README

项目 Git 仓库地址：https://github.com/lixk28/kat

项目 Git 提交历史：https://github.com/lixk28/kat/commits

环境和依赖

目前该项目只在 OpenSUSE Tumbleweed 下编译、测试过，其他 Linux 发行版未做任何测试。如果你使用的是 RPM 系 Linux Distro，可能会可以直接运行可执行文件。其他发行版大概率不能直接运行，可能会出现共享库路径错误的问题。

下面列出我编译时的工具链：

- gcc (SUSE Linux) 12.1.0
- GNU Make 4.3 Built for x86_64-suse-linux-gnu

不使用 gcc 的话，请使用至少支持 C11 和 GNU Extension 的编译器。

如果你使用 OpenSUSE Tumbleweed 或 OpenSUSE Leap 来构建和测试我的项目，需要使用 `zypper` 安装以下的包和库，以及它们所依赖的库 (会自动安装依赖)：

```
sudo zypper in libgcc_s1-32bit gcc12-32bit glibc-32bit
```

如果你使用 Ubuntu，参考依赖安装如下 (没测试过，不确定能成功)：

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install libc6:i386
sudo apt-get install multiarch-support
sudo apt-get install gcc-multilib
```

下载安装好依赖后，在项目根目录下执行 `make` 即可：

```
make clean && make
```

项目结构

```
.
├── src
│   └── include
└── test
    ├── error
    │   ├── semantic
    │   └── syntax
    ├── lex
    └── parse
```

- `src` 目录下包含源代码文件，`src/include` 包含头文件

- `test` 是测试和示例 `kat` 程序
  - `test/error` 是错误处理的示例程序
    - `test/error/syntax` 是语法错误程序
    - `test/error/semantic` 是语义错误程序
  - `test/lex` 包含词法分析示例程序和输出的 token 文件 `tokens.txt`
  - `test/parse` 包含语法分析示例程序和输出的 AST 文件 `ast.txt`
  - `test` 下有 `hello.kat` 的示例程序，以及生成的汇编 `hello.s` 和可执行文件 `hello`

kat 的使用

使用 `make` 构建完项目后，会在项目根目录下生成 `kat` 可执行文件。

在 `src/main.c` 中，`main` 函数中有以下代码 (非常抱歉时间有限没有做命令行参数解析)

- 运行 **词法分析** 示例程序，注释掉后面两段，parse 和 codegen 部分的代码，然后重新 `make`
- 运行 **语法分析** 或者 **错误处理** 示例程序，注释最后一段，codegen 部分的代码，然后重新 `make`
- 编译运行 `hello.kat`，不用注释任何的以下代码

```
token_t *tokens = lex(buf);
dump_token_list(tokens);

node_t *ast = parse(tokens);
dump_ast(ast);

if (argc >= 3) {
  output_file_path = malloc(sizeof(char) * (strlen(argv[2]) + 5));
  strcpy(output_file_path, argv[2]);
  strcat(output_file_path, ".s");
  output_file = fopen(output_file_path, "w");

  codegen(ast);

  fclose(output_file);

  execl("/usr/bin/gcc", "gcc", "-m32", output_file_path, "-o", argv[2], (char *) NULL);
}
```

下面是测试的具体命令：

- 词法分析

  根据上面的指引 `make` 后，运行下面的命令：

  ```
  ./kat test/lex/lex.kat > test/lex/tokens.txt
  ```

  输出的 token 序列保存在 `test/lex/tokens.txt` 中，实际上已经有这个文件了，如果觉得没有必要可以直接查看 `tokens.txt`。

- 语法分析

  根据上面的指引 `make` 后，运行下面的命令：

```
./kat test/parse/parse.kat > test/parse/ast.txt
```

输出的语法树 AST 保存在 `test/parse/ast.txt` 中，实际上已经有这个文件，如果觉得没有必要可以直接查看 `ast.txt`。

- 错误处理和语义分析

  根据上面的指引 `make` 后，运行下面的命令：

  ```
  ./kat test/error/syntax/<test-file-name>  # syntax error handling test
  # or
  ./kat test/error/semantic/<test-file-name>  # semantic error handling test
  ```

  其中 `<test-file-name>` 是测试文件名。

- 代码生成和运行

  根据上面的指引 `make` 后，运行下面的命令：

  ```
  ./kat test/hello.kat test/hello
  ```

  会在 `test` 下生成 `hello.s` 和可执行文件 `hello`，然后直接运行即可，如果不出意外，会输出下面的结果：

  ```
  $ test/hello
  hello, friends :^)
  -15
  ```

  对于 `hello.kat` 的代码、代码生成、运行时的说明请查看报告 `report.pdf`。