CC41B Sistemas Operativos
Control 1 – Semestre Primavera 2005
Prof.: Luis Mateu

Pregunta 1

La federación de tenis de mesa de Chile organiza un torneo en donde todos los jugadores se enfrentan con todos. En el torneo participan 8 jugadores, numerados de 0 a 7, que se enfrentan en 7 rondas de 4 partidos. Para disminuir la duración del torneo, se requiere que los partidos se jueguen en paralelo, sujeto a que se dispone de 3 mesas identificadas como A, B y C. Para hacer enfrentarse los jugadores x e y en la mesa `m` Ud. dispone del procedimiento `enfrentar(x, y, m)`. Este procedimiento toma mucho tiempo y solo retorna cuando el partido finalizó. Algunos partidos pueden ser muy cortos (3 sets) mientras otros pueden ser muy extensos (5 sets muy disputados).

Se ha programado el siguiente procedimiento para realizar el torneo pedido:

```
struct { int x, y; } partidos[4*7]= /* 7 fechas, 4 partidos por ronda */
  { {0,4}, {1,5}, {2,6}, {3,7}, /* 0 enfrenta a 4, 1 a 5, etc. por la 1era. ronda, */
    {0,1}, {2,4}, {3,5}, {7,6}, /* 0 enfrenta a 1, 2 a 4, etc. por la 2da. ronda, */
    {0,2}, {3,1}, {7,4}, {6,5}, ... }; /* 3era. ronda ... y así hasta la 7ma. */
int prox; /* índice del próximo partido que se debe jugar */
```

```
void torneo() {
  prox= 0;
  nTask m1= nEmitTask(mesa, "A");
  nTask m2= nEmitTask(mesa, "B");
  nTask m3= nEmitTask(mesa, "C");
  nWaitTask(m1);
  nWaitTask(m2);
  nWaitTask(m3);
}
```

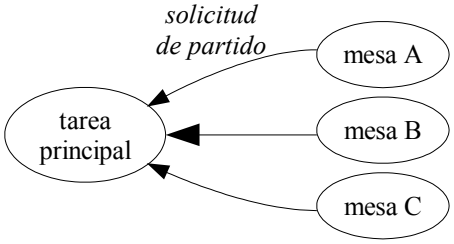
```
int mesa(char *m) {
  while (prox<4*7) {
    int x= partidos[prox].x;
    int y= partidos[prox].y;
    prox++;
    enfrentar(x, y, m);
  }
  return 0;
}
```

(a) (1.5 puntos) Para demostrar que esta solución es incorrecta, haga diagramas de threads que muestren que ella:
   i. podría ordenar que el jugador 0 se enfrente con el jugador 4 en las mesas A y B simultáneamente y no jugarse el partido entre los jugadores 1 y 5.
   ii. podría ordenar que el jugador 0 se enfrente con el jugador 1 por la segunda ronda cuando el jugador 0 todavía está jugando con 4 por la primera ronda en otra mesa.

(b) (4.5 puntos) Modifique esta solución para que funcione correctamente. Considere la siguientes restricciones:
- Si en una mesa se debe comenzar un partido, pero alguno de los jugadores todavía está jugando un partido previo en otra mesa, la mesa debe *esperar* hasta que ese partido concluya.
- Ud. debe usar como única herramienta de sincronización los monitores de nSystem.
- El procedimiento torneo retorna sólo una vez que todos los partidos finalizaron.

Pregunta 2

Resuelva el problema anterior utilizando como *única* herramienta de sincronización los mensajes de nSystem. Utilice la organización de tareas que se indica en la figura de más abajo. Cada mesa es una tarea que solicita partidos a la tarea principal. Para ello la mesa envía un mensaje a la tarea principal, la que responde con el partido que se debe jugar. Entonces, la mesa realiza el enfrentamiento. Al finalizar, la mesa envía un nuevo mensaje a la tarea principal indicando qué partido se jugó. La tarea principal responderá este mensaje con el nuevo partido que se debe jugar a continuación (-1 si no quedan partidos para jugar), etc.



Restricción: si una mesa está disponible para jugar un nuevo partido, pero el próximo partido en el arreglo no se puede jugar porque uno de los jugadores todavía está jugando en otra mesa, Ud. debe buscar algún otro partido que sí se pueda jugar y entregárselo a la mesa. Como en la pregunta anterior, `torneo` solo retorna cuando todos los partidos finalizaron.

Observación: Si por alguna condición de borde en su solución se pierde un poco de paralelismo hacia el final del torneo explique por qué, pero no resuelva el problema.