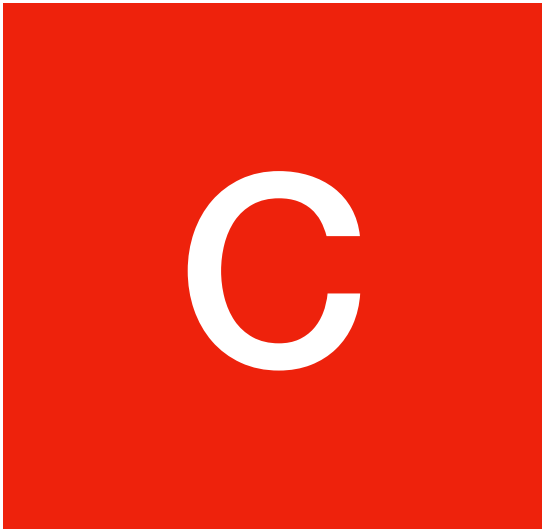
Use two buffers: one for the realtime thread, one for the non-realtime thread

Realtime Copy


A


C

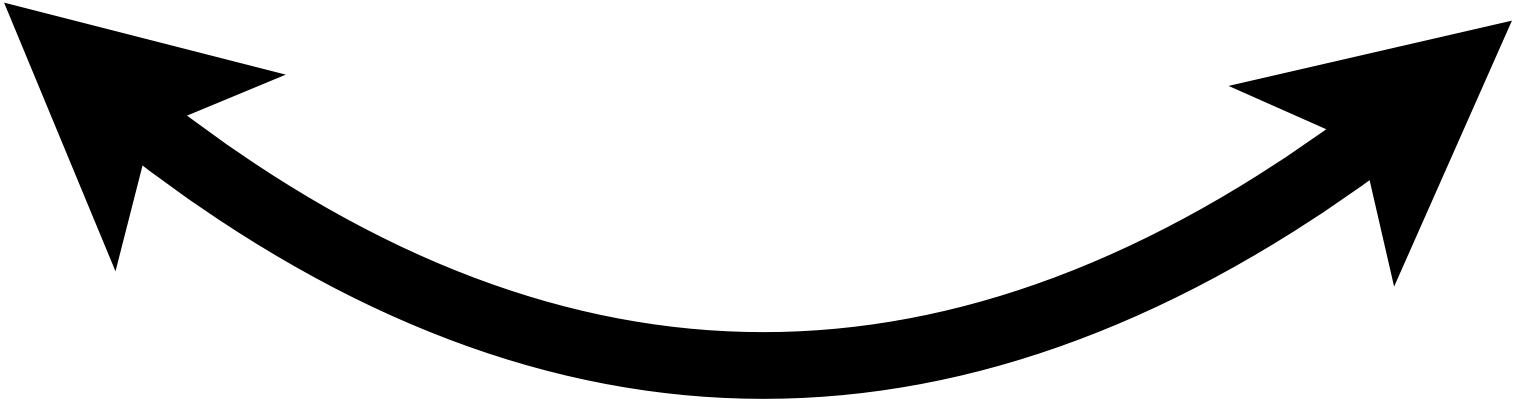
Double Buffering

Non-realtime Copy

Realtime thread only ever writes to this slot.

Non-realtime thread only reads from this slot.

Swap slots just before reading



B

D


E

1. Both slots are pre-initialised with valid data

2. Realtime thread can write to realtime slot without interference

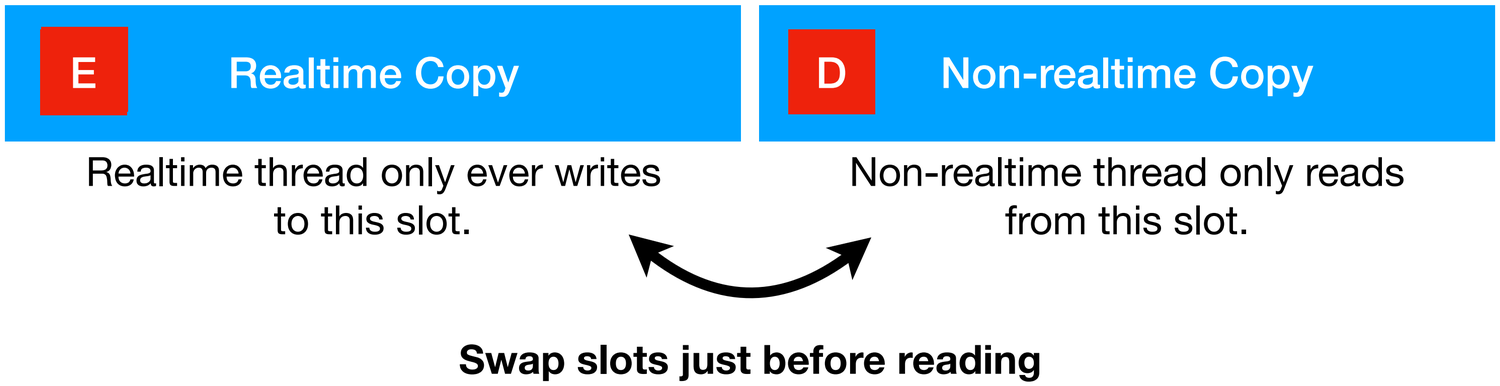
3. When non-realtime thread wants to read data, the slots are swapped

4. Realtime thread can continue to write to realtime thread while non-realtime thread reads

75

Double Buffering

Use two buffers: one for the realtime thread, one for the non-realtime thread



1. Both slots are pre-initialised with valid data
2. Realtime thread can write to realtime slot without interference
3. When non-realtime thread wants to read data, the slots are swapped
4. Realtime thread can continue to write to realtime thread while non-realtime thread reads

Double Buffering

```
using FrequencySpectrum = std::array<float, 512>;

std::array<FrequencySpectrum,2> mostRecentSpectrum;
std::atomic<int> idx = {0};

void processAudio (const float* buffer, size_t n)
{
    auto freqSpec = calculateSpectrum (buffer, n);

    mostRecentSpectrum[idx.load()] = freqSpec;
}

void updateSpectrumUIButtonClicked()
{
    auto i = idx.fetch_xor (1);
    displaySpectrum (mostRecentSpectrum[i]);
}
```

idx denotes current slot of realtime thread (idx XOR 1 denotes slot of non-realtime thread)