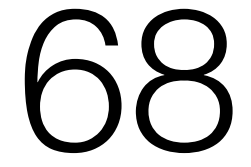
```
struct BiquadCoeffecients  {  float b0, b1, b2, a1, a2; };
BiquadCoeffecients* coeffs;
std::atomic<bool> isInAudioThread { false };

BiquadCoeffecients calculateLowPassCoeffecients (float freq);

void audioThread (const float* src, float* dst, size_t n)
{
    isInAudioThread = true;
    auto* coeffsCopy = coeffs;
    processBiquad (src, dst, n, coeffsCopy);
    isInAudioThread = false;
}

void updateFrequencyParameter (float newValue)
{
    auto* ptr = new BiquadCoeffecients (calculateLowPassCoeffecients (newValue));

    while (isInAudioThread.load())
        ;

    std::swap (ptr, coeffs);
    delete ptr;
}
```

The CAS Exchange Loop

ABA problem!

Spin whilst in audio thread

isInAudioThread could be changed here


68

The CAS Exchange Loop

```
struct BiquadCoeffecients  {  float b0, b1, b2, a1, a2; };
BiquadCoeffecients* coeffs;
std::atomic<bool> isInAudioThread { false };

BiquadCoeffecients calculateLowPassCoeffecients (float freq);

void audioThread (const float* src, float* dst, size_t n)
{
    isInAudioThread = true;
    auto* coeffsCopy = coeffs;
    processBiquad (src, dst, n, coeffsCopy);
    isInAudioThread = false;
}

void updateFrequencyParameter (float newValue)
{
    auto* ptr = new BiquadCoeffecients (calculateLowPassCoeffecients (newValue));

    while (isInAudioThread.load())
        ;

    std::swap (ptr, coeffs);
    delete ptr;
}
```

isInAudioThread could be changed here

ABA problem!

The CAS Exchange Loop

```
struct BiquadCoeffecients  {  float b0, b1, b2, a1, a2; };
std::unique_ptr<BiquadCoeffecients> storage { std::make_unique<BiquadCoeffecients>() };
std::atomic<BiquadCoeffecients*> biquadCoeffs;

void processAudio (float* buffer)
{
    auto* coeffs = biquadCoeffs.exchange (nullptr); // set biquadCoeffs to nullptr while in processing audio

    processBiquad (*coeffs, buffer);


    biquadCoeffs = coeffs;
}

void changeBiquadParameters (BiquadCoeffecients newCoeffs)
{
    auto newBiquad = std::make_unique<BiquadCoeffecients> (newCoeffs);

    for (auto* expected = storage.get(); // spin while the realtime thread is processing
         ! biquadCoeffs.compare_exchange_strong (expected, newBiquad.get());
         expected = storage.get());

    storage = std::move (newBiquad);
}
```