CC4302 Sistemas Operativos – Control 1 – Semestre Primavera 2024
Profs.: Mateu, Torrealba, Arenas

Pregunta 1

La función *pbb* del cuadro de la derecha recibe como parámetros (i) la función *gen*, que genera por ejemplo 5 cartas al azar para jugar al poker, (ii) *n*, (iii) la función *eval*, que entrega verdadero si por ejemplo las 5 cartas corresponden a 2 pares, y (iv) un puntero a una estructura con parámetros adicionales para *gen* y *eval*. La función *pbb* genera *n* juegos de cartas al azar y entrega una estimación de la probabilidad de obtener por ejemplo 2 pares. Reprograme la función *pbb* paralelizando de manera que se ejecute la primera mitad del ciclo *for* en un nuevo thread, y la segunda mitad del ciclo en el mismo thread que invocó *pbb*.

```
typedef struct cards Cards;
typedef void (*GenFun)(
   void *ptr, Cards *pcards);
typedef int (*EvalFun)(
   void *ptr, Cards *pcards);
double pbb(GenFun gen, int n,
   EvalFun eval, void *ptr) {
  long long sum= 0;
  for (int i= 0; i<n; i++) {
    Cards cards;
    (*gen)(ptr, &cards);
    if ((*eval)(ptr, &cards))
      sum++;
  }
  return (double)sum / n;
}
```

Pregunta 2

Un hotel posee 20 habitaciones. Los clientes del hotel invocan la función *checkIn( )* para solicitar una de las habitaciones. Esta función debe retornar el identificador de la habitación otorgada (entre 0 y 19). Cuando un cliente se va del hotel invoca *checkOut(id)*, en donde *id* es siempre el identificador de la habitación que le fue otorgada previamente. Si al invocar *checkIn* no hay ninguna habitación disponible, el cliente debe esperar hasta que se libere una. Programe ambas funciones de manera que las habitaciones sean **otorgadas por orden de llegada** y evitando que una misma habitación sea ocupada por 2 clientes simultáneamente. Los encabezados de las funciones son los que muestra el cuadro de la derecha. Para la sincronización Ud. debe ocupar un mutex y **una sola condición** de pthreads.

```
int checkIn();
void checkOut(int id);
```

Pregunta 3

Un banco requiere las siguientes 2 funciones: La función *saldo* que recibe como parámetro el número de una cuenta y entrega la cantidad de dinero disponible en esa cuenta y la función *retirar* que recibe un monto de dinero que un cliente solicita retirar, su número de cuenta y entrega el monto efectivo retirado. Si el monto excede el dinero disponible en esa cuenta no se retira nada, y por lo tanto se retorna 0. Se propone la siguiente implementación de estas funciones:

```
Diccionario dicc;

int saldo(int cuenta) {
  enterRead();
  int rem= consultar(dicc, cuenta);
  exitRead();
  return rem;
}
```

```
int retirar(int cuenta, int monto) {
  int rem= saldo(cuenta);
  rem = rem - monto;
  if (rem<0)
    return 0;
  enterWrite();
  modificar(dicc, cuenta, rem);
  exitWrite();
  return monto;
}
```

Las funciones *enterRead*, *exitRead*, *enterWrite* y *exitWrite* se atienden por orden de llegada usando el patrón request, tal como se vió en cátedra. La función *consultar* obtiene el valor asociado a una cuenta en un diccionario y *modificar* establece un nuevo valor asociado a una cuenta. Las cuentas ya se encuentran creadas en *dicc* y sus saldos son positivos.

Parte a.- Muestre con un diagrama de threads un datarace en donde el banco pierde dinero.

Parte b.- Corrija esta solución con los siguientes requerimientos:

- Para la sincronización Ud. debe usar las mismas funciones *enterRead*, *exitRead*, *enterWrite* y *exitWrite*. No puede modificarlas ni usar mutex, semáforos u otro mecanismo de sincronización.
- Cuando hay solo invocaciones de *saldo*, las llamadas a *consultar* deben ocurrir en paralelo.