Compiler: Dolmetscher zwischen Mensch und Maschine

Übersetzung von Maschinensprache

Der Mensch beschäftigt sich mit Computern, da diese programmierbar sind und uns viel Rechenaufwand abnehmen (weshalb sie im Deutschen gerne als Rechner bezeichnet werden [Anmerkung beim Schreiben: Bin mir unsicher, woher ich den Wortlaut habe, glaube aber, dass das aus Digitale Systeme oder aus einem anderen Grundlagenmodul kommt, finde es nur passend hier]). Bevor dabei entstehende Programme, sogenannte Software, ausgeführt werden können, müssen sie zunächst zuerst von der Programmiersprache in eine Form überführt werden, die eine Maschine deuten kann. Diese Aufgabe übernimmt der Compiler. Ein Compiler bezeichnet hierbei auschließlich einen Übersetzer, der eine ausführbare Datei produziert (meist eine .exe). Allerdings besteht auch die Möglichkeit die Befehle eines geschriebenen Programmes direkt auszuführen. Bei dieser Art von Übersetzer handelt es sich um einen sogenannten Interpreter (zu deutsch: Interpretierer, also jemand, der etwas deutet). Vorteil bei der Nutzung von interpretierten Skripten (wie Ruby, Perl oder OS-Shells) ist eine schnellere und effizientere Programmierung aufgrund von einfacherer Syntax und besserer Fehlerdiagnose. Allerdings bringen Sprachen dieser Art den Nachteil, dass sie üblicherweise langsamer im Umgang mit Ein- und Ausgabedateien sind. Jedem Programmierer sollte auch ein Präprozessor ein Begriff sein. Dieser kann abgekürzte Begriffe ("shorthands", gerne auch als Macros bezeichnet) in Quellsprachbefehle übersetzen. Ein Compiler übersetzt auch nicht zwangsweise direkt von der Hochsprache in Maschinencode, sondern geht den Zwischenschritt via Assembly, welche ihrerseits mittels eines Assemblers zu Maschinencode übersetzt wird.
Der Aufbau eines Compilers ist in Abbildung (ToDo: backslash:refToDo:Figure 1.6 aus dem Drachen) dargestellt und lässt sich auf abstrakter Ebene recht schnell auf den Punkt bringen. Da es sich bei einem Quelltext immer um reine Textdateien handelt, können solche Texte recht einfach in einzelne Teile, sogenannte Token, aufgrund bestimmter Formalismen (üblicherweise regulären Ausdrücken), zerlegt werden. Hierbei spricht man von der lexikalischen Analyse, deren Benennung sich von dem Wort Lexikon ableitet. Hierbei wird überprüft, ob alle Token (also Wörter) im Text ein Teil der gesamten Menge an zugelassenen Token ist, bei welcher es sich um ein vordefiniertes Quantum handelt. Man kann sich hierbei gut einen Lehrer vorstellen, der eine Klausur korrigiert, und diese zunächst auf Rechtschreibung überprüft. Danach, bzw. dabei würde ebendieser Lehrer auch nach grammatikalischen Fehlern Ausschau halten. In einem Compiler geschieht dies in der syntaktischen Analyse. Neben dem rein Sprachlichen, ist natürlich (und eigentlich vorrangig) der Inhalt einer Klausur relevant. Das Beispiel des Lehrers ist hierbei leider nicht zu einhundert Prozent anwendbar, aber im Prinzip ist das Vorgehen eines Compilers ähnlich deutbar. Die der syntakischen Analyse folgenden semantischen Analyse prüft, ob das Programm in Einheit mit den Sprachparadigma in Hinsicht auf (Variablen- oder

Datei-) Typen ist und sich allgemein an sprachspezifische Vorgaben hält. Wie dies genau umgesetzt wird, wird in den folgenden, namentlich korrespondierenden Kapiteln behandelt.