Determinism and Race Conditions

Determinism

- Lets define some processes.
- Each prints one letter once as soon as it starts.
- It will then pause for 1 second before printing the same letter again

Print A

Print B

Print C

Print D

Determinism

- Parallel starts all processes together and blocks progression until they are finished

- Spawn starts all processes together without waiting for any to finish

- Sequence will run processes one at a time until they are all done

Determinism

```
process_list = [
    printA(),
    printB(),
    printC(),
    printD()
]
Sequence (process_list)
print(“Done”)
```

Determinism

```
process_list = [
    printA(),
    printB(),
    printC(),
    printD()
]
Sequence (process_list)
print(“Done”)
```

Expected Output:

```
A
A
B
B
C
C
D
D
Done
```

Determinism

```
process_list = [
    printA(),
    printB(),
    printC(),
    printD()
]
Spawn (process_list)
print(“Done”)
```

Determinism

```
process_list = [
    printA(),
    printB(),
    printC(),
    printD()
]
Spawn (process_list)
print("Done")
```

Expected Output:

```
A
B
C
D
Done
A
B
C
D
```

Determinism

```
process_list = [
    printA(),
    printB(),
    printC(),
    printD()
]
Parallel (process_list)
print(“Done”)
```

Determinism

```
process_list = [
    printA(),
    printB(),
    printC(),
    printD()
]
Parallel (process_list)
print(“Done”)
```

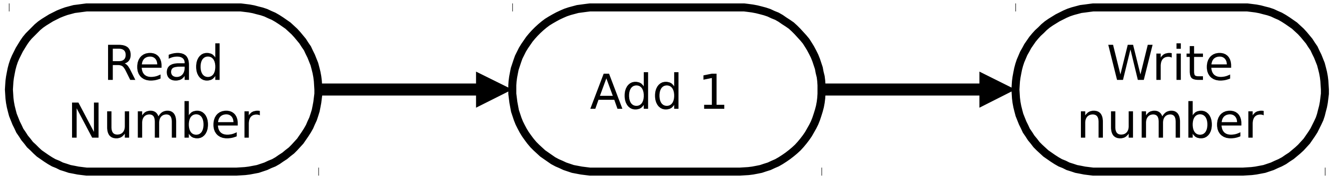
Expected Output:

```
A
B
C
D
A
B
C
D
Done
```

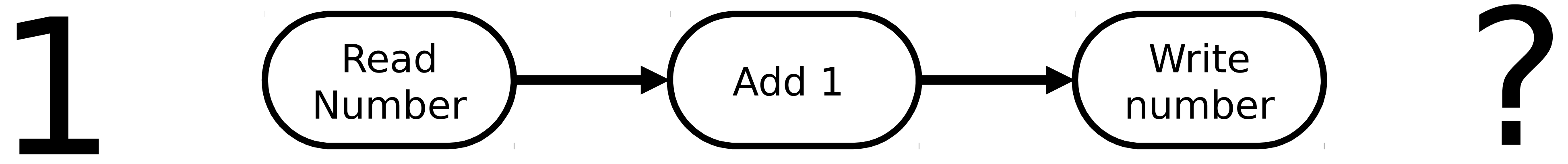
Determinism

- If you can work out the outcome of a programme before it runs, it is *Deterministic* in nature.
- CSP programmes are almost never deterministic.
- Process scheduling is (effectively) hidden and unknowable.

Race Conditions



Race Conditions



1

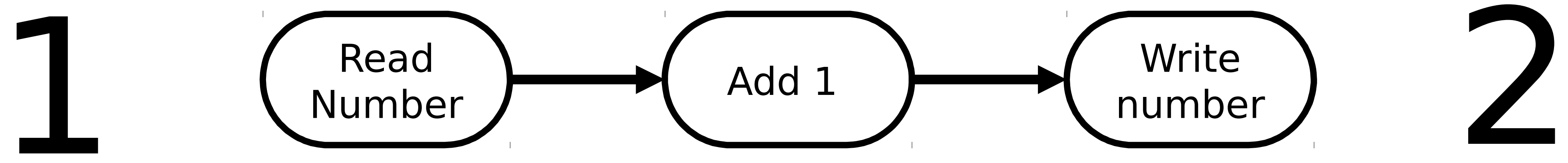
Read Number

Add 1

Write number

?

Race Conditions



1

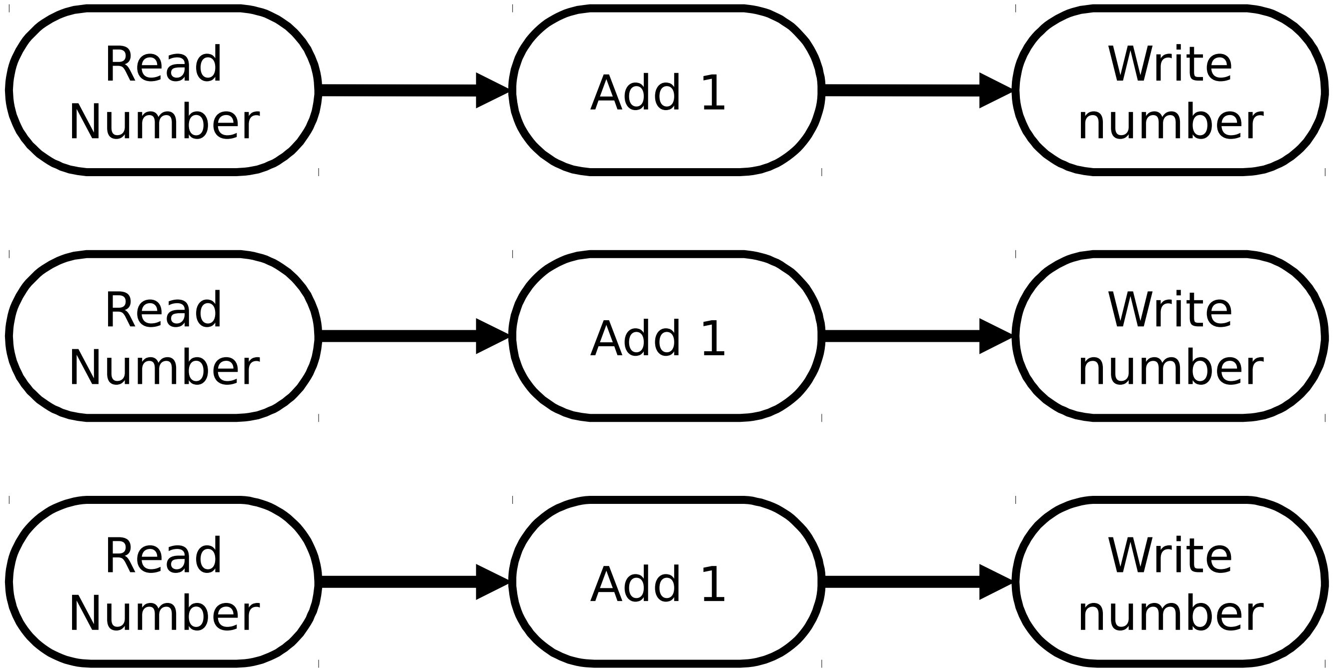
Read Number

Add 1

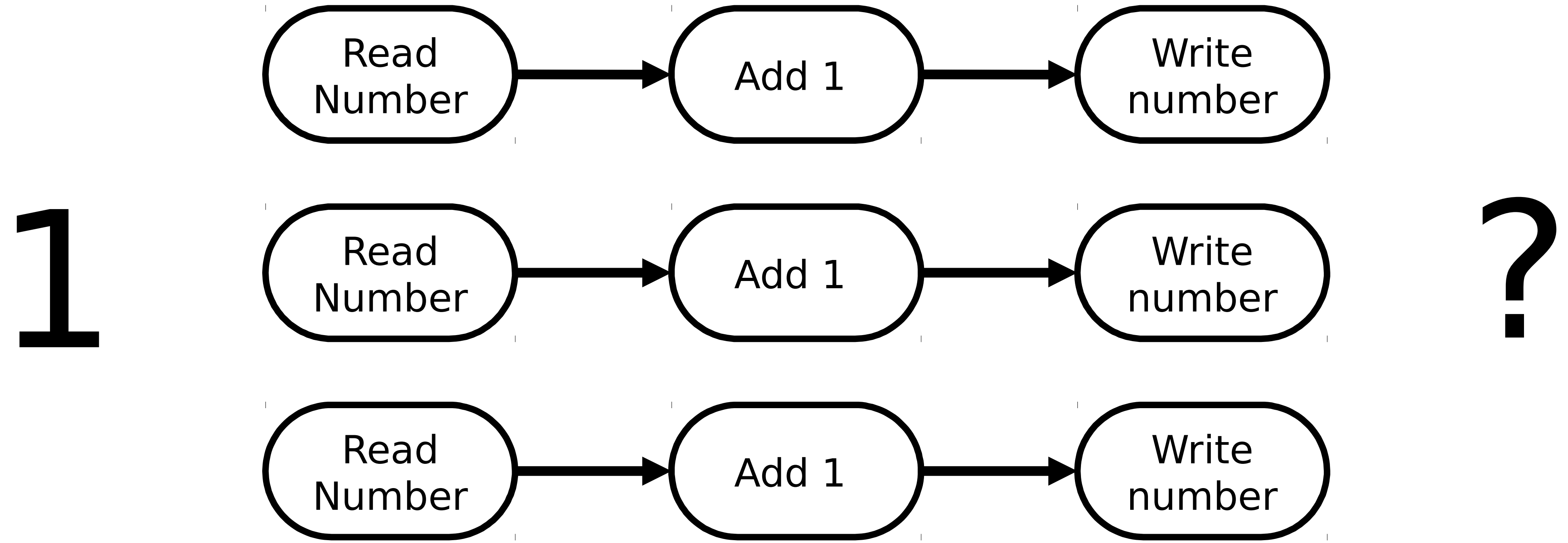
Write number

2

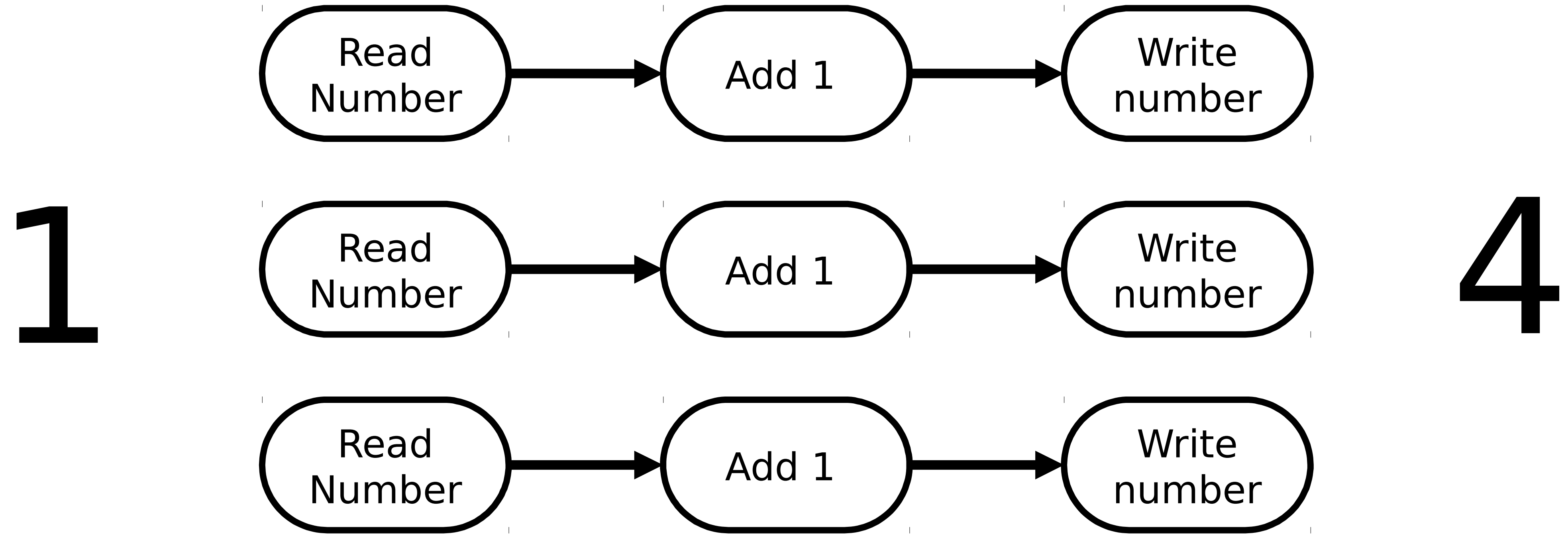
Race Conditions



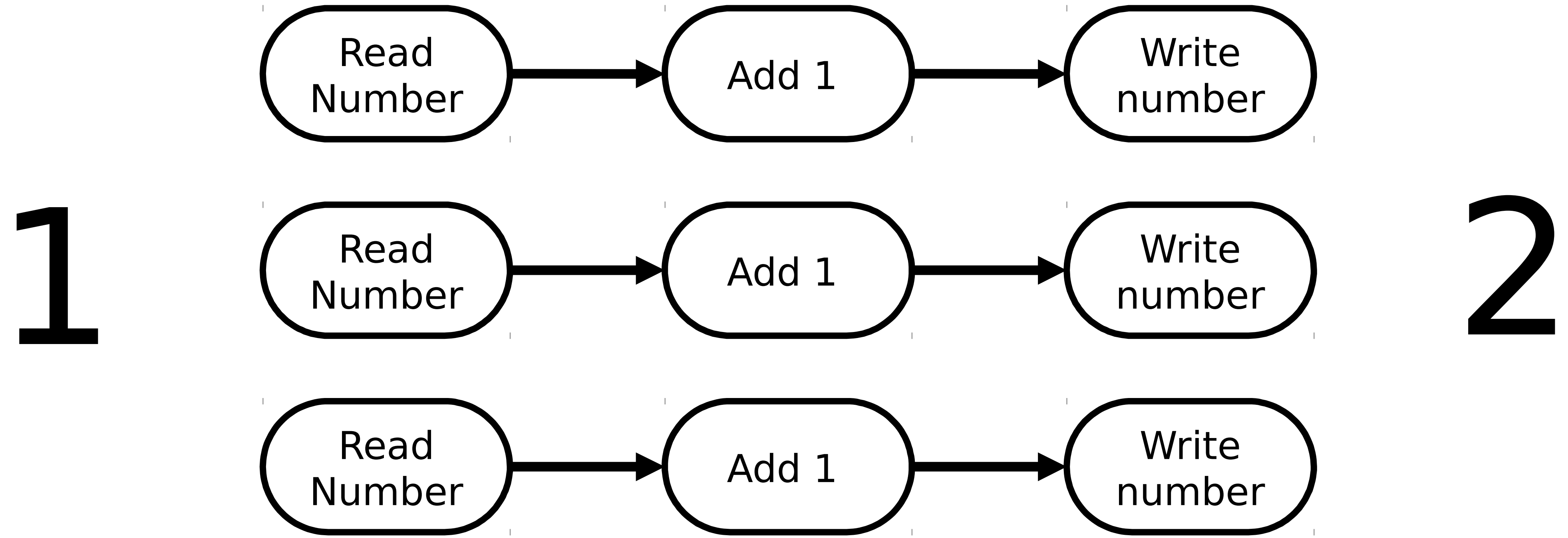
Race Conditions



Race Conditions

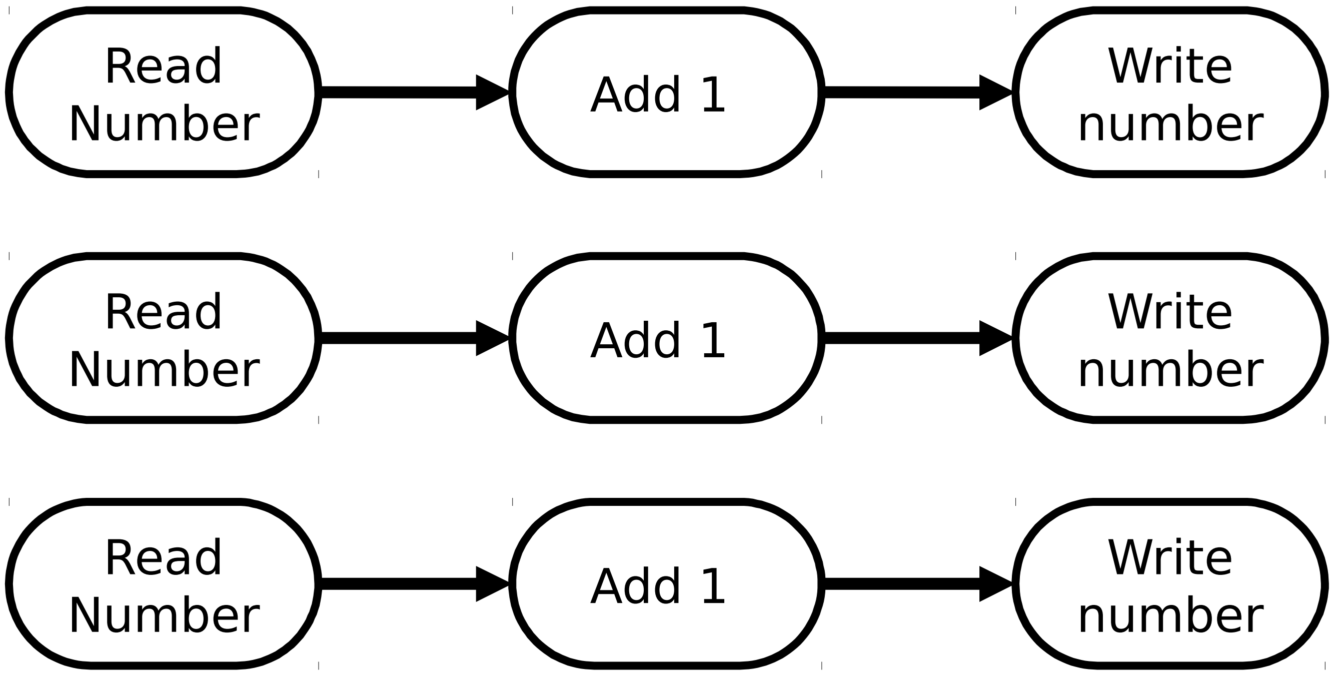


Race Conditions



Race Conditions

1



2-4

Race Conditions

- To avoid race conditions, never share memory.
- You could be cunning and use a system of locks.
- Cunning is one step away from stupid though.
- So just avoid sharing resources.

Debugging CSP

- Systems will break in unexpected ways.
- Systems can be large and difficult to reason about.
- Systems will behave non-deterministically.
- So how can we debug them?

Debugging CSP

- There is no set way!

- Debuggers built into IDEs might help as a starting point, maybe.

- You’ll probably need to rely on print statements.

Debugging CSP

- Print statements are tricky though.

- They can occur out of order.

- They can take a long time to print.

- They will also be overwritten.