Hacettepe University Department of Computer Engineering BBM301: Programming Languages Fall 2017 Project

Instructors	:	Dr. Pınar Duygulu Şahin, Dr. Nazlı İkizler Cinbiş
Research Assistants	:	Cumhur Yiğit Özcan, Feyza Nur Kılıçaslan
Due Date	:	06.11.2017

Description

With the recent advancements in technology, users must now manage files dispersed across several locations such as their PCs, laptops and mobile devices etc. Although there are a few cloud computing solutions such as Google Drive and Dropbox, offering only a limited storage option for free, they are not the ultimate solution. With the availability of inexpensive external storage options such as external drives, most users take up the habit of archiving their files instead of deleting them. In this project, you are expected to design a language to wrap the functionalities of the command line programming features of different operating systems as well as offering built-in functionality to easily use the API of some of the well-known cloud storage services. The language you will design should also have a direct support for managing files and folders through FTP protocol. The language should easily be used for example programming a synchronization software to regularly synchronize contents of two directories in a hard-disk of a PC and an external hard drive. You can define built-in functions for your programming language such as

- listContents(directory),
- getSize(file),
- create(directory),
- copy(file),
- compare(file1, file2),
- connectTo(cloudservice) etc.

Use your imagination and improve your own programming language for this problem.

Example Program

Assume that a programmer wants to develop a back-up program that can access the cloud storage services and FTP servers and create/manage a back-up directory on a local drive (i.e. an external drive). This program should be able to filter the files by their type (such as “text”, “music”, “picture” etc.) and archive them on a weekly manner. The programmer also wants to be able to check the size of the text files daily to create an immediate back-up for the ones that

have changed more than 1MB by size since the last back-up. In order not to back up duplicate files from different services and not to back up an unchanged file, programmer should also be able compare the files byte-by-byte (not only by their names and sizes). The program should have a simple interface for the users to alter the parameters like services and locations to check for changes, archiving period, locations etc.

Scope of Your Programming Language

You will design a programming language to program storage devices/services like in the sample back-up program described above. You should take into consideration that the built-in functions of your programming language should serve as a generalized wrapper so that a synchronization program would be able to synchronize the directories in different locations (i.e. local storage, FTP and cloud storage). Specifically it should include all of the following features:

- File and directory definition.
- Meta data of files and folders (i.e. size, type, date last modified etc.).
- Connection to cloud storage service and any FTP service.
- Wrapping both the local operating system functionality and remote services in a way to ensure easy manipulation of files and folder both in the remote and local storages.
- Defining and altering tasks to run periodically.
- Language level support for navigating through directories and subdirectories.

Nonetheless, the requirements given in this document are only intended as a base for the language you will design. You are expected to extend this feature list.

Part One: Design of the Language

For the first part of the project, you will design the syntax of your new programming language. The programming language has to implement these ground rules:

- Built-in functions
- Variable names
- Assignments
- Arithmetic operations
- Array data type
- Conditional statements
- Loop

Along with this document you will be provided a base token list for an imaginary c-like language, a base BNF grammar of that language and a sample code. These items meet the requirements given above to a large extend, yet, they are not complete. You are expected to improve on these base items and create a language that specializes on file operations. Since

these base items are pre-given, your work will only be graded by the additions you make on them. Therefore, do not forget to mention the additions you made in your report clearly.

Your design should also include a complete representation of the language in BNF grammar format. The programming language needs to have a name and short explanations for all structures of it.

Part Two: Lexical Analysis

In the second part of the project you are expected to design a lexical analyzer with lex/flex tools. This lexical analyzer will read an input program and generate pre-defined language keywords and tokens.

Part Three: Parser

For the third part of the project, you are expected to implement a parser for your language using yacc/bison tool. This parser should read an input code and as the output it should state whether the code is written correctly in your language or not. If the input file is written according to you language's syntax rules, it should print a success message and if it is not, then it should print an error message stating the related line. All the conflicts must be eliminated before implementing a parser. Please make sure that your language is unambiguous clearing all possible ambiguity related definitions.

Project Requirements

Your work will be evaluated with regard to the following requirements:

- Your report should include a complete BNF grammar of your programming language.
- The language you design should cover all the base prerequisites stated in this document.
- You are expected to extend the feature list of the language beyond the ones stated in this document.
    - You are expected to define new language specific data structures and built-in functions in accordance with the language scope. You are also strongly encouraged to think new use cases for the language to come up with new feature ideas.
    - Aside from the BNF, your report should contain descriptions for all the language specific features (either specified in this document or made up by you).
- You should create a lex file using the token list provided as a base.
- You should create a yacc file that represents your BNF grammar. You can use the BNF grammar provided as a base.

About Submit

- The project should be done by groups consisting of three persons. If you cannot find a group send an e-mail to one of the assistants to find you one.
- What to submit
    - Project Report (Soft copy): Project report should include the following:
        - Names and school IDs of the group members
        - Short descriptions of the defined tokens
        - The complete BNF grammar of your new programming language
        - Short description about syntax decisions
        - A list of additional functionality you have included in your language (in addition to the ones provided to you with the base token list)
        - A short tutorial on your language, explaining how each construct can be defined in your language
        - What are challenges of developing a parser for your language? For example what are the internal ambiguity solutions of yacc that was useful to solve ambiguity of your language, etc.
    - Lex file for your programming language (*.l file)
    - Yacc file of your parser (*.y file)
    - An example test file which is prepared with your programming language. This test file should include a sample of the defined structures and functionalities.
    - A makefile for the parser that will create a compiler executable with name <language_name>_parser
    - A makefile for the interpreter which will create a compiler executable with name <language_name>_interpreter
- **Submit:** You have to submit your project with the submit system which will be activated before due date. No other type of submission will be allowed. Please be aware that submission of one person from each group is sufficient, not all the group members have to make a submission.
- **Submission Format**
    - <studentid>.zip
        - project(folder)
            - ✓ report.pdf
            - ✓ project.l
            - ✓ project.y
            - ✓ makefile
            - ✓ test.txt (including code prepared by you that includes a showcase of your language specific features)