CC4302 Sistemas Operativos
Examen– Semestre Otoño 2012
Prof.: Luis Mateu

Pregunta 1

La función de abajo se encarga de llevar a cabo un torneo de squash. Recibe en `players` un arreglo con los nombres de los `n` jugadores que participan en el torneo y entrega el nombre del ganador del torneo.

```
char *torneo(char **players, int n) { // n: potencia de 2
  char **draw= (char**)nMalloc(n*2*sizeof(char*));
  int k;               // Ojo: draw[0] no se usa
  for (k= 0; k<n; k++) {
    draw[n+k]= players[k];
  }
  for (k= n-1; k>=1; k--) {
    draw[k]= play(draw[2*k], draw[2*k+1]); /* muy lento */
  }
  return draw[1]; /* entrega el ganador del torneo */
}
```

El torneo se resuelve en `n-1` partidos. El jugador que pierde un partido queda eliminado. La función torneo realiza un partido invocando `play`, que recibe los nombres de los 2 jugadores que se enfrentan y entrega el nombre del ganador. `Play` *es dado* y toma mucho tiempo en ejecutarse. Esta solución es ineficiente porque ejecuta secuencialmente los `n-1` partidos.

i. Suponiendo que hay suficientes canchas disponibles, se le pide a Ud. que reescriba la función `torneo` de tal forma que se jueguen los partidos simultáneamente. Use las tareas y monitores de nSystem. Ud. debe ser eficiente: 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*.

ii. Suponga ahora que hay solo 4 canchas disponibles, numeradas de 1 a 4. Reescriba la función `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.

Pregunta 2

Se desea agregar a nSystem la siguiente API para manejar locks de lectura/escritura:

Procedimiento	***Significado***
`void nEnterRead(nRWLock* l);`	Solicitud para leer
`void nExitRead(nRWLock* l);`	Notificación de salida de lectura
`void nEnterWrite(nRWLock* l);`	Solicitud para escribir
`void nExitWrite(nRWLock* l);`	Notificación de salida de escritura

Estos locks son una solución para el problema de los lectores/escritores visto en clases.

Implemente esta API, es decir la estructura nRWLock y los procedimientos nEnterRead, nExitRead, nEnterWrite y nExitWrite, como mecanismo de sincronización nativo de nSystem. Esto significa que Ud. debe implementar esta API usando los procedimientos de bajo nivel de nSystem (START_CRITICAL, Resume, PutTask, etc.). Ud. *no puede usar* otros mecanismos de sincronización ya disponibles en nSystem, como semáforos, monitores, mensajes, etc.

Requerimiento de simplicidad: su implementación *debe* dar prioridad a los escritores aún cuando esto signifique una eventual hambruna para los lectores.

Pregunta 3

i. Se tiene un archivo que no requiere bloques de indirección doble en una partición Unix con bloques de 1 KB. Se agrega un byte a este archivo y se crea el bloque de indirección doble. Haga un diagrama mostrando inodo, bloques de datos y de indirección. ¿De qué tamaño es el archivo?

ii. Ud. encuentra 2 archivos con nombres 1 y 2 en el directorio `lost+found` de una partición. Explique por qué están ahí y el porqué de los nombres 1 y 2.

iii. Considere un sistema Unix que se acaba de encender y por lo tanto no hay nada en el caché de disco en el núcleo. Estime cuantos accesos a disco se necesitan para leer el primer bloque del archivo "`/home/jgonzalez/notas.txt`". En su conteo considere inodos, directorios y bloques de datos. Haga un dibujo para explicar su conteo.