



A classical example - Producer Consumer

```
void producer () {  
    while(1) {  
        item := produce()  
        while(full(buffer)) {  
            /* do nothing */  
        }  
        write(buffer, item)  
    }  
}
```

```
void consumer () {  
    while(1) {  
        while(empty(buffer)) {  
            /* do nothing */  
        }  
        item := read(buffer)  
        consume(item)  
    }  
}
```



Critical Section

# A classical example - Producer Consumer

```
void producer () {  
    while(1){  
        item := produce()  
        while(full(buffer)) {  
            /* do nothing */  
        }  
        write(buffer, item)  
    }  
}
```

```
void consumer () {  
    while(1){  
        while(empty(buffer)) {  
            /* do nothing */  
        }  
        item := read(buffer)  
        consume(item)  
    }  
}
```

Critical Section

# Requirements

## 1. **Mutual exclusion**

If one thread is in the critical section, then no other is

➔ Mutual exclusion ensures **safety property** (nothing bad happen)

## 2. **Progress**

If some thread T is not in the critical section, then T cannot prevent some other thread S from entering the critical section. A thread in the critical section will eventually leave it.

## 3. **Bounded waiting** (no starvation)

If some thread T is waiting on the critical section, then T will eventually enter the critical section

➔ Progress and bounded waiting ensures the **liveness property** (something good happen)

## 4. **Performance**

The overhead of entering and exiting the critical section is small with respect to the work being done within it