Projektdokumentation

30. November 2014

Inhaltsverzeichnis

1 Architektur

Der Compiler besteht aus mehreren Komponenten:

- *Buffer* - Zuständig für das Lesen aus einer Datei. Der Buffer implementiert, wie der Name sagt, einen eigenen Zeichenpuffer, d.h. er greift nicht auf dem vom Betriebssystem bereitgestellten Puffer zurück. Deshalb kann der Buffer nur eine begrenzte Anzahl von Zeichen rückwärts lesen - nämlich geneu die, die noch im Puffer sind. Man sollte deshalb möglichst nur neue Zeichen lesen und nicht in der Datei zurückspringen.
- *Token* - Die Token des Compilers.
- *Automat* - Produziert Token aus einem Zeichenstrom. Der Automat besitzt einen internen Zustand, der je nach eingelesenem Zeichen wechseln kann. Jedes an den Automaten übergebene Zeichen kann deshalb einen Seiteneffekt haben - man muss die Ausgabe des Automaten also zwischenspeichern wenn man sie öfter benötigt.
- *Symboltable* - Speichert gefundene Identifier. Um erkennen zu können ob ein Identifier schon ein mal gefunden wurde und ob es sich bei dem Identifier um ein Schlüsselwort handelt werden diese in der Symboltable gespeichert. Sie ermöglicht ein schnelles abspeichern und wiederfinden der Identifier. Außerdem gibt sie ein sogenanntes Symbol zurück, das als Referenz auf einen Identifier dient und zusätzliche nützliche Informationen über den Identifier speichert.
- *ErrorHandler* - Speichert die Position und eine Fehlermeldung zu einem gefundenen Fehler. Der ErrorHandler kann gefundene Fehler zusammen mit ihrer Position, der Fehlermeldung und der Zeile, in der der Fehler vorkam, auf der Konsole ausgeben.
- *Scanner* - Kommuniziert mit Buffer, Automat, Symboltable und ErrorHandler um einem Tokenstrom zu produzieren. Dabei nimmt der Scanner so lange Zeichen aus dem Buffer entgegen und gibt diese an den Automat weiter bis dieser einen Token produzieren kann. Der produzierte Token wird daraufhin an die Komponente, die den Scanner aufruft, weitergegeben. Tritt ein Fehler beim produzieren eines Tokens auf, z.B. weil ein ungültiges Zeichen gefunden wurde, dann wird dieser Fehler im ErrorHandler gespeichert und der Scanner gibt ein Error Token aus. Wird ein Identifier Token erkannt prüft der Scanner mit Hilfe der Symboltable ob es sich um ein Schlüsselwort handelt und gibt dementsprechend ein anderes Token zurück. Sobald der Buffer keine weiteren Zeichen mehr zur Verfügung stellt, gibt der Scanner ein EOF Token aus.

2 Wichtige Verzeichnisse und Dateien

```
|-- Automat
|   |-- makefile
|   |-- src
|-- Buffer
|   |-- makefile
|   |-- src
|-- Scanner
|   |-- makefile
|   |-- src
|-- Symboltable
|   |-- makefile
|   |-- src
|-- sharedlib
|-- doc
|-- makefile
|-- run_tests.sh
|-- tests
```

- Die Verzeichnisse *Automat*, *Buffer*, *Scanner*, *Symboltable* sind auf die Verzeichnisse *debug*, *lib* und *src* aufgeteilt, wobei die ersten beiden automatisch vom *makefile* erzeugt werden wenn sie nicht vorhanden sind.
- *sharedlib* beinhaltet alle gelinkten library Dateien, die zur späteren Ausführung des Compilers benötigt werden.
- *doc* beinhaltet die Projektdokumentation.
- *run_tests.sh* ist die Testsuite. Sie durchsucht das Verzeichnis *tests* nach vorhandenen Tests und führt diese aus.
- *tests* beinhaltet alle Tests.

Um das Projekt zu bauen genügt es das oberste *makefile* auszuführen. Um zu überprüfen ob alles funktioniert sollte nach dem Bauen des Projekts zuerst die Testsuite ausgeführt werden.

Möchte man den Compiler manuell ausführen muss man zuallererst die library Dateien exportieren:

```
$ export LD_LIBRARY_PATH=sharedlib:$LD_LIBRARY_PATH
```

2.1 Testsuite

Die Datei *run_tests.sh* verfügt über folgende Funktionalität:

- Alle Dateien in *tests*, die die Endung *.test* besitzen werden als Eingabe für den Compiler verwendet.
- Zu jeder *.test* Datei muss es eine *.check* Datei geben. Diese beinhaltet die Ausgabe des Compilers. Die Testsuite prüft ob der Inhalt dieser Datei mit der tatsächlich produzierten Ausgabe des Compilers übereinstimmt.
- Optional kann noch eine *.flags* Datei angegeben werden. Diese beinhaltet Argumente für die Kommandozeile, die an den Compiler übergeben werden.
- Schlägt ein Test fehl wird eine *.run* Datei angelegt, die die tatsächliche Ausgabe der Compilers beinhaltet.

3 Codedokumentation

3.1 Automat

Die Funktionsweise des Automaten wird durch das folgende Diagramm verdeutlicht: