INFORMATIQUE & INFORMATIQUE INDUSTRIELLE
PREPARATION A L'EXAMEN D'UNIX
SESSION DE JANVIER 2004 - MERCENIER

Système d'exploitation :

+ Sous Unix, tous les processus sont des processus :

1) Démons.
2) Réentrants.
3) Zombies.

+ 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.

+ La fonction wait () permet :

1) D'attendre la fin d'un fils bien particulier.
2) D'attendre la fin d'un fils quelconque.
3) La frappe d'un caractère ou la réception d'un signal quelconque.

+ 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 du celui qui la émit.

+ Sous UNIX, tous les processus passent par l'état :

1) Démon.
2) Réentrant.
3) Zombie.

+ 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 *.*

+ Pour obtenir l'aide de la fonction fork (), il faut utiliser la commande suivante :

1) man fork.
2) man fork ().
3) man 2 fork.

+ Le code de retour de la fonction fork () est, pour le processus père :

1) 0.
2) Le pid du fils.
3) Son propre pid.

+ 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.

+ 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)
3) Vrai, mais il faut le préciser à la création.

Les fichiers :

Soit le fichier Examen.txt. Son contenu est :

Abcdefghijklmnopqrstuvwxyz & il sera utilisé pour les exemples de la suite.

+ Soit le programme suivant :

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

Le fichier Examen.txt est :

1) 123456ghijklmnopqrstuvwxyz.
2) 456defghijklmnopqrstuvwxyz.
3) 123defghijklmnopqrstuvwxyz.

+ Soit le programme suivant :

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

rc = ?

1) 3.
2) - 1. (Cas d'erreur)
3) 0.

+ 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) L'arrêt du programme.
4) ... 0

+ 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.

+ 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. (Ou XYZABCghijklmnopqrstuvwxyz)
2) abcdefghijklmnopqrstuvwxyzABCXYZ.
3) XYZdefghijklmnopqrstuvwxyz.

+ Soit le programme suivant :

```
hd = open ("Examen.txt",O_RDWR);
hd1 = dup (hd);
lseek (hd,3 ,0);
read (hd ,szBuffer,3);
read (hd1, szBuffer1,3);
exit (0);
```

szBuffer = ?

1) szBuffer = "def" szBuffer1 = "ghi".
2) szBuffer = "def' szBuffer1 = "def'.
3) szBuffer = "def' szBuffer1 = "abc".

+ 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.

+ Un proc a bloqué 1 fichier avec 1 verrou partagé & qu'il se termine suite à 1 SIGQUIT. Le fichier garde son verrou :

1) Vrai.
2) Faux.
3) Le verrou est levé uniquement si le programme l'a prévu.

+ 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.

+ Un fichier spécial, sous UNIX, caractérise :

1) Un exécutable.
2) Un fichier associé à un périphérique.
3) Un fichier système.

+ Un fichier FIFO (un tube) sert :

1) À stocker des données.
2) À mémoriser des informations utiles au noyau.
3) A la communication entre processus.

+ Un fichier peut- être supprimer 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.

+ Etant donné 1 fichier ouvert en O_RDWR, il est possible de se positionner au- delà de la fin de fichier via la fct lseek () :

1) Jamais.
2) Sans problème.

+ 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.

+ 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.

+ 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.

+ 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.

+ 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.

+ La fonction dup () alloue un nouveau descripteur dans la table des fichiers ouverts, ce dernier :

1) Il aura la même entrée dans la table des fichiers ouverts.
2) Il aura une entrée différente dans la table des fichiers ouverts, mais accèdera au même fichier.
3) Ca N'aura aucun effet, car le fichier est déjà ouvert une fois.

+ 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.

+ Dans le cadre de l'institut, un étudiant du même groupe que vous peut :

1) Lire et écrire dans vos 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)

+ 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) Aucuns paramètres, sauf si l'on est SU.

+ Lors du login, votre terminal se trouve :

1) En mode canonique.
2) En mode non canonique.

+ 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 :

+ 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.

+ 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.

+ 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.

+ 2 processus indépendants peuvent communiquer par pipe :

1) Faux, en aucun cas.
2) Vrai il suffit d'ouvrir le pipe par la fonction fcntl ().
3) Vrai le pipe étant propriété du système, on peut toujours l'ouvrir.

+ Un processus peut ouvrir un pipe après l'avoir fermé :

1) Vrai grâce à la fonction fnctl ().
2) Faux, en aucun cas.
3) Vrai grâce à la fonction open ().

+ 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.

+ 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.

Les I.P.C. :

+ 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.

+ 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 correctes, car l'opération est atomique.

+ Si l'on manipule 1 ensemble de sémaphores avec semop () & que sur 1 d'entre eux 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.
3) L'opération ne sera par faite, et le programme continue son exécution, il faut prévoir le cas.

+ On peut ouvrir un certain nombre de files de messages, ce nombre est :

1) Illimité, il suffit que les clés soient différentes.
2) Ce nombre est limité par le système.
3) Ce nombre est limité par le système et est le même pour tous les utilisateurs.

+ Le propriétaire peut supprimé une file de messages à tout moment :

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

+ Le propriétaire peut supprimé une mémoire partagée à tout moment :

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

+ Un sémaphore peut avoir :

1) Une valeur entière positive. (Ou 0)
2) Une valeur entière quelconque.
3) Une valeur réelle.

+ 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.

+ 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.

+ 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.

+ Lorsqu'un processus reçoit un signal, il peut déterminer le processus qui l'a émis :

1) Toujours vrai.
2) Impossible.

Notes :

- On ne peut modifier un verrou partagé en exclusif pour cause de deadlock. (Refusé par les nvx compilateurs)

- Deux processus ne peuvent jamais communiquer par pipe. (Mais bien par named pipe)

- Lorsque le père émet un signal, le père et le fils le reçoivent.

- Lors d'un FORK, les segments de données sont recopiés.

- Un propriétaire peut effacer une SHM quand il veut.

- Au moins la pluie tombe d'avantage, au moins il convient de réduire l'augmentation de la vitesse du balai. Et vice-versa.

- Un pipe s'hérite.