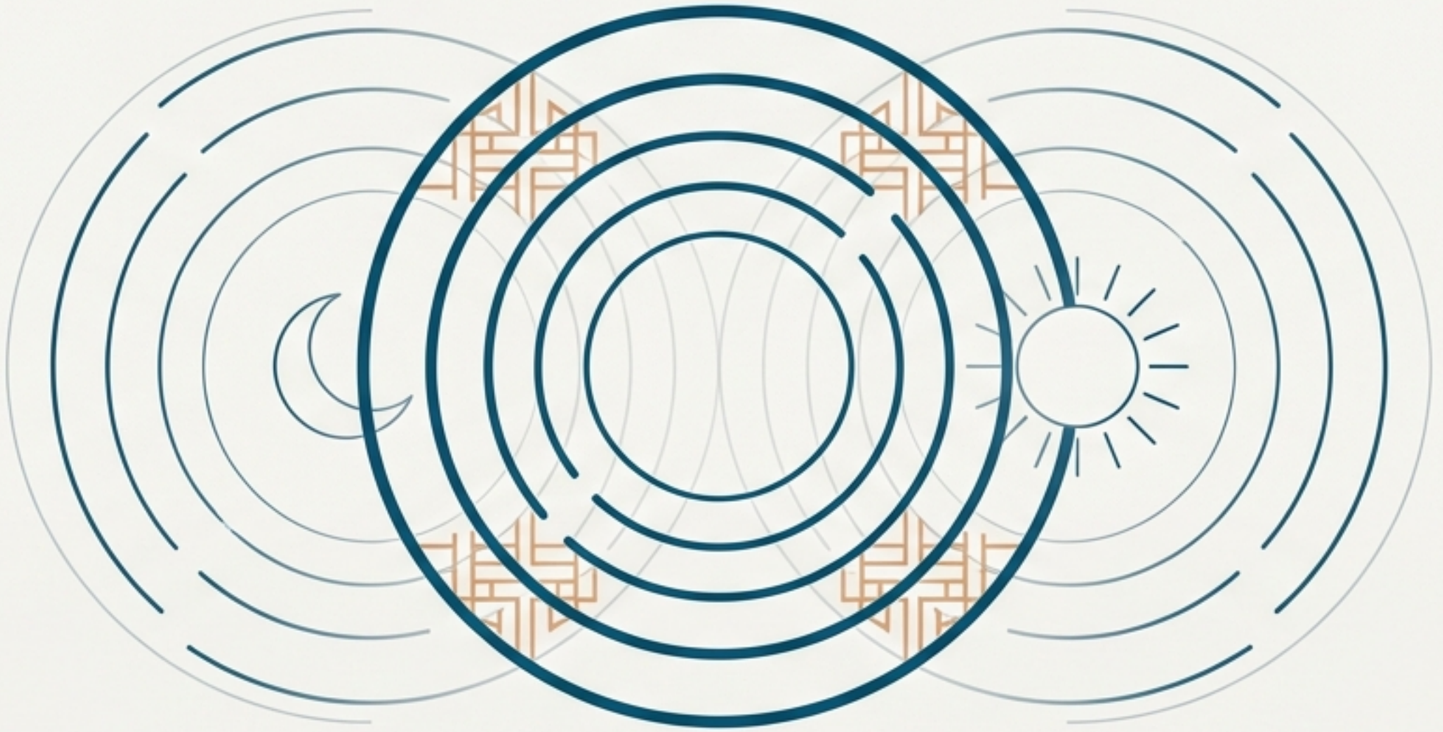
zig-klc: Mastering the Korean Lunisolar Calendar

A Detailed Guide to the Conversion Logic for Developers



A Zig port of the `rs-klc` Rust crate, providing accurate Korean Lunar-Solar calendar conversions.

Rust의 `rs-klc` 크레이트를 Zig로 포팅한, 정확한 한국 양음력 변환 라이브러리입니다.

Algorithm Verified by KASI
Korea Astronomy and Space Science Institute

More Than Just Conversion: A Complete Calendrical Toolkit

Accurate Conversions:
Between Gregorian (Solar) and Korean Lunar calendars.

KASI Verified:
Validated against the Korea Astronomy and Space Science Institute's data.

Gapja (간지):
Calculates the traditional Sexagenary cycle.

Memory Safety:
Leverages Zig's compile-time guarantees.

Intercalary Month (윤달):
Correctly handles lunar leap months.

Day of Week (요일):
Determines the day of the week for any date.

Julian Day Number (JDN):
Built on a foundation of astronomical calculations.

The Foundational Task: Converting Solar to Lunar

This is the most straightforward use case.

You initialize the converter, set a Gregorian (solar) date, and then you can access all corresponding lunar calendar information.

01_basic_conversion.zig

```zig
var converter = klc.LunarSolarConverter.new();

// Set the date to July 10, 2022
if (converter.setSolarDate(2022, 7, 10)) {
    // If the date is valid, you can now access lunar da
    // const lunar_year = converter.lunar_date.year;
    // const lunar_month = converter.lunar_date.month;
    // const lunar_day = converter.lunar_date.day;
}
```

한국어 설명: 양력 날짜를 음력으로 변환하는 가장 간단한 사용 예제입니다.

Finding Solar Dates from the Lunar Calendar

The library easily handles the reverse conversion from lunar to solar.

This is essential for finding the Gregorian date of traditional holidays, such as Seollal (설날), which falls on the first day of the first lunar month.

The `is_intercalary` parameter is crucial for handling leap months.

한국어 설명: 음력 날짜를 양력으로 변환합니다. 한국 설날(정월 초하루)과 같은 음력 기반 명절의 양력 날짜를 찾는 데 유용합니다.

02_lunar_to_solar.zig

```
var converter = klc.LunarSolarConverter.new();

// Find the Solar date for Korean New Year in 2024
// (Lunar date: Year 2024, Month 1, Day 1, not a leap mo
if (converter.setLunarDate(2024, 1, 1, false)) {
    // Now access the corresponding solar date.
    // const solar_year = converter.solar_date.year; //
    // const solar_month = converter.solar_date.month; /
    // const solar_day = converter.solar_date.day; // 10
}
```

Navigating a Key Complexity: The Intercalary Month (윤달)

An intercalary month, or 'Yundal' (윤달), is a leap month inserted into some years of the lunar calendar to keep it synchronized with the solar year. This happens approximately every 19 years.

`zig-klc` provides a direct way to check which month is the leap month for any given year.

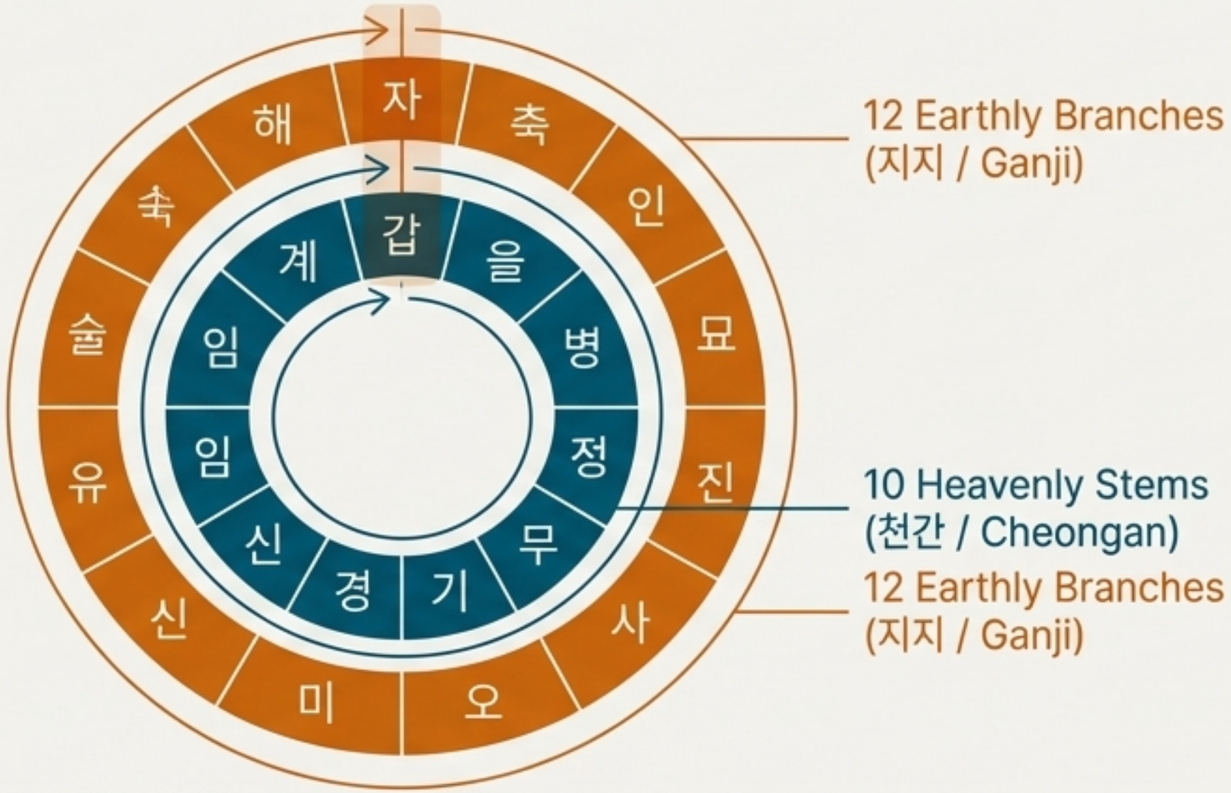
Example: The year 2023 has an intercalary 2nd month (윤 2월).

한국어 설명: 약 19년마다 나타나는 윤달(윤월)을 감지하고 처리하는 방법을 보여줍니다.

03_intercalary_month.zig

```zig
// Check for the intercalary month in
the year 2023.
if (let intercalary_month =
    klc.LunarSolarConverter.getLunar
   IntercalaryMonth(2023)) {
    // `intercalary_month` will be 2.
    // `intercalary_month` will be 2.
    std.debug.print("2023년 윤달: {d}월\n",
      .{intercalary_month});
} else {
    // This year has no intercalary
month.
}
```

Beyond Numbers: The Traditional Sexagenary Cycle (간지)



- The ‘Gapja’ (간지) or **Sexagenary Cycle** is a traditional date-naming system with a 60-year cycle.
- It combines two sub-cycles:
  - 10 Heavenly Stems (천간 / Cheongan)
  - 12 Earthly Branches (지지 / Ganji)
- `zig-klc` can compute the Gapja for the year, month, and day.
- The library provides output in both Korean (Hangul) and Chinese (Hanja) characters.

Bringing Tradition to Code: Calculating Gapja Strings

After setting a date, you can request the fully formatted Gapja strings. Memory management is required as these functions allocate new strings.

04_gapja_sexagenary.zig

```
// Assume `converter` is initialized and a date is set.
// `allocator` is an instance of std.mem.Allocator.

const korean_gapja = try converter.getGapjaString(allocator);
defer allocator.free(korean_gapja);

const chinese_gapja = try converter.getChineseGapjaString(allocator);
defer allocator.free(chinese_gapja);
```

Generated String Output

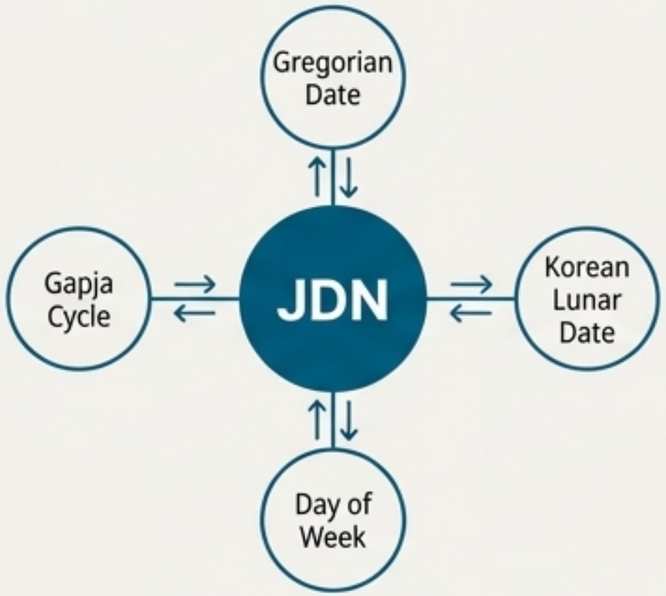
Korean (한글):

임인년 정미월 갑자일

Chinese (한자):

壬寅年 丁未月 甲子日

The Core Engine: The Julian Day Number (JDN)



What is it?

The JDN is an astronomical standard. It is the integer count of days that have passed since noon on January 1, 4713 BC.

Why is it the core?

It provides a single, unambiguous reference point for any date.

All conversions in `zig-klc` are performed by first converting the source date to its JDN, and then converting that JDN to the target calendar system.

Solar Date ⟺ JDN ⟺ Lunar Date

This approach ensures mathematical consistency and accuracy for all calculations.

JDN in Action: Precision and Historical Accuracy

The Gregorian Reform

The library's JDN calculation correctly handles the Gregorian calendar reform of 1582.

This means it knows that the dates **October 5-14, 1582, do not exist**.

This is a crucial detail for any historically accurate calendrical calculation.

05_julian_day_number.zig

```
// Calculate the JDN for a modern date
if (let jdn = klc.LunarSolarConverter.getJulianDayNu
     mber(2022, 7, 10)) {
    // jdn will be 2459771
    std.debug.print("JDN: {d}\n", .{jdn});
}

// The library would correctly handle date
validation around October 1582.
// converter.setSolarDate(1582, 10, 10) // Would
return failure.
```

Reference Point: JDN is the key to converting between different calendar systems.

A Powerful Byproduct of JDN: Calculating the Day of Week

Once the Julian Day Number for a date is known, determining the day of the week is a straightforward mathematical calculation (a modulo operation).

$$(JDN + 1.5) \bmod 7$$

This is far more reliable than complex rule-based algorithms.

The library returns an enum value representing the day, from `Monday` to `Sunday`.

한국어 설명: 율리우스 적일(JDN)을 기반으로 주어진 날짜의 요일을 정확하게 계산합니다.

07_day_of_week.zig

```zig
// What was the day of the week for Feb 10, 2024?
if (let dow = klc.LunarSolarConverter.getDayOfWeek(
     2024, 2, 10)) {
    // `dow` will be the enum value .Saturday
    std.debug.print("요일: {s}\n", .{@tagName(dow)});
}
```

Putting It All Together: A 360° View of a Single Date

February 10, 2024

from 08_comprehensive_example.zig

Solar Date

2024-02-10

Lunar Date

2024-01-01 (Not an intercalary month)

Day of Week

Saturday

Gapja (Korean)

갑진년 병인월 갑오일

Gapja (Chinese)

甲辰年 丙寅月 甲午日

Julian Day Number

2460351

Solar Leap Year?

Yes, 2024 is a leap year.

Lunar Intercalary Month?

No.

Developer Essentials: Safe and Robust Implementation

Error Handling (Date Validation)

The library validates all date inputs. Setting an invalid date will return a failure.

- **Valid Solar Range**: 1391-02-05 to 2050-12-31
- **Valid Lunar Range**: 1391-01-01 to 2050-11-18

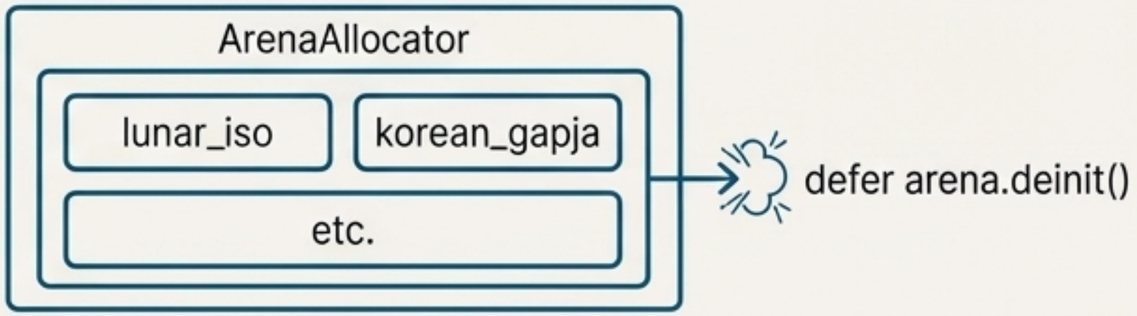
```
if (converter.setSolarDate(2022, 13, 10)) {
    // ... success
} else {
    std.debug.print("Invalid date.\n", .{});
}
```

Memory Management (Zig Idioms)

Functions that return strings require an `**Allocator**`.

This gives the developer full control over memory, preventing leaks.

The standard pattern is to use an `**ArenaAllocator**` for simplicity.




```
var arena = std.heap.ArenaAllocator.init(std.heap.page_allocator);
defer arena.deinit();
const allocator = arena.allocator();

const lunar_iso = try converter.getLunarIsoFormat(allocator);
// Use the string, it will be freed when the arena is deinitialized.
```

The zig-klc Philosophy: Accuracy, Safety, and Elegance

Accuracy

- Built on a solid astronomical foundation (JDN) and verified against the Korea Astronomy and Space Science Institute (KASI).

Safety

- Leverages Zig's explicit memory management and compile-time checks for robust, memory-safe code.

Elegance

- Provides a complete, culturally-aware toolkit for handling lunisolar calendar conversions and traditional date formats.

A reliable engine for any application requiring precise Korean calendrical calculations.

For detailed examples and source code, visit github.com/chunghha/zig-klc