BORDEAUX INP Enseirb-Matmeca

QCM – Tubes et exclusion mutuelle

IF 110 – Introduction aux systèmes d'exploitations

1. Qu'est-ce qu'un tube (pipe) dans un système d'exploitation ?

A. Un outil pour déboguer le code

B. Un mécanisme de communication inter-processus

C. Un protocole réseau

D. Une méthode de stockage de données

#QDLE#Q#AB*CD#15#

Un tube est un mécanisme de communication par

A. Envoi de message

B. Partage de mémoire

#QDLE#Q#A*B#10#

Un tube nommé est un fichier ?

A. Oui
B. Non

#QDLE#Q#A*B#10#

Un tube anonyme est un fichier ?

A. Oui
B. Non

#QDLE#Q#AB*#7#

Les données sur les tubes sont transmises :

A. De manière séquentielle

B. De manière aléatoire

C. En parallèle

D. En blocs de taille fixe

#QDLE#Q#A*BCD#15#

Les tubes nommés sous Unix/Linux :

A. Sont visibles uniquement pour le processus qui les crée

B. Peuvent être utilisés par n'importe quel processus ayant les bons droits

C. Sont toujours supprimés à la fin du processus

D. Ne peuvent transmettre que des données textuelles

#QDLE#Q#AB*CD#30#

Les messages envoyés dans un tube peuvent-t-ils être perdus ?

A. Oui

B. Non

C. Ça dépend

#QDLE#Q#AB*C#15#

Les données transmises sur les tubes sont :

A. Quelquonces
B. Définies par un intervalle de valeur

#QDLE#Q#A*B#10#

Un tube peut être utilisé pour :

A. Transmettre des signaux entre processus

B. Partager la mémoire entre processus

C. Transférer des fichiers entre machines

D. Implémenter des redirections dans le terminal

#QDLE#Q#ABCD*#20#

Les tubes nommés peuvent être utilisés pour communiquer entre :

A. Des processus qui ne sont pas liés entre eux

B. Uniquement des processus parents-enfants

C. Des processus exécutés sur des machines différentes

D. Uniquement des processus s'exécutant en parallèle

#QDLE#Q#A*BCD#25#

Quelle est la principale différence entre un tube nommé et un tube anonyme ?

A. La durée de vie

B. La capacité de stockage

C. Le type de données transmises

D. La méthode de création

#QDLE#Q#A*BCD#23#

Quelle commande bash sous Unix permet de créer un tube nommé ?

A. touch

B. mkpipe

C. mkfifo

D. pipe

#QDLE#Q#AB*CD#12#

Quelle est la limitation principale d'un tube anonyme ?

A. Il ne peut transmettre que du texte

B. Il ne peut être utilisé que par deux processus

C. Il a une capacité de stockage limitée

D. Il ne fonctionne que sur des systèmes d'exploitation spécifiques

#QDLE#Q#AB*CD#27#

Comment éviter les fermetures intempestives de tubes

A. En ouvrant un tube avec une redirection avancée en lecture + écriture
B. En ouvrant un tube avec une redirection avancée en lecture
C. En ouvrant un tube avec une redirection avancée en écriture
D. En ouvrant un tube avec la commande mkfifo

#QDLE#Q#A*BCD#30#

Pourquoi utiliser un tube pour la communication entre processus ?

A. Augmentation de la vitesse de traitement

B. Diminution de la charge CPU

C. Synchronisation entre les processus

D. Réduction de la consommation de mémoire

#QDLE#Q#ABC*D#23#

Un mutex est utilisé pour :

A. Assurer qu’un processus puisse exécuter une section critique sans interférence d’autres processus.

B. Bloquer définitivement un processus dans une section critique.

C. Permettre à plusieurs processus d’exécuter simultanément une section critique.

D. Accélérer l’exécution des sections critiques.

#QDLE#Q#A*BCD#25#

Si deux processus exécutent ce script simultanément, qu'observe-t-on ?

```
1   #!/bin/bash
2
3   critical_section() {
4       echo "Process $$ in critical section"
5       sleep 2
6       echo "Process $$ leaving critical section"
7   }
8
9   P.sh my_mutex
10  critical_section
11  V.sh my_mutex
```

A. Les deux processus accèdent à la section critique en même temps.
B. Un seul processus accède à la section critique à la fois.
C. Les deux processus restent bloqués.
D. Le mutex est ignoré, et le comportement est indéfini.

#QDLE#Q#AB*CD#40#

Que peut-il se passer ?

Script1.sh

```
1    P.sh mutex1
2    sleep 2
3    P.sh mutex2
4    critical_section
5    V.sh mutex2
6    V.sh mutex1
```

Script2.sh

```
1    P.sh mutex2
2    sleep 2
3    P.sh mutex1
4    critical_section
5    V.sh mutex1
6    V.sh mutex2
```

A. Aucun problème : les processus accèdent à la section critique sans conflit.
B. Les deux processus accèdent simultanément à la section critique.
C. Les deux processus peuvent rester bloqués (interblocage).
D. Le système résout automatiquement les conflits d'accès.

#QDLE#Q#ABC*D#40#

Lequel des scénarios suivants peut provoquer un interblocage ?

A. Deux processus tentent d’acquérir les mêmes mutex dans un ordre différent.

B. Un processus oublie de libérer un mutex après l’avoir acquis.

C. Les processus utilisent des mutex toujours dans le même ordre.

D. Les processus ne partagent pas de ressources communes.

#QDLE#Q#A*BCD#20#

Que garantit l'utilisation de P.sh et V.sh si ce script est executé plusieurs fois en simultané ?

```
1   #!/bin/bash
2
3   shared_file="shared_counter.txt"
4   mutex="my_mutex"
5
6   # Initialisation (à exécuter une seule fois)
7   if [ ! -f $shared_file ]; then
8       echo 0 > $shared_file
9   fi
10
11  P.sh $mutex
12  counter=$(cat $shared_file)
13  counter=$((counter + 1))
14  echo $counter > $shared_file
15  V.sh $mutex
16
17  echo "Process $$ incremented counter to $counter"
```

A. Que plusieurs processus peuvent écrire simultanément dans le fichier *shared_counter.txt*
B. Que l'accès au fichier *shared_counter.txt* est protégé contre les accès concurrents.
C. Que la valeur du compteur sera toujours incorrecte.
D. Que tous les processus s'exécuteront sans attendre.

#QDLE#Q#AB*CD#45#

Pourquoi ce code ne garantit-il pas l'exclusion mutuelle ?

```
1   critical_section() {
2       echo "Process $$ in critical section"
3       sleep 1
4       echo "Process $$ leaving critical section"
5   }
6
7   critical_section &
8   critical_section &
9   wait
10
```

A. Il n'utilise pas de mécanisme explicite pour protéger la section critique.
B. Il n'attend pas que les processus en arrière-plan se terminent.
C. Les deux processus utilisent la commande wait.
D. Les processus en arrière-plan sont automatiquement synchronisés.

#QDLE#Q#A*BCD#40#