CC41B Sistemas Operativos

Control 2 – Semestre Primavera 2004

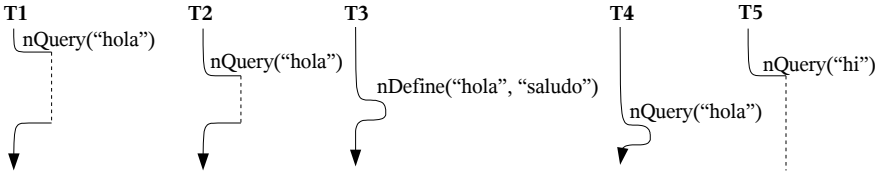
Prof.: Luis Mateu

Pregunta 1

La compañía TODODICC le pide a Ud. que elabore una nueva versión de nSystem en donde se reemplacen todas las herramientas de sincronización por un único diccionario compartido por todas las tareas. Dos procedimientos permiten manipular este diccionario:

- **`void nDefine(char *key, void *val)`**: asocia a la llave *key* la definición *val* en el diccionario global. Si *val* es NULL, se elimina la llave del diccionario.
- v**`oid *nQuery(char *key)`**: entrega la definición asociada a la llave *key*. Si la llave no existe en el diccionario, este procedimiento se bloquea hasta que alguna tarea defina esa llave.

Ejemplo:



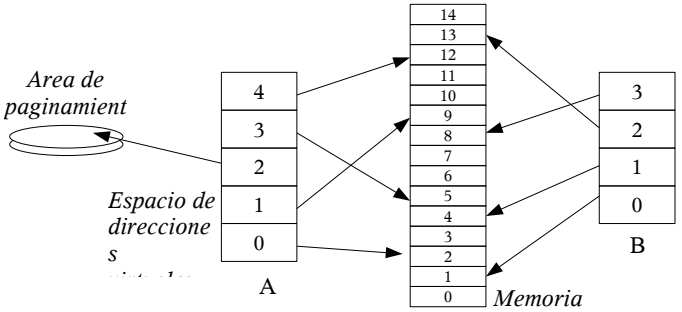
Implemente **`nDefine`** y **`nQuery`** en nSystem usando los procedimientos de bajo nivel (START_CRITICAL, Resume, PutTask, etc.). Ud. no debe basar su implementación en las herramientas de sincronización que TODODICC desea eliminar (semáforos, mensajes, monitores, etc.). Utilice el tipo de datos *Dict* para programas secuenciales:

- **`Dict makeDict()`**: crea un diccionario para programas secuenciales.
- **`void define(Dict d, char *key, void *val`**): asocia a la llave *key* la definición *val* en el diccionario *d*. Si *val* es NULL, se elimina la llave del diccionario.
- **`void *query(Dict d, char *key)`**: entrega la definición asociada a *key* en el diccionario *d*. Si la llave no existe se retorna NULL.

Pregunta 2

i. Explique por qué en un núcleo de sistema operativo *multi-threaded* (i.e. para multiprocesadores) la única herramienta de sincronización que puede ser usada para manipular la cola de procesos *ready* es el *spin-lock*. ¿Por qué no es posible usar un semáforo o un lock normal?

ii. El diagrama muestra la asignación de páginas en un sistema Unix que ejecuta los procesos A y B. Las páginas son de 4 KB. Suponga que el proceso A invoca *sbrk* pidiendo 8 KB adicionales.



- Haga un nuevo diagrama para la asignación de páginas justo después de invocar *sbrk*.
- Incluya en su diagrama la tabla de páginas para el proceso A indicando los atributos de cada página.

iii. Considerando el mismo diagrama del punto anterior (sin considerar *sbrk*), suponga que el proceso B invoca *fork* y que el núcleo utiliza la estrategia *copy-on-write* para implementar *fork*:

- Haga un nuevo diagrama para la asignación de páginas después de invocar *fork* y luego de que el proceso B modificó la página 2.
- Incluya en su diagrama la tabla de páginas para el padre y el hijo indicando los atributos de cada página.

iv. El siguiente diagrama muestra (con una x) los accesos a la memoria para un proceso que se ejecuta en un sistema Unix que utiliza la estrategia del *working set*.

	A	B	C	D	E	F	G
6	x	x	x	x		x	
5	x			x			
4	x	x			x	x	x
3	x	x		x	x	x	
2	x		x		x	x	
1					x		
0							

La letras A, B, C, etc. denotan los intervalos en los que se calcula el working set. Los números 0, 1, 2, etc. denotan las páginas del proceso.

- Indique para los períodos C a F qué accesos pueden producir page-faults (utilice coordenadas del estilo *(A, 4)*).
- Indique el valor del atributo *Referencia* para todas las páginas al inicio del intervalo E y al final de ese intervalo.
- Considere que el computador en donde se ejecuta este programa posee solo 5 páginas reales. Expliqué en qué momento la estrategia del working set no funcionará adecuadamente.