04-headr

目标：实现简单版本的 head 命令

-n：显示的行数

-b：显示的字符数

实现

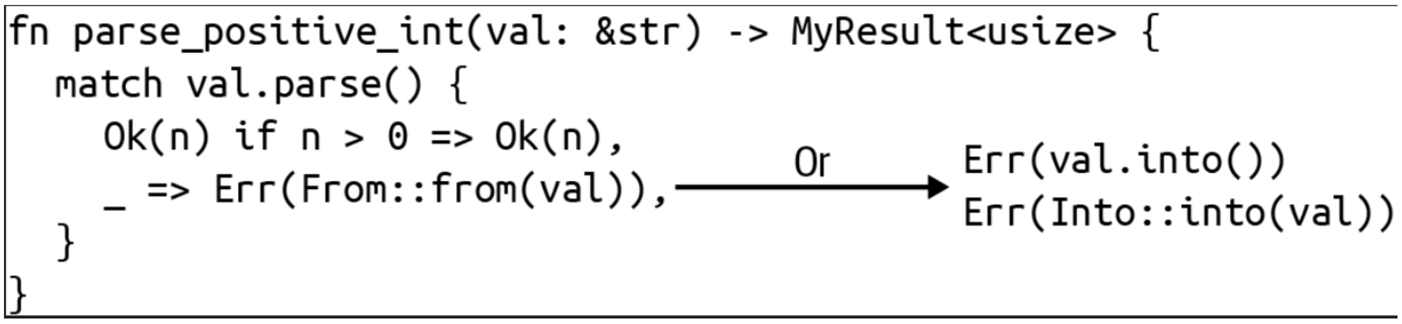
The unimplemented! macro will cause the program to panic or prematurely terminate with the message not implemented.

- 字符串转成数字

Rust

```rust
fn parse_positive_int(val: &str) -> MyResult<usize> {
    match val.parse() {
        Ok(n) if n > 0 => Ok(n),
        _ => Err(From::from(val)),
    }
}
```

- `From::from` to turn the input &str value into an Error.（用 `From::from` 将 &str 转成 Error）
- 也有其他方法：



```rust
fn parse_positive_int(val: &str) -> MyResult<usize> {
  match val.parse() {
    Ok(n) if n > 0 => Ok(n),
    _ => Err(From::from(val)),
  }
}
```

```rust
Err(val.into())
Err(Into::into(val))
```

- 解析参数

Rust

```rust
let lines = matches
    .value_of("lines")
    .map(parse_positive_int)
    .transpose()
    .map_err(|e| format!("illegal line count -- {}", e))?;
```

- `ArgMatches::value_of` returns an `Option<&str>`.
- Use `Option::map` to unpack a &str from Some and send it to `parse_positive_int`.
- The result of `Option::map` will be an `Option<Result>`, and `Option::transpose` will turn this into a `Result<Option>`.
- `map_err` 传入一个函数，如果成功直接透传，失败则会调用传入的函数返回错误

Rust

```rust
let mut handle = file.take(num_bytes as u64);
let mut buffer = vec![0; num_bytes];
let bytes_read = handle.read(&mut buffer)?;
print!(
    "{}",
    String::from_utf8_lossy(&buffer[..bytes_read])
);
```

- converting bytes to characters could fail because strings in Rust must be valid UTF-8. The `String::from_utf8` function will return an Ok only if the string is valid, but `String::from_utf8_lossy` will convert invalid UTF-8 sequences to the unknown or replacement character