Examen Administration des Systèmes et des Réseaux
Durée 2 heures
Documents autorisés
Portables, ordinateurs et téléphones, éteints

Vendredi 21 mars 2008

Attention

Les questions sont indépendantes. Vous devez expliciter et argumenter vos réponses. Par ailleurs il n'y a pas forcément de « bonne » réponse, l'argumentation que vous adosserez à vos commentaires n'en sera que plus importante.

1 **La partition correspondant à *mnt* est du type *FFS*, la taille du bloc étant de 8K et celle du fragment de 1k. La commande *dd* est utilisée dans la dernière partie de l'exemple pour rajouter 7168 caractères *null* (ASCII 0) au fichier *fichier*. En examinant les différentes modifications du fichier *fichier* dans la suite de commandes, que pouvez vous déduire sur la politique d'allocation du système de fichiers ? Expliquez celle-ci et montrez son intérêt s'il y en a un.**

```
ramassis# df --block-size=8192 . ; df -k . ; df -i .
Filesystem            8k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0        257104       260    249131   1% /mnt
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0       2056831      2074   1993053   1% /mnt
Filesystem             Inodes   IUsed   IFree IUse% Mounted on
/dev/dsk/c0t2d0s0      332992       5  332987    1% /mnt
ramassis# touch fichier ; ls -l fichier
-rw-r--r--    1 root     root           0 mars 18 17:28 fichier
ramassis# df --block-size=8192 . ; df -k . ; df -i .
Filesystem            8k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0        257104       260    249131   1% /mnt
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0       2056831      2074   1993053   1% /mnt
Filesystem             Inodes   IUsed   IFree IUse% Mounted on
/dev/dsk/c0t2d0s0      332992       6  332986    1% /mnt
ramassis# date > fichier ; ls -l fichier
-rw-r--r--    1 root     root          31 mars 18 17:29 fichier
ramassis# df --block-size=8192 . ; df -k . ; df -i .
Filesystem            8k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0        257104       260    249131   1% /mnt
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0       2056831      2075   1993052   1% /mnt
Filesystem             Inodes   IUsed   IFree IUse% Mounted on
/dev/dsk/c0t2d0s0      332992       6  332986    1% /mnt
ramassis# dd if=/dev/zero bs=1024 count=7 >> fichier ; ls -l fichier
7+0 records in
7+0 records out
-rw-r--r--    1 root     root        7199 mars 18 17:29 fichier
ramassis# df --block-size=8192 . ; df -k . ; df -i .
Filesystem            8k-blocks      Used Available Use% Mounted on
```

Nbr inode Libre

Espace utilisé du Disk

Bloc 8K

Bloc 1K

Ajout de 7 x 1024 caractères NUL

Espace libre
Dispo dans le Système de fichier
disk free

```
/dev/dsk/c0t2d0s0       257104       261    249130   1% /mnt
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0       2056831      2082   1993045   1% /mnt
Filesystem             Inodes   IUsed   IFree IUse% Mounted on
/dev/dsk/c0t2d0s0      332992       6  332986    1% /mnt
ramassis#
```

2 Les liens symboliques permettent de pointer sur un fichier, au sens large (catalogue, device, etc.), en s'affranchissant des limites des liens standards (*hard links*), qu'on peut très grossièrement considérer comme de simples alias. L'exemple ci-dessous est réalisé sur un système *HP-UX*, que pourriez-vous dire sur la constitution des liens symboliques et sur la méthode d'allocation de ceux-ci sur ce système ?

```
brumaire HP-UX# df -k . ; df -i .
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/vg01/lvtest        102400      1133     94945   2% /essai
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/vg01/lvtest       25320       5   25315    1% /essai
brumaire HP-UX# ln -s /./././/././././././././././././././etc/issue lien
brumaire HP-UX# df -k . ; df -i .
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/vg01/lvtest        102400      1133     94945   2% /essai
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/vg01/lvtest       25320       6   25314    1% /essai
brumaire HP-UX# ls -l lien
lrwxr-xr-x 1 root sys 42 Mar 17 11:47 lien -> /./././/././././././././././././././etc/issue
brumaire HP-UX# echo -n "/./././/././././././././././././././etc/issue" > fichier
brumaire HP-UX# df -k . ; df -i .
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/vg01/lvtest        102400      1134     94944   2% /essai
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/vg01/lvtest       25320       7   25313    1% /essai
brumaire HP-UX# ls -l fichier
-rw-r--r-- 1 root sys 42 Mar 17 11:48 fichier
brumaire HP-UX#
```

L'option *-n* de la commande *echo* permet de ne pas inclure le saut de ligne en fin de chaîne, plus exactement le caractère *newline*.
L'exemple suivant, similaire au précédent, est lui réalisé sous *Solaris*. Sachant qu'*HP-UX* utilise un mécanisme dit de *Fast Symlinks*, liens symboliques rapides, que pourriez-vous en déduire en comparant les deux exemples ?

```
ramassis SunOS# df -k . ; df -i .
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0       2056831      2074   1993053   1% /mnt
Filesystem             Inodes   IUsed   IFree IUse% Mounted on
/dev/dsk/c0t2d0s0      332992       5  332987    1% /mnt
ramassis SunOS# ln -s /./././/././././././././././././././etc/issue lien
ramassis SunOS# ls -l lien
lrwxrwxrwx 1 root root 42 mars 17 11:52 lien -> /./././/././././././././././././././etc/issue
ramassis SunOS# df -k . ; df -i .
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/dsk/c0t2d0s0       2056831      2075   1993052   1% /mnt
Filesystem             Inodes   IUsed   IFree IUse% Mounted on
/dev/dsk/c0t2d0s0      332992       6  332986    1% /mnt
ramassis SunOS#
```

3 La commande *touch* est ici utilisée pour créer un fichier vide. Connaissez vous un autre moyen ? Quel résultat donne selon vous la suite de commandes suivantes :

```
touch date ; find /usr -type f -newer date -print |& tee /var/tmp/fichiers
```

La suite précédente peut-elle jouer un rôle dans la détection d'intrusions ? Si non pourquoi et si oui comment ? Quelles en seraient les limites et précautions à prendre ? Le remplacement par *...; find /usr -type d -newer ...* vous paraît-il plus pertinent ?

4 Le programme *p.c* donné à la suite se contente simplement de créer un fichier de nom *sans-objet* en y ajoutant quelques caractères dans une boucle à chaque passage dans celle-ci. Quels en sont les effets ?

```
/ p.c /
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
int f;
main()
{
   if((f=open("sans objet", O_WRONLY|O_CREAT,0600))== 1)
        {perror("y'a un probleme"); exit(1);};
   while(1){
        if(write(f,"Lapin",5)== 1){perror("y'a encore un probleme"); exit(1);}
   }
}
```

Le programme est compilé pour en obtenir le binaire *p* (l'option *-w* n'est là que pour supprimer quelques messages d'avertissement du compilateur sans intérêt ici). Que pensez-vous de la suite de commandes ci-après :

```
cc -w p.c -o p ; ./p & ; sleep 1 ; rm sans-objet
```

Décrivez l'impact sur le fonctionnement du système s'il y en a un. Parmi les utilitaires que vous connaissez y en a t-il un ou plusieurs qui seraient utiles dans le cas présent ? Si oui pourquoi et comment. Que ce passera t-il si on supprime le fichier binaire *p*, à l'aide de *rm p*, c'est-à-dire si l'ensemble de commandes devient :

```
cc -w p.c -o p ; ./p & ; sleep 1 ; rm sans-objet ; rm p
```

Pensez vous que cela puisse poser des problèmes ? Si oui lesquels, à qui et comment y remédier ? On supposera que ni *sans-objet* ni *p* n'ont de lien physique (*hard link*) associé.

5 En examinant la table *ARP* donnée ci-après que pouvez-vous en déduire ? Donnez tous les renseignements que vous pourrez en extraire. Quelles informations vous manquerait-il et le cas échéant quelles sont les commandes qui vous permettraient de les obtenir ?

```
moumoune SunOS# arp -an
Device   IP Address               Mask      Flags      Phys Addr
------ -------------------- --------------- -------- ---------------
skge0  194.254.199.16       255.255.255.255 o        00:c0:b7:c9:cc:6a
skge0  194.254.199.85       255.255.255.255          00:03:ba:0f:d3:79
skge0  192.168.70.47        255.255.255.255 SPLA     00:0e:a6:8c:f4:71
skge0  192.168.79.47        255.255.255.255 SPLA     00:0e:a6:8c:f4:71
skge0  194.254.199.96       255.255.255.255 o        00:03:ba:a2:45:39
skge0  172.16.70.82         255.255.255.255 SPLA     00:0e:a6:8c:f4:71
skge0  194.254.199.47       255.255.255.255 SPLA     00:0e:a6:8c:f4:71
skge0  194.254.199.36       255.255.255.255 o        00:0a:95:87:2e:42
moumoune SunOS#
```

Le drapeau *S* indique une adresse statique, *o* une adresse dynamique obsolescente.

6 La suite de compilation *GCC* est largement répandue et utilisée, certains environnements dits « propriétaires » la fournissent même avec la distribution de leur système (SUN et *Solaris*, HP et *HP-UX*, etc.).
Quelles sont selon vous les raisons de ce succès et de cet engouement ? Ceux-ci vous paraissent-ils justifiés ?
Il est cependant aussi de plus en fréquent que les constructeurs mettent à disposition leur propre version de compilateur C et parfois gratuitement.
Selon vous y a-t-il un intérêt à utiliser l'une des distributions plutôt qu'une autre ? Quelles sont les raisons de choisir l'une ou l'autre.

7 Certains systèmes UNIX intègrent ce que l'on appelle le *desperation swapping*, un processus inactif pendant un temps donné est placé automatiquement en zone de *swap*.
Quel est le rôle de la zone de *swap* ? Connaissez vous des systèmes utilisant le *desperation swapping* ?
Quels sont, selon vous, les inconvénients et avantages de cette pratique ?
D'une façon plus générale d'autres systèmes n'utilisent pas le *swapping*, qu'en pensez vous ?
À quels risques s'expose t-on à ne pas disposer de zone de *swap*. Quels avantage peut-on tirer à ne pas en avoir ?

8 Au vu des trois exemples présentés à la suite, que pouvez-vous dire de la machine nommée *miasmes* ? Quelle sont la ou les qualités qui sont mises en relief ici ? Comment qualifier les relations de *moule* et *gaufre* avec celle-ci (*miasmes*) ?

```
miasmes# uname -a
HP-UX miasmes B.10.20 A 9000/720 miasmes two-user license
miasmes# ps -eaf | egrep 'portmap|nfsd' | grep -v grep
    root   391     1  0  Feb  1  ?         0:00 /usr/sbin/portmap
    root   634   633  1  Feb  1  ?         1:58 /usr/sbin/nfsd 4
    root   633     1  3  Feb  1  ?         1:56 /usr/sbin/nfsd 4
    root   638   633  0  Feb  1  ?         1:57 /usr/sbin/nfsd 4
    root   640   633  0  Feb  1  ?         2:04 /usr/sbin/nfsd 4
miasmes#
```

Quelle est la fonction du processus *portmap* ? Est-il nécessaire ici ?
Que pensez vous des processus *nfsd* et de leur nombre ? Comment analysez-vous le temps d'exécution de ceux-ci.
Cela pose-t-il un problème ? Comment pourrait-on y remédier ?

```
moule # cd /mnt/c
moule # df -k -F nfs
Filesystem            kbytes    used   avail capacity  Mounted on
miasmes:/essai         75733   20545   47614    31%    /mnt
moule # rm -f fichier-moule ; touch fichier-moule
moule # ls -l fichier-moule
-rw-r--r--   1 nobody   root           0 Mar 18 12:21 fichier-moule
moule #
```

Comparez les attributs des fichiers *fichier-moule* et *fichier-gaufre*, que pouvez-vous en déduire ?
La configuration de la machine *gaufre* vous paraît-elle porteuse de risques ? Si oui pourquoi et comment ?

```
gaufre# cd /mnt/c
gaufre# df -k -F nfs
Filesystem            kbytes    used   avail capacity  Mounted on
miasmes:/essai         75733   10297   57862    16%    /mnt
gaufre# rm -f fichier-gaufre ; touch fichier-gaufre
gaufre# ls -l fichier-gaufre
-rw-r--r--   1 root     root           0 Mar 18 09:21 fichier-gaufre
gaufre#
```

9 L'exemple qui suit est un extrait du rapport d'activité (*accounting*) cumulé pour le mois de février 2008. Les informations concernant les entrées/sorties sur disque on été supprimées parce que sans objet dans le cas présent. Rappelons que *TOTAL CPU-MIN* est le temps d'utilisation du processeur cumulé ; *TOTAL REAL-MIN* est la durée de vie cumulée des processus, de leur lancement jusqu'à la fin de leur exécution ; *MEAN SIZE-K* est la valeur moyenne de l'utilisation mémoire d'un processus en Ko ; *MEAN CPU-MIN* l'utilisation du processeur par un processus, obtenu par division de *TOTAL CPU-MIN* par le nombre de processus ; *HOG FACTOR* le taux d'occupation du processeur par un processus, calculé en divisant *TOTAL CPU-MIN* par *TOTAL REAL-MIN*. Le champ *KCOREMIN*, donne un « débit d'utilisation », par minute censé déterminer la gourmandise des processus en ressources.

```
Mar  1 05:15 2008  TOTAL COMMAND SUMMARY FOR FISCAL 03 Page 1
```

TOTAL COMMAND SUMMARY

COMMAND NAME	NUMBER CMDS	TOTAL KCOREMIN	TOTAL CPU-MIN	TOTAL REAL-MIN	MEAN SIZE-K	MEAN CPU-MIN	HOG FACTOR
TOTALS	3492763	27976192.00	11502.06	3698431.75	2432.28	0.00	0.00
java	848	10739412.00	103.88	77140.94	103385.03	0.12	0.00
xemacs-2	147	5906736.50	7102.57	11091.87	831.63	48.32	0.64
javac	947	1263297.12	38.09	32.62	33166.25	0.04	1.17
cc1	97133	1256085.50	154.32	155.59	8139.29	0.00	0.99
rsync	182	1048072.12	57.00	1120.18	18387.88	0.31	0.05
bash	1613089	619562.75	138.23	62694.93	4482.08	0.00	0.00
sshd	70984	288443.66	106.65	191668.70	2704.54	0.00	0.00

En analysant le total cumulé, *TOTALS*, que pouvez-vous dire sur l'utilisation du système (processeur et mémoire) ? Que pensez-vous de l'utilisation des ressources par l'éditeur *xemacs*, celle-ci vous paraît-elle normale ? *cc1* est une des principales passes du compilateur *C*, comparez avec *javac*. Que pouvez-vous dire sur le processus *java* ? à quoi sert-il ? Comment évaluez-vous sa consommation de ressources ? Commentez les statistiques des autres processus.
Les temps sont exprimés en minutes, pour mémoire : 1 jour compte 1 440 minutes, le mois de février 2008 à compté 41 760 minutes.