Promises and Multiple Waiting Threads Solutions

Single Producer with Multiple Consumers

- What is meant by "Single Producer, Multiple Consumers"?
  - One thread which produces a result
  - Many threads which wait for this result
- Give an example where this could be useful
  - Financial application
  - One thread fetches the latest share price
  - Other threads wait to send the latest price to brokers, newsfeeds, trading systems, etc

std::future and Multiple Waiting Threads

- Why is it not safe to use std::future when there are multiple waiting threads?
  - std::future is not designed for this
  - Cannot be copied
  - Assumes it has exclusive access to the shared state
  - Sharing an std::future object across threads creates a data race

Obtaining an std::shared_future object

- Give three ways to obtain a std::shared_future object
  - Move create it from an existing std::future object

    `std::shared_future<int> shared_fut1 =  std::move(fut);`
  - Call share() on a std::future object

    `std::shared_future<int> shared_fut2 = fut.share();`
  - Call get_future() on a std::promise object

    `shared_future<int> shared_fut3 = prom.get_future();`

std::shared_future Example

- Rewrite the program from the previous lecture
- Use multiple consumer threads with std::shared_future