2 consumer , 1 producer

- Required two different CVs (conditional variables)

5 consumer, 2 Producer

- Required two unique cvs (conditional variables)

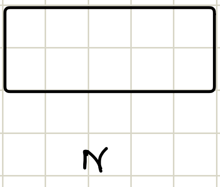
Summary :

(1) use different conditional variables for different types of thread.

(2) Always check the conditions in while loop

MID 243 / Fall 24

5



count $\longrightarrow$ shared variable

Child Thread:

```
while (true) {
  lock (m);
  while (count == 0) {
     signal (emptyBox);
     wait (fullbox, m);
  }
```

```
    getChocolateFromBox();
    eat();
    count--;
    unlock(m);
}
```

PS Concurrency

31

```
// Guest:
lock(m)
guest_count++
if (guest_count == N)
    signal(cv_host)
wait(cv_guest, m)
signal(cv_guest)
enterHouse()
unlock(m)
```

30

```
// Rider
lock(mutex)
rider_count++
if (rider_count == N)
    signal(cv_operator1)
    wait (cv_rider, mutex)
    enter_ride ()
    enter_count ++
    if (enter_count == N)
        signal (cv_operator2)
    unlock (mutex)
```

1 Finding Output From a Given code

```
mutex  m
cv     c[3]
int i = 0
void *func (void *args)
{
    lock (m)
    int id = *(int *) args
    if (id != i)
        wait (c[id], m)
    printf ("Thread %d", id);
    i = (i + 1) % 3;
    signal (cv[i])
    unlock (m)
}

int main ()
{
    pthread_t p[3];
    for (int i=0; i<3; i++) {
        create (p[i], func, i);
    }

    for (int i=0; i<3; i++)
    {
        join (p[i]);
    }
}
```

Output

```
  Thread 0
  Thread 1
  Thread 2
```

②

```
mutex m
cv    c[3]
int  i = 1
void *func ( void * args)
{
    lock (m)
    int id = * (int *) args
    if (id != i)
        wait (c[id], m)
    printf ("Thread %d", id);
    i = (i + 2) % 3;
    signal (cv[i])
    unlock (m)
}

int main()
{
    pthread_t  p[3];
    for (int i = 0; i < 3; i++) {
        create (p[i], func, i);
    }
    for (int i = 0; i < 3; i++)
    {
        join (P[i]);
    }
}
```

Output

```
Thread 0
Thread 1
Thread 2
```