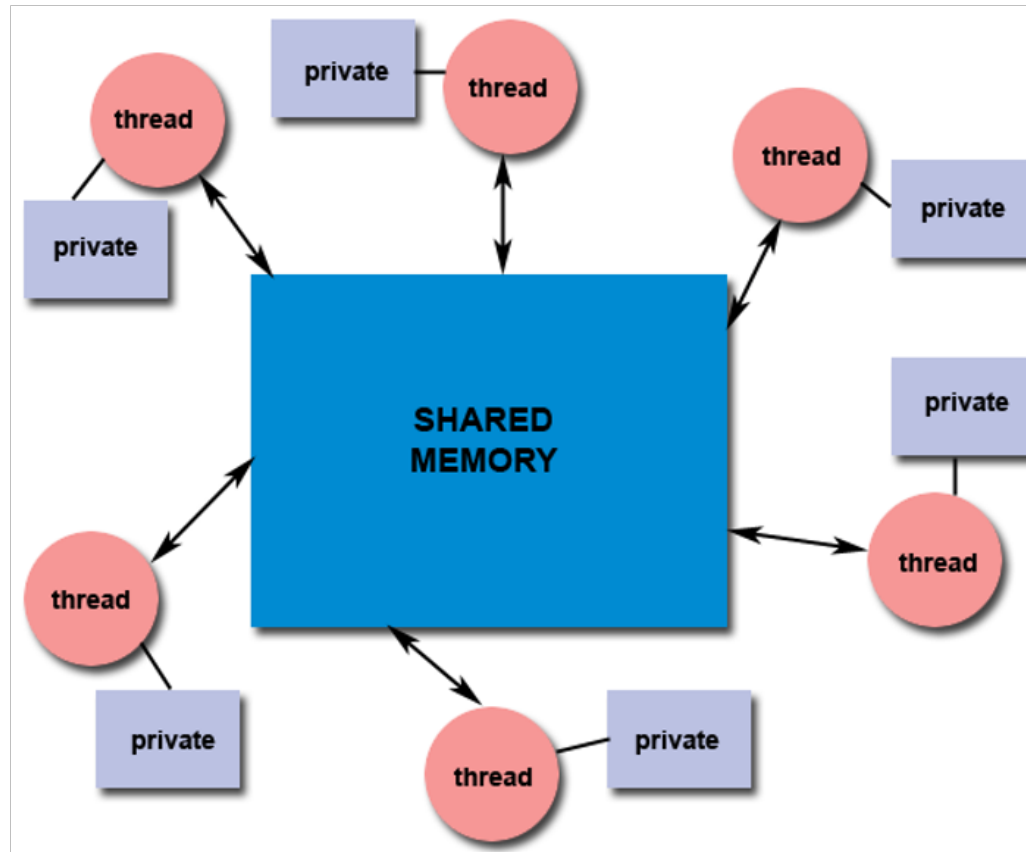


Recursos compartidos: Threads

Los programadores son responsables de sincronizar el acceso a los datos compartidos.

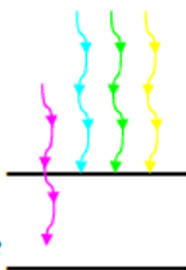
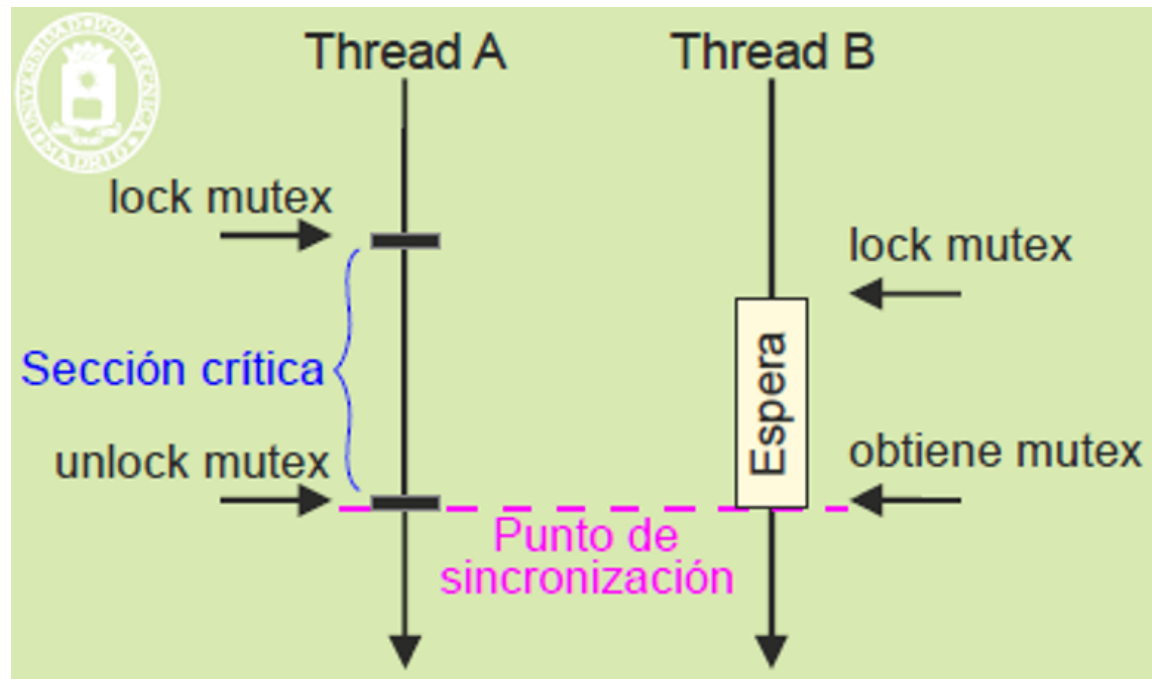


Mutex y variables condicionales

- Un mutex es un mecanismo de sincronización indicado para procesos ligeros.
- Es un semáforo binario con dos operaciones atómicas:
 - **lock(m)** Intenta bloquear el mutex, si el mutex ya está bloqueado el proceso se suspende.
 - **unlock(m)** Desbloquea el mutex, si existen procesos

```

mutex_lock(mutex);
<<sección crítica>>
mutex_unlock(mutex);
  
```

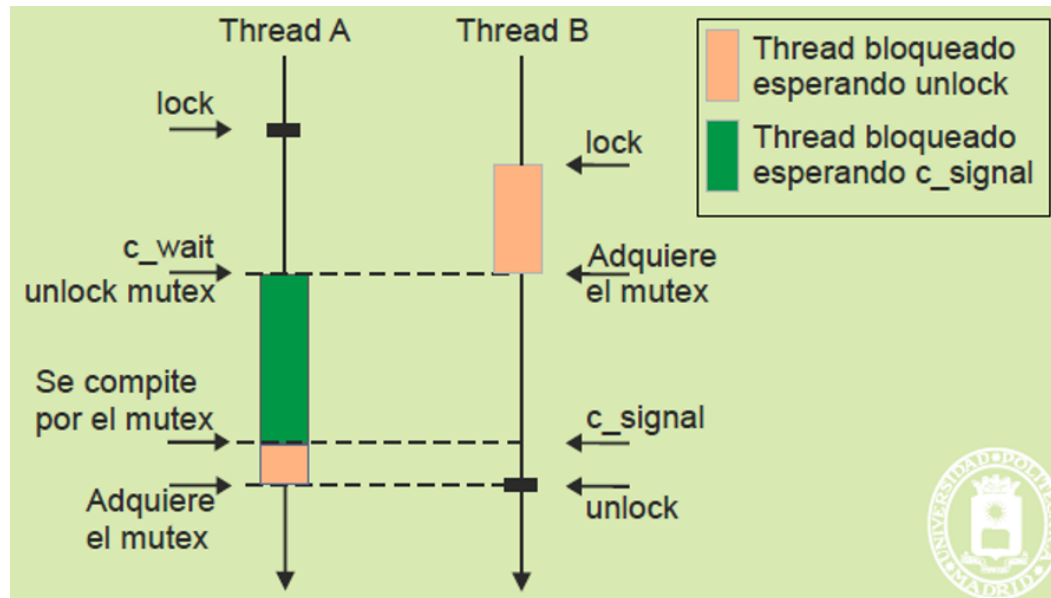
Mutex y variables condicionales

Variable Condicional: Variable de sincronización asociada a un mutex y se usa para bloquear a un thread hasta que ocurra un suceso.

Operaciones:

- **wait():** Liberar el mutex y bloquear el proceso. Cuando el proceso se despierta readquiere el mutex.
- **signal/notify():** Envía un aviso a un thread bloqueado en la variable condicional. Si no hay ningún thread bloqueado no hace nada.
- **broadcast():** Envía un aviso a todos los threads bloqueados en la variable condicional. Si no hay ningún thread bloqueado no hace nada.

Mutex y variables condicionales



```
mutex lock(mimutex);
while (<<no puedo continuar (condición variable control)>>)
    c_wait(cond, mimutex);
<<modifica variables de control>>
mutex_unlock(mimutex);
```

<<acceso al recurso compartido. El acceso puede ser múltiple>>

```
mutex lock(mimutex);
<<modifica variables de control>>
c_signal(cond);
mutex_unlock(mimutex);
```

- Ambos hilos intentan adquirir el **mutex** (representado por la palabra `lock`). Solo uno de los hilos puede mantener el mutex al mismo tiempo.
- **Thread A** adquiere el mutex primero y luego llama a `c_wait` (una función que bloquea el hilo en espera de una señal). Cuando se llama a `c_wait`, el hilo libera el mutex (`unlock mutex`) y se bloquea esperando una señal de otro hilo.
- Mientras **Thread A** está bloqueado esperando la señal (`c_wait`), el **Thread B** puede adquirir el mutex.
- **Thread B** realiza su tarea y, eventualmente, envía la señal `c_signal` para despertar al **Thread A**. Luego, **Thread B** libera el mutex (`unlock`).
- Cuando **Thread A** recibe la señal `c_signal`, compete nuevamente por el mutex para continuar su ejecución. Una vez que lo adquiere, sigue ejecutándose.

Mutex y variables condicionales: Productor/Consumidor

```
int buffer[BUFSIZE], in, out;  
condition empty, full;  
mutex mimutex;  
main()  
{ in=0; out=0;  
  ....  
}
```

```
void productor()  
{ ele = producir_elemento();  
  lock(mimutex);  
  while ((out+1)%BUFSIZE == in)  
    wait(full);  
  buffer[out]=ele;  
  out=(out+1)%BUFSIZE;  
  signal(empty);  
  unlock(mimutex);  
}
```

```
void consumidor()  
{  
  lock (mimutex);  
  while (in == out)  
    wait(empty);  
  int ret=buffer[in];  
  in=(in+1)%BUFSIZE;  
  signal(full);  
  unlock (mimutex);  
}
```

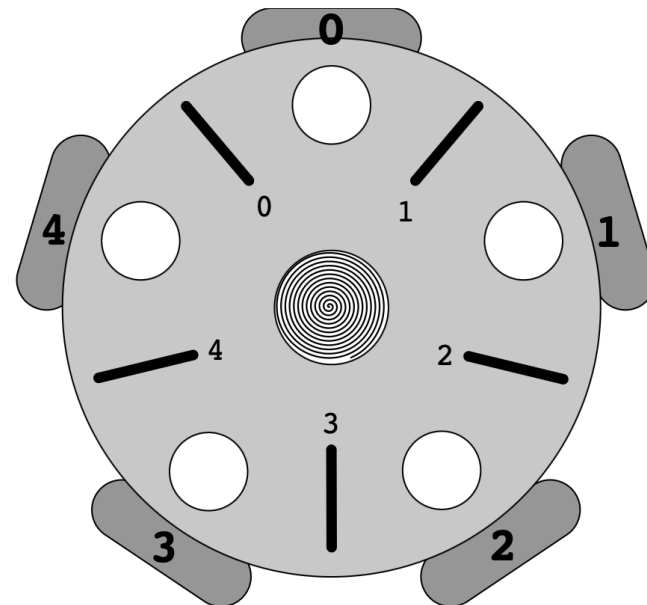
Ejercicio

- Desarrolle una solución al problema de los lectores y escritores con prioridad para los procesos escritores empleando Semáforos
- Desarrolle una solución al problema productor consumidor utilizando Mutex y variables condicionales.

Problema Clásico

Filósofos comensales

El problema radica en que se dispone de una mesa con cinco platos y cinco palillo, uno situado entre cada plato. Para comer un filósofo necesita dos palillos; uno situado a la derecha y otro situado a la izquierda del plato, con el condicionante de que un filósofo no puede coger un palillo si este ha sido cogido por otro filósofo hasta que este ultimo lo deje libre. El problema consiste en indicar el código que ejecutaría los filósofos de forma que estos puedan pensar y comer y que se cumplan las tres condiciones que debe cumplir cualquiera solución al problema de la sección crítica: exclusión mutua, progreso y espera limitada.



Problema Clásico

- Barbero dormilón
 - Problema planteado por Dijkstra en 1971
 - Una peluquería en la que hay un barbero, una silla de peluquero y N sillas para que se sienten los clientes en espera, si es que los hay.
 - Si no hay clientes presentes, el barbero se sienta en su silla de peluquero y se duerme.
 - Cuando llega un cliente, éste debe despertar al barbero dormilón.
 - Si llegan más clientes mientras el barbero corta el cabello de un cliente, se sientan (si hay sillas desocupadas) o salen de la peluquería (si todas las sillas están ocupadas).