8_Conditions

2022-11-02

8. Conditions

8.1 Introduction

8.1.1 Prerequisites

```
library(rlang)
```

8.2 Signalling conditions

```
stop("This is what an error looks like")
```

```
## Error in eval(expr, envir, enclos): This is what an error looks like
```

```
warning("This is what a warning looks like")
```

```
## Warning: This is what a warning looks like
```

```
message("This is what a message looks like")
```

```
## This is what a message looks like
```

8.2.1 Errors

```
f <- function() g()
g <- function() h()
h <- function() stop("This is an error!")

f()
```

```
## Error in h(): This is an error!
```

```
h <- function() stop("This is an error!", call. = FALSE)
f()
```

```
## Error: This is an error!
```

```r
h <- function() abort("This is an error!")
f()
```

```
## Error in `h()`:
## ! This is an error!
```

8.2.2 Warnings

```r
fw <- function() {
  cat("1\n")
  warning("W1")
  cat("2\n")
  warning("W2")
  cat("3\n")
  warning("W3")
}
```

```r
# Setting warn to 0 still doesn't fix immediate printing
fw()
```

```
## 1
```

```
## Warning in fw(): W1
```

```
## 2
```

```
## Warning in fw(): W2
```

```
## 3
```

```
## Warning in fw(): W3
```

```r
formals(1)
```

```
## Warning in formals(fun): argument is not a function
```

```
## NULL
```

```r
file.remove("this-file-doesn't-exist")
```

```
## Warning in file.remove("this-file-doesn't-exist"): cannot remove file 'this-
## file-doesn't-exist', reason 'No such file or directory'
```

```
## [1] FALSE
```

```
lag(1:3, k = 1.5)
```

```
## Warning in lag.default(1:3, k = 1.5): 'k' is not an integer

## [1] 1 2 3
## attr(,"tsp")
## [1] -1  1  1
```

```
as.numeric(c("18", "30", "50+", "345,678"))
```

```
## Warning: NAs introduced by coercion

## [1] 18 30 NA NA
```

8.2.3 Messages

```
fm <- function() {
  cat("1\n")
  message("M1")
  cat("2\n")
  message("M2")
  cat("3\n")
  message("M3")
}

fm()
```

```
## 1

## M1

## 2

## M2

## 3

## M3
```

```
cat("Hi!\n")
```

```
## Hi!
```

```
message("Hi!")
```

```
## Hi!
```

8.2.4 Exercises

1. Write a wrapper around file.remove() that throws an error if the file to be deleted does not exist.

```
better.file.remove <- function(file){
  if(file.exists(file)){
    file.remove(file)
  } else{
    stop("File does not exist")
  }
}

better.file.remove("test.txt")
```

```
## Error in better.file.remove("test.txt"): File does not exist
```

```
file.create("test.txt")
```

```
## [1] TRUE
```

```
better.file.remove("test.txt")
```

```
## [1] TRUE
```

2. What does the appendLF argument to message() do? How is it related to cat()?

```
message("No Breaks for me", appendLF = F)
```

```
## No Breaks for me
```

```
message("Now a break", appendLF = T)
```

```
## Now a break
```

```
message("Woo", appendLF = T)
```

```
## Woo
```

```
cat("Hi")
```

```
## Hi
```

```
cat("Hello")
```

```
## Hello
```

Basically you can decide to extend the message you already have by setting appendLF to False. `cat()` by default is similar to `message(appendLF = F)`

8.3 Ignoring conditions

```r
f1 <- function(x) {
  log(x)
  10
}
f1("x")
```

```
## Error in log(x): non-numeric argument to mathematical function
```

```r
f2 <- function(x) {
  try(log(x))
  10
}
f2("a")
```

```
## Error in log(x) : non-numeric argument to mathematical function
```

```
## [1] 10
```

```r
default <- NULL
try(default <- read.csv("possibly-bad-input.csv"), silent = TRUE)
```

```
## Warning in file(file, "rt"): cannot open file 'possibly-bad-input.csv': No such
## file or directory
```

```r
suppressWarnings({
  warning("Uhoh!")
  warning("Another warning")
  1
})
```

```
## [1] 1
```

```r
suppressMessages({
  message("Hello there")
  2
})
```

```
## [1] 2
```

```r
suppressWarnings({
  message("You can still see me")
  3
})
```

```
## You can still see me
```

```
## [1] 3
```

8.4 Handling conditions

```
tryCatch(
  error = function(cnd) {
    # code to run when error is thrown
  },
   code_to_run_while_handlers_are_active
)
```

```
## NULL
```

```
withCallingHandlers(
  warning = function(cnd) {
    # code to run when warning is signaled
  },
  message = function(cnd) {
    # code to run when message is signaled
  },
    code_to_run_while_handlers_are_active
)
```

```
## Error in withCallingHandlers(warning = function(cnd) {: object 'code_to_run_while_handlers_are_activ
```

8.4.1 Condition objects

```
cnd <- catch_cnd(stop("An error"))
str(cnd)
```

```
## List of 2
##  $ message: chr "An error"
##  $ call   : language force(expr)
##  - attr(*, "class")= chr [1:3] "simpleError" "error" "condition"
```

8.4.2 Exiting handlers

```
f3 <- function(x) {
  tryCatch(
    error = function(cnd) NA,
    log(x)
  )
}

f3("x")
```

```
## [1] NA
```

```
f3(10)
```

```
## [1] 2.302585
```

```r
tryCatch(
  error = function(cnd) 10,
  1 + 1
)
```

```
## [1] 2
```

```r
tryCatch(
  error = function(cnd) 10,
  {
    message("Hi!")
    1 + 1
  }
)
```

```
## Hi!
```

```
## [1] 2
```

```r
tryCatch(
  message = function(cnd) "There",
  {
    message("Here")
    stop("This code is never run!")
  }
)
```

```
## [1] "There"
```

```r
tryCatch(
  error = function(cnd) {
    paste0("--", conditionMessage(cnd), "--")
  },
  stop("This is an error")
)
```

```
## [1] "--This is an error--"
```

```r
path <- tempfile()
tryCatch(
  {
    writeLines("Hi!", path)
    # ...
  },
  finally = {
    # always run
    unlink(path)
  }
)
```

8.4.3 Calling handlers

```r
tryCatch(
  message = function(cnd) cat("Caught a message!\n"),
  {
    message("Someone there?")
    message("Why, yes!")
  }
)
```

```
## Caught a message!
```

```r
withCallingHandlers(
  message = function(cnd) cat("Caught a message!\n"),
  {
    message("Someone there?")
    message("Why, yes!")
  }
)
```

```
## Caught a message!

## Someone there?

## Caught a message!

## Why, yes!
```

```r
withCallingHandlers(
  message = function(cnd) message("Second message"),
  message("First message")
)
```

```
## Second message

## First message
```

```r
# Bubbles all the way up to default handler which generates the message
withCallingHandlers(
  message = function(cnd) cat("Level 2\n"),
  withCallingHandlers(
    message = function(cnd) cat("Level 1\n"),
    message("Hello")
  )
)
```

```
## Level 1
## Level 2

## Hello
```

```r
# Muffles the default handler which prints the messages
withCallingHandlers(
  message = function(cnd) {
    cat("Level 2\n")
    cnd_muffle(cnd)
  },
  withCallingHandlers(
    message = function(cnd) cat("Level 1\n"),
    message("Hello")
  )
)
```

```
## Level 1
## Level 2
```

```r
# Muffles level 2 handler and the default handler
withCallingHandlers(
  message = function(cnd) cat("Level 2\n"),
  withCallingHandlers(
    message = function(cnd) {
      cat("Level 1\n")
      cnd_muffle(cnd)
    },
    message("Hello")
  )
)
```

```
## Level 1
```

cnd doesn't pass to tlevel 2 and is muffled by level 1

8.4.4 Call stacks

```r
f <- function() g()
g <- function() h()
h <- function() message("!")
```

```r
withCallingHandlers(f(), message = function(cnd) {
  lobstr::cst()
  cnd_muffle(cnd)
})
```

```
##      x
##   1. +-base::withCallingHandlers(...)
##   2. +-global f()
##   3. | \-global g()
##   4. |   \-global h()
##   5. |     \-base::message("!")
##   6. |       +-base::withRestarts(...)
##   7. |       | \-base withOneRestart(expr, restarts[[1L]])
##    8. |        |   \-base doWithOneRestart(return(expr), restart)
##    9. |        \-base::signalCondition(cond)
##   10. \-global `<fn>`(`<smplMssg>`)
##   11.   \-lobstr::cst()
```

```
tryCatch(f(), message = function(cnd) lobstr::cst())
```

```
##      x
##   1. \-base::tryCatch(f(), message = function(cnd) lobstr::cst())
##   2.   \-base tryCatchList(expr, classes, parentenv, handlers)
##   3.     \-base tryCatchOne(expr, names, parentenv, handlers[[1L]])
##   4.       \-value[[3L]](cond)
##   5.         \-lobstr::cst()
```

8.4.5 Exercises

1. What extra information does the condition generated by `abort()` contain compared to the condition generated by `stop()` i.e. what's the difference between these two objects? Read the help for `?abort` to learn more.

```
catch_cnd(stop("An error"))
```

```
## <simpleError in force(expr): An error>
```

```
catch_cnd(abort("An error"))
```

```
## <error/rlang_error>
## Error:
## ! An error
## Backtrace:
```

abort shows you the backtrace of what triggered the error

2. Predict the results of evaluating the following code

```
show_condition <- function(code) {
  tryCatch(
    error = function(cnd) "error",
    warning = function(cnd) "warning",
    message = function(cnd) "message",
    {
      code
      NULL
    }
  )
}

show_condition(stop("!"))
```

```
## [1] "error"
```

```r
show_condition(10)
```

```
## NULL
```

```r
show_condition(warning("?!"))
```

```
## [1] "warning"
```

```r
show_condition({
  10
  message("?")
  warning("?!")
})
```

```
## [1] "message"
```

“error”, “NULL”, “warning”, “message”

3. Explain the results of running this code:

```r
withCallingHandlers(
  message = function(cnd) message("b"),
  withCallingHandlers(
    message = function(cnd) message("a"),
    message("c")
  )
)
```

```
## b
```

```
## a
```

```
## b
```

```
## c
```

`message("c")` triggers the first handler which calls `message(a)` which triggers the second handler. The second handler then messages “b”, then the message from the first handler comes through with “a” which retriggers the second handler which sends “b” again. Finally the originally message of “c” is printed

4. Read the source code for `catch_cnd()` and explain how it works.

```r
catch_cnd
```

```
## function (expr, classes = "condition")
## {
##     stopifnot(is_character(classes))
##     handlers <- rep_named(classes, list(identity))
##     eval_bare(rlang::expr(tryCatch(!!!handlers, {
##         force(expr)
##         return(NULL)
##     })))
## }
## <bytecode: 0x0000000013071b48>
## <environment: namespace:rlang>
```

First checks to see if classes is provided as a character object. Creates a list object called handlers which holds the different types of conditions to be caught. Then evaluates the expression using `tryCatch`

5. How could you rewrite `show_condition()` to use a single handler?

```
show_condition
```

```
## function(code) {
##   tryCatch(
##     error = function(cnd) "error",
##     warning = function(cnd) "warning",
##     message = function(cnd) "message",
##     {
##       code
##       NULL
##     }
##   )
## }
## <bytecode: 0x000000002146d100>
```

```
show_condition2 <- function(code) {
  tryCatch(
    condition = function(cnd) {
      if (inherits(cnd, "error"))   return("error")
      if (inherits(cnd, "warning")) return("warning")
      if (inherits(cnd, "message")) return("message")
    },
    {
      code
      NULL
    }
  )
}
```