CC4302 Sistemas Operativos – Examen – Semestre Otoño 2025 – Profs.: Mateu, Torrealba, Arenas

Pregunta 1

I. (5 puntos) En un torneo de ajedrez se enfrentarán *N* jugadores. Cada jugador tiene un ranking asignado único entre *0* (el peor ranqueado) y *N-1* (el mejor ranqueado). La mitad elige jugar con las piezas blancas, la otra mitad con las piezas negras. Jugar con las negras es más difícil que jugar con las blancas y por lo tanto el que juega con las negras debe tener mejor ranking que el juega con las blancas. La siguiente es una implementación incompleta de la función *play* que retorna el adversario con el que se enfrentará el jugador *me*:

```
enum { BLACK=0, WHITE=1 };
typedef struct {
  Player *me, **padv;
  int color;
  …
} Request;
…
Request *preqs[N];
Player *play(Player *me, int r,
             int color) {
  …
  Player *adv= NULL;
  int i, incr, end;
  if (color==BLACK) {
    // Si juego con las negras busco
    // un jugador con ranking
    // r-1, r-2, …, 0
    i= r-1; incr= -1; end= -1;
  } else {
    // Si juego con las blancas busco
    // un jugador con ranking
    // r+1, r+2, …,N-1
    i= r+1; incr= 1; end= N;
  }
  while (i!=end) {
    if (preqs[i]!=NULL &&
        preqs[i]->color!=color)
      break;
    i += incr;
  }
  if (i!=end) {
    Request *preq= preqs[i];
    // El jugador me se enfrenta con
    // preq->me
    … asigne adv y *preq->padv …
  }
  else {
    // Debe esperar en preqs[r]
    Request req={me,&adv,color,
                 …};
    preqs[r]= &req;
    … espere …
  }
  …
  return adv;
}
```

En la función play *me* es el identificador de quien solicita jugar, *r* es su ranking y *color* es WHITE o BLACK. **Complete** esta implementación. Para la sincronización Ud. debe usar un mutex y múltiples condiciones de pthreads. Además debe evitar cambios de contexto inútiles.

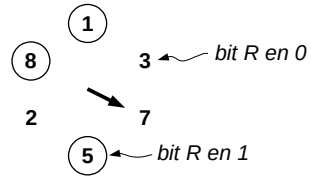
Restricciones: Cuando un jugador A solicita jugar con las negras, entre todos los que esperan en ese instante jugar con las blancas, A debe enfrentarse con el de mejor ranking pero que no exceda el ranking de A. Si A tiene ranking 5 y esperan con las blancas jugadores con ranking 1, **3**, 7, 9 se debe elegir al de ranking 3. Si ningún contrincante cumple esta restricción, A debe esperar. Cuando un jugador B solicita jugar con las blancas, entre todos los que esperan jugar con las negras, B debe enfrentarse con el de peor ranking pero asegurándose que posea un ranking mayor al de B. Si ningún contrincante cumple esta restricción, B debe esperar. Si B tiene ranking 6 y están esperando jugadores con las negras con ranking 2, 4, **8**, 10, se debe elegir a 8.

II. (1 punto) Explique si con las restricciones establecidas puede o no puede haber deadlock.

Pregunta 2

Programe la función *nPlayer *nPlay(nPlayer *me, int color, int r)* con los mismos requerimientos de la pregunta 1, pero como herramientas 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 de sincronización pre-existentes en nThreads como semáforos, mutex o condiciones. Use el estado WAIT_PLAY.

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: 3, 9, 7, 5, 2, 8.

B) Se tiene un archivo de 99 KB en una partición Unix con bloques de 8 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) 5 procesos se encuentran en estado de espera porque hicieron peticiones para leer sectores del disco (máximo 1000 sectores) en el siguiente orden: 200, 900, 500, 100, 700. El último sector leído fue el 300 y el penúltimo el 100. Indique en qué orden se harán las lecturas de estos 5 procesos cuando la estrategia de scheduling de disco es: (i) shortest seek first, (ii) método del ascensor (o look) y (iii) método del ascensor circular (c-look). Además, indique cuál demora menos tiempo, asumiendo que el tiempo que demora es proporcional a la distancia que recorre, es decir, mover el cabezal desde el sector 100 al 300 puede asumir que demora 2 ms.

D) Una ráfaga de un proceso se ejecuta en 3 tajadas (*slices*). ¿De qué tipo de scheduling se trata? ¿*Preemptive* o *non-preemptive*? Explique.

E) Si tuviese que programar desde cero un núcleo de sistema operativo para máquinas mono-core, ¿elegiría un núcleo clásico o un núcleo moderno? Explique la razón. ¿Por qué cambiaría su decisión si se tratase de máquinas multi-core?