marquet    665  0.0  7.6  5212  2360   1 S    09:16   0:01 kdvi -restore kdvirc.
marquet    667  1.7 29.1 10860  8984   1 S    09:16   0:38 emacs -geometry -0+0
marquet    668  0.0  5.0  4568  1548   1 S    09:16   0:01 kvt -restore kvtrc.8
marquet    669  0.0  2.3  4044   728   1 S    09:16   0:00 korn -icon korn.xpm -
marquet    671  0.0  2.3  4324   736   1 S N  09:16   0:00 /opt/kde/bin/kblanksc
marquet    674  0.0  2.4  1576   764  p2 S    09:17   0:00 -csh
marquet    675  0.0  2.1  1572   656  p8 S    09:17   0:00 -csh
marquet    708  0.0  0.2   864    88  ?  S    09:17   0:00 (emacsserver)
marquet    788  0.2  5.9  2424  1828  ?  S    09:51   0:00 /usr/local/lib/texmf/
marquet    791  0.0  1.5   828   492  p8 R    09:53   0:00 ps ux
```

FIG. 1 – Commande `ps`

```
% (set dix="123456789 " ; echo $dix$dix$dix$dix$dix$dix$dix$dix)
123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
%ps uxl
 FLAGS   UID   PID  PPID PRI  NI   SIZE   RSS WCHAN       STA TTY TIME COMMAND
100100  5408   376     1   0   0   1468     0 wait4       SW    1 0:00 (login)
100100  5408   671     1  19  19   4324   736 nanosleep   S N   1 0:00 /opt/kde/
     0  5408   384   376   0   0   1628    28 sigsuspend  S     1 0:00 (tcsh)
100000  5408   564   384   0   0   1136    28 wait4       S     1 0:00 (startx)
     0  5408   565   564   0   0   1820    88 wait4       S     1 0:00 (xinit)
100000  5408   569   565   0   0   1132    36 wait4       S     1 0:00 (startkde
100000  5408   571   569   0   0   5616  1420 do_select   S     1 0:02 kfm
100000  5408   573   569   0   0   4104   660 do_select   S     1 0:00 kbgndwm
100000  5408   574   569   0   0   4084   628 do_select   S     1 0:00 krootwm
100000  5408   575   569   0   0   4888  1444 do_select   S     1 0:03 kpanel
100000  5408   576   569   0   0   4192  1452 do_select   S     1 0:00 knotes
100000  5408   577   569   0   0   3756  1524 do_select   S     1 0:03 kwm
100000  5408   578   577   0   0   1336    52 sigsuspend  S     1 0:00 (tcsh)
100000  5408   580   577   0   0   1336    28 sigsuspend  S     1 0:00 (tcsh)
100000  5408   581   577   0   0   1336    44 sigsuspend  S     1 0:00 (tcsh)
100000  5408   582   577   0   0   1336    32 sigsuspend  S     1 0:00 (tcsh)
100000  5408   583   577   0   0   1336    44 sigsuspend  S     1 0:00 (tcsh)
100000  5408   584   577   0   0   1336    32 sigsuspend  S     1 0:00 (tcsh)
100000  5408   669   578   0   0   4044   728 do_select   S     1 0:00 korn -ico
100000  5408   663   581   3   0   4576  1452 do_select   S     1 0:01 kvt -rest
100000  5408   668   582   0   0   4568  1548 do_select   S     1 0:01 kvt -rest
100000  5408   665   583   0   0   5212  2360 do_select   S     1 0:01 kdvi -res
100000  5408   667   584   0   0  10860  8984 do_select   S     1 0:38 emacs -ge
     0  5408   675   663  18   0   1572   676 sigsuspend  S    p8 0:00 -csh
100000  5408   708   667   0   0    864    88 do_select   S    ?  0:00 (emacsser
100000  5408   788   667   0   0   2424  1828 pipe_read   S    ?  0:00 /usr/loca
     0  5408   674   668   0   0   1576   764 read_chan   S    p2 0:00 -csh
100000  5408   795   675  10   0    932   500             R    p8 0:00 ps uxl
```

FIG. 2 – Commande ps

```
% ps uxlw
 FLAGS   UID   PID  PPID PRI  NI   SIZE   RSS WCHAN       STA TTY TIME COMMAND
100100  5408   376     1   0   0   1468     0 wait4       SW   1  0:00 (login)
100100  5408   671     1  19  19   4324   736 nanosleep   S N  1  0:00 /opt/kde/bin/kblankscrn.kss -
delay 3 -install -corners ilis -nice 19
     0  5408   384   376   0   0   1628    28 sigsuspend  S    1  0:00 (tcsh)
100000  5408   564   384   0   0   1136    28 wait4       S    1  0:00 (startx)
     0  5408   565   564   0   0   1820    88 wait4       S    1  0:00 (xinit)
100000  5408   569   565   0   0   1132    36 wait4       S    1  0:00 (startkde)
100000  5408   571   569   0   0   5616  1420 do_select   S    1  0:02 kfm
100000  5408   573   569   0   0   4104   660 do_select   S    1  0:00 kbgndwm
100000  5408   574   569   0   0   4084   628 do_select   S    1  0:00 krootwm
100000  5408   575   569   0   0   4888  1444 do_select   S    1  0:03 kpanel
100000  5408   576   569   0   0   4192  1452 do_select   S    1  0:00 knotes
100000  5408   577   569   1   0   3756  1524 do_select   S    1  0:04 kwm
100000  5408   578   577   0   0   1336    52 sigsuspend  S    1  0:00 (tcsh)
100000  5408   581   577   0   0   1336    44 sigsuspend  S    1  0:00 (tcsh)
100000  5408   582   577   0   0   1336    32 sigsuspend  S    1  0:00 (tcsh)
100000  5408   583   577   0   0   1336    44 sigsuspend  S    1  0:00 (tcsh)
100000  5408   584   577   0   0   1336    32 sigsuspend  S    1  0:00 (tcsh)
100000  5408   669   578   0   0   4044   728 do_select   S    1  0:00 korn -
icon korn.xpm -icon korn.xpm
100000  5408   663   581   0   0   4576  1456 do_select   S    1  0:01 kvt -restore kv-
trc.34 -icon kvt.xpm
100000  5408   668   582   0   0   4568  1548 do_select   S    1  0:01 kvt -
restore kvtrc.8
100000  5408   665   583   0   0   5212  2360 do_select   S    1  0:01 kdvi -restore kd-
virc.1 -caption Visualiseur DVI -icon kdvi.xpm -miniicon kdvi.xpm
     0  5408   667   584   1   0  10860  9012 do_select   S    1  0:39 emacs -geometry -
0+0 -f vm
     0  5408   675   663  15   0   1572   680 sigsuspend  S   p8  0:00 -csh
100000  5408   708   667   0   0    864    88 do_select   S   ?   0:00 (emacsserver)
100000  5408   788   667   0   0   2424  1828 pipe_read   S   ?   0:00 /usr/local/lib/texmf/bin/i86_linux/
a -m -d francais -p /users/phm/.ispell_francai
     0  5408   674   668   0   0   1576   764 read_chan   S   p2  0:00 -csh
100000  5408   799   675  17   0    932   500             R   p8  0:00 ps uxlw
```

FIG. 3 – Commande ps

Question 6.2 (Commande `utilisateurs`) Écrire un shell-script `utilisateurs` qui liste l'ensemble des utilisateurs ayant au moins un processus en cours d'exécution.

Question 6.3 (Commande `topmemoire`) La commande `topmemoire` affiche la liste des utilisateurs consommant le plus de mémoire.

Exercice 7 (Tuer les processus)
La commande

```
kill [-signal] pid...
```

envoie le signal précisé (`9` ou `KILL` par défaut) aux processus désigné par leur numéro.

L'effet de cette commande (avec le signal `9`) est de tuer le processus désigné.

On désire développer une commande `killname` qui envoie un signal à un processus que l'on ne désigne non pas par son numéro, mais par son nom. Le nom du processus est donné dans le champs `COMMAND` du résulat de la commande `ps`.

```
killname [-e] [-signal] name...
```

envoie le signal précisé à tous les processus désigné par le nom `name`. Notons qu'un processus `killname` ne se tue jamais lui même, mais il peut tuer d'autres processus `killname`.

L'option `-e` précise de considérer exactement le nom des processus. Par défaut, `killname` tue tous les processus dont les premiers caractères du nom correspondent avec l'argument.

Exercice 8 (Arborescence de processus)
Les processus sont organisés sous forme d'une arborescence dont la racine est le processus `init` de numéro 1.

La commande `pstree [pid]` affiche l'arborescence des processus issus du processus de numéro `pid` (par défaut 1). Par exemple, suite à la commande `ps uxlw` de la page 5 :

```
% pstree 376
PID  PPID   COMMAND
376     1   (login)
384   376   \_ (tcsh)
564   384       \_ (startx)
565   564           \_ (xinit)
569   565               \_ (startkde)
571   569                   \_ kfm
573   569                   \_ kbgndwm
574   569                   \_ krootwm
575   569                   \_ kpanel
576   569                   \_ knotes
577   569                   \_ kwm
578   577                       \_ (tcsh)
669   578                       |   \_ korn -icon korn.xpm -icon korn.xpm
581   577                       \_ (tcsh)
663   581                       |   \_ kvt -restore kvtrc.34 -icon kvt.xpm
675   663                       |       \_ -csh
798   675                       |           \_ pstree
582   577                       \_ (tcsh)
668   582                       |   \_ kvt -restore kvtrc.8
674   668                       |       \_ -csh
583   577                       \_ (tcsh)
665   583                       |   \_ kdvi -restore kdvirc.1 -caption Visualiseur DVI -
icon kdvi.xp
584   577                       \_ (tcsh)
667   584                       |   \_ emacs -geometry -0+0 -f vm
708   667                       |       \_ (emacsserver)
788   667                       |       \_ /usr/local/lib/texmf/bin/i86_linux/ispell -
a -m -d franca
```

Exercice 9 (Commande `mmv`)
Écrire une commande `mmv` qui renomme une liste de fichiers en une autre liste de fichiers. Par exemple :

```
mmv f1 f2 f3 g1 g2 g3
```

renomme `f1` en `g1`, `f2` en `g2`, et `f3` en `g3`. On ne testera pas la validité du renommage (existence de `f1`, `f2`... et non-existence de `g1`, `g2`...).

Exercice 10 (Commande `logstat`)
La commande `last` affiche toutes les connexions réalisées sur la machine par l'ensemble des utilisateurs. Le format d'affichage est le suivant :

```
levaire   ftp          labatt.lifl.fr   Tue Nov 18 18:36 - 18:37  (00:00)
webadmin  pts/5        labatt.lifl.fr   Tue Nov 18 14:45   still logged in
levaire   pts/5        labatt.lifl.fr   Thu Nov 13 13:45 - 18:40 (1+04:55)
webadmin  pts/5        pcdessgi24       Fri Nov  7 11:49 - 11:50  (00:00)
webadmin  console                       Tue Nov  4 17:04   still logged in
reboot    system boot                   Tue Nov  4 14:55
levaire   pts/5        stu12            Tue Jun  3 12:43 - down   (01:55)
```

Écrire un shell-script `logstat` qui affiche pour chaque utilisateur le nombre total de ses connexions et sa durée totale de connexion en jours, heures et minutes. La durée d'une connexion est donneée dans le dernier champ. Si un paramètre est spécifié, on n'affichera que les informations concernant l'utilisateur dont le nom a été passé en paramètre. Si l'utilisateur est toujours connecté, calculer la durée de cette connexion en utilisant la date courante. Pour simplifier, on supposera que ces connexions ont commencé dans le même mois que le mois courant.

Exercice 11 (Enquête)
Les stations Unix ne doivent pas être arrêtées violemment. En particulier une coupure de courant brutale peut avoir des conséquences néfastes : pour les utilisateurs connectés à ce moment là et pour la machine elle-même.

Cet exercice vise à développer des commandes qui vont permettre d'analyser ce genre d'événements malheureux.

La commande Unix `last` permet de prendre connaissance des informations de connexion des utilisateurs sur une machine. Elle écrit sur la sortie standard une ligne par connexion et par événement du type `reboot`, `shutdown`... Cette ligne comporte (entre autre) le nom, la date et l'heure de connexion.

```
% set dix="123456789 " ; echo $dix$dix$dix$dix$dix$dix$dix$dix
123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
% last
bourgois  console                       Mon Dec  8 08:26   still logged in
simon     console                       Fri Dec  5 20:02 - 22:23  (02:20)
lemaitre  pts/0        stu3             Fri Dec  5 19:59 - still logged in
marfjan   console                       Fri Dec  5 19:52 - 19:53  (00:01)
marfjan   pts/2        stu12            Fri Dec  5 18:12 - 19:52  (01:39)
quanquoi  pts/2        stu12            Fri Dec  5 18:10 - 18:11  (00:00)
marfjan   console                       Fri Dec  5 17:47 - 17:47  (00:00)
yroos     pts/6        choulette.lifl.f Fri Dec  5 17:03 - 17:04  (00:00)
gay       pts/6        stu12            Fri Dec  5 16:39 - 16:40  (00:00)
fasquelf  console                       Fri Dec  5 16:16 - 17:46  (01:30)
reboot    system boot                   Fri Dec  5 16:10
gay       pts/0        stu20            Fri Dec  5 16:06 - 16:08  (00:02)
fasquelf  console                       Fri Dec  5 16:06 - 16:08  (00:01)
gay       pts/4        stu20            Fri Dec  5 16:01 - 16:05  (00:04)
[...]
```

```
robert    console                       Thu Nov  6 19:43 - 22:27  (02:44)
salomon   console                       Thu Nov  6 19:42 - 19:42  (00:00)
marfjan   console                       Thu Nov  6 18:54 - 19:27  (00:33)
reboot    system boot                   Thu Nov  6 18:54
marfjan   pts/0         stu19           Thu Nov  6 18:48 - 18:50  (00:01)
lecat     pts/4         pcdessgi12      Thu Nov  6 16:42 - down
marfjan   console                       Thu Nov  6 15:46 - down
delenclo  console                       Thu Nov  6 15:25 - 15:43  (00:17)
delenclo  console                       Thu Nov  6 15:09 - 15:25  (00:16)
rebena    ftp           stu11           Thu Nov  6 14:50 - 15:03  (00:13)
```

Question 11.1 (Commande `temoin`) La commande `temoin` affiche sur la sortie standard le(s) nom(s) de celui(ux) qui peu(ven)t-être témoin(s) du ¡¡ crime ¿¿. Les témoins potentiels sont au nombre de deux :

– celui qui était connecté sur la console juste avant le reboot. On se limitera à la même journée que le reboot ;
– celui qui s'est reconnecté sur la console juste après le reboot. On se limitera aussi au jour du reboot.

On ne s'occupera que du dernier reboot en date. Attention, la première connexion qui suit un reboot n'est pas forcément effectuée sur la console ; dans ce cas, l'usager qui se connecte n'est pas considéré comme témoin.

Question 11.2 (Commande `suspect`) Si le nom du premier témoin est égal au nom du second témoin alors le témoin devient suspect. La commande `suspect` affiche dans ce cas son nom sur la sortie standard. Elle ne fait rien dans le cas contraire.

Question 11.3 (Commande `victimes`) La commande victime affiche sur la sortie standard le nom des victimes. On appelle victime un usager qui a vu une de ses connexions coupée au moment de ce reboot.