Отчёт по учебной практике на тему "Тестирование информационных систем"

ПОУ "Волго-Вятский колледж информатики, финансов, права и управления"
Выполнил студент ИСП-41
Поздеев Владислав Евгеньевич

Введение

Целью практики было обучение тестированию программного обеспечения. Во время практики я научился писать юнит-тесты по принципу белой коробки на языке программирования Rust. Тестирование это важный этап создания надёжного программного обеспечения, ведь в течении разработки сложной взаимосвязанной системы, результат изменения одной части приложения может затронуть остальные. Этап тестирования позволяет предостеречь появление таких неожиданных багов.

Описание проекта

В ходе разработки парсера для парсинга особого формата конфигурации для VPN.
Конфигурация имеет иерархическую структуру включающую себя:

1. Комментарии в любой части документа
2. Блоки декларации, которые включают в себя:
   1. Константы, и другие декларации
   Такая структура имеет также название "Рекурсивная композиция"[1]
   Из-за того что сложно написать корректный парсер, который парсит всю структуру целиком было принято решение раздробить парсер на отдельные части, которые удобно разрабатывать и тестировать. В качестве методологии разработки была выбрана TDD (test-driven development)[2].

Описание тестов

1. Парсинг комментариев

```
...

pub fn parse_hash_comment(input: &str) -> IResult<&str, &str> {
    let (rest, _) = multispace0(input)?;

    let (rest, result) = preceded(
        tag("#"),
        take_while(|char| char != '\n' && char != '\r')
    )(rest)?;
```

```
    let (rest, _) = multispace0(rest)?;

    Ok((rest, result))
}

pub fn parse_hash_comments0(input: &str) -> IResult<&str, &str> {
    let (rest, _) = many0(parse_hash_comment)(input)?;

    Ok((rest, ""))
}

#[test]
fn test_hash_comment_parsing() {
    assert_eq!(parse_hash_comment("#comment"), Ok(("", "comment")));
    assert_eq!(parse_hash_comment("#comment\r"), Ok(("", "comment")));

    assert_eq!(parse_hash_comment("# comment"), Ok(("", " comment")));
    assert_eq!(parse_hash_comment("#comment\r\n"), Ok(("", "comment")));

    assert_eq!(
        parse_hash_comment("#comment\r\nsome_check"),
        Ok(("some_check", "comment"))
    );

    assert_eq!(
        parse_hash_comment("#comment\nsome_check"),
        Ok(("some_check", "comment"))
    );

    assert_eq!(
        parse_hash_comment("     #comment\nsome_check"),
        Ok(("some_check", "comment"))
    );
}
```

2. Парсинг констант

```
#[derive(PartialEq, Debug, Clone)]
#[pyclass(module = "vpn_config_parser", get_all)]
pub struct Constant {
    r#type: String,
    key: String,
    value: String,
}

pub fn parse_constant(input: &str) -> IResult<&str, Constant> {
    fn take_until_delimiter(input: &str) -> IResult<&str, &str> {
        take_while(|char: char| !matches!(char, ' ' | '\r' | '\n' | '#' |
```

```
'}'))(input)
    }

    let (rest, _) = multispace0(input)?;

    let (rest, r#type) = take_until_delimiter(rest)?;
    let (rest, key) = preceded(space1, take_until_delimiter)(rest)?;
    let (rest, value) = preceded(space1, take_until_delimiter)(rest)?;

    let (rest, _) = multispace0(rest)?;

    Ok((
        rest,
        Constant {
            r#type: r#type.to_string(),
            key: key.to_string(),
            value: value.to_string(),
        },
    ))
}

#[test]
fn test_constant_parsing() {
    assert_eq!(
        parse_constant(&"input someKey someValue"),
        Ok((
            "",
            Constant {
                r#type: "input".to_string(),
                key: "someKey".to_string(),
                value: "someValue".to_string(),
            }
        ))
    );

    assert_eq!(
        parse_constant(&"input someKey someValue rest_things"),
        Ok((
            "rest_things",
            Constant {
                r#type: "input".to_string(),
                key: "someKey".to_string(),
                value: "someValue".to_string(),
            }
        ))
    );

    assert_eq!(
        parse_constant(&"input   someKey  someValue rest_things\r\n"),
        Ok((
```

```
            "rest_things\r\n",
            Constant {
                r#type: "input".to_string(),
                key: "someKey".to_string(),
                value: "someValue".to_string(),
            }
        ))
    );
}
```

3. Парсинг определения

```
pub fn parse_definition_name(input: &str) -> IResult<&str, &str> {
    let (rest, _) = multispace0(input)?;
    let (rest, _) = tag("declare")(rest)?;
    let (rest, result) = delimited(
        multispace1,
        take_while(|char| !matches!(char, ' ' | '\r' | '\n' | '\t' |
'{')),
        multispace0,
    )(rest)?;

    let (rest, _) = char('{')(rest)?;
    let (rest, _) = multispace0(rest)?;

    Ok((rest, result))
}

#[test]
fn test_parsing_declare_definition() {
    assert_eq!(
        parse_definition_name("declare some_name {"),
        Ok(("{", "some_name"))
    );

    assert_eq!(
        parse_definition_name("declare    some_name  \n\r\t
{some_stuff"),
        Ok(("{some_stuff", "some_name"))
    );

    assert_eq!(
        parse_definition_name("declare  \n\t\r  some_name  \n\r\t
{some_stuff"),
        Ok(("{some_stuff", "some_name"))
    );

    assert_eq!(
        parse_definition_name("declare some_name {"),
```

```
        Ok(("{", "some_name"))
    );

    assert_eq!(
        parse_definition_name("declare some_name{"),
        Ok(("{", "some_name"))
    );
}
```

Запуск юнит-тестов

Для запуска тестов необходимо иметь установленный тулчейн rustup.
В консоли в директории проекта необходимо ввести следующую команду:

```
cargo test
```

В результате в консоль выведется следующая информация о результатах проведения тестов:

```
wennerryle@fedora:~/Рабочий стол/projects/vpn_config_parser_core$ cargo
test
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.02s
     Running unittests src/lib.rs (target/debug/deps/vpn_config_parser-
c21bb072b2f8bc76)

running 3 tests
test lexems::declare::constant::test_constant_parsing ... ok
test lexems::hash_comment::test_hash_comment_parsing ... ok
test lexems::declare::definition::test_parsing_declare_definition ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered
out; finished in 0.00s

wennerryle@fedora:~/Рабочий стол/projects/vpn_config_parser_core$
```

Компиляция проекта

Так как парсер является библиотекой для языка программирования Rust, он требует особых этапов компиляции.
Для компиляции под manylinux требуется ввести следующую команду:

```
docker run --rm -v .:/io ghcr.io/pyo3/maturin build --release
```

Для компиляции под windows требуется ввести следующие команды:

```
pip install maturin # если не установлен maturin
python -m venv .venv
source .venv/Scripts/activate
pip install -e
maturin develop -r
```

Заключение

В ходе практики были успешно достигнуты поставленные цели по изучению тестирования программного обеспечения. В процессе разработки парсера для VPN конфигураций был получен практический опыт написания юнит-тестов по методологии "белого ящика" на языке программирования Rust. Применение подхода TDD (разработка через тестирование) позволило эффективно разделить сложную задачу парсинга на маленькие подзадачи и обеспечить надежность каждого компонента системы.

Были успешно реализованы и протестированы ключевые компоненты парсера:

- Парсинг комментариев с различными вариантами форматирования
- Обработка констант с учетом типа, ключа и значения
- Парсинг определений с корректной обработкой пробельных символов

Все разработанные тесты успешно выполняются, что подтверждает корректность работы созданных компонентов. Полученные навыки тестирования и практический опыт применения TDD будут полезны в дальнейшей разработке надежного программного обеспечения.

Использованная литература

1. Design Patterns. Elements of Reusable Object-Oriented Software 1995 by Addison Wesley Longman, Inc. Перевод на русский язык ООО Издательство "Питер", 2022
2. Интернет ресурс Wikipedia https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5 - Разработка через тестирование, дата обращения 10.12.2024