Double Buffering

➔ Unfortunately, calling `updateSpectrumUIButtonClicked` twice in a row will show old data!

Realtime thread writes to it's slot (let's say slot 0)

`idx` is swapped. Any new writes now go to slot 1. Old value (`0`) is assigned to `i`.

slot `0` is displayed. As writes now go to slot 1, realtime thread can’t overwrite us while displaying.

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

```cpp
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]);
}
```

55

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)

Realtime thread writes to it's slot (let's say slot `0`)

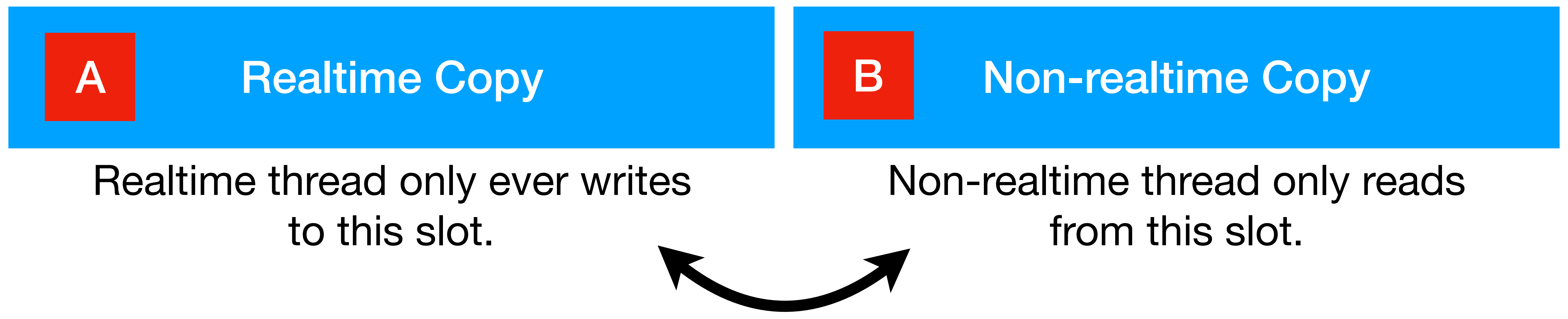
`idx` is swapped. Any new writes now go to slot 1. Old value (`0`) is assigned to `i`.

slot `0` is displayed. As writes now go to slot 1, realtime thread can't overwrite us while displaying.

➔ Unfortunately, calling `updateSpectrumUIButtonClicked` twice in a row will show old data!

Double Buffering

Problem: reading twice in a row:



Realtime thread only ever writes to this slot.

Non-realtime thread only reads from this slot.

Swap slots just before reading