Tema 3-Interbloqueo e inanición

Es una situación donde dos o más procesos están esperando por recursos que el otro tiene ocupados. Por ejemplo:

```
void fun(int *v1, int *v2, mtx_t *m1, mtx_t *m2){
        int x;

        mtx_lock(m1);
        mtx_lock(m2);
        x= *v1 + *v2;
        mtx_unlock(m1);
        mtx_unlock(m2);

        return x;
}
```

Si llamamos a la función desde 2 threads de forma que los valores se intercambien:

- fun(a, b, m1, m2)
- fun(b, a, m2, m1)

Ambos threads se quedarían esperando al mútex que tiene el otro bloqueado.

Puede darse interbloqueo cuando:

- Exclusión mútua: existen recursos no compartibles.
- Hold and wait: un proceso mantiene recursos reservados mientras espera para reservar otros.
- No apropiación: el SO no puede liberar recursos reservados.
- Espera circular: los procesos esperan por recursos reservados por procesos que esperan por los primeros.

El interbloqueo se puede evitar mediante detección o prevención. Como el sistema operativo conoce que recursos están siendo utilizados y en espera, puede saber qué y cuándo produce un interbloqueo. Al detectarlo puede matar a los procesos involucrados o puede apropiarse de los recursos y entregárselos a otro proceso. Pero el programador puede prevenirlo (hold and wait o reserva ordenada).

Hold and wait

```
void fun(int *v1, int *v2, mtx_t *m1, mtx_t *m2){
        int x;
```

```
        while(1){
                mtx_lock(m1);
                if(mtx_trylock(m2)){
                        mtx_unlock(m1);
                        continue;
                }
        }
        x= *v1 + *v2;
        mtx_unlock(m1);
        mtx_unlock(m2);
        break;

        return x;
}
```

Reserva ordenada

```
void fun(int *v1, int *v2, mtx_t *m1, mtx_t *m2, int orden1, int orden 2){
        int x;

        if(orden1<orden2){
                mtx_lock(m1);
                mtx_lock(m2);
        } else{
                mtx_lock(m2);
                mtx_lock(m1);
        }
        x= *v1 + *v2;
        mtx_unlock(m1);
        mtx_unlock(m2);

        return x;
}
```