# Let's Write a Shell!

2012-2013 CSCI 3150 - Programming Assignment 1

Supplementary notes - 23:15, 2012 Oct 2

**Abstract**

The original specification fails to remove ambiguities; it produces more. Therefore, the specification requires a supplement to explain ambiguities.

Phase 1 is for breaking an input command line into tokens so that it can assist you in implementing Phase 2. Dr. Wong considers the ambiguities happened in this specification as a disaster! He whole-heartedly apologizes for the mistakes.

## Ambiguities in the Specification

We are going to list all the ambiguities in the specification version 1.0.2 point-by-point.

1. [**Problem**] In Page 22, the specification says "*The interpreter recognizes built-in commands whenever a correctly-placed built-in command is encountered.*".

   However, in Page 7, there is an example command "`cd / > out.txt`" indicated as wrong! An incorrectly-placed built-in commands is recongized, too!

   [**Resolution**] Dr. Wong thinks that the assignment is not there to cause you trouble (and as a matter of fact, it DID).

   - He withdraws the wrong example on Page 7.
   - Only the correctly-placed built-in commands are classified with the type "`Built--in Command`".

- For other wrongly-placed built-in commands, you are free to have your own implementation. In addition, our tutor will avoid having such test cases.
- Those free-to-implement commands include:
  - "`cd / > out.txt`",
  - "`ls | cd /`",
  - "`fg | cd`", etc.
- Such a relaxation applies to both Phases 1 and 2.

---

2. [**Problem**] Again, in Page 22, the specification says "*The interpreter recognizes built--in commands whenever a correctly-placed built-in command is encountered.*". What is the meaning of "recognize"? Moreover, should we also check the number of input arguments in those built-in commands?

   [**Resolution**] The word "recognize" means to know that a particular command is a built-in command. However, we did not say anything about the checking of the number of built-in commands in Section 3.1.2 on Page 22 of the specification (*OH NO...*).

   - In Phase 1, you have to recongize all four built-in commands by printing out its "`Type`" as "`Built-in Command`" only when they are correctly-placed.
   - [**cd and exit**] On top of knowing that it is a built-in command, you have to execute them:
     - If the number of arguments matches the requirement in the specification, then execute the task defined for that built-in command.
     - Else (if the number of arguments does not match), you should report an appropriate error message.
     - What is the meaning of "an appropriate error message? We will talk about it later.
   - [**jobs and fg**] We did not say anything about them:

  - You have to recongize their types as "`Built-in Command`".
  - You are not required to check the number of arguments.
  - What if you have implemented such a checking? Just leave the implementation there.

---

3. **[Problem]** Again, in Page 22, The box named "Phase 1 input and output example" contains contradicting outputs.

   - "`cd /`" causes the interpreter to list the tokens. Then, "`cd`" executes.
   - "`cd / /`" does not causes the interpreter to list any tokens. Yet, "`cd`" knows that the number of arguments is wrong!

   The question is: when to list the token?

   **[Resolution]** Sorry that this is the worst part of the specification. The reasonable understanding of the interpreter is:

   (a) By following Point 2 on Page 21, both "`cd /`" and `cd / /`" should cause the interpreter to list the tokens.

   (b) Since the language does not define the number of arguments for built-in commands, the interpreter should not consider "`cd / / `" as not matching our language.

   (c) Next, after knowing that the input command line is in a good shape, the interpreter will execute the tasks defined for "`cd`".

   (d) Eventually, the interpreter reports "`cd: wrong number of arguments`".

   So, what implementations are considered as correct? Since we are too late to face our ambiguities, we will consider the following outputs as correct in Phase 1 when your interpreter encounters the command "`cd / /`".

   - **[Best choice.]** List tokens. Then, print "`cd:  wrong number of arguments`".

- **[Second-best choice.]** Print "`cd: wrong number of arguments`" only.
- **[Least-favored choice.]** Print "`Error: invalid input command line`" only (because Dr. Wong accidentally accepted such an answer in the Facebook).

All in all, the rule of thumb about the case "`cd / /` " is to report an error. For Phase 2, please output "`cd: wrong number of arguments`" because you no longer need to list the tokens.

The above resolution also applies to the built-in command "`exit`".

---

**– END –**