Restarts in Julia

Beyond Exception Handling

Grupo 25

Miguel Parece - ist1103369

Problem Statement & Goals

- **Problem:** Traditional exception handling (e.g., try/catch) aborts computation. Common Lisp’s restarts allow recovery without unwinding the stack.

- **Goal:** Implement Lisp-style restarts in Julia, enabling:
  - Signaling vs. resumable exceptions (signal vs. error).
  - Dynamic restarts (e.g., :return_value, :retry).
  - Interactive user prompts for restarts.

Core Architecture of Exceptional.jl

- **Core Functions:**
  - **to_escape**
  - **handling**
  - **with_restart**

- **Signaling Modes:**
  - **signal**
  - **error**

- **Task-Local Storage:**
  - Stores handlers and restarts dynamically.
  - Enables stack-like management for nested handling/with_restart blocks.
  - **Why?** Allows dynamic scoping, ensuring handlers/restarts are visible only within their lexical scope.
- **to_escape Function:**
  - Uses gensym() tokens to create unique exit points.
  - Prevents accidental cross-scope exits (e.g., ExitException token matching).
  - **Why?** Safely implements non-local exits without leaking scope.
- **Separation of signal vs. error:**
  - signal(exception): Non-fatal; allows propagation if unhandled.
  - error(exception): Fatal; throws an error if no handler is found.
  - **Why?** Provides flexibility for recoverable vs. non-recoverable errors.
- **with_restart / Restart Invocation:**
  - Restarts are stored in task-local storage and invoked via invoke_restart.
  - **Why?** Allows restarts to be defined at lower levels but invoked higher up.

Extensions in ExceptionalExtend.jl

- **restart_data Struct:**
  - Encapsulates restarts with metadata:
  - test: Conditionally enables a restart.
  - report: Describes the restart to users.
  - interactive: Collects user input (
  - funct: The restart's action.
  - **Why?** Aligns with Common Lisp's restarts.

- **Interactive Error Handling:**
  - Overhauled error function:
  - Lists available restarts and prompts users to select one.
  - Example: reciprocal(0) prompts for a value via interactive().
  - **Why?** Mimics Common Lisp's interactive debugging, enhancing usability.