10-commr

目标：实现 comm 命令

实现

- 枚举 Column

Rust

```rust
#[derive(Debug)]
enum Column<'a> {
    Col1(&'a str),
    Col2(&'a str),
    Col3(&'a str),
}
```

- 处理主流程

Rust

```rust

pub fn run(config: Config) -> MyResult<()> {
    let file1 = &config.file1;
    let file2 = &config.file2;

    if file1 == "-" && file2 == "-" {
        return Err(From::from("Both input files cannot be STDIN (\"-\")"));
    }

    let case = |line: String| {
        if config.insensitive {
            line.to_lowercase()
        } else {
            line
        }
    };

    let mut lines1 = open(file1)?.lines().filter_map(Result::ok).map(case);
    let mut lines2 = open(file2)?.lines().filter_map(Result::ok).map(case);

```

```
    let print = |col: Column| {
        let mut columns = vec![];
        match col {
            Col1(val) => {
                if config.show_col1 {
                    columns.push(val);
                }
            }
            Col2(val) => {
                if config.show_col2 {
                    if config.show_col1 {
                        columns.push("");
                    }
                    columns.push(val);
                }
            }
            Col3(val) => {
                if config.show_col3 {
                    if config.show_col1 {
                        columns.push("");
                    }
                    if config.show_col2 {
                        columns.push("");
                    }
                    columns.push(val);
                }
            }
        };

        if !columns.is_empty() {
            println!("{}", columns.join(&config.delimiter));
        }
    };

    let mut line1 = lines1.next();
    let mut line2 = lines2.next();

    while line1.is_some() || line2.is_some() {
        match (&line1, &line2) {
            (Some(val1), Some(val2)) => match val1.cmp(val2) {
                Equal => {
                    print(Col3(val1));
                    line1 = lines1.next();
                    line2 = lines2.next();
                }
                Less => {
                    print(Col1(val1));
                    line1 = lines1.next();
```

```
                    line1 = lines1.next();
                }
                Greater => {
                    print(Col2(val2));
                    line2 = lines2.next();
                }
            },
            (Some(val1), None) => {
                print(Col1(val1));
                line1 = lines1.next();
            }
            (None, Some(val2)) => {
                print(Col2(val2));
                line2 = lines2.next();
            }
            _ => (),
        }
    }

    Ok(())
}
```

- You can choose when to advance any iterator by using `Iterator::next`. For instance, when used with a filehandle, you can manually select the next line.
- You can use match on combinations of possibilities by grouping them into a tuple.（用 tuple 做 match）
- You can use the cmp method of the Ord trait to compare one value to another. The result is a variant of `std::cmp::Ordering`.