PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

2da práctica (tipo a)
(Segundo semestre de 2017)

Horario 0781: prof. V. Khlebnikov

Duración: 1 h. 50 min.
Nota: No se puede usar ningún material de consulta.
La presentación, la ortografía y la gramática influirán en la calificación.
Puntaje total: 20 puntos

Pregunta 1. (**5 puntos – 25 min.**) Consideremos el siguiente código:

```
 1      int rc,wc; semaphore x=1, y=1, z=1, wsem=1, rsem=1;
 2
 3      void reader()
 4      {
 5          while (true) {
 6              semWait(z);
 7                  semWait(rsem);
 8                      semWait(x);
 9                          rc++;
10                          if (rc == 1)
11                              semWait(wsem);
12                      semSignal(x);
13                  semSignal(rsem);
14              semSignal(z);
15              read();
16              semWait(x);
17                  rc--;
18                  if (rc == 0) semSignal(wsem);
19              semSignal(x);
20          }
21      }
22
```

```
23      void writer()
24      {
25          while (true) {
26              semWait(y);
27                  wc++;
28                  if (wc == 1)
29                      semWait(rsem);
30              semSignal(y);
31              semWait(wsem);
32                  write();
33              semSignal(wsem);
34              semWait(y);
35                  wc--;
36                  if (wc == 0) semSignal(rsem);
37              semSignal(y);
38          }
39      }
40      void main() {
41          rc=wc=0;
42          parbegin(reader,writer); }
```

a) (**1 punto – 5 min.**) Si en el sistema hay solamente los procesos de lectores y son muchos, ¿cuántos de ellos pueden bloquearse en el semáforo `rsem`? ¿Y en el semáforo `x`?

b) (**2 puntos – 10 min.**) Si en el sistema hay los lectores y los escritores pero el primero es un lector: R1, R2, R3, W1, R4, R5, W2, R6, W3, etc., ¿en qué semáforos se forman las colas de qué procesos y cuál será el comportamiento del sistema?

c) (**2 puntos – 10 min.**) Si en el sistema hay los lectores y los escritores pero el primero es un escritor: W1, R1, R2, R3, W2, R4, R5, W3, R6, W4, etc., ¿en qué semáforos se forman las colas de qué procesos y cuál será el comportamiento del sistema?

Pregunta 2. (**5 puntos – 25 min.**) Se suponía que el siguiente código debería garantizar la exclusión mutua pero no lo hace. Encuentre la secuencia de ejecución tal que ambos procesos pasan a sus secciones críticas juntos.

```
 1      boolean blocked[2];
 2      int turn;

 3      void P(int id)
 4      {
 5          while (TRUE) {
 6              blocked[id] = TRUE;
 7              while (turn != id) {
 8                  while (blocked[1-id]);
 9                  turn = id;
10              }
11              /* critical section */
12              blocked[id] = FALSE;
13              /* remainder */
14          }
15      }
```

```
16      void main()
17      {
18          blocked[0] = blocked[1] = FALSE;
19          turn = 0;
20          parbegin(P(0),P(1));
21      }
```

Pregunta 3. (**5 puntos – 25 min.**) Se proporcionan 2 soluciones al problema de los filósofos comensales. ¿Cuál será su evaluación de estas soluciones (**2 puntos** + **3 puntos**)? Con detalles, por favor, y recordando la solución vista en clase.

Solución 1:

```
1     semaphore fork[5] = {1};
2     int i;

3     void Ph(int i)
4     {
5         while (TRUE) {
6             think();
7             wait(fork[i]);
8             wait(fork[(i+1) mod 5]);
9             eat();
10            signal(fork[(i+1) mod 5]);
11            signal(fork[i]);
12        }
13    }

14    void main()
15    {
16        parbegin(Ph(0),Ph(1),Ph(2),Ph(3),Ph(4));
17    }
```

Solución 2:

```
1     semaphore fork[5] = {1};
2     semaphore room = {4};
3     int i;

4     void Ph(int i)
5     {
6         while (TRUE) {
7             think();
8             wait(room);
9             wait(fork[i]);
10            wait(fork[(i+1) mod 5]);
11            eat();
12            signal(fork[(i+1) mod 5]);
13            signal(fork[i]);
14            signal(room);
15        }
16    }

17    void main()
18    {
19        parbegin(Ph(0),Ph(1),Ph(2),Ph(3),Ph(4));
20    }
```

Pregunta 4. (**5 puntos – 25 min.**) Aquí está la solución del mismo problema pero con un monitor:

```
1     monitor dining_controller {
2         cond forkReady[5];
3         boolean fork[5] = {TRUE};

4         void getForks(int pid)
5         {
6             int l = pid;
7             int r = (++pid) % 5;

8             if (!fork[l]) cwait(forkReady[l]);
9             fork[l] = FALSE;
10            if (!fork[r]) cwait(forkReady[r]);
11            fork[r] = FALSE;
12        }

13        void releaseForks(int pid)
14        {
15            int l = pid;
16            int r = (++pid) % 5;

17            if (empty(forkReady[l])) fork[l] = TRUE;
18            else csignal(forkReady[l]);
19            if (empty(forkReady[r])) fork[r] = TRUE;
20            else csignal(forkReady[r]);
21        }
22    }
```

```
23      void philosopher [k=0 to 4]
24      {
25          while(TRUE) {
26              think();
27              getForks(k);
28              eat();
29              releaseForks(k);
30          }
31      }
```

¿Cuándo es posible *deadlock*? Si no sucede *deadlock*, ¿en qué orden comerán todos y cada uno de los filósofos si todos los filósofos ejecutan la línea 27 al mismo tiempo pero en orden 0, 1, 2, 3 y 4? También considere que de todos el filósofo 0 es el más lento para comer, el filósofo 1 es más rápido que el filósofo 0, el 2 es más rápido que el 1, etc., y el filósofo 4 es el más rápido de todos. Para todos los filósofos el tiempo para meditar es mucho más largo que el tiempo para comer.

La práctica ha sido preparada por VK
con LibreOffice Writer en Linux Mint 18.2 Sonya.

Profesor del curso: (0781) V. Khlebnikov

Pando, 29 de septiembre de 2017