CC41B - Auxiliar 2

Profesor: Luis Mateu

Auxiliares: Hernán Arroyo, Juan Manuel Barrios

Control 1, 2006, pregunta 1, parte a

Se propone la siguiente implementación para un semáforo:

```
int sc= 0, wc= 0;
```

```
void signal(){
  sc++;
}
```

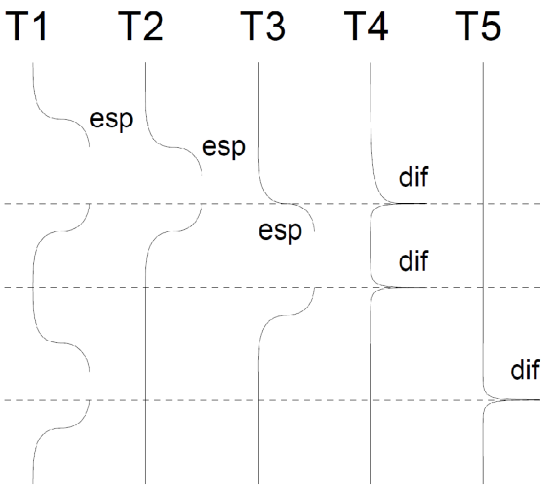
```
void wait() {
  while (wc<=sc)
    ;
  wc++;
}
```

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

1) Un solo thread invoca `signal` y otro thread invoca `wait`.
2) Un proceso pesado invoca `signal` y otro proceso pesado invoca `wait`.
3) Varios threads pueden invocar tanto `signal` como `wait`.

Control 1, 2008, pregunta 1

Para los procedimientos `esperar` y `difundir` se especifica que cuando cualquier thread invoca `esperar`, ese thread debe quedar bloqueado hasta que algún thread invoque `difundir`, retornando la información suministrada por `difundir`. Si la invocación de `esperar` ocurre simultáneamente con la invocación de `difundir` (es decir, si hay algún traslape en las dos invocaciones), el thread puede continuar de inmediato o bien puede bloquearse hasta la siguiente invocación de `difundir`. Si ocurren 2 invocaciones simultáneas de `difundir`, esperar puede retornar la información suministrada en cualquiera de esas 2 invocaciones de `difundir`.
Se propone la siguiente implementación:




```
int cont= 0;
Info *info;
void difundir(Info *infoP) {
  cont++;
  info= infoP;
}
```

```
Info *esperar() {
  int micont= cont;
  while (micont==cont)
    ;
  return info;
}
```

1) Haga un diagrama de threads que muestre que la solución propuesta es inconsistente. Por ejemplo un thread podría retornar información equivocada. (Considere que nunca se produce el desborde de la variable `cont`.)
2) Indique si es posible corregir esta solución haciendo una pequeña modificación de forma tal que ya no se produzcan inconsistencias.
3) Critique esta solución desde el punto de vista de la eficiencia.
4) Escriba una solución consistente y *eficiente* de `esperar` y `difundir` usando como herramienta de sincronización los semáforos de nSystem. Considere que estos semáforos garantizan que los nWaitSem serán atendidos en orden FIFO.

Control 1, 2007, pregunta 1

```
typedef struct Nodo {
  char *llave, *def;
  struct Node *izq, *der;
  nSem semIzq, semDer;
} Nodo;
typedef struct {
  Nodo *raiz;
  nSem semRaiz;
} ConcDict;
void addDef(ConcDict *dict,
     char *llave, char *def)
{
  insertar(dict->semRaiz,
           &dict->raiz,
           llave, def);
}
```

```
void insertar(nSem sem, Nodo **ppnodo,
              char *llave; char *def) {
  Nodo *pnodo;
  if (*ppnodo==NULL) {
    *ppnodo= crearNodoHoja(llave,
                           def); /* dado */
  }
  else { /*** Supuesto: la llave ***
         *** no está en el arbol ***/
    pnodo= *ppnodo;
    if (strcmp(llave, pnodo->llave)<0) {
      insertar(pnodo->semIzq,
               &pnodo->izq, llave, def);
    }
  else {
    insertar(pnodo->semDer,
             &pnodo->der, llave, def);
} } }
```

El programa anterior es la implementación incompleta de un diccionario concurrente en base a un árbol de búsqueda binaria. El procedimiento `crearNodoHoja` es dado e inicializa correctamente todos los campos. Esto incluye la creación de los semáforos que contiene el nodo, cada uno con un ticket. Cuando se crea el diccionario, se crea automáticamente el semáforo para la raíz en la estructura `ConcDict`, aún cuando el diccionario esté vacío. Suponga que no existe una operación para eliminar definiciones en el diccionario.

1) Haga un diagrama de threads que muestre que el diccionario puede quedar en un estado inconsistente cuando 2 threads llaman a `addDef` concurrentemente. (2 puntos.)
2) Complete el programa de más arriba agregando las líneas que faltan. Ud. necesita garantizar la exclusión mutua cuando 2 threads están trabajando en el mismo nodo, pero Ud. *debe* permitir que continúen en paralelo una vez que trabajan en ramas distintas del árbol. El código dado es correcto, sólo agregue las líneas que faltan para la sincronización. La declaración de los semáforos y su paso como parámetros es una ayuda para que resuelva el problema. (4 puntos.)

Importante: Implementar la exclusión mutua a nivel del árbol completo tiene 0 puntos.

Propuesto: implementar la búsqueda en paralelo.