11-tailr

目标：实现 tail 命令

实现

- 解析正负数

Rust

```rust
static NUM_RE: OnceCell<Regex> = OnceCell::new();

fn parse_num(val: &str) -> MyResult<TakeValue> {
    let num_re =
        NUM_RE.get_or_init(|| Regex::new(r"^([+-])?(\d+)$").unwrap());

    match num_re.captures(val) {
        Some(caps) => {
            let sign = caps.get(1).map_or("-", |m| m.as_str());
            let num = format!("{}{}", sign, caps.get(2).unwrap().as_str());
            if let Ok(val) = num.parse() {
                if sign == "+" && val == 0 {
                    Ok(PlusZero)
                } else {
                    Ok(TakeNum(val))
                }
            } else {
                Err(From::from(val))
            }
        }
        _ => Err(From::from(val)),
    }
}
```

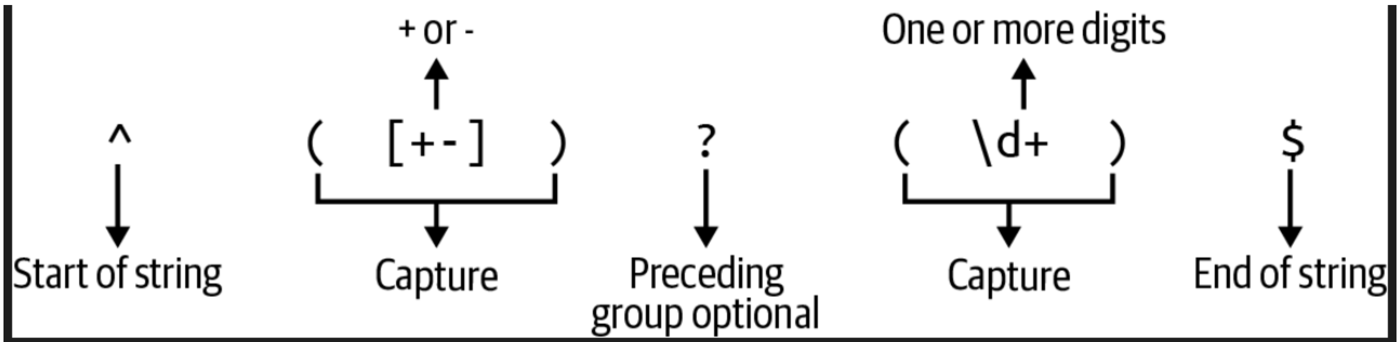

Figure 11-1. This is a regular expression that will match a positive or negative integer.

- 计算文件的行数和字符数

Rust

```rust
fn count_lines_bytes(filename: &str) -> MyResult<(i64, i64)> {
    let mut file = BufReader::new(File::open(filename)?);
    let mut num_lines = 0;
    let mut num_bytes = 0;
    let mut buf = Vec::new();
    loop {
        let bytes_read = file.read_until(b'\n', &mut buf)?;
        if bytes_read == 0 {
            break;
        }
        num_lines += 1;
        num_bytes += bytes_read as i64;
        buf.clear();
    }
    Ok((num_lines, num_bytes))
}
```

- 打印字符

Rust

```rust
fn print_bytes<T: Read + Seek>(
    mut file: T,
    num_bytes: &TakeValue,
    total_bytes: i64,
) -> MyResult<()> {
    if let Some(start) = get_start_index(num_bytes, total_bytes) {
        file.seek(SeekFrom::Start(start))?;
        let mut buffer = Vec::new();
        file.read_to_end(&mut buffer)?;
        if !buffer.is_empty() {
            print!("{}", String::from_utf8_lossy(&buffer));
        }
    }

    Ok(())
}
```