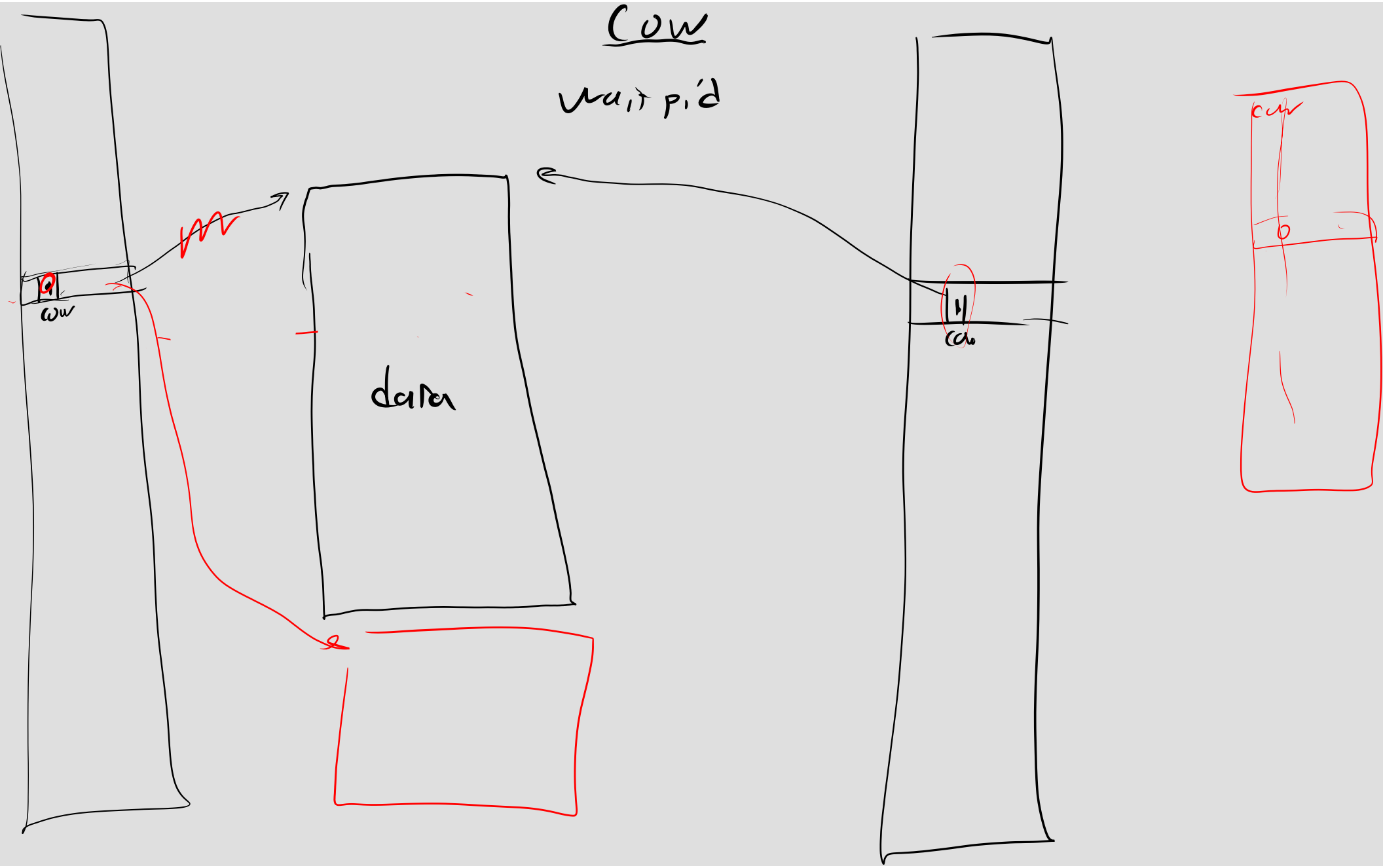
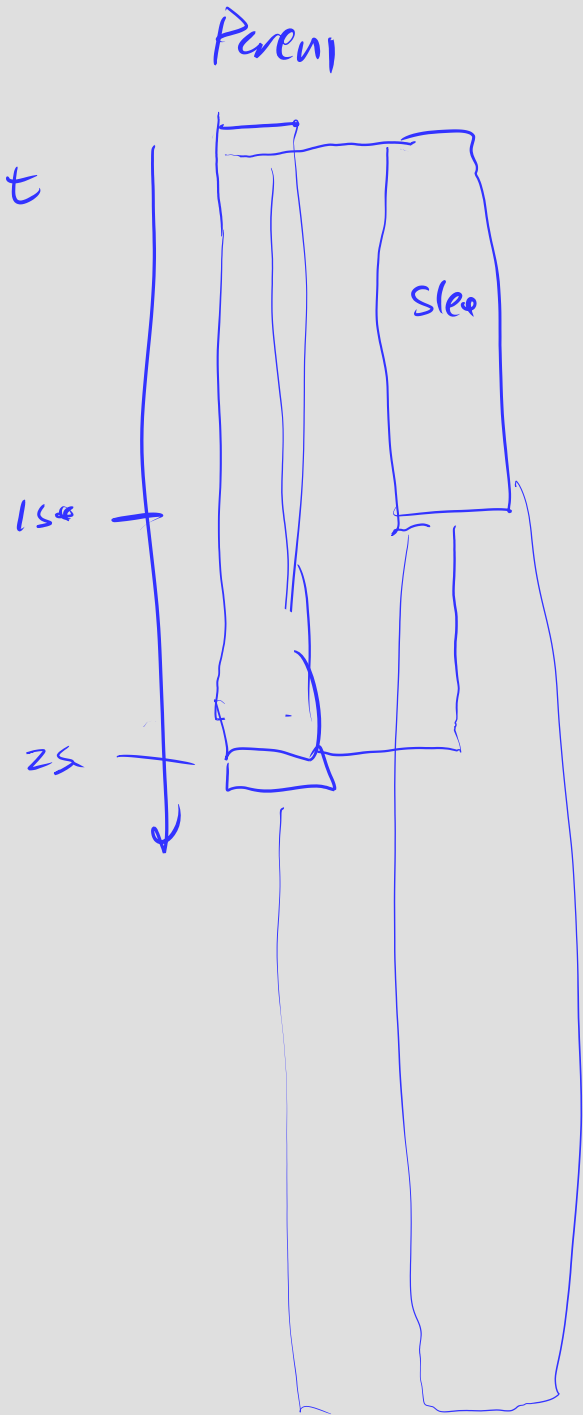

Cow
wait pid
0
cow
data
1
cow
cow

```
pid_t x
if ((x = fork()) == 0) {
    //child
    sleep(10)

} else {
    int retval;
    sleep (2)
    waitpid ( X , &retval , 0);
}
```

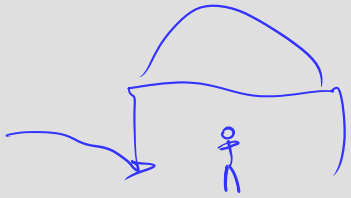


Sychronize

group eat

barrier

- not alone



arrival

MUTEX — mutual exclusion

One at a time

bathroom stall

enter

exit

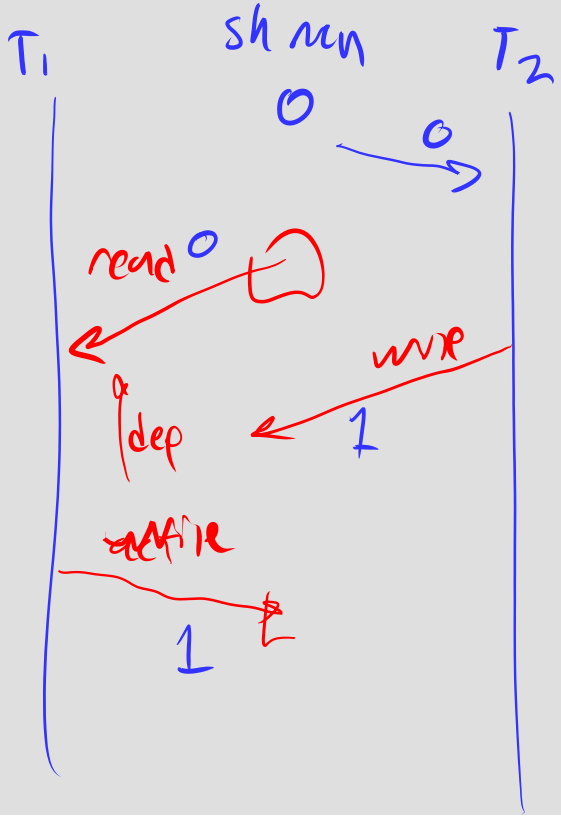
Mutex

race condition

order dependence

Shared memory

↳ write



x += 1

1. 2+ threads

2. non-stack address
   lock
   read
   ↓ dara dep
   write
   unlock

3. One Thread can write that address

mutex
lock
unlock

```
lock (mutex) {
  lock
  while(mutex.locked) {
      // do nothin
  }
  mutex.locked = true
  unlock
  return
}
```

```
t = 1
while (!cas(0, L, t)){}
```

```
asm{
  =
  =
  =
}
```

```
Synchronized (L){

}
```

atomic operation

+=

test and set

success = compare and swap (x , mem_1, mem_2)

```
if (mem1 == x) {
    swap ( mem1 , mem2)
}
```