CC4302 – Sistemas Operativos
Auxiliar 5

Profesor: Luis Mateu
Auxiliar: Diego Madariaga

22 de abril de 2020

1 P1 Control 1, 2013/2

Un estadio posee un único baño que debe ser compartido por hinchas rojos e hinchas verdes. El baño es amplio y admite un número ilimitado de personas. El problema consiste en evitar que los hinchas rojos se encuentren con los hinchas verdes dentro del baño. Los hinchas rojos solicitan entrar al baño invocando `entrarRojo` y notifican su salida con `salirRojo`, mientras que los hinchas verdes invocan `entrarVerde` y `salirVerde`. Se plantea la siguiente solución incorrecta para este problema:

```
int mutex = FALSE;
int rojos = 0, verdes = 0;


void entrarRojo() {
  if (rojos == 0) {
    while (mutex)
      ;
    mutex = TRUE;
  }
  rojos++;
}
void salirRojo() {
  rojos--;
  if (rojos == 0)
    mutex = FALSE;
}
```

```
void entrarVerde() {
  if (verdes == 0) {
    while (mutex)
      ;
    mutex = TRUE;
  }
  verdes++;
}
void salirVerde() {
  verdes--;
  if (verdes == 0)
    mutex = FALSE;
}
```

1. Muestre mediante un *diagrama de threads* que un hincha rojo puede entrar al baño cuando hay hinchas verdes presentes. Detalle bien las invocaciones de procedimientos, los instantes en que se hacen los `if`, se incrementan variables, etc.

2. Escriba una *solución correcta* y *eficiente* para este problema utilizando 3 semáforos de nSystem (hint: utilice la estructura de la solución incorrecta). No importa que en su solución algunos procesos sufran "hambruna".

3. Suponga que se agrega una nueva restricción: el baño tiene capacidad solo para 4 personas. Reescriba su solución para evitar que en algún momento hayan más de 4 personas en el baño. Use un semáforo adicional.

2 P1 Control 1, 2006/2

1. Se propone la siguiente implementación para un semáforo:

```
int sc = 0, wc= 0;
void signal() {
  sc++;
}
```

```
void wait() {
  while (wc <= sc)
    ;
  wc++;
}
```

   Para los siguientes casos, diga cuándo esta implementación es correcta o muestre mediante un diagrama de threads que es incorrecta:

   - Un solo thread invoca `signal` y otro thread invoca `wait`.
   - Un proceso pesado invoca `signal` y otro proceso pesado invoca `wait`.
   - Varios threads pueden invocar tanto `signal` como `wait`.

2. Implemente el semáforo de la parte (1) usando los mensajes de nSystem.

3 P2 Control 1, 2006/2

La ATP Tour ha organizado un tipo de torneo de tenis que se caracteriza por durar un solo día. Para ello los partidos son mucho más cortos, con el objeto de jugar un cuadro clásico de tenis en una sola jornada. Actualmente existe una solución secuencial escrita en nSystem:

```
char *torneo(char **players, int n) {
  char **draw = (char**) nMalloc(2*n*sizeof(char*));
  int k, r, i;

  for (k = 0; k < n; k++)
    draw[n+k] = players[k];

  for (r = n; r > 1; r = r/2) {
    /* ronda de r jugadores */
    for (i = r/2; i < r; i++)
      draw[i] = play(draw[2*i], draw[2*i+1]); /* muy lento */
  }

  return draw[1]; /* entrega el ganador del torneo */
}
```

En donde `players` es un arreglo con los nombres de los `n` jugadores del torneo, donde `n` es una potencia de 2. El procedimiento `play` es dado y toma mucho tiempo en enfrentar 2 jugadores, cuyos nombres se reciben como parámetros. El procedimiento `torneo` retorna el nombre del ganador. El problema es que como se juega un solo partido a la vez, el torneo termina usualmente en la madrugada.

1. Suponiendo que hay suficientes canchas disponibles, reescriba el procedimiento `torneo` de tal forma que se jueguen los partidos simultáneamente. En caso de requerir herramientas de sincronización use los monitores de nSystem. Ud. debe ser eficiente: si el ganador de Moya - Massu se debe enfrentar con el ganador de Gonzalez - Federer, el partido debe comenzar de inmediato una vez que terminen ambos partidos, ¡y no antes! (es decir que el partido entre `draw[2*k]` y `draw[2*k+1]` debe comenzar en cuanto se conozcan los nombres que van en esas dos posiciones del draw).

2. Suponga ahora que hay solo 4 canchas disponibles, numeradas de 1 a 4. Modifique `torneo` de modo que ahora se utilice `char *play(char *jug1, char *jug2, int cancha)`, que indica en qué cancha se jugará el partido. Desde luego, Ud. no puede usar una misma cancha para dos partidos simultáneamente. Para la sincronización Ud. debe usar los monitores de nSystem.