semaphores not yet in g++ and clang 10
but should be in 2021!
↳ relevant to assign 5!

```
Semaphore s(10); // 10 permission slips
s.wait();
s.signal();
```

← will wait & block on 0 until a permission slip is available

"thread rendezvous" - a generalization of thread.join()

also: can imitate multiprocessing model & signal that I've gotten to same point in my execution
"pass a baton in a relay race"
like in trace assignment:

parent:
```
waitpid(pid, NULL, WUNTRACED);
```

child
```
raise(SIGSTOP);
```

example: the "internet" implemented as char buffer[8];
reader-writer program

```
int main(—>) {
    char buffer[8];
    thread w(writer, buffer);
    thread r(reader, buffer);
    r.join(); w.join();
    return 0;
}
```

Thread safe! ⟹

```
char buffer[8];
semaphore empty(8);
semaphore full(0);
thread w(writer, buffer, ref(empty), ref(full));
thread r(reader, buffer, ref(empty), ref(full));
w.join(); r.join();
return 0;
```

write can't overwrite any data
reader can't read any redundant data

shared by reference automatically

semaphore& empty,
semaphore& full

```
static void writer(char buffer[]) {
    for (size_t i=0; i<320; i++) {
        char ch = randomChar('A', 'Z');
        cout << oslock << "Writer: " << ch << endl << osunlock;
        buffer[i % 8] = ch;   ← empty.wait();
    }  ↖ full.signal();
}
```

no coordination! writer could be overwriting characters before reader even reads them
want to make sure writer is never more than 8 steps ahead of reader
likewise, reader never more than 0 steps ahead of writer

```
static void reader(char buffer[]) {
    for (size_t i=0; i<320; i++) {   ← full.wait();
        char ch = buffer[i % 8];   ← empty.signal();
        cout << oslock << "\tReader: " << ch << endl << osunlock;
        processData(ch);
    }
}
```

"1 more slot you can write to that won't cause problems"

Load Balancer? → Myth-buster example

sequential vs. concurrent
most of the time on myth spent waiting to be scheduled
function compileCS110ProcessCountMap
very brute force counting sunets in a ps command output