Practice and Experience of Teaching Reform for Compiler Principle*

SHU Zhong-mei, LI Wen-jun, ZHOU xiao-cong
(School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510275, China)

Abstract: Compiler principle course is the typical representative which perfectly combines the theory and practice among the courses of computer majors. But the contents of the teaching and experimentation of compiler principle are stale all along. The new idea that "language is an effective way to solve problems" is introduced into the compiler principle teaching, the development of experimental curriculum system, library of experimental items, and experimental software equipment are also discussed.

Key words: compiler principle; theory teaching and practice; experiment design
CLC number: TP31 Document code: A Article ID: 0529-6579 (2007) S2-0101-04

1 Preface

Programming Languages play an important role in computer majors. Many research results in the areas such as software engineering, artificial intelligence, database and distributed computing etc. usually are converted into some specific mechanism of a language. The courses such as "Programming", "Compiler Principle" and "Programming Language Fundamentals" have been set up respectively according to the perspective of the user, the implementer, and the designer of a programming language.

Compiler principle course is mainly about the technologies related to the implementation of languages. CC1991 (Computing Curricula 1991) designed by ACM/IEEE-CS has proposed that a qualified graduates majored at Computer Science should grasp 12 important concepts, including "conception and formal model"[1]. On the other hand, from the development of software automation and the present situation, it is easily to be seen that the important conception has been covered and embodied in compiler principle course.

Compiler principle course is the typical representative which perfectly combines the theory and practice among the courses of computer majors. The good results of the formal theory should be emphasized, and the application of these good results to software development should be paid more attention to in the compiler principle course. The models and the algorithms of compiling applied to software architecture, data structure design of auxiliary form, and variant syntax and semantic analysis, have been stereotyped. The contents of compiler principle teaching and its experiments are very stale from the perspective of the students. Especially, the unstructured programming technology adopted in the algorithms of the Compiler principle textbooks make students who learn Object-Oriented Programming from the beginning feel very strange. Based on the above analysis and according to the teaching experience in recent years, a new teaching idea is introduced, and some new, applicable experimentation strategies have been designed in this paper.

2 New Idea of Compiler Principle Teaching

2.1 The Existing Problem

At present, implementing technologies of the high level programming language are emphasized by the main storm compiler principle textbooks, such as the classic one written by A. Aho etc[2]. Compiler principle textbooks in domestic almost adopt this teaching idea. According to this idea, from compiler principle course, students could grasp the whole process of translating a source program written in high level programming language into machine language. On the one hand, they could understand the lexical analysis techniques, its formal language and automaton theory fundamentals. They could grasp top-down and bottom-up syntax analysis methods, such as LL(K), LR(K), and Operator Precedence analysis techniques. They

*收稿日期: 2007-10-28
基金项目: 广东省自然科学基金资助项目 (06300527); 中山大学实验室科研基金资助项目 (YJ200712)
作者简介: 舒忠梅 (1974年生), 女, 博士, 讲师; E-mail: isszm@mail.sysu.edu.cn

could learn the basic idea of semantic analysis, the design and implementation of syntax-directed definition and translation scheme. On the other hand, they could know some basic compiling techniques about runtime memory allocation and management, intermediate code generation, code optimization and target code generation etc.

But there are some obstacles to compiler principle teaching brought by the above guideline which only emphasizes the implementing technologies. For example many teachers and students thought compiler principle course as teaching students how to design and implement a compiler. The research and development of compiler have been professionalized in the software industry. The main compiler products in the world are monopolized by several corporations abroad. Domestic software corporations lagging in system software area seldom have the opportunity to develop a brand-new compiler or to rebuild an existed compiler. All of this make student have no interest and no motivation to learn it. It is even bad that some people in the colleges think compiler principle is useless.

2.2 Language is An Effective Way to Solve Problems

Actually although the main point of compiler principle is about implementing technologies of high level programming languages, these technologies also can be applied to other languages. Therefore, compiling is a processing technique of languages, not only of high level programming languages. Language is well recognized as an important and effective way to solve problems in computer science and technology. Besides high level programming languages, there are many other languages used in computer software area.

The common solution by use of language has many classic examples, such as typesetting languages (eg. LaTex, Postscript, PDF, RTF), database manipulating and query language (SQL), label language (HTML, XML), user defined language etc. In recent years, it is easily to be seen that language is an effective way to solve problems from the development of software technologies. For example, Web service technique which plays an important role in enterprise applications, just use languages to define the service composition (e.g. BPEL) and service query (Xpath). The common languages in computer software area are classified in Fig. 1. Furthermore, the processing of programming languages is not only limited to compiling, but also to explore automatic typeset, static analysis and structured edit etc.

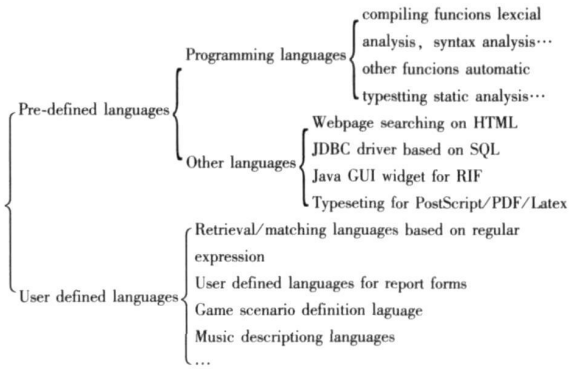

Fig. 1 The classification of common languages processing in computer software area

2.3 New ideas effect on Compiler Principle teaching

Language is an effective way to solve problems, this idea is not brand-new, but is seldom emphasized in the compiler principle teaching. From the teaching practice of recent years, it is found that "language is an effective way to solve problems" is impenetrate in compiler principle teaching, which can explore the perspective of students to research and develop, thus promote their interest and motivation to study.

From the perspective of "pipe/filter" software architecture which commonly adopted by compiler, the front end techniques have broader application than the back end[3]. For example, lexical analysis and syntax analysis which are belong to front end have many applications, but code optimization and target code generation which are belong to back end have few applications. The applications of every compiling process present the shape of shrunken trumpet as shown in Fig. 2.

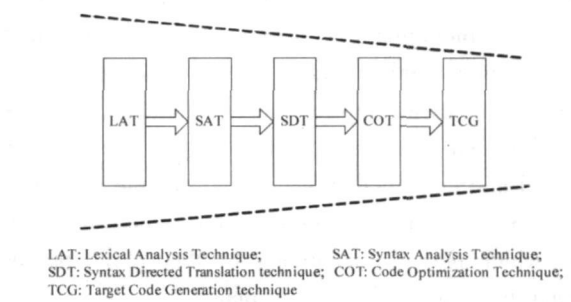

Fig. 2 The applications of compiling process

Based on the new idea of teaching, the emphasis of compiler principle should be put on the front end techniques such as lexical analysis, syntax analysis and syntax directed translation etc, the back end techniques such as intermediate code generation, code optimization and target code generation should be simplified. Furthermore, the corresponding experiments should emphasize the kernel role of languages, that is how to

solve some practical problems by use of languages is determined firstly and then how to process the language by use of the techniques learning from compiler principle is considered as well.

3 Design of Compiler Principle Experiments

Experimental teaching plays an important role in the cultivation of high-quality IT professionals. With the guideline of compiler principle teaching has been innovated, many students states that their interest to learning compiler principle is far beyond the expectation. From this course, they really and truly understood how to apply the compiler principle and techniques to the various areas such as enterprise computing, interactive game, and development of software tools etc. Experimentation reform should correspond with theory teaching reform, both of them can be promoted with each other. As for experimentation, the following methods are adopted.

3.1 Layout Basic Experiments and Integrated Experiments Reasonably

In the new experimentation scheme, compiler principle experiments are classified into two parts, one is the basic experiments which can use compiling techniques directly, and the other is the integrated ones which should combine compiling techniques and some related techniques learned from other courses such as software engineering, database, and parallel and distributed computing etc.

The basic experiments can be carried out through "experiment on compiler principle" course, which should be conducted with the theory course at the same time. (The periods of the basic experiments are about 25% of the total theory course). The basic experiments that can strengthen students to understand the compiler principle methods, techniques and tools are elaborately embodied by a large and integrated experiment, which covers all the important knowledge of the theory course, including grammar and BNF, lexical analysis, syntax analysis and syntax directed translation, and automatic compiling tools.

In order to improve the quality of the basic experiments, the design of the basic experiments takes many aspects into account, such as the integration, applicability, research and exploration etc. All of the basic experiment is to request students to finish a general project which covers the important knowledge of the theory course. The experiment project is divided into 5 sub projects, and each of them are further refined to several experiment steps.

The integrated experiments emphasize particularly on large, synthesized, applicable, innovative, and even cross-disciplinary experiments, could be combined with the other course experiments, which are carried out after finishing the theory course. The purpose of these experiments is to train students how to propose questions, analyze and solve problems, which could cultivate the ability of application, collaboration, exploration and innovation.

The integrated experiments usually involve several courses, and the experiment projects are designed based on some research and develop background, such as engine for report forms, simple workflow engine, implementation of JDBC driver, and game engine based on scenario languages etc.

3.2 Build up Library of Experiment Items and Experimental Software Equipment

To satisfy the different cultivating need, experiment projects and experiment items are separated. So far, there is one project for the basic experiment of compiler principle course, which has two variables, that is "languages" and "processing".

Under this circumstance, a series of experiment items are developed by means of substituting the two variables. For example, as for "languages" variable is concerned, either the simplified structured programming languages such as TINY, SP, Wren, PL/0, Oberon-0 etc., or the object oriented languages such as Decaf, MiniJava, COOL, SOOL etc. can be chosen in the experiments. As for "processing" is concerned, formatting a source program, reverse engineering etc. can be taken in account. There are several experiment items having been successfully conducted in compiler principle course, including automatic typesetting tool for programming language SP, automatic typesetting tool for programming language Wren, measurement of program complexity etc. The library of experimental items and experimental software equipment can be built up based the above experiment items.

4 Conclusions

Compiler principle is a typical course which perfectly combines the theory and practice, which both focuses on the theory foundation, and emphasizes the application to software develop. Based on the accumulated teaching experience, the new idea that "language is an effective way to solve problems" is adopted during the course of teaching compiler principle, which could not only interest student to study this course better, but also make for the design of the experiments. All of these strategies can help students to understand the ap-

plication of compiling techniques to software develop, and improve students ability to solve problems by use of compiling techniques in practice.

References:

[1] CC1991 (Computing Curricula 1991). Report of the ACM/IEEE-CS Joint Task Force on Computing Curricula December 1990, http://www.computer.org/education/cc1991/.

[2] AHO A, LAM M, SETHI R, et al. Compilers: Principles, Techniques, and Tools: 2nd Ed[M]. Addison-Wesley, 2006.

[3] SHAW M, GARLAN D. Software Architecture: Perspective on an Emerging Discipline[C]. Prentice-Hall International, 1996.

编译原理教学改革实践初探

舒忠梅，李文军，周晓聪

（中山大学软件学院，广东 广州 510275）

摘 要：编译原理是计算机科学与技术专业众多专业课程中理论与实践相结合的典范，但长期以来该课程的教学与实验内容陈旧，难以跟上软件技术迅速发展的步伐。根据作者近年的教学实践经验，提出在该课程的教学活动中应贯穿一种新的指导思想，即强调“语言也是求解问题的一种有效途径”。阐述了这一教学改革的新思路，并以此为基础重新设计了该课程的实验环节。

关键词：编译原理；教学改革；实验设计