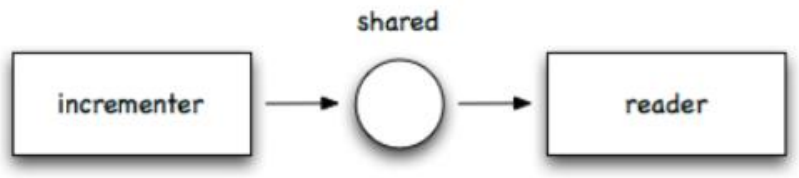
Exercise 2 - Sharing data between threads

In this exercise, there will be created a program that creates two threads; an *incrementer* and a *reader*. They will share an unsigned integer variable, called shared, which is initially 0.
incrementer must increment shared every second.
reader must read shared every second, and send an output to stdout.



Visual representation of how *incrementer* and *reader* must function.

Implementation

The following code is written so support the requirements:

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

using namespace std;

unsigned int shared = 0;

void* incrementer(void* data)
{
  while(1)
    {
      unsigned int* shared = static_cast<unsigned int*>(data);
      (*shared)++;
      //printf("test: %d \n", *(shared));
      sleep(1);
    }
}

void* reader(void* data)
{
  while(1)
    {
      unsigned int* shared = static_cast<unsigned int*>(data);

      printf("Counter is: %d \n", *(shared));
      sleep(1);
    }
}

int main()
{
  // unsigned int shared = 0;

    // define threads
    pthread_t t1,t2;

    // create thread instance
              //pthread_create(&t2, NULL, reader, &shared);
    pthread_create(&t1, NULL, incrementer, &shared);
    pthread_create(&t2, NULL, reader, &shared);
```

```
    // join threads (Wait until threads are done)
    pthread_join(t1,nullptr);
    pthread_join(t2,nullptr);

    return 0;
}
```

The incrementer increments shared while the reader then reads reader and then prints it.

Results

It's obvious that since both functions have a 1 second delay, and they are running at the same time, the reader will not always be able to "keep up".

Discussion

The problem with the code is that *incrementer* and *reader* runs at the same time, and with the same time interval. As the functions take time, it can cause *reader* to read right before or right after *incrementer* has incremented the shared variable.

Filer

Inc_read.png	17,1 KB	2018-09-13	Brian Nymann
threads_2.png	31,5 KB	2018-09-13	Brian Nymann