Clase Auxiliar VIII

Prof: L. Mateu
Aux: M. Leyton

1 de octubre de 2004

1. Semáforos con Timemout

Implemente los Super Semáforos en `nSystem`. Los Super Semáforos permiten manejar *timeouts*. La API de estos semáforos es la siguiene:

- `nSuperSem nMakeSuperSem(int count);`
- `void nWaitSuperSem(nSuperSem sem, int timeout);`
- `void nSignalSuperSem(nSuperSem sem);`
- `void nDestroySuperSem(nSuperSem sem);`

2. Ejercicio: nExchange

El método *nExchange* permite intercambiar enteros entre dos tareas. Para que el intercambio funcione, las dos tareas deben invocar a *nExchange* respectivamente. El API del método es el siguiente:

- `int nExchange(nTask task, int value);`

El valor retornado por este método corresponderá al `value` enviado por `task`. Si `task` aún no ha invocado a *nExchange*, con mi tarea, entonces este método se debe quedar bloqueado hasta que `task` invoque a *nExchange*.

Implemente este método utilizando los procedimientos de bajo nivel de `nSystem`. Modifique el descriptor `nTask` en caso de ser necesario.

3. Solución Super Semáforos

```
#define MAGICFALSE (0x2367a2f1)
#define MAGICTRUE  (0x78a214f5)

typedef struct nSuperSem {
int count;
FifoQueue queue;
} *nSuperSem;


nSuperSem nMakeSuperSem(int count)
{
  nSuperSem sem;

  sem= (nSuperSem) nMalloc(sizeof(*sem));
  sem->count= count;
  sem->queue= MakeFifoQueue();

  return sem;
}

int nWaitSuperSem(nSuperSem sem, int timeout)
{
  int success= TRUE;

  START_CRITICAL();

    if (sem->count>0)
      sem->count--;
    else if (timeout==0)
      success= FALSE;
    else if (timeout==-1)
    {
      current_task->status= WAIT_SUPERSEM;
      PutObj(sem->queue, current_task);
      ResumeNextReadyTask();
    }
    else
    {
      current_task->status= WAIT_SUPERSEM_TIMEOUT;
      current_task->rc= MAGICFALSE; /* success? */
      ProgramTask(timeout);

      PutObj(sem->queue, current_task);
      ResumeNextReadyTask();
      if (current_task->rc!=MAGICTRUE && current_task->rc!=MAGICFALSE)
        nFatalError("nWaitSuperSem", "Inconsistencia, rc incorrecto\n");
      success= current_task->rc==MAGICTRUE;
        /* nSignalSuperSem lo puede colocar en MAGICTRUE */
      if (!success) DeleteObj(sem->queue, current_task);
      else if (QueryObj(sem->queue, current_task))
        nFatalError("nWaitSuperSem",
                    "Inconsistencia, la tarea todavia esta en la cola\n");
    }

  END_CRITICAL();
```

```
  return success;
}

void nSignalSuperSem(nSuperSem sem)
{
  nTask wait_task;

  START_CRITICAL();

    do
    {
      wait_task= (nTask)GetObj(sem->queue);
    }
      while (wait_task!=NULL && wait_task->status==READY);

    if (wait_task==NULL)
      sem->count++;
    else
    {
       if (wait_task->status==WAIT_SUPERSEM_TIMEOUT) CancelTask(wait_task);
       wait_task->status= READY;
       wait_task->rc= MAGICTRUE; /* success= TRUE */
       PushTask(ready_queue, current_task); /* Sigue estando ready */
       PushTask(ready_queue, wait_task);
       ResumeNextReadyTask(); /* wait_task es la primera en la cola! */
    }

  END_CRITICAL();
}

void nDestroySuperSem(nSuperSem sem)
{
  if (! EmptyFifoQueue(sem->queue) )
    nFatalError("nDestroySem",
      "Se intenta destruir un semaforo con tareas pendientes\n");
  DestroyFifoQueue(sem->queue);
  nFree(sem);
}
```

4. Solución nExchange

```
struct nTask{
    ...
    exchange_task;
    exchange_value;
} *nTask;


int nExchange(nTask task, int value){

    int rvalue;

    START_CRITICAL();
    {
        nTask this_task=current_task;
```

```
            /* Si task ya invoco nExchange conmigo */
            if(task->status = WAIT_EXCHANGE &&
                task->exchange_task==current_task){

                /* Extraigo el valor enviado por task */
                rvalue=task->exchange_value;
                /* (**)Guardo en task el valor que yo envio */
                task->exchange_value=value;

                /* Coloco a task en la cola Ready */
                task->status=READY;
                PushTask(readu_queue, this_task);
                PushTask(ready_queue, task);
                ResumeNextReadyTask();
            }
            /* task aun no invoca nExchange conmigo*/
            else{
                /* Guardo el task y value en mi descriptor*/
                this_task->exchange_task=task;
                this_task->exchange_value=value;

                /*  Cambio mi estado y sigo con la siguiente ejecucion */
                this_task->status=WAIT_EXCHANGE;
                ResumeNextReadyTask();

                /* Extraigo el valor guardado por task en (**) */
                rvalue=this_task->exchange_value;
            }
        }

        END_CRITICAL();

        return  rvalue;
    }
```