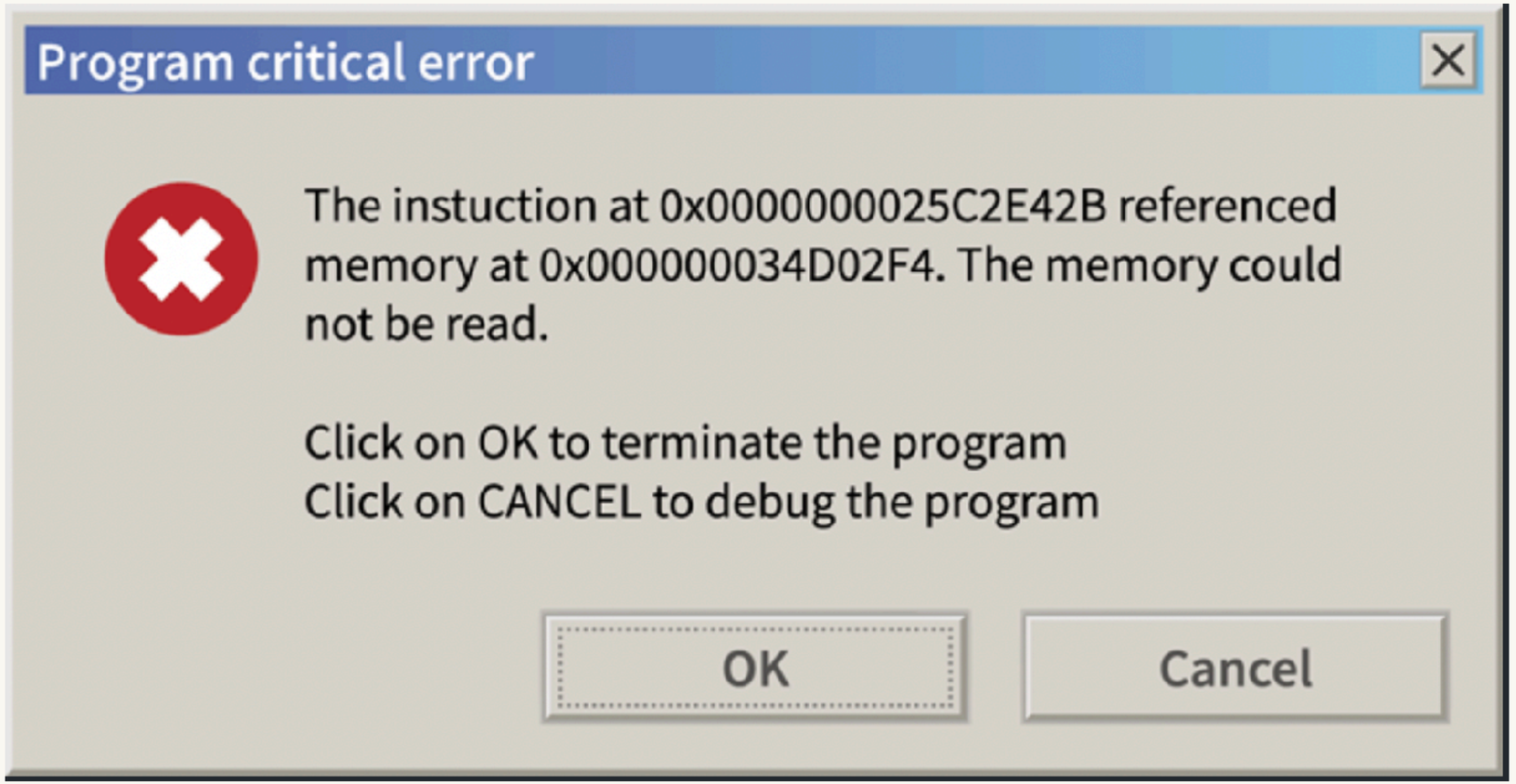
Custom errors


Program critical error
The instuction at 0x0000000025C2E42B referenced memory at 0x000000034D02F4. The memory could not be read.
Click on OK to terminate the program
Click on CANCEL to debug the program
OK
Cancel

Our error principles

1. A **consistent** structure makes it easier for users to scan for key details.
2. Try to **communicate** exactly what went wrong. If you know what right looks like, show that too.
3. Include **context**, like the function call and argument name.
4. Strive to be **concise** so you don't overwhelm the reader, but provide links to more details.

Generating errors

```
# Base R
stop()

# rlang
abort()

# cli
cli::cli_abort()
```

We consider rlang and cli to be “free” dependencies for most packages

cli_abort() makes it easy to mix text and values

```
# Glue interpolation
x ← 10
cli::cli_abort("x ({x}) must be less than 10.")

# With styling
path ← "foo.txt"
cli::cli_abort("{.arg path} ({.path {path}}) doesn't exist.")
cli::cli_abort(
  "{.arg x} must be a string, not {.obj_type_friendly {x}}."
)
# https://cli.r-lib.org/reference/inline-markup.html
```

Pluralisation is a breeze

```
n_files ← 1
cli::cli_abort("Can't supply {n_files} file{?s}.")

n_files ← 2
cli::cli_abort("Can't supply {n_files} file{?s}.")
```

It’s easy to add links

```
cli::cli_abort("See {.url https://cli.r-lib.org} for details.")
cli::cli_abort("See {.fun stats::lm} to learn more.")
cli::cli_abort("See the tibble options at {.help tibble::tibble_options}.")

# Including links that run code
cli::cli_abort("Run {.run testthat::snapshot_review()} to review.")

# More at https://cli.r-lib.org/reference/links.html#hyperlink-support
```

Bulleted lists allow you to present multiple details

```
cli::cli_abort(c(
  "Unexpected content type {.str {content_type}}.",
  "*" = paste0(
    "Expecting {.str {type}}",
     if (!is.null(suffix)) " or suffix {.str {suffix}}",
     "."
  ),
  i = "Override check with {.code check_type = FALSE}."
))


# https://cli.r-lib.org/reference/cli_bullets.html#details
```

Your turn

use_package("cli")

Convert all existing uses of stop() to cli::cli_abort().

Use inline markup (https://cli.r-lib.org/reference/inline-markup.html) where appropriate.

Test your work. Do you need new tests or are you existing tests sufficient?

Ensure R CMD check passes.

Some style notes

```
# We use "must" when you know what is a valid input
dplyr::lag(1:5, "x")
#> Error in `dplyr::lag()`:
#> ! `n` must be a whole number, not the string "x".

# We use "Can't" when you can't state exactly what is expected
dplyr::select(mtcars, xyz)
#> Error in `dplyr::select()`:
#> ! Can't subset columns that don't exist.
#> ✖ Column `xyz` doesn't exist.

# More at <https://style.tidyverse.org/error-messages.html>
```

Error helpers

cli::cli_abort() automatically includes the function name

```
my_function ← function() {
  cli::cli_abort("An error")
}

my_function()
#> Error in `my_function()`:
#> ! An error
```

But what if you write a helper?

```
my_error_helper ← function() {
  cli::cli_abort("An error")
}
my_function ← function() {
  my_error_helper()
}
my_function()
Error in `my_error_helper()`:
! An error
```

Not useful to mention a function that users can't see

```
str_sub("x", 1:2)
#> Error in `recycle()`:
#> ! Can't recycle `arg` to length 1.
```

You need to capture the caller environment and pass it along

```
my_error_helper ← function(call = parent.frame()) {
  cli::cli_abort("An error", call = call)
}
my_function ← function() {
  my_error_helper()
}
my_function()
Error in `my_function()`:
! An error
```

Your turn

use_package("rlang", min_version = "1.0.0") &

update snapshots

Add the call argument to recycle() and check_pattern().

How do the snapshots change?

Verify that R CMD check passes

```
my_error_helper ← function(call = parent.frame()) {
  cli::cli_abort("An error", call = call)
}
```