Architecture des **S**ystèmes d'**E**xploitation
Maîtrise d'Informatique

Janvier 2012

Devoir Surveillé

– tous documents autorisés –

Le langage C est requis pour les implémentations

Les différentes parties peuvent être traitées indépendamment, tous les documents sont autorisés, mais ne perdez pas trop de temps à chercher l'information utile. Notez encore que l'énoncé contient 24 points, je m'arrêterai à 20/20 ;-)

Exercice I. Mémoire virtuelle (8 points)

La MMU d'un processeur 32 bits repose sur un découpage des adresses virtuelles (32 bits) comme indiqué ci-dessous :

bit 31… bit 24	bit 23… bit 17	bit 16… bit 10	bit 9… bit 0
idx1	idx2	idx3	offset

Pour traduire une adresse virtuelle en adresse physique, le microprocesseur utilise un registre GTT (*Global Translation Table*). Ce registre donne l'entrée d'une table d'adresses. A l'index idx1 de cette table le microprocesseur trouve l'adresse d'une autre table (la table d'indirection de second niveau). A l'index idx2 de cette table le processeur trouve l'adresse d'une troisième table (la table d'indirection de troisième niveau). Finalement, à l'index idx3 de cette table le processeur trouve (enfin) l'adresse de la page de mémoire physique dans laquelle il peut aller lire.

Question I.1 (1 point)

En assumant que le processeur ne dispose pas de cache pour la traduction d'adresse virtuelle en adresse physique. Combien de lecture en RAM doit-il faire lorsque le programme demande à lire 1 donnée en mémoire virtuelle ?

Question I.2 (1 point)

Selon la décomposition de l'adresse virtuelle donnée précédemment, quelle est la taille d'une page de mémoire ?

Question I.3 (3 points)

En considérant que les adresses stockées dans cette table sont de 32 bits, quelle est la taille de la *Global Translation Table* au regard de la décomposition présentée précédemment ? Quel est la taille d'une table d'indirection de second niveau ? Et celle d'une table d'indirection de troisième niveau ?

Question I.4 (3 points)

Au total, quel sera la taille, en octets, de l'ensemble des tables stockées en mémoire pour permettre de mapper en mémoire physique un espace d'adressage virtuel de 1 Go ?

Exercice II. Ordonnancement des tâches (8 points)

Les mécanismes de synchronisation de tâche peuvent être la source de problèmes d'interblocage entre plusieurs programmes.

Question II.1 (1 point)

Expliquez quand qu'elle circonstance il est pertinent d'utiliser un mutex lors de la réalisation de logiciels ?

Question II.2 (1 point)

Expliquez les rôles des trois primitives d'un mécanisme de sémaphore, dont les prototypes sont donnés ci-dessous :

```
void initSemaphore(struct *sem, int initialValue);
void P(struct *sem, int v);
void V(struct *sem, int v);
```

Question II.3 (3 points)

Expliquez pourquoi les tâches T1 et T2, ci-dessous, peuvent entrer en interblocage (les sections « … » rapportent la présence de ligne de code qui ne peuvent pas être la source d'interblocage mais qui manipulent des données sensibles) :

```
/* Variables globales et fonctions partagées */
int d1,d2,d3 ;        /* données sensibles */
struct sem s1,s2,s3 ; /* ces sémaphores sont initialisées à 1 */
void f() {
  P(s1);
  ... /* lecture/modification de d1 */
  h();
  V(s1);
}
void g() {
  P(s2);
  h();
  ... /* lecture/modification de d2 */
  V(s2);
}
void h(){
  P(s3);
  ... /* lecture/modification de d3 */
  V(s3);
}
```

```
/* Tâche T1 */
void f1() {
  P(s2);
  ... /* lecture/modif de d2 */
  f();
  V(s2);
}
```

```
/* Tâche T2 */
void f2() {
  P(s1);
  ... /* lecture/modif de d1 */
  g();
  V(s1);
}
```

Question II.4 (3 points)

S2 est en fait utiliser pour éviter de faire des accès concurrents à une donnée d2. S1 est utilisé pour protéger une autre structure d1et s3 une structure d3. Corrigez le programme pour préserver ces protections et éviter tous risque d'interblocage.

Exercice III. Performance des disques durs (8 points)

Il est parfois utile de pouvoir cloner le contenu complet d'un disque ou d'une partition. On parle d'une image *ghost* qui est en fait une copie parfaite d'un disque sur un autre. Nous nous intéressons à la réalisation d'un tel programme.

Nous nous intéressons ici à la gestion d'une famille de disques aux propriétés particulières. Ces disques disposent de 4096 secteurs et de 8192 pistes. Chaque secteur contient 4096 octets (4 Ko). Ces disques tournent à la vitesse de 7320 tours par minute, c'est-à-dire 122 tours par seconde ou encore 2 microsecondes pour passer d'un secteur au secteur suivant. La rotation de ces disques est continue (elle ne s'arrête donc pas même si le disque a déjà atteint le bon secteur mais pas encore la bonne piste par exemple) et le sens de rotation du disque est imposé (la tête de lecture parcourt les secteurs dans l'ordre croissant : 0, 1, 2, …, 4094, 4095, 0,1…). Par ailleurs, la tête de lecture peut se déplacer de la piste 0 à la piste 8191 en 8,192 millisecondes, c'est-à-dire qu'il faut 1 microsecondes à la tête de lecture pour avancer ou reculer d'une piste. Le sens du déplacement sur les pistes n'est pas imposé par le matériel.

Les délais de lecture, d'écriture ou de formatage sont inscrits dans les délais de « survol » de la piste, et peuvent donc être considérés comme nul par rapport aux délais de déplacement.

Les registres d'accès au contrôleur de ces disques sont décrits dans l'extrait ci-dessous :

Extrait du fichier `Hardware.ini` qui décrit les ports associés aux disques.

```
#
# Configuration des disques durs
#

# > Disque dur IDE Maitre
ENABLE_HDA      = 1     # ENABLE_HD=0 =>
                        #             simulation du disque désactivée
HDA_CMDREG      = 0x3F6 # registre de commande du disque maître
HDA_DATAREGS    = 0x110 # base des registres de données
                        # (r,r+1,r+2,...r+7)
HDA_IRQ         = 14    # Interruption du disque



# > Disque dur IDE Esclave
ENABLE_HDB      = 1     # ENABLE_HD=0 =>
                        #             simulation du disque désactivée
HDA_CMDREG      = 0x376 # registre de commande du disque esclave
HDA_DATAREGS    = 0x170 # base des registres de données
                        # (r,r+1,r+2,...r+7)
HDA_IRQ         = 15    # Interruption du disque
```

La fonction `int _in(int port);` réalise la lecture sur le port désigné. La valeur retournée correspond à l'octet qui a été lu sur ce port. Les numéros de port sont identifiés dans le fichier de configuration du matériel `hardware.ini`. La fonction `void _out(int port, int value);` réalise l'écriture d'une valeur d'un octet sur le port désigné.

L'envoi de commandes se fait également en écrivant sur le port désigné comme port de commande dans le fichier de configuration. Cela permet au microprocesseur de solliciter une opération du disque magnétique. Une fois que le microprocesseur envoie une commande, l'exécution de celle-ci débute immédiatement. Si la commande nécessite des données, il faut obligatoirement les avoir fournies avant de déclencher la commande. De la liste des commandes ATA-2 nous avons retenu le sous-ensemble décrit dans la table 1. Comme il y a deux disques, un maître et un esclave il y a deux ports de commande et deux ports de données, et les deux disques peuvent fonctionner en parallèle.

Nom	Code	Port de données (P0; P1; ...; P15)	objet
SEEK	0x02	numCyl (int16); numSec (int16)	déplace la tête de lecture
READ	0x04	nbSec (int16)	lit nbSec secteurs
WRITE	0x06	nbSec (int16)	écrit nbSec secteurs
FORMAT	0x08	nbSec (int16); val (int32)	initialise nbSec secteurs avec val
STATUS	0x12	IRQFlags (int8)	Drapeau d'activation des interruptions.
DMASET	0x14	R.F.U.	R.F.U.
DSKNFO	0x16	nbCyl (int16); nbSec (int16); tailleSec (int16)	retourne la géométrie d'un disque
MANUF	0xA2	Id du fabricant du disque (16 octets)	Identifie le disque
DIAG	0xA4	Status	Diagnostic du disque : 0 = KO / 1 = OK

Table 1 : Commandes ATA-2

A l'aide des quatre procédures ci-dessous :

```
void seekA(int cyl, int sec) {
   /* set cylinder */
   _out(0x110,(cyl>>8)&0xFF);
   _out(0x111,cyl&0xFF);
   /* set cylinder */
   _out(0x112,(sec>>8)&0xFF);
   _out(0x113,sec&0xFF);
   /* set command */
   _out(0x3F6,0x02);
   /* wait seek */
   _sleep(14);
}
```

```
void seekB(int cyl, int sec) {
   /* set cylinder */
   _out(0x170,(cyl>>8)&0xFF);
   _out(0x171,cyl&0xFF);
   /* set cylinder */
   _out(0x172,(sec>>8)&0xFF);
   _out(0x173,sec&0xFF);
   /* set command */
   _out(0x376,0x02);
   /* wait seek */
   _sleep(15);
}
```

```
void read_sectorA(int value) {
   _out(0x110,0);
   _out(0x111,1);
   _out(0x112,(value>>24)&255);
   _out(0x113,(value>>16)&255);
   _out(0x114,(value>>8)&255);
   _out(0x115,(value)&255);
   /* set command read */
   _out(0x3F6,0x04);
   /* wait format */
   _sleep(14);
}
```

```
void write_sectorB(int value) {
   _out(0x170,0);
   _out(0x171,1);
   _out(0x172,(value>>24)&255);
   _out(0x173,(value>>16)&255);
   _out(0x174,(value>>8)&255);
   _out(0x175,(value)&255);
   /* set command read */
   _out(0x376,0x06);
   /* wait format */
   _sleep(15);
}
```

Les deux programmes suivants permettent de réaliser une copie fidèle du disque maître sur le disque esclave :

Programme 1 :

```
void copyDsk() {
 int maxC,maxS;
 int sec,cyl;
 /* Get disk geometry */
 _out(0x3F6,0x16);
 maxC=(_in(0x110)<<8)+_in(0x111);
 maxS=(_in(0x112)<<8)+_in(0x113);
 for(cyl=0;cyl<maxC;cyl++) {
  for(sec=0;sec<maxS;sec++) {
   seekA(cyl,sec);/*seek master*/
   readA();
   memcpy(MASTERBUFFER,
          SLAVEBUFFER,
          SIZEOFSECTOR);
   seekB(cyl,sec);/*seek slave*/
   writeB();
  }
 }
}
```

Programme 2

```
void copyDsk() {
 int maxC,maxS;
 int sec,cyl;
 /* Get disk geometry */
 _out(0x3F6,0x16);
 maxC=(_in(0x110)<<8)+_in(0x111);
 maxS=(_in(0x112)<<8)+_in(0x113);
 for(cyl=0;cyl<maxC;cyl++) {
  for(sec=maxS-1;sec>=0;sec--) {
   seekA(cyl,sec);/*seek master*/
   readA();
   memcpy(MASTERBUFFER,
          SLAVEBUFFER,
          SIZEOFSECTOR);
   seekB(cyl,sec);/*seek slave*/
   writeB();
  }
 }
}
```

Question III.1 (1 point)

Considérez les paramètres donnés au début de l'énoncé. Considérez aussi qu'avant l'exécution de la copie, les 2 têtes de lectures sont placées sur la piste zéro, secteur zéro. Selon ces hypothèses, calculez le temps que prendra la copie d'un disque sur l'autre selon le programme 1.

Question III.2 (1 point)

Selon les mêmes hypothèses combien de temps prendra la copie réalisée par le programme 2 ? (la seule différence est la ligne mise en gras) Expliquez cette importante différence dans le temps de réalisation de la copie.

Question III.3 (2 point)

En constatant que les deux disques fonctionnent de façon indépendante. Il est possible de passer une commande sur le disque maître et une autre sur le disque esclave, sans attendre la fin de la première. Les deux commandes seront alors réalisées simultanément. Ainsi il est possible de réaliser un programme réalisant une copie au moins deux fois plus rapidement que le meilleur des programmes présentés ci-dessus. Expliquez comment il faudrait modifier le programme en conséquence ?

Question III.4 (2 point)

En considérant la réponse que vous avez formulée dans la question précédente, donnez le code du programme le plus performant que vous pouvez imaginer pour copier un disque maître sur un disque esclave de même géométrie.

Question III.5 (2 point)

Les systèmes d'exploitation modernes copient un fichier (suffisamment volumineux) deux fois plus vite lorsqu'il s'agit d'une copie d'un disque source vers un disque destination différent, que lorsque le fichier est copié sur le même disque que celui dont il provient. Expliquez pourquoi il en est ainsi.