CC4302 Sistemas Operativos – Examen – Semestre Primavera 2023 – Prof.: Luis Mateu

Pregunta 1

a. (1 punto) ¿Para qué sirven los números *major* y *minor* de un dispositivo?

b. (1 punto) En el problema de la isla, muchos turistas usan sus vehículos para trabajar en el continente y pasear en una isla como lo muestra la función *turistaThread* de más abajo. Para ir desde el continente hacia la isla o viceversa invocan la función *cruzarA* que usa una única barcaza que puede transbordar un solo vehículo a la vez. La barcaza es controlada desde un thread que ejecuta la función *barcazaThread*.

```
typedef enum {CONT=0,ISLA=1} Lado;
Vehiculo *gv[2]= {NULL, NULL};
volatile int *volatile gpw[2];
void *turistaThread(void *ptr) {
  Vehiculo *v= ptr;
  for (;;) {
    trabajar(v);
    cruzarA(ISLA, v);
    pasear(v);
    cruzarA(CONT, v);
  }
  return NULL;
}
// Si lado=ISLA cruza desde CONT hacia ISLA
void cruzarA(Lado destino,
             Vehiculo *v) {
  volatile int w= 0;
  Lado origen= !destino;
  while (gv[origen]!=NULL)
    ; // esperar
  gpw[origen]= &w;
  gv[origen]= v;
  while (w==0)
    ; // esperar
}
```

```
// !CONT == ISLA && !ISLA==CONT
void *barcazaThread(void *ptr) {
  Barcaza *bar= ptr; // la barcaza
  Lado origen= CONT; // en continente
  for (;;) {
    Lado destino= !origen;
    while( gv[origen]==NULL &&
           gv[destino]==NULL)
      ; // esperar (*)
    Vehiculo *v= NULL;
    volatile int *pw= NULL;
    if (gv[origen]!=NULL) {
      v = gv[origen];
      pw = gpw[origen];
      gv[origen]= NULL;
    }
    if (v!=NULL)
      embarcar(bar, v);
    navegarA(bar, destino);
    if (v!=NULL) {
      desembarcar(bar, v);
      *pw= 1;
    }
    origen = destino;
} }
```

Observe que en (*) si no hay ningún vehículo esperando en la orilla en donde se encuentra la barcaza, pero sí hay uno esperando en la orilla opuesta, la barcaza debe navegar vacía a la orilla opuesta. Este código *funciona correctamente* hasta que falla debido a un *datarace*. **Muestre por medio de un diagrama de threads** que un vehículo podría quedar esperando en el continente para siempre.

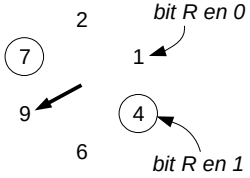
c. (4 puntos) Reescriba las funciones *cruzarA* y *barcazaThread* para que siempre funcionen correctamente. No puede modificar ninguna otra función. Ud. debe usar el patrón *request* y 2 colas para: (i) evitar cambios de contexto inútiles, (ii) que quienes esperan en la isla sean embarcados por orden de invocación de cruzarA(CONT, v) y (iii) que quienes esperan en el continente sean embarcados por orden de invocación de cruzarA(ISLA, v).

Pregunta 2

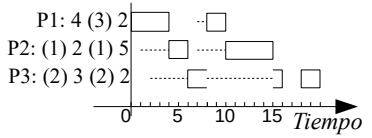
(i) (5 puntos) Programe las funciones *nCruzar* y *nBarcazaThread* como funciones nativas de nThreads con los mismos parámetros y funcionalidad de *cruzarA* y *barcazaThread* de la parte c de la pregunta 1. Ud. debe resolver este problema usando los procedimientos de bajo nivel de nThreads (*START_CRITICAL*, *END_CRITICAL*, *schedule*, *setReady*, *suspend*, *nth_putBack*, etc.). Ud. no puede usar otros mecanismos ya disponibles en nThreads como semáforos, mutex, mensajes, etc. Para satisfacer la atención por orden de llegada, *embarcar*, *navegar* y *desembarcar* deben ser llamadas fuera de toda sección crítica.

(ii) (1 punto) ¿Tiene sentido usar spin-locks en una máquina monocore? Explique. Responda entonces cómo implementaría la exclusión mutua en un núcleo moderno de Unix para una máquina monocore.

Pregunta 3



I. 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: 4, 9, 4, 5, 4, 3, 2.



II. El diagrama parcial muestra las decisiones de scheduling para 3 procesos. Para cada proceso se indica la duración de la ráfaga y la duración de su estado de espera entre paréntesis. En el diagrama la línea punteada indica que el estado del proceso es READY. Explique si la estrategia de scheduling es *preemptive* o *non preemptive*. Rehaga el diagrama completo considerando ahora la estrategia *shortest job first non preemptive*. Considere que el predictor de duración para la próxima ráfaga es la duración de la última ráfaga.

III. ¿Cuál es el tamaño máximo de un archivo que usa un solo bloque de indirección simple en una partición con bloques de 1 KB? ¿Y si la partición posee bloques de 4 KB?

IV. Considere que el sistema operativo nativo (el anfitrión) es Windows y un sistema operativo invitado es Debian. (a) ¿En qué modo se ejecuta el núcleo de Windows, usuario o sistema? Explique. (b) ¿En qué modo se ejecuta de verdad el núcleo de Debian? Justifique.