Tema 4-Productores y consumidores

Una condición permite a los procesos o threads suspender su ejecución hasta que se les despierte. A veces es necesario interrumpir la ejecución de estos en medio de una sección crítica hasta que otro proceso modifique el estado de un recurso compartido. Este otro proceso tiene que despertar a los que están esperando. Vamos a ver como quedarían los productores y consumidores ahora:

Productor

```
mtx_t buffer_lock;
cnd_t buffer_full;
cnf_t buffer_empty;

while(1){
        elemento e=crear_elemento();

        mtx_lock(&buffer_lock);
        while(number_of_elements()<BUFFER_SIZE){
                cnd_wait(&buffer_full, &buffer_lock);
        }
        insert(e);
        if(number_of_elements()==1){
                cnd_broadcast(&buffer_empty);
        }
        mtx_unlock(&buffer_lock);
}
```

Consumidor

```
while(1){
        elemento e;
        mtx_lock(&buffer_lock);
        while(number_of_elements()==0){
                cnd_wait(&buffer_empty, &buffer_lock);
        }
        remove(e);
        if(number_of_elements()==BUFFER_SIZE-1){
                cnd_broadcast(&buffer_full);
        }
        mtx_unlock(&buffer_lock);
```

```
        utilizar(e);
}
```

Apéndice

cnd_wait:
https://en.cppreference.com/w/c/thread/cnd_wait
cnd_broadcast:
https://en.cppreference.com/w/c/thread/cnd_broadcast

Problemas y usos

Si wait no fuera atómico, es decir, si no fuera vusta por el sistema como una única instrucción indivisible, el estado del buffer podría cambiar antes de que el thread se durmiera. Este problema se llama lost wakeup.

Las claves para utilizar condiciones son:

- Comprobar en un while , tanto antes como después de esperar, el estado del programa.
- Tener bloqueado el mútex que protege ese estado antes de la comprobación.
- Pasar ese mútex al wait para que otro thread pueda desbloquearlo.
- Evitar mantener otros mutex bloqueados durante la espera.

Esto visto en pseudocódigo sería así:

- Thread que comprueba y espera:

```
lock(m)
while(!estado válido para continuar)
        wait(cnd, m)
sección crítica
unlock(m)
```

- Thread que cambia el estado y notifica:

```
lock(m)
cambiar_estado()
if(estado ahora válido)
        broadcast(cnd)
unlock(m)
```

Usamos un while para esperar porque no sabemos cuántos productores despiertan. Si solo se ha retirado 1 elemento y se despierta más de 1 productor, intentaremos insertar con el buffer lleno.

Usamos broadcast en vez de signal porque signal no notifica a los consumidores si el buffer no está vacío.