CC4302 Sistemas Operativos – Examen – Semestre Otoño 2024 – Profs.: Mateu, Torrealba, Arenas

Pregunta 1

```
void initDisk();
int requestDisk(double pri);
void releaseDisk(int id);
// Cola de prioridades
PriQueue *makePriQueue();
double priBest(PriQueue *q);
void *priPeek(PriQueue *q);
void *priGet(PriQueue *q);
void priPut(PriQueue *q,
    void *elem, double pri);
int emptyPriQueue(PriQueue *q);
```

Parte a.- (4,5 puntos) Un servidor DNS utiliza varios pthreads para responder en paralelo consultas de nombres de dominio en Internet. Por razones de eficiencia los datos están almacenados en *p* discos replicados (todos tienen el mismo contenido). Los pthreads *deben* usar los discos en paralelo, pero el mismo disco no puede ser usado por 2 pthreads a la vez. Para garantizar la exclusión mutua de los discos Ud. debe programar las funciones *requestDisk* y *releaseDisk* del cuadro de arriba. Un pthread invoca la función *requestDisk* con una prioridad de acceso *pri* para solicitar un disco para su uso exclusivo. Esta función debe seleccionar un disco desocupado retornando su identificador entre 0 y *p*-1. Si no hay discos disponibles se debe esperar. El pthread notifica que ha desocupado el disco invocando *releaseDisk* con el identificador del disco que ocupaba como argumento. Si en ese momento hay llamadas pendientes de *requestDisk*, ese mismo disco se debe otorgar a la llamada que tenga la mejor prioridad. Use la cola de prioridades con las funciones que muestra el cuadro. Programe las inicializaciones en la función *initDisk*.

Restricciones: Para la sincronización Ud. debe usar un mutex y múltiples condiciones. Para evitar cambios de contexto inútiles use el patrón request.

Parte b.- (1,5 puntos) ¿Cuáles son los 3 errores clásicos al programar con múltiples threads? Ejemplifíquelos en palabras con el problema de la cena de filósofos.

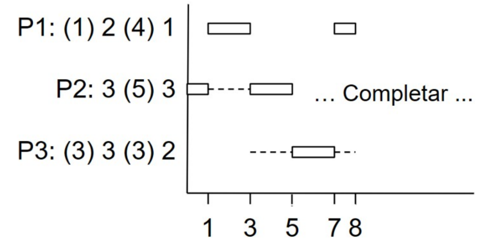
Pregunta 2

```
void nth_initDisk();
int nRequestDisk(double pri);
void nReleaseDisk(int id);
```

I. (4 puntos) Considere ahora que el servidor DNS usa nano threads. Programe las funciones *nRequestDisk* y *nReleaseDisk* con los encabezados del cuadro y con la misma funcionalidad de la pregunta 1, pero como herramienta de sincronización nativa de nThreads, es decir usando operaciones como *START_CRITICAL*, *setReady*, *suspend*, *schedule*, etc. No puede implementar esta API en términos de otras herramientas pre-existentes en nThreads como semáforos, mutex o condiciones.

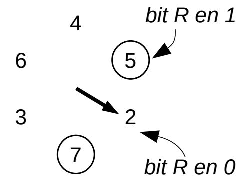
Ayuda: Programe las inicializaciones en la función *nth_initDisk*; use la misma cola de prioridades; use el estado WAIT_DISK, y; puede agregar nuevos campos al descriptor de los nano threads (no necesita declararlo, solo indique su tipo en un comentario).

II. (2 puntos) 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. Identifique qué estrategia es la utilizada. ¿Es esta estrategia de scheduling preemptive o non preemptive? Complete el diagrama.



Pregunta 3

A) 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, 1, 3, 1, 3, 2.

B) Se tiene un archivo de 73 KB en una partición Unix con bloques de 4 KB. Haga un diagrama mostrando inodos, bloques de datos y de indirección. Conteste además: ¿Cuanto espacio en disco se ocupa realmente para almacenar este archivo?

C) ¿Cuantos spin-locks se necesitan en un núcleo clásico de Unix para una máquina multi-core? ¿Y cuantos spin-locks se necesitan en un núcleo moderno para una máquina mono-core? Explique.

D) Ignorando la diferencia de costo entre un SSD y un disco duro, explique cuál es la principal ventaja de elegir un SSD en vez de un disco duro y cuál es la principal desventaja.