CC4302 Sistemas Operativos – Tarea 6 – Semestre Otoño 2024 – Profs.: Mateu, Torrealba, Arenas

Las acciones de la empresa ACME se transan en una bolsa de comercio cuyos operadores son representados mediante threads. Para comprar o vender una acción los operadores usan las funciones *compro* y *vendo*. Sus encabezados son:

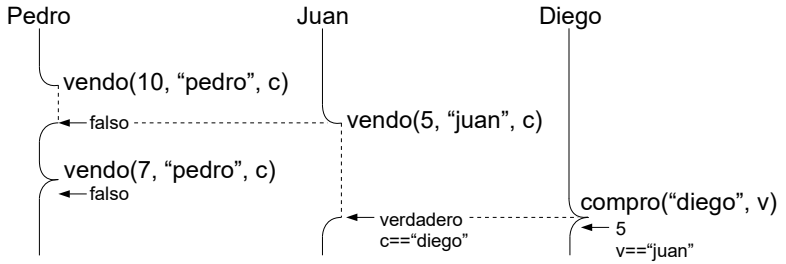
```
int vendo(int precio, char *vendedor, char *comprador);
int compro(char *comprador, char *vendedor);
```

En el cuadro siguiente se muestra a la izquierda el código usado por los múltiples threads vendedores de acciones y a la derecha el código de los múltiples compradores.

```
char *nom= miNombre();
int precio= miPrecio();
char comprador[100];
if (vendo(precio, nom,
          comprador)) {
  printf("vendi a %s\n",
         comprador);
}
```

```
char *nom= miNombre();
char vendedor[100];
sleep(tiempoAleatorio());
int precio= compro(nom, vendedor);
if (precio>0) {
  printf("compre a %s en %d\n",
         vendedor, precio);
}
```

La función *compro* transa una sola acción con el vendedor más barato de ese momento, retornando el precio pagado y copia el nombre del vendedor en el *2do parámetro*. Si en ese momento no hay ningún vendedor el precio será 0 y *compro* retorna de inmediato. La función *vendo* ofrece una acción al precio indicado y espera hasta que (i) aparezca un comprador, en cuyo caso retorna verdadero y copia el nombre del comprador en el 3er parámetro, o (ii) aparezca (o ya hay) un vendedor con un precio menor, retornando falso en tal caso. El siguiente diagrama muestra un ejemplo de ejecución. Una vez realizada la compra/venta, los vendedores ofrecen nuevamente un precio hasta que aparezca el siguiente comprador.



Pedro llama a *vendo*, que retorna falso cuando Juan llama a *vendo* con un precio menor. La segunda llamada de Pedro fracasa de inmediato porque su precio todavía es mayor al de Juan. Juan sí tiene éxito cuando Diego llama a *compro* y por lo tanto la llamada a *vendo* de Juan retorna verdadero, copiando el nombre “diego” en el parámetro *comprador*. Por su parte *compro* retorna 5 (el precio) y copia “juan” en el parámetro *vendedor*. Observe que nunca habrá más de un vendedor en espera.

Programe las funciones *vendo* y *compro* usando obligatoriamente esta metodología:

- Para programar la sincronización requerida Ud. debe usar spinlocks. No puede usar otras herramientas de sincronización como mutex/condiciones o semáforos. Tampoco puede crear nuevos threads con *pthread_create*.
- Use una variable global para el spinlock que garantiza la exclusión mutua.
- Use una variable global para guardar el precio más económico ofertado por un vendedor hasta el momento.
- Use una variable global para guardar la dirección del string con el nombre del vendedor y otra variable global para la dirección en donde el comprador debe copiar su nombre.
- Use una **variable local** *VL* para mantener el estado de la oferta de un vendedor: en espera, adjudicado o rechazado. Además use un puntero global que guarde la dirección de VL, para que el próximo vendedor o el comprador pueda cambiar el estado de VL. Recuerde que solo puede haber un vendedor en espera y por lo tanto no necesita una cola.
- Use una **variable local** para el spinlock en donde el vendedor debe esperar, y declare una variable global con la dirección del spinlock del vendedor en espera.

Prueba de la tarea: La verificación del funcionamiento correcto de su tarea se realizará primero implementando los spinlocks con mutex y condiciones, sin busy-waiting, y luego usando verdaderos spin-locks que sí esperan con busy-waiting y por lo tanto, se ocupa el 100% de la CPU. El primer método es útil para detectar dataraces.

El test de esfuerzo verifica la consistencia de su solución: que si la función *vendo* dice que a Juan le compró Diego, entonces la función *compro* debe decir que Juan le vendió a Diego.

Instrucciones

Descargue *t6.zip* de U-cursos y descomprímalo. Ejecute el comando *make* sin parámetros en el directorio *T6* para recibir instrucciones acerca del archivo en donde debe programar su solución (*bolsa.c*), cómo compilar, probar y depurar su solución, los requisitos que debe cumplir para aprobar la tarea y cómo entregar su tarea por U-cursos.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *bolsa.zip* generado por *make zip*. Este incluye *bolsa.c* y *resultados.txt*. Recuerde descargar el archivo que subió, descargar nuevamente los archivos adjuntos y volver a probar la tarea tal cual como la subió a U-cursos. Solo así estará seguro de no haber entregado archivos incorrectos. Se descuenta medio punto por día de atraso. No se consideran los días de receso, sábado, domingo o festivos.