CC4302 Sistemas Operativos – Examen – Semestre Otoño 2022 – Prof.: Luis Mateu

Pregunta 1

Parte a.- (2 puntos) La función *f* de más abajo invoca las funciones *g* y *h*. Las funciones *g* y *h* son dadas y toman mucho tiempo en calcularse.

```
double g(double **mat, double x);
double h(double **mat, double x);
double f(double **mat, double x) {
  return g(mat, x) + h(mat, x);
}
```

Reprograme la función *f* de manera que el cálculo de *g* y *h* se haga en paralelo. Para ello Ud. debe crear un nuevo thread (de pthreads) que calcule *h(mat, x)*. El thread original calcula *g(mat, x)* y retorna el resultado final. **¡Revise que su solución posee paralelismo!**

Parte b.- (4 puntos) El tipo *Portero* sirve para evitar que entren a una tienda más de *max* personas simultáneamente. Se le pide que defina el tipo *Portero* y programe las siguientes funciones:

- *void iniPortero(Portero *p, int max)*: Inicializa el portero *p* que permite que ingresen simultáneamente hasta *max* threads.
- *void entrar(Portero *p)*: Solicita al portero *p* ingresar a la tienda. Si ya hay *max* personas en el interior debe esperar.
- *void salir(Portero *p)*: Notifica al portero *p* la salida de la tienda. Si hay threads en espera se hace pasar al thread que lleva más tiempo esperando.

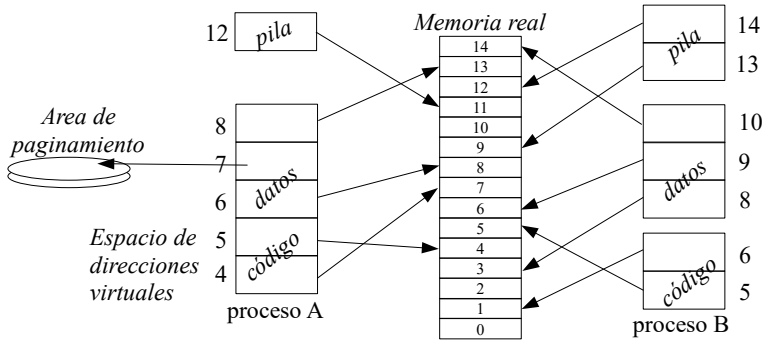
Para resolver este problema **Ud. debe usar los mutex y condiciones de pthreads**. No puede usar otras herramientas de sincronización como semáforos. Además su solución debe evitar cambios de contexto inútiles como que un thread se despierte solo para darse cuenta que debe continuar esperando, y por lo tanto debe usar el patrón *request*.

Pregunta 2

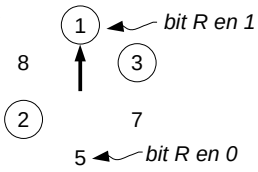
Resuelva nuevamente el problema del portero pero esta vez considerando una máquina multicore sin sistema operativo. **Por lo tanto la única herramienta de sincronización disponible es el spin-lock.** Ud. debe definir el tipo *Portero* y reprogramar las mismas funciones de la parte *b* de la pregunta 1. Para hacer esperar a los cores use una cola (tipo *Queue*) de spin-locks. Debe respetar el orden de llegada.

Pregunta 3

I. El diagrama de más abajo muestra la asignación de páginas en un sistema Unix que ejecuta los procesos A y B. Las páginas son de 4 KB. El núcleo utiliza la estrategia *copy-on-write* para implementar *fork*. (a) Construya la tabla de páginas del proceso A después de que este invoca *sbrk* pidiendo 6 KB adicionales. (b) A continuación el proceso B invocó *fork*. Construya la tabla de páginas para el proceso hijo justo después de que este modificó la página virtual 9. No construya la tabla del padre. En las tablas indique página virtual, página real y atributos de validez y escritura.



II. Considere un sistema Unix que implementa la estrategia del reloj. El sistema posee 6 páginas reales disponibles y corre un solo proceso. La figura indica el estado inicial de la memoria, mostrando las páginas residentes en memoria, la posición del cursor y el valor del bit R.



Dibuje los estados por los que pasa la memoria para la siguiente traza de accesos a páginas virtuales: 2, 5, 4, 5, 8, 6.

III. Haga un diagrama que muestre el inodo, los bloques de indirección y los bloques de datos de un archivo de 4152 KB (1038 * 4KB), ubicado en una partición Unix con bloques de 4 KB.

IV. Estime cuántos page faults por segundo se pueden atender cuando el área de paginamiento está (a) en un SSD y (b) en un disco duro. (c) Explique cuál sería la principal desventaja si el área de paginamiento está en un SSD y no en un disco duro.

V. 5 procesos se encuentra en estado de espera porque hicieron peticiones para leer bloques del disco en el siguiente orden: 900, 100, 800, 600, 700. El último bloque leído fue el 300 y el penúltimo el 200. Indique en qué orden se harán las lecturas de estos 5 procesos cuando la estrategia de scheduling de disco es: (a) *shortest seek first*, (b) método del ascensor (o *look*).