3/11 - Anderson Queue Lock

Note: Friday will be an experimental Zoom lecture. Remote lectures starting next week :(

We assume the processor gives us one special atomic operation, which takes a variable and increments its current value, and for this to happen atomically
> Don't need to worry about being preempted in the middle of incrementation

- We have an array of booleans, with a slot in that array for each process
- Start with a True for process 0 and a False everywhere else
    - A True means that if you are in this place in line, you are allowed to move forward
    - So processes sit and wait until they see a True in their cell

- Need to know how many threads you have
    - Algorithm breaks if the number of threads is 3
        - Why?
- Using macros rather than functions, because we don't need a return value; simply need to substitute some expression values
- A set of operations wrapped in a d_step is one deterministic transition, vs atomic which can be nondeterministic and the scheduler is told not to preempt it
- Two main variables: next (the current thread) and flag (true/false)
- We also keep track of where we expect the next slot to be, which will help us judge some properties
    - Plus ghost variables that exist only to help us do verification
- `active` means that you don't have to explicitly start them up from an init process, they will be there as soon as the machine starts
- Every process has their own `mySlot` variable, which are separate across all the processes
    - What you're spinning on until you see a `True`
- Before we return to the start of the loop, tell the next process that it can go -- set next slot to True