CC3301 Control 3

2 horas

18 Noviembre de 2010

Pregunta 2

Usted, que tiene visión de negocio, instaló una escalera en la mina san josé que le permite a los turistas bajar al refugio y subir a cambio de una módica suma. Para esto, usted, que en el fondo es más computín que negociante, para evitar colapsos programó un acceso exclusivo a la escalera.

```
pthread_mutex_t escalera;

int en_refugio = 0;
pthread_cond_t espacio;
pthread_mutex_t mutex;

/* un thread por turista */
un_turista(int id) {
  for (;;) {
    pthread_mutex_lock(&escalera);
    baja();
    while (en_refugio >= MAX)
      pthread_cond_wait(&espacio, &escalera);
    pthread_mutex_unlock(&escalera);
    en_refugio++;
    entra_y_mira();
    en_refugio--;
    pthread_cond_signal(&espacio, &escalera);
    pthread_mutex_lock(&escalera);
    sube();
    pthread_mutex_unlock(&escalera);
    respira();
  }
}
```

Como habrá notado, los turistas son un poco claustrofóbicos por lo que suben a tomar aire y vuelven a bajar. Además el refugio tiene un máximo MAX de personas que pueden estar y mirar al mismo tiempo.

1. Analice y argumente si la solución tiene alguno de los problemas típicos de multi-threading: data-race, dead-lock, hambruna, etc.

   Solución: Hay data-race porque la variable en_refugio puede ser escrita concurrentemente por dos o más threads. Hay dead-lock porque un turista puede estar bajando la escalera y no poder entrar al refugio porque está lleno; pero tampoco se puede desocupar porque la escalera está ocupada y nadie puede salir. Hay hambruna porque puede ser que un turista no logre nunca tomar la escalera; de hecho, un solo turista podría tomar la escalera todo el tiempo (subir y bajar a cada rato) y dejar en hambruna a todo el resto.

2. Si encontró problemas en 1, corrija su código para que no suceda **Solución**:

```
int en_refugio = 0;
pthread_cond_t espacio;
pthread_mutex_t mutex;

/* solo tomo la escalera si mi ticket es igual a la pasada, esto evita
   la hambruna */
int pasadas;
int tickets;

/* un thread por turista */
un_turista(int id) {
  int pase = 0;

  for (;;) {

    pthread_mutex_lock(&escalera);
    pase = tickets++;

    /* antes de bajar verifico si hay espacio para evitar el dead-lock */
    while (en_refugio >= MAX && pase != pasadas)
      pthread_cond_wait(&espacio, &escalera);
    baja(id);
    en_refugio++; /* protejo la variable de los data-race */
    pasadas++;
    pthread_mutex_unlock(&escalera);

    entra_y_mira(id);

    pthread_mutex_lock(&escalera);
    pase = tickets++;

    while (pase != pasadas)
```

```
        pthread_cond_wait(&espacio, &escalera);

      en_refugio--;
      pthread_cond_signal(&espacio);
      sube(id);
      pasadas++;
      pthread_mutex_unlock(&escalera);

      respira(id);
    }
  }
```