03-catr

目标：实现简单版本的 cat 命令，实现 -n 和 -b 的功能

- -n：打印行数
- -b：打印行数，但是行数不计空行

实现

开始采用 TDD 模式，先把测试用例拷贝一份

Rust

```rust
type MyResult<T> = Result<T, Box<dyn Error>>;
```

- Create a MyResult to represent an Ok value for any type T or some Err value that implements the Error trait.（创建一个 MyResult 类型，可以返回任意类型 T 或者返回一个实现了Error trait 的Err）

定义一个参数结构：

Rust

```rust
#[derive(Debug)]
pub struct Config {
    files: Vec<String>,
    number_lines: bool,
    number_nonblank_lines: bool,
}
```

- 继承 Debug trait 可以方便打印结构体
  - 可以使用 `dbg!()`

- `and_then`

Rust

```rust
fn get_args() -> MyResult<Config> {...}
fn run(config: Config) -> MyResult<()> {...}

if let Err(e) = catr::get_args().and_then(catr::run) {
    eprintln!("{}", e);
    std::process::exit(1);
}
```

- 把 Config 直接传给 catr::run

打开文件：

Rust

```rust
// 注意这里用到了Box来创建一个指向堆内存的文件句柄
// 如果不用Box的话会报错
fn open(filename: &str) -> MyResult<Box<dyn BufRead>> {
    match filename {
        "-" => Ok(Box::new(BufReader::new(io::stdin()))),
        _ => Ok(Box::new(BufReader::new(File::open(filename)?))),
    }
}
```

随机生成文件名字：

Rust

```rust
fn gen_bad_file() -> String {
    loop {
        let filename: String = rand::thread_rng()
            .sample_iter(&Alphanumeric)
            .take(7)
            .map(char::from)
            .collect();

        if fs::metadata(&filename).is_err() {
            return filename;
        }
    }
}

```

- `fs::metadata` returns an error when the given filename does not exist, so return the nonexistent filename.（如果文件不存在会返回错误）

Rust

```rust
pub fn run(config: Config) -> MyResult<()> {
    for filename in config.files {
        match open(&filename) {
            Err(err) => eprintln!("Failed to open {}: {}", filename, err),
            Ok(file) => {
                let mut last_num = 0;
                for (line_num, line_result) in file.lines().enumerate() {
                    let line = line_result?;

                    if config.number_lines {
                        println!("{:>6}\t{}", line_num+1, line);
                    } else if config.number_nonblank_lines {
                        if !line.is_empty() {
                            last_num += 1;
                            println!("{:>6}\t{}", last_num, line);
                        } else {
                            println!();
                        }
                    } else {
                        println!("{}", line);
                    }
                }
            },
        }
    }
    Ok(())
}
```

- `Iterator::enumerate`. This method will return a tuple containing the index position and value for each element in an iterable, which is something that can produce values until exhausted（它会返回一个元组，包含下标（从0开始）和对应的值）