UNIX

Ce document n'est pas garanti sans erreur !

Exemples de questions posées lors de l'examen

Lenart Nathan
01/01/2010

Lenart Nathan Janvier 2010

2ème Informatique de gestion
Examen de C (Unix) en janvier 2010
(Mercenier)

Les fichiers: 

1. Un fichier ordinaire sous UNIX est
   1 - un fichier source
   2 - un fichier exécutable
   3 - un fichier de données
   4 - aucun des trois
   5 - les trois

2. Un fichier spécial, sous UNIX, caractérise
   1 - un exécutable
   2 - un fichier associé à un périphérique
   3 - un fichier système

3. Un fichier FIFO (un tube) sert
   1 - a stocké des données
   2 - mémoriser des informations utiles au noyau
   3 - à la communication entre processus

4. Un fichier peut être supprimé sur disque par
   1 - uniquement le SU
   2 - uniquement le propriétaire
   3 - le propriétaire et le SU
   4 - par toutes personnes ayant les droits d'accès en écriture sur le fichier

5. Etant donné un fichier ouvert en O_RDWR, il est possible de se positionner au delà de la fin de fichier grâce à l'instruction lseek()
   1 - jamais (il te met a la fin mais pas plus loin)
   2 - sans problème

Soit le fichier Examen.txt. Son contenu est abcdefghijklmnopqrstuvwxyz et il sera utilisé pour les exemples de la suite.

6. Soit le programme suivant

```
…
hd = open("Examen.txt",O_RDWR);
hd1 = open("Examen.txt"
O_RDWR);
write(hd ,"123",3);
write(hdl,"456",3);
…
```

   Le fichier Examen.txt est
   1 - 123456ghijklmnopqrstuvwxyz
   2 - 456defghijklmnopqrstuvwxyz
   3 - 123defghijklmnopqrstuvwxyz

7. Soit le programme suivant

```
…
hd = open("Examen.txt",O_RDWR);
lseek(hd, 100,0);
rc = write(hd ,"123",3);
```

   rc = ?
   1 - 3
   2 - -l (cas d'erreur)
   3 - 0

8. Soit le programme suivant

```
…
hd = open("Examen.txt",O_RDWR);
lseek(hd, 100,0);
rc = read(hd ,szBuffer,3);
```

   rc =?
   1 - 3
   2 - -1 (cas d'erreur)
   3 - 1'arrêt du programme
   4 – 0 (j'ai testé ! il lit 0 byte)

9. Soit le programme suivant

```
…
hd=open("Examen.txt",O_RDWR|O_
APPEND);
lseek(hd,3 ,0)
write(hd ,"XYZ",3);
```

   Le fichier Examen.txt est
   1 - abcXYzghijklmnopqrstuvwxyz
   2 - abcdefghijklmnopqrstuvwxyz (car il y a une erreur)
   3 - abcdefghijklmnopqrstuvwxyzXYZ

10. Soit le programme suivant

```
…
hd = open("Examen.txt",O_RDWR);
idFils = fork();
if (idFils)
   {     write(hd ,"XYZ",3);
         exit(1);
   }
write(hd ,"ABC",3);
exit(0);
```

Le fichier Examen.txt est

1 - ABCXYZghijklmnopqrstuvwxyz

2- ahcdefghijklmnopqrstuvwxyzABCXYZ

3 - Xyzdefghijklmnopqrstuvwxyz

11. Soit le programme suivant

```
hd = open("Examen.txt",O_RDWR);
hdl = dup(hd);
lseek(hd,3 ,0)
read(hd ,szBuffer,3);
read(hdl ,szBuffer1 ,3);
exit(0);
```

1 - sZBuffer = "def" sZBuffer1 = "ghi"

2 - sZBuffer = "def' szBuffer1 = "def'

3 - sZBuffer = "def" sZBuffer1 = "abc"

12. Soit le programme suivant

```
hd = open("Examen.txt",O_RDWR);
lseek(hd,20,0);
rc = read(hd ,sZBuffer, 10);
```

rc =?

1 - -1 (cas d'erreur)

2 - 10

3 - 6 **26-20**

13. Lorsqu'un processus a bloqué un fichier avec un verrou partagé, et que ce processus se termine suite à la réception d'un signal SIGQUIT. Le fichier reste avec son verrou partagé

1 – Vrai

2 – Faux (locks are removed when process terminates – lockf)

3 - Le verrou est levé uniquement si le programme l'a prévu

14. La pose d'un verrou exclusif sur un enregistrement empêche physiquement un autre processus de manipuler cet enregistrement.

1 - Toujours vrai

2 - Vrai, uniquement si l'autre processus teste l'existence du verrou

3 - Faux, l'autre processus fait ce qu'il veut.

15. Un verrou exclusif posé sur un enregistrement empêche le processus propriétaire de manipuler cet enregistrement.

1 - Faux

2 - Vrai, c'est en fait une sécurité

3 - Vrai, dans certains cas

16. Lorsqu'un processus veut modifier un verrou partagé en un verrou exclusif, il peut le faire directement.

1 - Vrai, il fait ce qu'il veut de ce qui lui appartient

2 - Faux, car dans ce cas il crée une situation de dead lock

3 - Vrai, mais il doit être sûr qu'un autre processus ne manipule pas cet enregistrement

17. On peut placer sur un enregistrement un verrou partagé et un verrou exclusif

1 - Vrai, par définition d'un verrou partagé

2 - Faux, en aucun cas

18. Sur un même enregistrement, un processus peut placer un verrou partagé et un verrou exclusif

1 - Vrai

2 - Faux, car il ne peut exister 2 verrous différents sur le même enregistrement

3 - Vrai, mais seul le dernier verrou posé existe.

19. Lors de la création d'un fichier avec les droits d'accès 0777, on obtient, sur disque, les droits suivants:

1- 0777

2 - 0755

3 - 0700

20. La fonction dup() alloue un nouveau descripteur dans la table des fichiers ouverts, ce dernier:
    1 - aura la même entrée dans la table des fichiers ouverts
    2 - une entrée différente dans la table des fichiers ouverts, mais accédera au même fichier
    3 - n'aura aucun effet, car le fichier est déjà ouvert une fois

21. Dans le cadre de l'institut, un étudiant du même groupe que vous peut,
    1 - lire et écrire dans vous fichiers dont vous avez donné les droits 0660
    2 - ne peut pas lire vos fichiers même si vous avez donné les droits 0660, car les droits d'accès de votre répertoire lui interdisent

22. Les terminaux sont des périphériques dont on peut modifier
    1 - tous les paramètres indépendamment les uns des autres
    2 - certains paramètres à conditions qu'ils soient compatibles avec les autres
    3 - aucun paramètre, sauf si l'on est SU

23. Lors du login, votre terminal se trouve
    1 - en mode canonique
    2 - en mode non canonique

24. Lors du login, un terminal vous est attribué
    1 - de façon aléatoire mais vous l'aurez toujours, car vous êtes connu du système
    2 - de façon fixe, car il est réservé dans le fichier des mots de passe et vous obtiendrez toujours le même
    3 - de façon aléatoire, en fonction des ressources du système

Les Pipes:

25. Un processus reçoit un signal SIGPIPE
    1 - Lorsqu'il tente d'écrire dans un pipe plein
    2 - Lorsqu'il tente d'écrire dans un pipe non ouvert en lecture
    3 - Jamais

26. Un processus qui tente d'écrire dans un pipe plein
    1 - est interrompu, car il y a erreur d'écriture, et il reçoit un SIGPIPE
    2 - est bloqué sur l'instruction write
    3 - continue son exécution, mais n'a pas écrit dans le tube.

27. Un processus veut faire une écriture dans un pipe vide et fermé en lecture:
    1 - Le processus ouvre le pipe, écrit et retourne le nombre de bytes écrit
    2 - Il attend qu'un autre processus ouvre le pipe en lecture
    3 - Il reçoit sigpipe

28. Deux processus indépendants peuvent communiquer par pipe.
    1 - Faux, en aucun cas
    2 - Vrai, il suffit d'ouvrir le pipe par la fonction fcntlO
    3 - Vrai, le pipe étant propriété du système, on peut toujours l'ouvrir

29. Un processus peut ouvrir un pipe après l'avoir fermé.
    1 - Vrai, grâce à la fonction fcntl()
    2 - Faux, en aucun cas
    3 - Vrai, grâce à la fonction open()

30. Un processus exécute l'instruction

```
…
read(hdPipe,szBuffer, 10);
```

alors qu'il n'y a que 5 caractères dans le pipe.

1 - le processus est en attente, car la lecture est bloquante.
2 - le processus lit les 5 caractères sans problème est continue son exécution
3 - le processus s'interrompt, car il y a une erreur.

31. Un processus sait qu'il n'y a rien à lire dans un tube et par conséquent ne reste pas sur un read() car

1 - il n'y a plus assez de caractères dans le tube
2 - le tube est fermé en écriture par tous les processus qui le manipulent
3 - il ne le sait pas, il faut lui envoyer un signal

32. Un étudiant de l'institut pourra communiquer par tube nommé avec son coéquipier

1 - toujours, Si le tube nommé à les droits correspondants
2 - oui, à condition que le tube possède les droits d'accès et que le tube soit crée dans un répertoire commun
3 - jamais, car c'est voulu ainsi dans l'institut

33. La pose d'un verrou exclusif sur un enregistrement empêche un autre processus de manipuler cet enregistrement :

1 - Toujours vrai.
2 - Vrai, uniquement si l'autre processus teste l'existence du verrou.
3 - Faux, l'autre processus fait ce qu'il veut.

Les I.P.C.

34. On peut ouvrir un nombre quelconque de files de messages
Ce nombre est

1 - illimité, il suffit que les clés soient différentes
2 - limité par le système
3 - limité par le système et est le même pour tous les utilisateurs

35. Les I.P.C. sont supprimés automatiquement à la fin du processus.

1 - Vrai
2 - Faux
3 - Vrai, mais uniquement lorsque tous les processus qui les manipulent sont terminés

36. Le propriétaire peut supprimer une file de messages à tout moment.

1 - Vrai
2 - Faux
3 - Vrai, mais uniquement lorsque tous les processus qui les manipulent sont terminés

37. Le propriétaire peut supprimer une mémoire partagée à tout moment.

1 - Vrai
2 - Faux
3 - Vrai, mais uniquement lorsque tous les processus qui lui sont attachés sont terminés

38. On peut manipuler un ensemble de sémaphores en une seule opération.

1 - Faux, il faut boucler sur tous les éléments de l'ensemble
2 - Vrai, mais si le programme est interrompu par la réception d'un signal, l'ensemble peut être incohérent
3 - Vrai, et toutes les valeurs de l'ensemble seront corrects, car l'opération est atomique

39. Si l'on manipule un ensemble de sémaphores avec la fonction semop(), et que sur un des sémaphores de l'ensemble, on ne peut effectuer l'opération,

1 - les autres sémaphores seront modifiés
2 - l'opération est bloquante jusqu'à la possibilité de manipuler l'ensemble complet

40. Un sémaphore peut avoir

- **une valeur entière positive (ou = 0)**
- une valeur entière quelconque
- une valeur réelle

41. Si un processus endormit sur un appel système reçoit un signal :
    1 - Il ne fait rien, le signal est suspendu pendant l'appel système.
    2 - Il interprète le signal, et renvoie une erreur.
    3 - Il interprète le signal, renvoie une erreur et positionne errno.

42. Lorsqu'un processus reçoit un signal :
    1 - Il interprète le signal.
    2 - Il interrompt la fonction, interprète le signal et positionne errno à EINTR.
    3 - Il positionne errno à EINTR et continue son exécution.

43. Un proc exécute le handler assigné à un signal, le comportement du signal n'est pas modifié, il reçoit encore le signal :
    1 - Ce 2ème signal est suspendu.
    2 - Ce 2ème signal est perdu.
    3 - Ce 2ème signal directement interprété.

44. Lorsqu'un processus reçoit un signal, il peut déterminer le processus qui l'a émis :
    1 - Toujours vrai.
    2 - Impossible.

Système

45. Sous Unix, tous les processus sont des processus :
    1 - Démons.
    2 - Réentrants.
    3 - Zombies.

46. Si un processus possède les droits d'accès 0555, son propriétaire est « root » et est dans le répertoire « /usr /bin » :
    1 - Tous les utilisateurs peuvent l'exécuter.
    2 - Seul le SU peut l'exécuter.
    3 - Seul les utilisateurs appartenant au groupe « système » peuvent l'exécuter.

47. La fonction wait () permet :
    1 - D'attendre la fin d'un fils bien particulier.
    2 - D'attendre la fin d'un fils quelconque.

48. Lorsqu'un signal est envoyé à un processus père par un de ses fils :
    1 - Tous les processus (père et fils) reçoivent le signal.
    2 - Seul le père reçoit le signal.
    3 - Tous les processus (père et fils) reçoivent le signal, sauf de celui qui l'a émit.

49. Sous UNIX, tous les processus passent par l'état :
    1 - Démon.
    2 - Réentrant.
    3 - Zombie.

50. Un processus zombie est :
    1 - Un processus dont le père est mort.
    2 - Un processus mort qui occupe encore son espace mémoire.

51. Pour rechercher une chaîne de caractères « xyz » dans un ensemble de fichiers, il faut utiliser la commande suivante :
    1 - find . –name « xyz » - print.
    2 - grep xyz *.*

52. Pour obtenir l'aide de la fonction fork (), il faut utiliser la commande suivante :
    1 - man fork.
    2 - man fork ().
    3 - man 2 fork.

53. Le code de retour de la fonction fork () est, pour le processus père :
    1 - 0.
    2 - Le pid du fils.
    3 - Son propre pid.

54. Après un exec (), il faut :
    1 - Tester le code de retour de la fonction, pour s'assurer qu'il n'y a pas de problème.
    2 - Rien, car soit elle s'est bien déroulée, soit le programme s'arrête.
    3 - Traiter l'erreur.

55. Un I- nœud est associé à un fichier de façon unique :
    1 - Vrai.
    2 - Faux, il peut y en avoir plusieurs. (Obtenu par un lien)

56. Suite à la réception d'un signal 9 le programme :
    1 - est terminé sans vider les buffers.
    2 - est terminé après avoir vidé les buffers.
    3 - ne faire rien car le signal est réservé au SU.

57. Si taille d'un bloc est de 1024bytes. Un fichier de 11000bytes est constitué de
    1 - 11 blocs.
    2 - 12 blocs.
    3 - 10 + 1 bloc de 780 bytes

58. Un pipe sert :
    1 - à stocker des données.
    2 - mémoriser des informations utiles au noyau.
    3 - à la communication entre processus.

59. Lors de l'exécution de la fonction fork() dans le processus fils **tous** les fichiers ouverts précédemment restent toujours ouverts :
    1 - Vrai.
    2 - Vrai, à la condition que le flag FD_CLOEXEC ne soit pas positionné.
    3 - Vrai, à l'exception du fichier standard stdin.

1. Je veux compiler MonProg.c qui utilise math.h:

cc -o MonProg MonProg.c -lm ???

2. Pour qu'un processus attende sur un sémaphore, il faut la commande:

a - semctl()
b - semget()
c - semop()

3. Pour avoir le MAN de la fonction exit:

a - man -s 2 exit
b - man -s 2 exit()
c - man exit
d - man exit()

4. Quelle commande permet de vérifier la cohérence du système:

a - dump
b - fsck
c - df

5. Quelle est la commande pour restaurer a partir de la bande magnétique:

a - ufsrestore
b - ufsdump
c - mount

6. Qu'est-ce qui permet de mettre un processus en attente:

a - sleep
b - wait
c - pause
d - Pause()

7. Un pipe contient 20 bytes. On utilise un buffer qui est une chaîne de 40 caractères.
On fait rc = read(hdPipe[0],szBuffer,30), que contient rc ?:

a - 20
b - 40
c - la lecture est bloquante

8. En considérant la hiérarchie suivante:
aa: rwx r-x r-x
|-> dd: rwx r-x rwx

L'étudiant Toto pourra-t-il écrire dans le répertoire dd ?

a - Il ne pourra pas car il n'a pas les droits sur le répertoire aa
b - Il peut car il a les droits d'écriture sur le fichier dd

Partie Administration :

9. Que contient /etc/hosts ? (histoire de nom de serveur sur le réseau ?)

liste d'hôtes du réseau local

10. .bash_login, .bash_logout doivent se trouver dans:

a - obligatoirement dans la HOME
b - il faut mettre le chemin
c - n' importe où dans le répertoire de l'utilisateur

11. Sur une machine Sun, les informations relatives aux systèmes de fichiers d'une machine UNIX sont définies:

a - dans le fichiers /etc/vfstab
b - dans le fichiers /etc/disktab
c - nulle part, car elles sont fixées par le constructeur

12. Tout utilisateur qui connait le mot de passe du S.U. peut se connecter en tant que root:

a - Sans problème
b - Jamais
c - Il doit être connu par le système, c.-à-d., indiqué dans un fichier particulier

13. Comment écrire un message à tout les utilisateurs:

a - wall
b - write
c - writeall

14. Comment ajouter un utilisateur:

a - par la commande useradd
b - éditer le fichier dans /etc/passwrd
c - le compte se créé automatiquement lors du premier LOG