CS 315
Project 1

Assigned: Feb. 17, 2024
Due: Feb. 25, 2024, 23:59

A Programming Language for IoT Devices and its Lexical Analyzer

This semester's project is about the design of a new language for the Internet of Things (IoT) edge devices, also called nodes. Assume that you are working in the software department of a company that builds IoT devices. The hardware department constructs IoT nodes that has

- sensors; eg., temperature, humidity, air pressure, air quality, light, sound level (for a given frequency),
- 10 switches that can be set as on/off to control some actuators,
- a timer that returns the current time as six integer values, year, month, day, hour, minute, second. The values of predefined variables or the results of function calls,
- connection to the internet.

Your team is asked to design a special purpose Programming Language to program the IoT nodes produced by the hardware department. People who purchase these devices may not be computer engineers; so it is important that the programming language is readable, writable and reliable. This newly designed language should be similar to the classical imperative languages, since the users are assumed to have taken a basic programming course in college.

Part A - Language Design (30 points)

First, you will **give a name** to your language and design its syntax. Note that the best way to hand in your design is its grammar in BNF form, followed by a description of each of your language components. The following is a list of features that are required in your language:

- Primitive functions: read data from each sensor, timestamp from the timer,
- Mechanism to define a connection to a given URL,
- Mechanisms to send/receive an integer value at a time, from/to a connection,
- Variable names,
- Assignment operator, arithmetic operators,
- Expressions,
- Conditional statements; e.g., if-then, if-then-else,
- Loops; e.g., for, while,
- Mechanisms for defining and calling functions,
- Comments.

You are encouraged to use your imagination to extend the list given above.

You will have a chance to do minor revisions on your syntax design for Project 2, later in the semester. Language designs are almost never exactly right in the first iteration. Just try your best to make it as readable/writable/reliable as you can and keep your eyes open for what does and what does not work :)

Part B - Lexical Analysis (40 points)

In the second part of this project, you will design and implement a lexical analyzer for your language, using the lex tool available on all Unix style systems. Your scanner should read its input stream and output a sequence of tokens corresponding to the lexemes you will define in your language. Since at this stage you will not be able to connect the output to a parser, your scanner will print the names of the tokens on the screen. For instance, if we were designing a C like syntax, for the input

```
if ( temp > 35 ) { ...
```

the lexical analyzer should produce the output

IF LP IDENTIFIER GR_EQ NUMBER RP LBRACE ...

Part C - Example Programs (30 points)

Finally, you will prepare 3 test programs of your choice that exercise all of the language constructs in your language, including the ones that you may have defined in addition to the required list given above. Be creative, and have some fun.

Additional to your test programs, write the programs for the following two programs given as pseudo-code:

Program 1:

```
Define a function that computes the day of the week. It is easy to compute it as given here.
Test the function by sending the current day of week information as a log message to a connection with a given URL
```

Program 2:

```
1. continuously check,
  1.1. the time and if it is in between the working hours turn the lights on by setting the specific switch on, an
You determine the working hours, the switch for the lights. The company does not work during weekend.
  1.2. the temperature and if it is below 18 degrees celcius, turn the heater on, and it it is above 30 degrees ce
  1.3  if a sound with a frequency higher than 15mHz is detected, send an alarm message to a connection with a giv
```

Make sure your lex implementation correctly identifies all the tokens. The TA will test your lexical analyzer with these example programs along with different programs written in your language.

You are not required to write an interpreter or compiler for this language. Just write a few programs in the language you designed and make sure that the lexical analyzer produces the right sequence of tokens.

Teams

The project will be implemented in teams of two or three students. The members of the teams will be the same for this and the next project, in which you will write a parser for your programming language. Note that all members of a team will get the same grade in this part. We will assume each member has contributed the project equally.

Submission

- There are several parts that you will hand in.
  1. A project report including the following components:
     - Name, ID and section for all of the project group members.
     - The name of your programming language.
     - The complete BNF description of your language.
     - About one paragraph explanation for each language construct (i.e. variables and terminals) detailing their intended usage and meaning, as well as **all of the associated conventions.**
     - Descriptions of how nontrivial tokens (comments, identifiers, literals, reserved words, etc) are defined in your language. For all of these, explain what your motivations and constraints were and how they relate to various language criteria such as readability, writability, reliability, etc.
  2. Your lex description file
  3. The example test program described above, written in your language
- Make sure your lexical analyzer compiles and runs on `dijkstra.cs.bilkent.edu.tr`. The TA will test your project on the dijkstra machine, and any project that does not compile or run on this machine will get 0 on Part-B.
- Your password is the same as the one for your @ug.bilkent.edu.tr e-mail account.
- Please upload all of the above items to *Moodle* before the due date. PDF format is preferred for the project reports.
- Late submissions will be accepted, with 20 points (out of 100) deduction for each extra day.

Resources

- Running lex and yacc on Linux systems (accessible in Bilkent Campus)
- A Compact Guide to Lex & Yacc

Good Luck! - Have fun.
