Producer-Consumer Problem

- Two kinds of entities:
  - Producers: Generate new objects
  - Consumers: Consume these objects
- Intermediate buffer
  - A data structure containing objects
- Producers and consumers running concurrently
  - May be running at different speeds
- Examples?

Producer-Consumer Problem

Producer:
while(1)
{
    produce(item);
    put(item, buffer);
}

Buffer_type buffer;

Consumer:
while(1)
{
    item = get(buffer);
    consume(item);
}

- What’s the critical section?
- How do we protect the critical section?

Producer-Consumer Synchronization: Using Mutex Locks

Producer:
while(1)
{
    produce(item);
    lock(mut);
    put(item, buffer);
    unlock(mut);
    consume(item);
}

Buffer_type buffer;
mutex mut;

Consumer:
while(1)
{
    lock(mut);
    item = get(buffer);
    unlock(mut);
    consume(item);
}

- What happens if buffer is empty?
**Handling Empty Buffer: Using Mutex Locks**

Buffer_type buffer;  int num_items=0;
mutex mut;

Producer:
while(1)
{
    produce(item);
    lock(mut);
    put(item, buffer);
    num_items++;
    unlock(mut);
}

Consumer:
while(1)
{
    lock(mut);
    while (num_items==0);
    item = get(buffer);
    num_items--;
    unlock(mut);
    consume(item);
}

**Handling Empty Buffer: Using Condition Variables**

Buffer_type buffer;  int num_items=0;
mutex mut;  cond_var cond;

Producer:
while(1) {
produce(item);
lock(mut);
while (num_items==0)
item = get(buffer);
num_items++;
signal(cond);
unlock(mut);
}

Consumer:
while(1) {
lock(mut);
while (num_items==0)
wait(cond, mut);
item = get(buffer);
num_items--;
consume(item);
}

**Producer-Consumer: Using Semaphores**

Buffer_type buffer;  int num_items=0;
Semaphore mut=1;  Semaphore items=0;

Producer:
while(1) {
produce(item);
wait(mut);
put(item, buffer);
num_items++;
signal(mut);
signal(items);
}

Consumer:
while(1) {
wait(items);
wait(mut);
item = get(buffer);
um_items--;
signal(mut);
consume(item);
}

- Do we need num_items?

**Bounded Buffer Problem**

- Suppose Buffer is of finite size: capacity of N items
- What happens if buffer is full?
Bounded Buffer: Using Semaphores

```c
Buffer_type buffer;
Semaphore mut=1; Semaphore items=0, slots=N;
```

Producer:
```
while(1) {
    produce(item);
    wait(slots);
    wait(mut);
    put(item, buffer);
    signal(mut);
    signal(items);
}
```

Consumer:
```
while(1) {
    wait(items);
    wait(mut);
    item = get(buffer);
    signal(mut);
    signal(slots);
    consume(item);
}
```

Readers-Writers Problem

- Reader: Process that only reads the shared data
- Writer: Process that modifies the shared data

Examples:
- Shared access to a file or database
- Access to a Webpage

Readers-Writers Problem (Contd.)

- Reader/writer accesses are not symmetric:
  - Any number of readers can be concurrently reading
  - Only one writer can be writing at a time (no other readers or writers)

Prioritizing Access

- Who should get higher priority of access?
  - First RW problem: Readers given higher priority
    - A reader can join if other readers already reading
    - Writer given access only when all readers done
  - Second RW problem: Writers given higher priority
    - A reader cannot join if a writer is waiting
    - Writer given access when all existing readers done
First Readers-Writers Problem With Semaphores

```c
Semaphore wrt=1, mutex=1;  int readcount=0;

Writer:
while(1) {
    wait(wrt);
    write_data();
    signal(wrt);
}

Reader:
while(1) {
    wait(mutex);
    readcount++;
    if (readcount==1)
        wait(wrt);
    signal(mutex);
    Read_data();
    wait(mutex);
    readcount--;
    if (readcount==0)
        signal(wrt);
    signal(mutex);
}
```

Reader-writer locks

- Special locks
  - Implement a reader-writer synchronization protocol
  - Acquired in read or write mode
- What types of applications can benefit from these?