Sync-advanced

Suplement !

1-Producer 1-Consumer Problem

(lockfree)

```
int front = 0;          /* buf[(front+1) % SIZE] is first item */
int rear  = 0;          /* buf[rear % SIZE] is last item */
int buf[SIZE];

void insert(x) {    /* Insert x onto the rear of shared buffer items */
    while(rear - front == SIZE) ;       /* Wait for available slot */
    buf[++rear % SIZE] = x;             /* Insert the item */
}

int remove() {       /* Remove and return the first item from buffer */
    while(rear == front) ;              /* Wait for available item */
    return(buf[++front % SIZE]);        /* remove the item */
}
```

1st Readers-Writers Problem

Readers:

```
int readcnt;    /* Initially 0 */
sem_t mutex, w; /* Both initially 1 */

void reader(void)
{
  while (1) {
    P(&mutex);
    readcnt++;
    if (readcnt == 1) /* First in */
      P(&w);
    V(&mutex);

    /* Reading happens here */

    P(&mutex);
    readcnt--;
    if (readcnt == 0) /* Last out */
      V(&w);
    V(&mutex);
  }
}
```

Writers:

```
void writer(void)
{
  while (1) {
    P(&w);

    /* Writing here */

    V(&w);
  }
}
```

2nd Readers-Writers Problem(1)

```
int readcnt,writecnt;        /* Initially 0 */
sem_t mutex1, mutex2, r, w; /* initially 1 */
```

Readers:

```
Reader() {

P(&r);
P(&mutex1);
readcnt = readcnt +1;
if(readcnt == 1)
            P(&w);
V(mutex1);
V(&r);

access resource;

P(&mutex1);
readcnt = readcnt -1;
if(readcnt == 0)
            V(&w);
V(&mutex1);
}
```

Writers:

```
Writer(){

P(&mutex2);
writecnt = writecnt +1;
if (writecnt == 1)
            P(&r);
V(&mutex2);
P(&w);

access resource;

V(&w);
P(&mutex2);
writecnt - writecnt - 1;
if (writecnt == 0)
            V(&r);
V(&mutex2);
}
```

2nd Readers-Writers Problem(1)

```
int readcnt,writecnt;     /* Initially 0 */
sem_t mutex1, mutex2, r, w; /* initially 1 */
```

Readers:

```
Reader() {

P(&r);
P(&mutex1);
readcnt = readcnt +1;
if(readcnt == 1)
            P(&w);
V(mutex1);
V(&r);

access resource;

P(&mutex1);
readcnt = readcnt -1;
if(readcnt == 0)
            V(&w);
V(&mutex1);
}
```

Reader 2

Reader 1

Writers:

```
Writer(){

P(&mutex2);
writecnt = writecnt +1;
if (writecnt == 1)
            P(&r);
V(&mutex2);
P(&w);

access resource;

V(&w);
P(&mutex2);
writecnt - writecnt - 1;
if (writecnt == 0)
            V(&r);
V(&mutex2);
}
```

Writer 1

2nd Readers-Writers Problem(2)

Readers:

```
int readcnt,writecnt;     /* Initially 0 */
sem_t writePending, mutex1, mutex2, r, w; /* initially 1 */
```

```
Reader() {

P(&writePending);
P(&r);
P(&mutex1);
readcnt = readcnt + 1;
if(readcnt == 1)
            P(&w);
V(mutex1);
V(&r);
V(&writePending);

access resource;

P(&mutex1);
readcnt = readcnt - 1;
if(readcnt == 0)
            V(&w);
V(&mutex1);
}
```

Reader 2

Reader 1

Writers:

```
Writer(){

P(&mutex2);
writecnt = writecnt + 1;
if (writecnt == 1)
            P(&r);
V(&mutex2);
P(&w);

access resource;

V(&w);
P(&mutex2);
writecnt = writecnt - 1;
if (writecnt == 0)
            V(&r);
V(&mutex2);
}
```

Writer 1

3rd Readers-Writers Problem (Fair)

```
int readcnt;        /* Initially 0 */
sem_t mutex, r, w; /* initially 1 */
```

Readers:

```
Reader() {

P(&r);
P(&mutex);
readcnt = readcnt +1;
if(readcnt == 1)
            P(&w);
V(mutex);
V(&r);

access resource;

P(&mutex);
readcnt = readcnt -1;
if(readcnt == 0)
            V(&w);
V(&mutex);
}
```

Writers:

```
Writer(){

P(&r);
P(&w);
            access resource
V(&w);
V(&r);
}
```

We need FIFO semaphore for FIFO operation.

Fast solution to 3rd Readers-Writers Problem

```
int incnt, outcnt, wait; /* Initially 0 */
sem_t in, out;           /* initially 1 */
Sem_t w;                 /* Initially 0 */
```

Readers:

```
Reader() {

P(&in);
incnt = incnt +1;
V(&in);

access resource;

P(&out);
outcnt = outcnt +1;
if(wait==1 && incnt == outcnt)
            V(&w);
V(&out);
}
```

Writers:

```
Writer(){

P(&in);
P(&out);
if(incnt == outcnt)
      V(&out);
Else
       wait = 1;
       V(&out);
       P(&w);
       wait = 0;

 access resource;

V(&in);
}
```