Sistemes Operatius II
Avaluació parcial – Parcial part 2 – 20 de desembre del 2018

Nom i Cognoms: _____

En total hi ha 3 problemes. Podeu fer servir apunts o portàtil per fer consultes a les transparències però no es pot programar cap codi. Només cal respondre breument a les preguntes. No es puntuaran preguntes no raonades.

Problema 1 (concurrència, dificultat baixa, 2.5 punts). Suposeu el següent codi

```
01    variables globals:
02          pthread_mutex_t mutex1, mutex2;
03
04    funcio1:
05          lock(&mutex1);
06          /* funcio 1, part 1 */
07          lock(&mutex2);
08          /* funcio 1, part 2 */
09          unlock(&mutex2);
10          /* funcio 1, part 3 */
11          unlock(&mutex1);
12
13    funcio2:
14          lock(&mutex2);
15          /* funcio 2, part 1 */
16          lock(&mutex1);
17          /* funcio 2, part 2 */
18          unlock(&mutex1);
19          /* funcio 2, part 3 */
20          unlock(&mutex2);
```

Es demana

1. En aquest codi les variables mutex1 i mutex2 són variables globals. Fa falta que siguin globals? No poden ser locals a cadascuna de les dues funcions, funcio1 i funcio2? Raona la resposta.
2. Indica quin problema es pot produir en aquest codi i per què. Raona la resposta indicant els canvis de context que s'han de produir perquè es produeixi el problema.
3. Comenta (breument) com solucionar el problema que es dóna al punt 2. Indica com ha de ser el codi de funcio1 i funcio2 perquè el codi funcioni correctament.

Problema 2 (semàfors i monitors, dificultat mitjana, 3.5 punts). A la biblioteca de la Facultat de Matemàtiques i Informàtica hi ha ordinadors disponibles per als alumnes de grau. En total hi ha M ordinadors disponibles i un estudiant qualsevol pot agafar un ordinador sempre que n'hi hagi un de lliure. Un cop l'ha fet servir el torna a la biblioteca.

Disposem d'una variable global que ens indica si l'ordinador està lliure (disponible) o no. El codi de la declaració, així com el d'inicialització, agafar un ordinador i retornar-lo es mostra a continuació.

```
01    variables globals:
02          int M = ...; // un valor sencer, pex 10
03          boolean lliure[M];
04
05    inicialitzacio:
06          int i;
07          for(i = 0; i < M; i++)
08                lliure[i] = true;
09
10    agafar_ordinador:
11          int i;
12          for(i = 0; i < M; i++)
13                if (lliure[i]) {
14                      lliure[i] = false;
15                      return i;
16                }
17          exit(ERROR);
18
19    retornar_ordinador(int i):
20          lliure[i] = true;
21
22    prestec_ordinador:
23          int i;
24          i = agafar_ordinador;
25          /* L'estudiant treballa amb l'ordinador */
26          retornar_ordinador(i);
```

Suposem que s'ha executat la funció d'inicialització per indicar que tots els ordinadors estan disponibles. Respecte la darrera funció, prestec_ordinador, es demana el següent

1. Comenta el problema que es pot produir en cas que múltiples estudiants vulguin fer un préstec d'un ordinador. Indica quina(es) funció(ns) cal protegir, a la funció prestec_ordinador, perquè no es produeixin problemes. Raona la resposta!
2. Indica i comenta com s'ha de modificar el codi de prestec_ordinador perquè no es produeixin problemes en cas que múltiples estudiants vulguin fer un préstec d'un ordinador al mateix temps. S'han de fer servir semàfors.
3. Atès el fet que només hi ha M ordinador disponibles, comenta com s'ha de modificar el codi de prestec_ordinador per assegurar que l'estudiant trobi un ordinador lliure sempre que en vulgui agafar un (és a dir, que no s'executi la línia 17 del codi). S'han de fer servir semàfors.
4. En cas que es vulguin fer servir monitors comenta breument com solucionar els punts 2 i 3. Pots fer referència a una transparència directament si així ho desitges.

Problema 3 (monitors, dificultat alta als darrers exercicis, 4 punts). A continuació es mostra el següent algorisme, vist a classe de teoria, dels lectors i escriptors justos per a monitors.

```
01    variables globals:
02          int nr = 0; boolean w = false;
03          pthread_mutex_t mutex; pthread_cond_t cond;
04
05    read_lock:
06          lock(&mutex);
07          while (w) {
08                wait(&cond, &mutex);
09          }
10          nr++;
11          unlock(&mutex);
12
13    read_unlock:
14          lock(&mutex);
15          nr--;
16          if (nr == 0) broadcast(&cond);
17          unlock(&mutex);
18
19    write_lock:
20          lock(&mutex);
21          while (w) {
22                wait(&cond, &mutex);
23          }
24          w = true;
25          while (nr != 0) {
26                wait(&cond, &mutex);
27          }
28          unlock(&mutex);
29
30    write_unlock:
31          lock(&mutex);
32          w = false;
33          broadcast(&cond);
34          unlock(&mutex);
```

Es demana contestar a les següents preguntes breument:

1. Què és el que fa que aquest algorisme sigui considerat un algorisme just? En altres paraules, com s'assegura que els escriptors impedeixin l'entrada als lectors a la seva zona crítica?
2. Suposem que hi ha un escriptor escrivint i que hi ha lectors adormits al wait de la línia 8 així com escriptors adormits al wait de la línia 22. En sortir l'escriptor de la seva secció crítica crida la funció write_unlock, la qual crida al broadcast de la línia 33. Se sap qui dels dos, lectors o escriptors, podrà entrar a la secció crítica? Comenta breument la resposta.
3. (Difícil) Es proposa modificar el codi per donar preferència als escriptors en cas que es doni el cas descrit a la pregunta anterior. Indiqueu clarament les modificacions a realitzar al codi. Les podeu realitzar directament sobre el codi d'aquesta pàgina.
4. (Difícil) Comenteu per què les modificacions proposades donen preferència als escriptors.