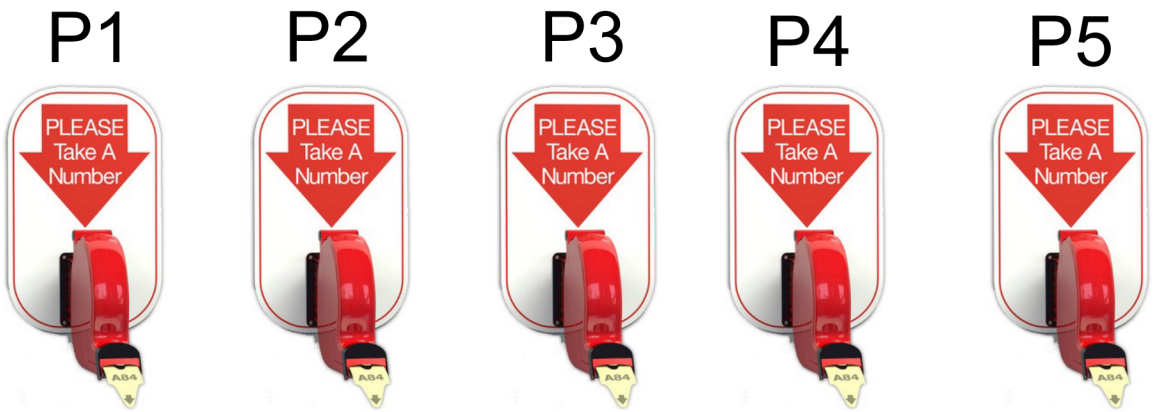
Lamport's Bakery Algorithm

- One general *n*-process solution for mutual exclusion
- Based loosely on "Please take a number for service"
- Assign each requesting process a number
- Smallest number gets the critical section
- One problem: Processes might get the same number, since they choose and increment their value concurrently



Bakery, Cont.

Shared variables / Initialization:

```
// is process i taking a #?
bool choosing[n] = {false};
// what number i did choose?
int number[n] = {0};
```

Bakery, Cont.

- Define a new way to compare the numbers on the tickets so that ties can be broken
- Each process `Pi` has a unique integer identifier `i`
- Use this number to break ties as follows:
  - Define "<" as: `(n1, i) < (n2, j) if (n1 < n2) or n1 = n2 and i < j`
- More notation: Let `max(number[i])` = the maximum element in the `number` array

Bakery, Cont.

```
while (1) {     // each process i
  choosing[i] = true;
  number[i] = max(number) + 1;
  choosing[i] = false;
  for (j = 0; j < n; j++) {
    while (choosing[j]);
      while (number[j] != 0 &&
       (number[j], j) < (number[i], i));
  }
  // critical section
  number[i] = 0;
  // do normal work
}
```

Bakery, Cont.

- Why is it necessary to wait until a process has finished choosing in the Baker's algorithm?

P0: (slow):

```
choosing[0] = true;
number[0] = max(number) + 1;   /*1*/
choosing[0] = false;
```

P1: (fast):

```
choosing[1] = true;
number[1] = max(number) + 1;   /*1*/
choosing[1] = false;
```