The humble FIFO

Single Consumer, Single Producer

```
template <typename T> class fifo {
public:
    bool push (T && arg) {
        auto pos  = writepos.load();
        auto next = (pos + 1) % slots.size();

        if (next == readpos.load())
            return false;

        slots[pos] = std::move (arg);
        writepos.store (next);
        return true;
    }

    bool pop(T& result) {
        auto pos = readpos.load();

        if (pos == writepos.load())
            return false;

        result = std::move (slots[pos]);
        readpos.store ((pos + 1) % slots.size());
        return true;
    }
private:
    std::vector<T> slots = {}; std::atomic<int> readpos = {0}, writepos = {0};
};
```

David Rowland
Fabian Renn-Giles
Real-time 101

Costs of various FIFOs