CS4402 Learning journal 2

The Learning Journal is a tool for self-reflection on the learning process. In addition to completing directed tasks, you should use the Learning Journal to document your activities, record problems you may have encountered and to draft answers for Discussion Forums and Assignments. The Learning Journal should be updated regularly (on a weekly basis), as the learning journals will be assessed by your instructor as part of your Final Grade.

Your learning journal entry must be a reflective statement that considers the following questions:

* Describe what you did. This does not mean that you copy and paste from what you have posted or the assignments you have prepared. You need to describe what you did and how you did it.
* Describe your reactions to what you did
* Describe any feedback you received or any specific interactions you had. Discuss how they were helpful
* Describe your feelings and attitudes
* Describe what you learned

Another set of questions to consider in your learning journal statement include:

* What surprised me or caused me to wonder?
* What happened that felt particularly challenging? Why was it challenging to me?
* What skills and knowledge do I recognize that I am gaining?
* What am I realizing about myself as a learner?
* In what ways am I able to apply the ideas and concepts gained to my own experience?

Finally, describe one important thing that you are thinking about in relation to the activity.

Answer all questions that apply, however, you are expected to complete the Learning Journal assignment in 400-600 words.

During the learning of this week, there are a few items that I have gained more understanding of the compilation process.

Besides that, I learned the virtual box model of Unix and hypervisor.

I learned the basic compiling process that breaks into four steps lexical, syntax, and type. Then at last there is a code generation.

It’s surprising to learn the differences between an interpreter and a compiler.

One is to run every time of execution and the other runs once as job complete.

I also got the definition of the language of programming,

The set of terminal symbols and those terminals cannot be divided anymore.

There is a set of nonterminal symbols. The non-terminal symbols can be substituted into the terminal symbols.

The set of syntactic equations or production. The equation is specified for each nonterminal symbol. The last one is the start symbols. This is called the Backus Nau Form. The extended version includes express repetition and optionality.

The interpretation of sentences always rests on the recognition of their syntactic structure

Every sentence must have a single structure to be unambiguous.

A language is regular, if its syntax can be expressed by a single EBNF expression.

The recognition of regular sentences is a finite automaton, also called a state machine.

Then I gained exposure to two context-free languages, one is the recursive descent and the other is the table-driven top-down parsing.

Both the recursive-descent and table-driven shows the top-down parsing. It's also possible to proceed according to a complementary principle.

I also gained more understanding of static and dynamic typing. Static typing requires the definition of a data type when it is created.

String and weak typing define the flexibility of a language that allows operations between data types.

Strong-typed languages will not allow you to add a float with an integer. The weak typing will try its best to accommodate that request.

I gained some understanding of how type-checking helps the quality control of code.

It's better to have a strong type of checking than to have a bug that is hard to trace.

The virtual machine is a simulation used to resemble the instructions. The computer maps the virtual instruction and gives a real feeling of execution on top of a real computer.

IBM is nearly one-to-one mapping in the virtual machine model. The second is JAVA virtual machines. The Unix and OSI virtual machine calls the operation system’s functions directly.

Overall, I gained more understanding of how language transformed into machine instructions. This understanding of the low-level compiling processes boosts my confidence in understanding how the system works.