Systemnahes Programmieren

Michael Bührer 43279, Stefan Fink 43446, Janet Do 47486

May 22, 2016

Contents

1 Aufgabenstellung

Es soll die Funktionsweise eines Scanners sowie die Einordnung innerhalb eines Compilers verstanden werden. Ziel ist es einen funktionieren Compiler zu programmieren. Die Funktionalitäten eines Compilers liegen grundsätzlich im Einlesen einer Datei, verarbeiten des eingelesenen Inputs und deren Auswertung. Die Auswertung besteht darin den Code zu zerteilen und in diesem nach vorgegeben Bezeichner, Grundsymbolen, Zahlen etc. auch Tokens genannt, zu filtern. Kommen Symbole vor, die nicht in unserem Vokabular existieren, so werden sie als ungültig gewertet. Hierfür ist der Zustandsautomat verantwortlich. Wurden nun alle Tokens gefunden, so muss ihre Reihenfolge nach einer korrekten Grammatik überprüft werden. Dies übernimmt der Scanner.

2 Erläuterung

Die Einlese-Komponente wird später der Buffer. Der Buffer gibt die eingelesenen Zeichen an die Hauptkomponente, den Scanner weiter. Der Scanner hat Zugriff auf den Automaten, Buffer und Symboltabelle. Diese Komponente steuert alles. Der Scanner steuert den Buffer an, wenn er mehr Zeichen braucht, gibt diesen an den Automaten weiter, damit dieser sich immer im korrekten Zustand befindet und überprüft permanent, ob ein Token erkannt wurde. Die Symboltabelle beinhaltet unsere Tokens.

3 Scanner

3.1 Buffer

1. Konstruktor
   Im Konstruktor wird die Datei mittels des übergebenen Pfades eingelesen und auf fileDescriptor gespeichert. Ansonsten werden nur die notwendigen Erstinitialisierung, Bufferfüllung und Speicherallozierung vorgenommen.

```
Buffer::Buffer(const char *path) {
    #ifdef NODIRECT
        if ((this->fileDescriptor = open(path, O_RDONLY)) < 0) {
    #else
     if ((this->fileDescriptor = open(path, O_RDONLY | O_DIRECT)) < 0) {
    #endif
                perror("Öffnen fehlgeschlagen");
                exit(-1);
        }
allocMem(&memptr1, BUFFERSIZE, BUFFERSIZE + sizeof(char));
this->buffer1 = (char*)this->memptr1;
this->buffer1[BUFFERSIZE] = '\0';
this->totalBufferIndex = this->currentBufferIndex = 0;
this->column = this->oldColumn = 1;
this->line = 1;
this->atEof = false;

if (this->readBytes(this->buffer1, BUFFERSIZE) == 0) {
        fprintf(stderr, "Datei ist leer.\n");
        exit(-1);
}
        this->currentBuffer = &this->buffer1;
```

}

3.1.1 Erklärung

3.2 Automat

3.3 Scanner

3.4 Symboltabelle