The standard library and error handling

23 October 2026

Instructions

1. Download the starter code file.
2. Implement the functionality described in the next section.
3. Check that the program passes the tests (`cargo test`).
4. Fix all compiler warnings.
5. Find common problems in the code using `cargo clippy` and fix them.

Functionality

1. Write a function
   `divisors(n: NonZero<u32>) -> BTreeSet<NonZero<u32>>` that computes the divisors of the given number.
2. Write a function `assert_sorted(values: &[u32])` that panics if `values` is not sorted. Use the `windows` method and `panic`. Try this function out in `main`.
3. Benchmark the `divisors` function using `time::Instant`, `time::Duration` and `hint::black_box`. Call the function 100 times with the arguments `1..=100` and compute the average running time. Print it in milliseconds, with the fractional part.
4. Write new versions of your favourite functions, in variants for TCP streams:
   `bulk_write(stream: &mut TcpStream, buf: &[u8]) -> io::Result<()>` and
   `bulk_read(stream: &mut TcpStream, size: usize) -> io::Result<Vec<u8>>`
   (reminder: `https://sop.mini.pw.edu.pl/en/sop1/lab/l2/`). Do not use the library functions `write_all` and `read_exact`.
5. Start a TCP server in the `main` function.
6. Listen for new clients in a `for` loop.
7. Pass the stream of each new client to a new function
   `handle_client(stream: TcpStream) -> io::Result<()>`, which:
   (a) handles errors related to the TCP connection with the `?` operator.
   (b) receives from the client the length of a path as a 4-byte big-endian number (`u32::from_be_bytes`), followed by that many bytes of the path. Use `bulk_read` for receiving.
   (c) converts the received bytes to a file path (`String::from_utf8`, `PathBuf::from`).
   (d) Handle the conversion error with `match` (only `String::from_utf8` can return one).

(e) If an error occurs, send the client the string `Bad path\n` (`str::as_bytes`) and return `Ok(())` from the function.

(f) If there is no error, send back the names of the files in that path, each on a separate line, assuming that it leads to a directory (`fs::read_dir`). Use `bulk_write` for sending.

(g) If an error occurs while listing the contents, send the client the string `Bad dir\n`, print the error on the server's console and return `Ok(())`.

Print errors from `handle_client` on the console in `main`.

Requirements

- The code compiles with the stable Rust compiler.
- No warnings from the compiler or `clippy`.
- All tests pass.
- Full implementation of the specified functionality.

Grading (3 pts)

- 1 pt: Work during the laboratory.
- 1 pt: Full functionality (complete implementation meeting the requirements).
- 1 pt: Presentation of the solution and answers to the instructor's questions.