Biblioteka standardowa i obsługa błędów

24 października 2025

Instrukcja

1. Zaimplementuj funkcjonalność opisaną w kolejnej sekcji.
2. Napraw wszystkie ostrzeżenia kompilatora.
3. Znajdź powszechne problemy z kodem używając `cargo clippy` i je napraw.

Funkcjonalność

1. Napisz funkcję
   `divisors(n: NonZero<u32>) -> BTreeSet<NonZero<u32>>`, która dla zadanej liczby wyznacza jej dzielniki.
2. Napisz funkcję `assert_sorted(vec: &Vec<u32>)`, która panikuje, jeśli `vec` nie jest posortowany. Użyj `Vec::windows` i `panic`. Wypróbuj tę funkcję w `main`.
3. Wykonaj benchmark funkcji `divisors` używając `time::Instant`, `time::Duration` i `hint::black_box`. Wywołaj funkcję 100 razy z argumentami 1..100 i oblicz średni czas działania. Wypisz go w milisekundach z ułamkami.
4. Napisz nowe wersje swoich ulubionych funkcji w wariantach dla strumieni TCP:
   `bulk_write(stream: &mut TcpStream, buf: &[u8]) -> io::Result<()>` i
   `bulk_read(stream: &mut TcpStream, size: usize) -> io::Result<Vec<u8>>`
   (przypomnienie: `https://sop.mini.pw.edu.pl/pl/sop1/lab/l1/`). Nie używaj funkcji `write_all` i `read_exact` z biblioteki.
5. Uruchom w funkcji `main` serwer TCP.
6. W pętli `for` nasłuchuj na nowych klientów.
7. Strumień każdego nowego klienta przekazuj do nowej funkcji
   `handle_client(stream: TcpStream) -> io::Result<()>`, która:
   (a) błędy związane z połączeniem TCP obsługuje operatorem `?`.
   (b) odbiera od klienta ciąg bajtów i konwertuje go na ścieżkę do pliku. (`String::from_utf8`, `PathBuf::from_str`).
   (c) Błędy w powyższej konwersji obsłuż przez `match`.
   (d) Jeśli błąd się pojawi, to wyślij klientowi napis `Bad path\n` (`str::as_bytes`) i zwróć z funkcji `Ok(())`.
   (e) Jeśli błędu nie ma, to odeślij listę plików w tej ścieżce, zakładając, że prowadzi do katalogu (`fs::read_dir`).
   (f) Jeśli nastąpi błąd przy wyświetlaniu zawartości, wyślij klientowi napis `Bad dir\n`, wypisz na konsoli serwera otrzymany błąd i zwróć `Ok(())`.

   Błędy z `handle_client` w `main` wyświetlaj na konsoli.

Wymagania

- Kod kompiluje się przy użyciu stabilnego kompilatora Rust.
- Brak ostrzeżeń z kompilatora i `clippy`
- Pełna implementacja zadanej funkcjonalności.

Ocena (3 pkt)

- 1 pkt: Praca w trakcie laboratorium.
- 1 pkt: Pełna funkcjonalność (pełna implementacja zgodna z wymaganiami).
- 1 pkt: Prezentacja rozwiązania i odpowiedź na pytania prowadzącego.