A captured variable in a closure may not live long enough.

Erroneous code example:

```
fn foo() -> Box<Fn(u32) -> u32> {
    let x = 0u32;
    Box::new(|y| x + y)
}
```

This error occurs when an attempt is made to use data captured by a closure, when that data may no longer exist. It's most commonly seen when attempting to return a closure as shown in the previous code example.

Notice that `x` is stack-allocated by `foo()`. By default, Rust captures closed-over data by reference. This means that once `foo()` returns, `x` no longer exists. An attempt to access `x` within the closure would thus be unsafe.

Another situation where this might be encountered is when spawning threads:

```
fn foo() {
    let x = 0u32;
    let y = 1u32;

    let thr = std::thread::spawn(|| {
        x + y
    });
}
```

Since our new thread runs in parallel, the stack frame containing `x` and `y` may well have disappeared by the time we try to use them. Even if we call `thr.join()` within foo (which blocks until `thr` has completed, ensuring the stack frame won't disappear), we will not succeed: the compiler cannot prove that this behavior is safe, and so won't let us do it.

The solution to this problem is usually to switch to using a `move` closure. This approach moves (or copies, where possible) data into the closure, rather than taking references to it. For example:

```
fn foo() -> Box<Fn(u32) -> u32> {
    let x = 0u32;
    Box::new(move |y| x + y)
}
```

Now that the closure has its own copy of the data, there's no need to worry about safety.

This error may also be encountered while using `async` blocks:

```
use std::future::Future;

async fn f() {
    let v = vec![1, 2, 3i32];
    spawn(async { //~ ERROR E0373
        println!("{:?}", v)
    });
}
```

```
fn spawn<F: Future + Send + 'static>(future: F) {
    unimplemented!()
}
```

Similarly to closures, `async` blocks are not executed immediately and may capture closed-over data by reference. For more information, see https://rust-lang.github.io/async-book/03_async_await/01_chapter.html.