

Inhaltsverzeichnis

1	Einleitung	1
2	Allgemeines zu lex und yacc	3
2.1	Phasen eines Compilers	3
2.2	Lexikalische Analyse	6
2.2.1	Syntaxanalyse	11
3	lex – Ein Werkzeug für die lexikalische Analyse	25
3.1	Einführendes Beispiel zu lex	26
3.1.1	Aufbau eines einfachen lex-Programms	26
3.1.2	Das Arbeiten mit lex	27
3.2	Die Struktur eines lex-Programms	29
3.2.1	Definitionsteil	29
3.2.2	Regelteil	30
3.2.3	Benutzerdefinierte Routinen	30
3.3	Die lex-Regeln	32
3.3.1	Pattern-Angaben in lex-Regeln	32
3.3.2	Aktionen in den lex-Regeln	48
3.3.3	Zweideutige Regeln	78
3.4	Der Definitionsteil eines lex-Programms	84
3.4.1	Festlegung der Wirtssprache	85
3.4.2	Angabe von C- oder Ratfor-Code	85
3.4.3	Angabe von C- oder Ratfor-Code in <code>%{.}%</code>	86
3.4.4	Reguläre Definitionen	89
3.4.5	Startbedingungen	92
3.4.6	Zeichensatztabellen (nicht unter Linux)	97
3.4.7	Verstellen von lex-internen Tabellengrößen	99
3.5	Benutzerdefinierte Routinen in lex-Programmen	104

3.6	Das Arbeiten mit lex und seine Aufrufsyntax	104
3.6.1	Die lex-Aufrufsyntax	104
3.6.2	Übersetzen eines lex-Programms in einen ablauffähigen Code	106
3.7	Einblick in die Arbeitsweise von lex	107
3.7.1	Fest vorgegebene externe Definitionen	109
3.7.2	Umwandlung des lex-Programms in die Wirtssprache	110
3.7.3	Die Dateien nform und nrform (nicht unter Linux)	114
3.8	lex in Zusammenarbeit mit yacc	122
3.9	lex-Anwendungsbeispiele	125
3.9.1	Analyse der Schachtelung von C-Programmen	125
3.9.2	Erstellen einer Cross-Referenz-Liste für C-Module	128
3.9.3	Realisierung eines einfachen Assemblers	133
3.9.4	Komplexitätsanalyse von Shellskripts	141
3.9.5	Eine Sprache für graphische Darstellungen	143
3.9.6	Taschenrechner mit polnischer Notation	148
3.10	Schlußbemerkungen zu lex	151
4	yacc – Ein Werkzeug für die Syntaxanalyse	153
4.1	Einführendes Beispiel	154
4.1.1	Aufbau eines einfachen yacc-Programms	156
4.1.2	Das Arbeiten mit yacc	160
4.2	Die Struktur eines yacc-Programms	162
4.3	Der Regelteil eines yacc-Programms	166
4.3.1	Aufbau einer Regel	166
4.3.2	Namen in Regeln	166
4.3.3	Literale in Regeln	167
4.3.4	Zusammenfassen von Regeln	169
4.3.5	Leere rechte Seite	169
4.3.6	Kommentare in Regeln	170
4.3.7	Token und nichtterminale Symbole	172
4.3.8	Startsymbol	172
4.3.9	Eingabeende	172
4.3.10	Aktionen in Regeln	174
4.3.11	Fehlerbehandlung	181
	Der Definitionsteil eines yacc-Programms	196
4.4.1	Angabe von C-Code in %{..%}	196
4.4.2	Token-Definition mit %token	199
4.4.3	Festlegung des Startsymbols mit %start	204

4.4.4	Angabe von Prioritäts- und Assoziativitätsbedingungen	205
4.4.5	Typdefinitionen für Rückgabewerte der lexikalischen Analyse und der Aktionen	214
4.5	Benutzerdefinierte Routinen in yacc-Programmen	222
4.6	Die Arbeitsweise von yacc-Parsern	223
4.6.1	Erstellen der Datei y.output	223
4.6.2	Der Parser als Zustandsautomat mit Stack	226
4.7	Konflikte und Mehrdeutigkeiten	233
4.7.1	Konflikte in yacc-Regeln	233
4.7.2	Konfliktanzeige in y.output	238
4.7.3	Vermeiden und Beseitigen von Konflikten	249
4.8	Die Zusammenarbeit von yacc mit der lexikalischen Analyse	275
4.8.1	Kommunikation zwischen lexikalischer Analyse und yacc	275
4.8.2	Programmgenerierung	278
4.8.3	Die Aufgabenverteilung zwischen Parser und Scanner	281
4.9	Umgang mit nicht kontextfreien Grammatiken	289
4.10	Debugging-Möglichkeiten bei yacc	293
4.11	Sonstige yacc-Konstrukte	298
4.11.1	Simulieren von accept und error	298
4.11.2	Zugriff auf Werte in vorherigen Regeln	298
4.11.3	Konstrukte aus früheren yacc-Versionen	301
4.12	Vorschläge zum Stil von yacc-Grammatiken	302
4.13	yacc-Anwendungsbeispiele	303
4.13.1	sort-Generator	303
4.13.2	Automatisches Flechten/Entflechten von komplexen C-Datentypen	309
4.13.3	Taschenrechner mit Intervallarithmetik	315
4.13.4	Menügenerator	320
4.13.5	Profiler für C-Programme	357
4.14	Schlußbemerkungen zu yacc	375
A	Übersicht über die regulären Ausdrücke	377
	Stichwortverzeichnis	379