Other Synchrnization Problems

- Dining Philosophers
- Producer Consumer
- Readers Writers
  - reader’s priority, writer’s priority

Readers/Writers with R priority

- **Reader**

```
P(mutex)
if (nr == 0) {
   nr++; P(notaccessed);
} else
   nr++;
V(mutex);

// Read Operations
   P(mutex);
   nr --;
   if (nr == 0) V(notaccessed);
   V(mutex);
```

- **Writer**

```
   P(exclw);
   P(notaccessed);

//Write Operations
   V(notaccesed);
   P(exclw);
```

Serializers

- Monitor Problems
  - If monitor encapsulates resource, then concurrency is reduced even where it is possible
  - If resource is outside, then rouge processes can bypass the monitor.
- Serializers try to avoid this:
  - They are still an ADT with defined operations that encapsulate data, and enforce mutual exclusion.
  - Procedures ma have “hollow” regions where they may allow other processes to access the serializer.
    - **join-crowd** (crowdid) **then** body **end**
    - **enque** (prio,qname) **until** (condition)
  - all events that gain and release the serializer are totally ordered.

Serializer to solve Readers/Writers

- **Read**

```
Enque (rq) until empty(wcrowd)
Joincrowd(rc) then
      //Read operation
end
```

- **Write**

```
Enque (wq) until ( empty(wc) && empty(rc) && empty(rq) )
Joincrowd (wc) then
            //Write Operation
end
```

Path Expressions

- Defines possible “valid” execution histories of the operations
  - Sequencing: a;b – a precedes b, no concurrency.
  - Selection: a+b – either a or b is done, but not both and in any order.
  - Concurrency: {a} – any number of instances of a can be done at the same time.
- **Path** {read} + write **end** gives a weak reader’s priority solution.

CSP

- P 2?v
  - Get the value of v from P2 as an input
- P1!10
  - Output value 10 to P1
- The input and output are synchronized if they name each other as source/destination, and the types match
- G-> CL – execute commands in list CL if guard G is true.
- Alternative command – execute one of the choices where is guard is true.
  - G1 -> CL1 o G2 -> CL2 … o … Gn -> CLn
- Repetitive Command *[Alternative] – repeat until all guards are false.