13-calr

目标：实现 cal 命令

实现

- 字符串解析数字

Rust

```rust
fn parse_int<T: FromStr>(val: &str) -> MyResult<T> {
    val.parse()
        .map_err(|_| format!("Invalid integer \"{}\"", val).into())
}
```

- month要转成u32，year要转成i32

- 格式化

Rust

```rust

fn format_month(
    year: i32,
    month: u32,
    print_year: bool,
    today: NaiveDate,
) -> Vec<String> {
    let first = NaiveDate::from_ymd(year, month, 1);
    let mut days: Vec<String> = (1..first.weekday().number_from_sunday())
        .into_iter()
        .map(|_| "  ".to_string()) // two spaces
        .collect();

    let is_today = |day: u32| {
        year == today.year() && month == today.month() && day == today.day()
    };

    let last = last_day_in_month(year, month);
    days.extend((first.day()..=last.day()).into_iter().map(|num| {
```

```
        let fmt = format!("{:>2}", num);
        if is_today(num) {
            Style::new().reverse().paint(fmt).to_string()
        } else {
            fmt
        }
    }));

    let month_name = MONTH_NAMES[month as usize - 1];
    let mut lines = Vec::with_capacity(8);
    lines.push(format!(
        "{:^20}  ", // two trailing spaces
        if print_year {
            format!("{} {}", month_name, year)
        } else {
            month_name.to_string()
        }
    ));

    lines.push("Su Mo Tu We Th Fr Sa  ".to_string()); // two trailing spaces

    for week in days.chunks(7) {
        lines.push(format!(
            "{:width$}  ", // two trailing spaces
            week.join(" "),
            width = LINE_WIDTH - 2
        ));
    }

    while lines.len() < 8 {
        lines.push(" ".repeat(LINE_WIDTH));
    }

    lines
}
```

- Use `Vec::chunks` to get seven weekdays at a time.

Rust

```
let slice = ['l', 'o', 'r', 'e', 'm'];
let mut iter = slice.chunks(2);
assert_eq!(iter.next().unwrap(), &['l', 'o']);
assert_eq!(iter.next().unwrap(), &['r', 'e']);
assert_eq!(iter.next().unwrap(), &['m']);
assert!(iter.next().is_none());
```

- 实现

Rust

```rust
pub fn run(config: Config) -> MyResult<()> {
    match config.month {
        Some(month) => {
            let lines = format_month(config.year, month, true, config.today);
            println!("{}", lines.join("\n"));
        }
        None => {
            println!("{:>32}", config.year);
            let months: Vec<_> = (1..=12)
                .into_iter()
                .map(|month| {
                    format_month(config.year, month, false, config.today)
                })
                .collect();

            for (i, chunk) in months.chunks(3).enumerate() {
                if let [m1, m2, m3] = chunk {
                    for lines in izip!(m1, m2, m3) {
                        println!("{}{}{}", lines.0, lines.1, lines.2);
                    }
                    if i < 3 {
                        println!();
                    }
                }
            }
        }
    }

    Ok(())
}
```

- The `Iterator::zip` method will combine the elements from two iterators into a new iterator containing a tuple of values from the sources. The `itertools::izip` macro allows you to expand this to any number of iterators.

Rust

```rust
let mut results = [0, 0, 0, 0];let inputs = [3, 7, 9, 6];
for (r, index, input) in izip!(&mut results, 0..10, &inputs) {
    *r = index * 10 + input;
}
assert_eq!(results, [0 + 3, 10 + 7, 29, 36]);
```