MANCHESTER 1824
The University of Manchester

COMP35112 Chip Multiprocessors

Synchronisation in Parallel Programming - Condition Variables

Pierre Olivier

Event Signalling

- **Sleeping or busy-waiting for the buffer to be non-full/non-empty is suboptimal**
  - Example with thread trying to deposit in a full buffer:

```
while(full) {
    pthread_mutex_unlock(&b->lock);
    usleep(100);
    pthread_mutex_lock(&b->lock);
    full = (b->count == b->max_elements);
}
```

Event Signalling

- **Sleeping or busy-waiting for the buffer to be non-full/non-empty is suboptimal**
  - Example with thread trying to deposit in a full buffer:

```
while(full) {
    pthread_mutex_unlock(&b->lock);
    usleep(100);
    pthread_mutex_lock(&b->lock);
    full = (b->count == b->max_elements);
}
```

```
while(full) {
    pthread_mutex_unlock(&b->lock);
    /* busy wait */
    pthread_mutex_lock(&b->lock);
    full = (b->count == b->max_elements);
}
```

Event Signalling

- **Sleeping or busy-waiting for the buffer to be non-full/non-empty is suboptimal**
  - Example with thread trying to deposit in a full buffer:

```
while(full) {
    pthread_mutex_unlock(&b->lock);
    usleep(100);
    pthread_mutex_lock(&b->lock);
    full = (b->count == b->max_elements);
}
```

```
while(full) {
    pthread_mutex_unlock(&b->lock);
    /* busy wait */
    pthread_mutex_lock(&b->lock);
    full = (b->count == b->max_elements);
}
```

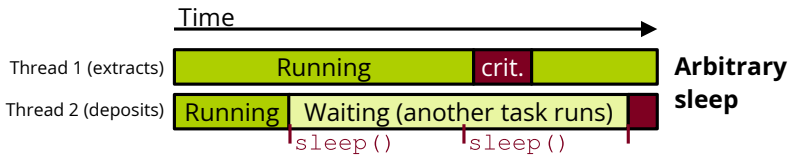
- With the `usleep` set to an arbitrary time, may sleep for a much longer time than needed
- Without usleep, keep trying non-stop, monopolising the CPU and wasting cycles

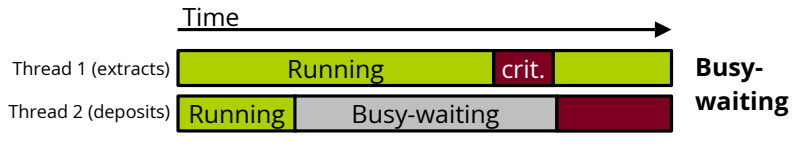
Event Signalling

- Sleeping or busy-waiting for the buffer to be non-full/non-empty is suboptimal
  - Example with thread trying to deposit in a full buffer:

```
while(full) {
    pthread_mutex_unlock(&b->lock);
    usleep(100);
    pthread_mutex_lock(&b->lock);
    full = (b->count == b->max_elements);
}
```

```
while(full) {
    pthread_mutex_unlock(&b->lock);
    /* busy wait */
    pthread_mutex_lock(&b->lock);
    full = (b->count == b->max_elements);
}
```



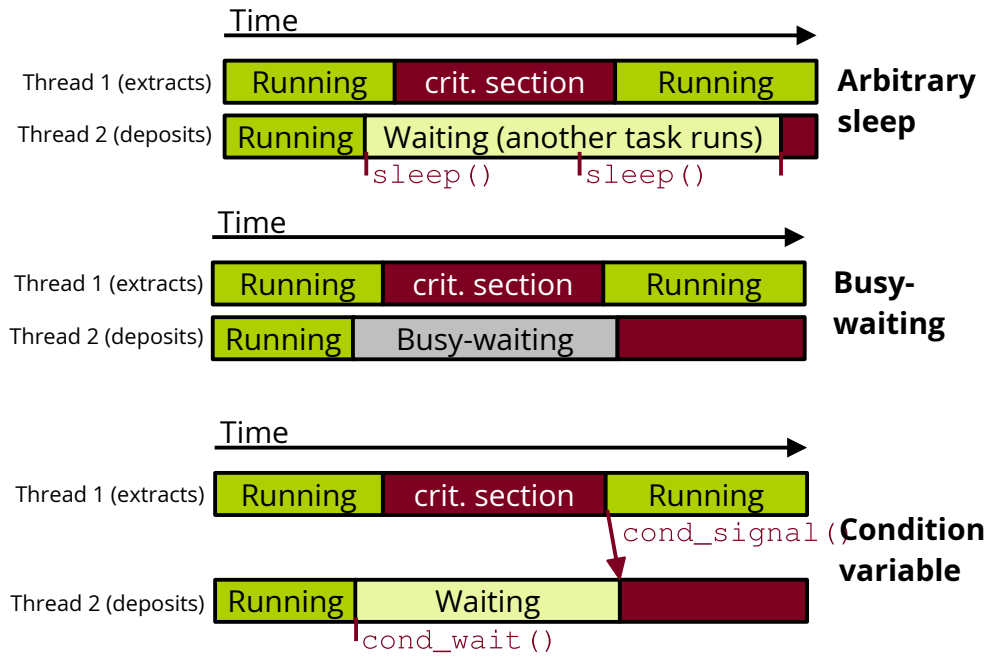


Condition Variables

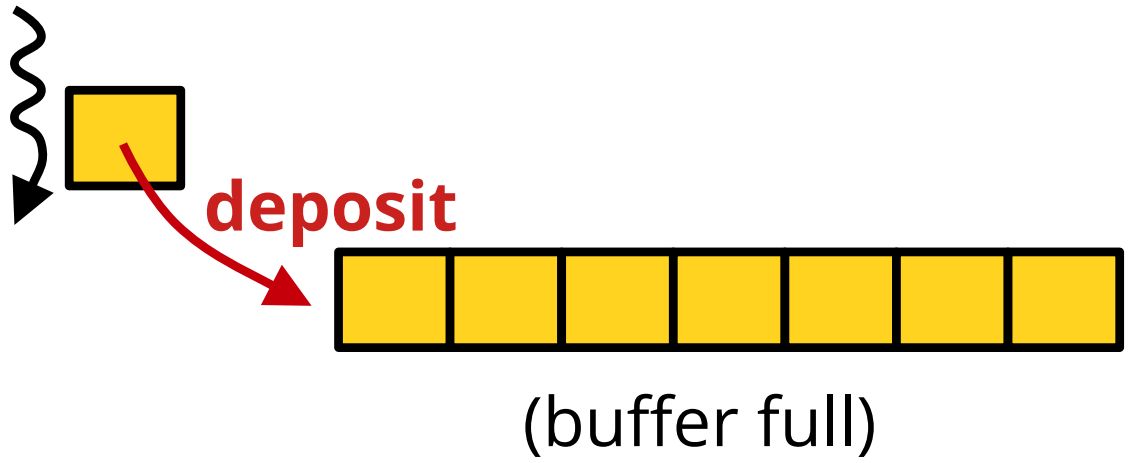
- Used to signal threads waiting for a conjunction of:
  1. **A lock becoming free** and
  2. **An arbitrary event** (e.g. buffer becoming non-full)
- Best of both worlds: thread wakes up right when needed, without monopolising the CPU by spinning

Condition Variables

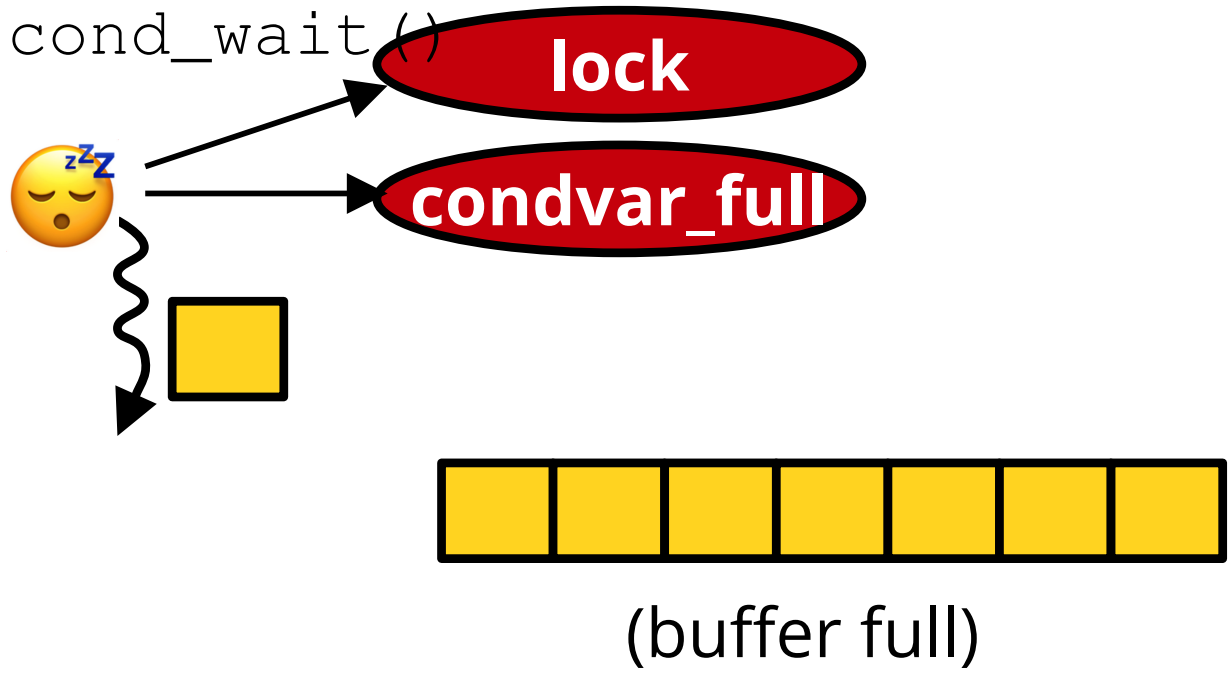
- Used to signal threads waiting for a conjunction of:
  1. **A lock becoming free** and
  2. **An arbitrary event** (e.g. buffer becoming non-full)
- Best of both worlds: thread wakes up right when needed, without monopolising the CPU by spinning



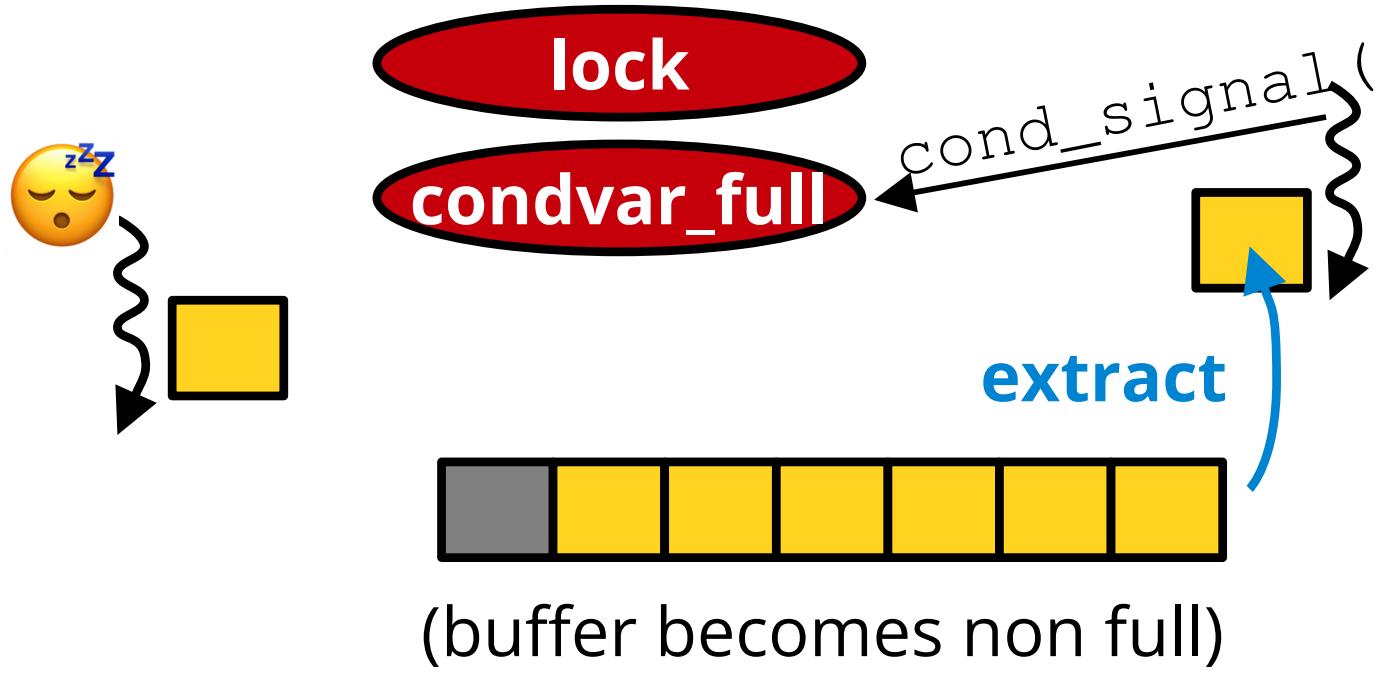
Applying Condition Variables to Our Example



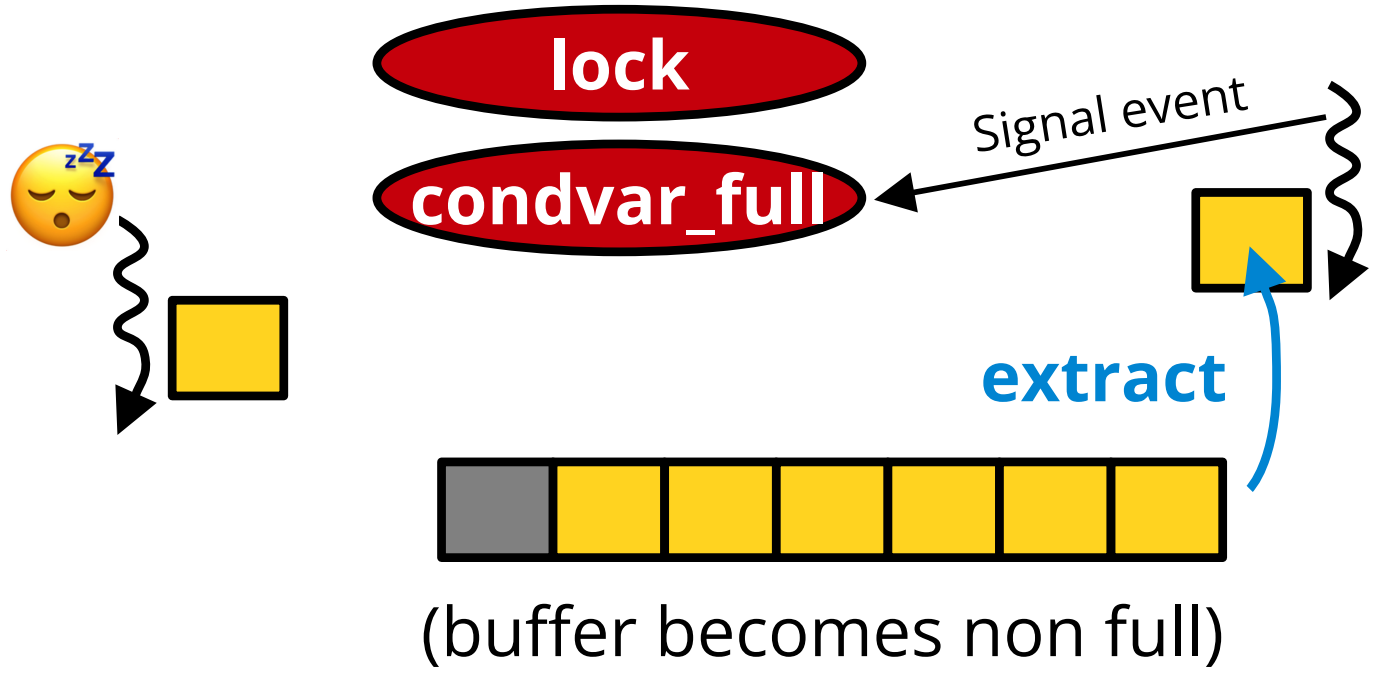
Applying Condition Variables to Our Example



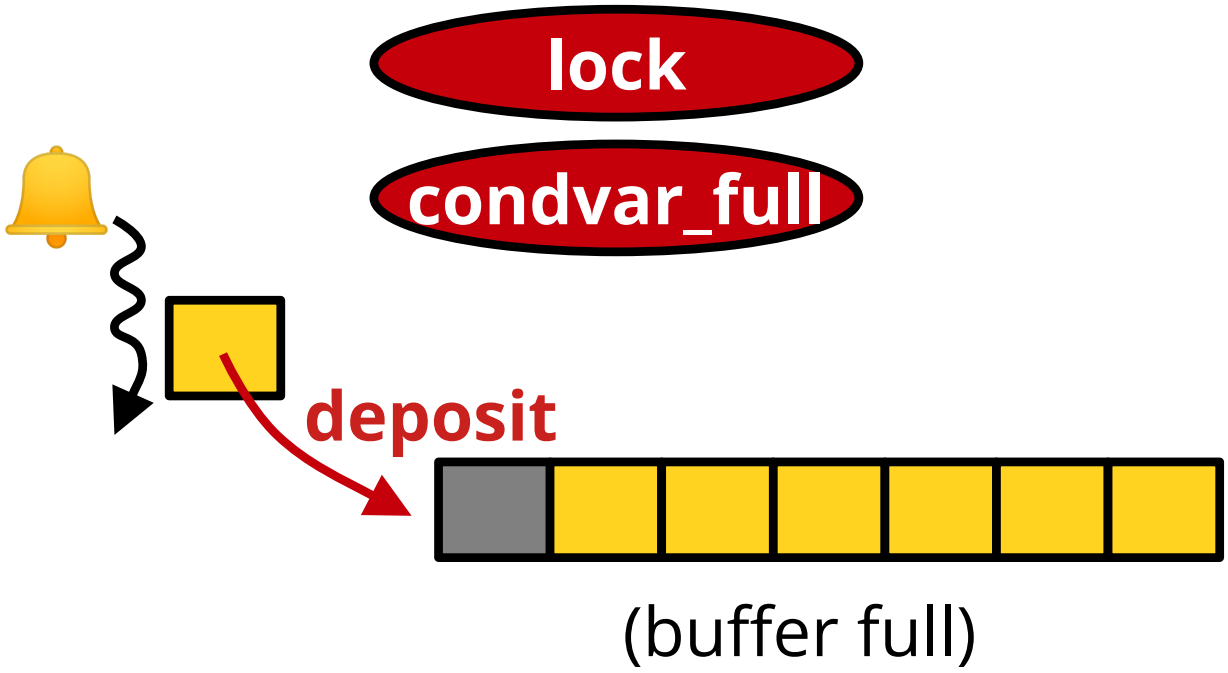
Applying Condition Variables to Our Example



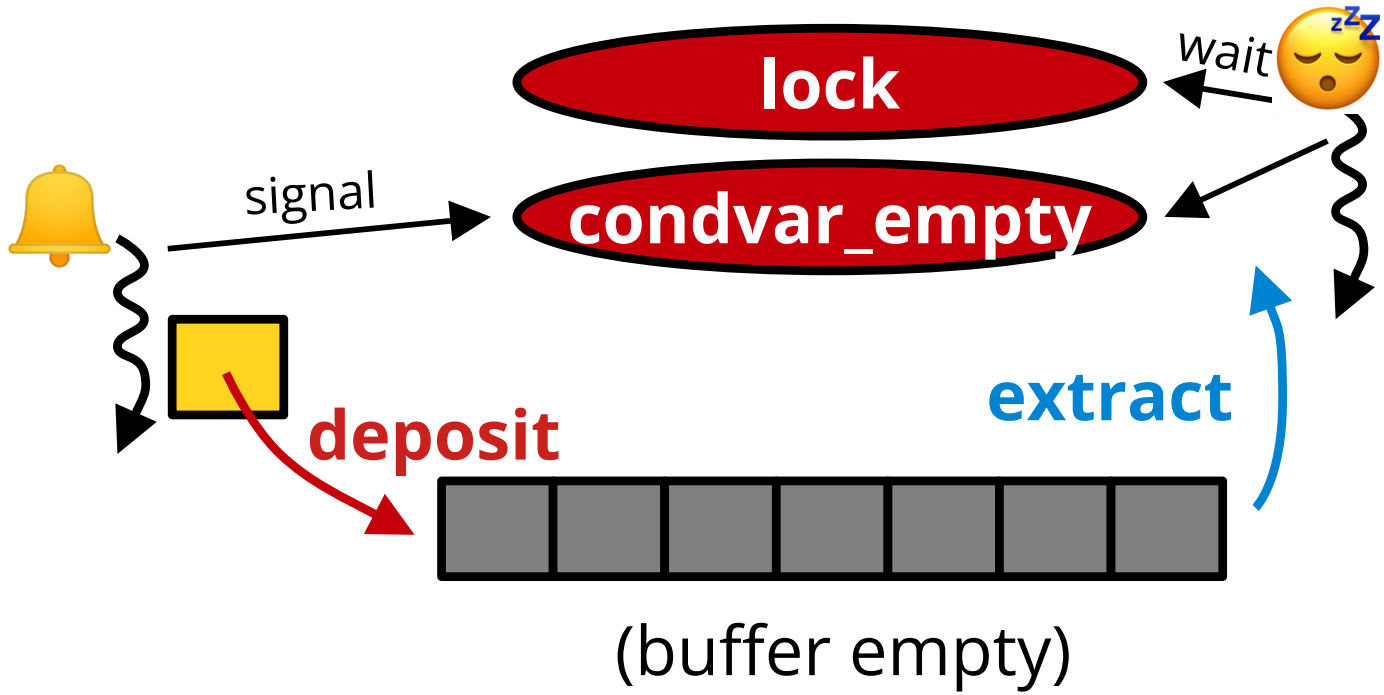
Applying Condition Variables to Our Example



Applying Condition Variables to Our Example



Applying Condition Variables to Our Example



Condition Variables

```c
void deposit(bounded_buffer *b, int message) {
  pthread_mutex_lock(&b->lock);

  int full = (b->count == b->max_elements);
  while(full) {
    // wait on condfull to be signalled
    // when the buffer becomes non-full
    // and the lock is free
    pthread_cond_wait(&b->condfull, &b->lock);
    // we hold the lock here
    full = (b->count == b->max_elements);
  }

  b->buffer[b->in_index] = message;
  b->in_index = (b->in_index + 1) %
    b->max_elements;

  //signal condempty if buf. becomes non-empty
  if(b->count++ == 0)
    pthread_cond_signal(&b->condempty);

  pthread_mutex_unlock(&b->lock);
}
```

```c
int extract(bounded_buffer *b) {
  pthread_mutex_lock(&b->lock);

  int empty = !(b->count);
  while(empty) {
    // wait on condempty to be signalled when
    // the buffer becomes non-empty
    // and the lock is free
    pthread_cond_wait(&b->condempty, &b->lock);
    empty = !(b->count);
  }

  int message = b->buffer[b->out_index];
  b->out_index = (b->out_index + 1) %
    b->max_elements;

  //signal condfull if buf. becomes non-full
  if(b->count-- == b->max_elements)
    pthread_cond_signal(&b->condfull);

  pthread_mutex_unlock(&b->lock);
  return message;
}
```

08b-condition-variables/condvar.c

Summary

- **Condition variables**: useful mechanism for event signalling
  - Avoid delayed wakeup resulting from arbitrary sleeps
  - Avoid wasting CPU cycle due to busy waiting
- Next lecture: more about locks