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

Pregunta 1

Parte a.- La función *palindro* del cuadro de la derecha entrega 1 si un string *s* de *n* caracteres es palíndrome, es decir que al invertirlo se lee el mismo string. En caso contrario entrega 0. Paralelice la función *palindro* de manera que se ejecute la primera mitad del ciclo en un nuevo thread (de pthreads) y la segunda mitad en el thread original. **¡Revise que su solución posee paralelismo!**

```
int palindro(char *a,
             int n) {
  int i= 0;
  while (i<n/2) {
    if(a[i]!=a[n-1-i])
      return 0;
    i++;
  }
  return 1;
}
```

Parte b.- Programe las siguientes funciones:

void ocupar(int pri);
void desocupar( );

La función *ocupar* solicita el uso exclusivo de un recurso compartido con prioridad *pri*, que puede ser 1 o 0. Solo un thread puede ocupar el recurso en un instante dado. Si el recurso está ocupado, se espera. La función *desocupar* notifica que el recurso fue desocupado. Si en ese momento hay threads en espera que lo solicitaron con prioridad 1, se otorga a cualquiera de ellos. De lo contrario, se otorga a cualquiera que lo haya solicitado con prioridad 0.

Restricción: Para resolver este problema **Ud. debe usar un mutex y una sola condición de pthreads**. No puede usar otras herramientas de sincronización, como por ejemplo semáforos.

Pregunta 2

Parte i.- Resuelva nuevamente el problema de la parte *b* de la pregunta 1 pero esta vez considerando una máquina multicore sin sistema operativo. **Por lo tanto la única herramienta de sincronización disponible es el spin-lock.** En la función *desocupar*, el recurso se debe otorgar al core que lleva más tiempo esperando entre los que lo solicitaron con prioridad 1. Si ningún core en espera lo solicitó con prioridad 1, se otorga al que lleve más tiempo esperando. Para hacer esperar a los cores use una cola (tipo *Queue*) de spin-locks. No hay límite en la cantidad de spin-locks que puede usar.

Parte ii.- Programe las siguientes funciones como herramientas de sincronización nativas de nThreads, es decir usando operaciones como *START_CRITICAL*, *setReady*, *suspend*, *schedule*, *nth_putBack*, etc.:

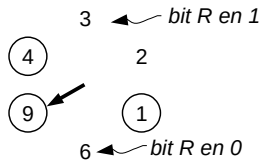
void nOcupar(int pri);
void nDesocupar( );

La funcionalidad debe ser la misma de la parte *a*. Ud. ***no*** puede implementar la API solicitada en términos de otras herramientas de sincronización pre-existentes en nThreads.

Pregunta 3

I. El proceso A posee código en el intervalo de direcciones virtuales [20 KB, 32 KB[, datos en [32 KB, 40 KB] y pila en [56 KB, 64 KB[. El proceso B posee código en [12 KB, 20 KB[, datos en [20 KB, 28KB[y pila en [60 KB, 64 KB[. (a) Confeccione las tablas de páginas de ambos procesos, indicando número de página real y atributos V y W. La máquina posee 128 KB de memoria real organizada en páginas de 4 KB. Decida Ud. la asignación de páginas reales. (b) Explique qué cambia en la tabla de páginas del proceso B cuando desborda su pila. (c) Explique qué cambia en la tabla de páginas del proceso A cuando este invoca *malloc* solicitando 100 bytes de memoria, pero esa cantidad de memoria no está disponible en su área de datos actual.

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: 9, 2, 5, 9, 7, 4.

III. Se tiene un archivo de 397 KB en una partición Unix con bloques de 4 KB. Haga un diagrama mostrando inodo, bloques de datos y de indirección. Conteste además: ¿Cuanto espacio en disco se ocupa realmente para almacenar este archivo?

IV. Considere una partición con bloques de 4 KB en un disco duro tradicional con un tiempo de acceso de 10 milisegundos y una velocidad de 100 MB/seg. Estime cuanto tomaría leer 100 archivos de 4 KB y cuanto tomaría leer un solo archivo de 400 KB.

V. Compare ventajas y desventajas de las 3 estrategias de scheduling de disco vistas en cátedra: (a) *shortest seek first* (b) *LOOK* o método del ascensor (c) *C-LOOK* o método del ascensor circular.

VI. ¿Cómo se garantiza la exclusión mutua al acceder a la cola de procesos *ready* en un núcleo de sistema operativo para un computador single-core? ¿Y si el computador es multi-core?