

Master mention informatique — Systèmes d'exploitation

Session de septembre

Durée : 3h — Notes de cours autorisées

1 Gestion mémoire

Question 1 Qu'est-ce qu'une MMU ? À quoi sert la TLB d'une MMU ? Cette dernière a-t-elle une forte capacité mémoire ? Pourquoi ?

Question 2 De quelles fonctionnalités (opérations déclenchées de manière logicielle) doit-on disposer au niveau du système pour gérer correctement une MMU ? Pensez aux changements de contexte de processus ainsi qu'aux défauts de pages...

Question 3 Lorsqu'une page est évacuée vers la mémoire secondaire, il faut mémoriser quelque part le numéro du bloc disque correspondant. En particulier, on pourrait décider d'utiliser le champ qui sert d'ordinaire à spécifier le numéro de page physique (puisque la page n'est plus en mémoire). En supposant que la page physique numéro 0 n'est jamais attribuée à un processus, expliquez comment le système pourrait alors simplement distinguer une page non-valide d'une page présente sur le disque mais pas en mémoire.

Discutez des inconvénients de cette façon de mémoriser l'emplacement sur le disque. Quelle autre alternative pourrait-on choisir ? Quels seraient les inconvénients de cette seconde approche ?

Question 4 La plupart des systèmes d'exploitation permettent aux processus de partager physiquement des zones mémoires. Dans un système basé sur la pagination, cela se traduit par le fait que plusieurs tables de pages peuvent référencer la même page physique.

Sans support spécifique du matériel, comment gérer correctement le partage des pages ? En particulier, comment veiller à ce que tout se passe bien lorsqu'une telle page est évacuée sur le disque ?

Question 5 Rappelez en quoi consiste le mécanisme appelé "*Copy-on-Write*" et à quoi il sert. Expliquez comment un système peut mettre en place un tel mécanisme en ce qui concerne les structures de données du noyau (tables des pages ou autres). Comment le système sait-il qu'il doit déclencher un "COW" dans son traitement d'interruption ?

Question 6 Certains pensaient que l'introduction du mécanisme de pagination dans les systèmes d'exploitation allait rendre le recours à un mécanisme de va-et-vient de processus inutile. En fait, ce dernier mécanisme est toujours présent dans les systèmes multiprogrammés actuels. Pourquoi ?

1.1 Synchronisation et section critique

Nous l'avons vu dans Nachos : la plupart des synchronisations visant à entrer en section critique pour une courte durée utilisent simplement le masquage des interruptions pour inhiber tout risque de préemption intempestive. Toutefois, dans un noyau s'exécutant sur plusieurs processeurs, ce mécanisme devient bien évidemment insuffisant !

Question 1 Quel est le mécanisme utilisé dans ce cas (en complément du masquage des interruptions sur le processeur courant) ? Sur quelle instruction machine repose-t-il typiquement ?

Question 2 Détailler les opérations à effectuer lorsque l'on désire ainsi protéger une section critique, en veillant bien à l'ordre dans lequel il est nécessaire d'utiliser les deux mécanismes complémentaires.

Question 3 Selon la taille des sections critiques, on pourrait hésiter entre utiliser la stratégie précédente ou utiliser des outils tels que les sémaphores, qui provoqueraient parfois le blocage des processus. Pourquoi y a-t-il dilemme ?

Question 4 Expliquez comment on pourrait combiner les deux mécanismes pour obtenir une méthode de synchronisation assez efficace en moyenne.

2 Moniteurs de Hoare

On s'intéresse à résoudre le classique problème des "lecteurs-rédacteurs" à l'aide des moniteurs de Hoare. On suppose l'existence des définitions suivantes :

```
typedef ... lock_t ;
typedef ... cond_t ;
void acquire(lock_t *l);
void release(lock_t *l);
void cond_wait(cond_t *c, lock_t *l);
void cond_signal(cond_t *c);
```

On suppose également l'existence des primitives `lire()` et `ecrire()` qui imposent les contraintes suivantes : un appel à la fonction `lire` peut être effectué en parallèle avec d'autres appels à `lire`, mais un appel à la fonction `ecrire` doit être effectué en exclusion mutuelle de tout autre appel à `ecrire` ou à `lire`.

Il s'agit donc d'implanter la fonction `lecteur` (resp. `redacteur`) effectuant un appel à la fonction `lire` (resp. `ecrire`) en respectant les contraintes précédentes.

Question 1 Donnez une version des fonctions `lecteur` et `redacteur` privilégiant les processus lecteurs.

Question 2 Même question en privilégiant les processus rédacteurs.