CC41B Sistemas Operativos

Examen – Semestre Otoño 2010

Prof.: Luis Mateu

Pregunta 1

Se necesita buscar un dato `d` en un conjunto de `NLIB` libros. La información no se almacena en un orden específico por lo que en el peor caso habría que consultar todos los libros. Para acelerar la búsqueda se dispone de `NEMP` empleados que pueden trabajar simultáneamente. La siguiente implementación sirve como referencia de la funcionalidad pedida, pero ocupa un solo empleado:

```
Empleado *empls[];
Libro *libros[];

int buscar(Dato *d) {
  int i;
  for (i= 0; i< NLIB; i++) {
    /* Solo ocupa el primer empleado en empls */
    if (consultar(empls[0], libros[i], d)) {
      return TRUE; /* dato encontrado! */
    }
  }
  return FALSE; /* dato no encontrado */
}
```

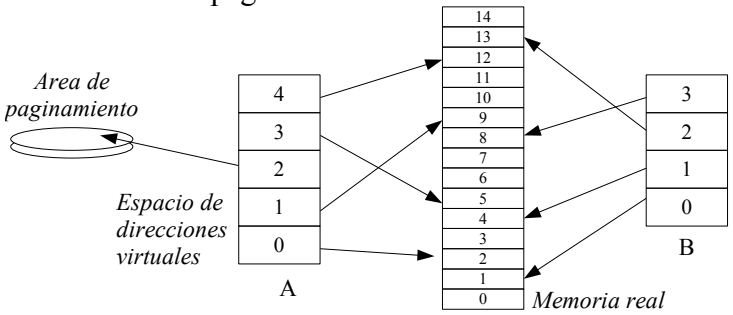
El procedimiento `consultar(e, l, d)` ordena al empleado `e` que busque `d` en el libro `l` y sólo retorna una vez que el empleado tiene la respuesta, lo cual puede tomar bastante tiempo. `Consultar` y las estructuras `Empleado` y `Libro` son dados. El procedimiento `buscar` es invocado varias veces, pero desde un único *thread*.

Se le pide a Ud. reescribir el procedimiento `buscar` de manera que los `NEMP` empleados trabajen en paralelo para buscar `d`. Restricciones:

- Un libro no puede ser consultado simultáneamente por 2 empleados.
- Un empleado puede consultar un libro a la vez.
- Ningún empleado podrá iniciar una consulta a partir del momento en que otro empleado encuentra el dato buscado. Sin embargo Ud. sí debe esperar que las consultas ya iniciadas terminen.
- Un empleado no debe permanecer ocioso si hay un libro que puede ser consultado (si no interfiere con el punto anterior).
- Resuelva el problema de sincronización usando los monitores de nSystem, asegurándose con `nWaitTask` que todas las tareas terminaron adecuadamente antes de que `buscar` retorne.

Pregunta 2

a) 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 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.



b) Construya la tabla de páginas para el padre y el hijo indicando los atributos de cada página (bits de validez y escritura).

c) Suponga que el hardware de un computador no implementa el bit *Dirty* (D). Explique si sería posible implementar estrategias de remplazo sin este bit. ¿Y si el hardware no implementara el bit de *referencia* (R)?

d) En un sistema Unix todos los procesos avanzan extremadamente lento. El uso de CPU es apenas un 1% y el disco de paginamiento funciona al 100%. ¿Qué estrategia de remplazo de páginas usa este sistema? Explique. ¿Qué estrategia sería más recomendable para esta situación?

Pregunta 3

i. Un sistema posee 1024 archivos de 512 bytes c/u cuya creación fue muy distante en el tiempo. Estime cuanto tiempo total (¡en segundos!) costaría buscar la palabra "hello" en todos estos archivos. ¿Y cuanto costaría buscar la misma palabra en un solo archivo de 512 KB, que fue creado en una sola sesión? Explique sus supuestos considerando un disco moderno.

ii. Considere un sistema Unix que se acaba de encender y por lo tanto no hay nada en el cache 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.

iii. Suponga que un programa invoca lseek(N) y luego lee un solo bloque de datos (de 1024 bytes). Explique con un dibujo para qué rango de valores de N se deberá leer adicionalmente *un y solo un* bloque de indirección simple.