Interpreted Language and Compiled Language

Terry Yin

October 31, 2014

1 Interpreted language

```
> !turn left
```

```
/bin/sh: turn: command not found
```

```
> !go left
```

```
/bin/sh: go: command not found
```

```
> !find direction
```

```
find: direction: No such file or directory
```

```
> !make ourway
```

```
make: *** No rule to make target ‘ourway’.  Stop.
```

Now you are stuck in the middle of your way to your friend's place in a foreign city. And you are getting frustrated with this unix shell (`/bin/sh`) based vehicle, Wikipedia (2014*c*). It doesn't seem to understand your command. But at least the shell program tried to interpret your command immediately and give you some feedback.

```
> !find grocery_store
```

```
grocery_store
```

Ok, it seems the smart vehicle has found the `grocery_store` you pointed to it. It's a good start. Now you learned more commands you can use on the vehicle. You can even program the vehicle with multiple commands:

```
!echo "trip start..."
!if find grocery_store; then echo "I saw it"; else echo "what?";fi
!echo "thanks for using our service."
```

```
trip start...
grocery_store
I saw it
thanks for using our service.
```

The way computer runs the above shell script program is the same as the interactive commands we use in the beginning. The shell (here it's the **interpreter**) first read one line from the script, next interpret it to understand your intent for the computer, then do it, and last it provides the feedback. The same process will continue until the end of the script. Wikipedia (2014*b*)

2 Compiled language

You might be a person who likes to plan your trip up-front. You are not satisfied with finding that your smart car doesn't understand your command 'turn left' only when your trip started. You want to make sure the car understands all your commands and knows the direction already before you start the journey.

Then you'd better **compile** all your instructions before you run the program by translating all the instructions to the basic commands smart cars can understand. It's just like using a **compiled language** Wikipedia (2014*a*), like C:

```
%%writefile trip.c
#include "stdio.h"
int main() {
    printf("start trip\n");
    if( find("grocery_store", F_OK ) != -1 ) {
        printf("I saw it.\n");
    } else {
        printf("what?\n");
    }
    printf("thanks for using our service\n");
}
```

```
Overwriting trip.c
```

The above C code is "equivalent" to the shell script code in the previous section. But you cannot use it directly. You need to compile it first. Let's compile it using the GCC compiler:

```
!gcc trip.c -o trip
```

```
trip.c:4:9: warning: implicit declaration of function 'find' is invalid in C99 [-Wimplicit-function-decl
    if( find("grocery_store", F_OK ) != -1 ) {
        ^
trip.c:4:31: error: use of undeclared identifier 'F_OK'
    if( find("grocery_store", F_OK ) != -1 ) {
                              ^
1 warning and 1 error generated.
```

As you can see, the compilation wasn't successful. The compiler told me that it doesn't understand my function `find` and it doesn't understand what `F_OK` means. The good news is, at least I don't have to be stuck in the middle of my journey, I know there's problem already.

Let's fix the C code by including the necessary header file and using the correct function name. Then try again:

```
%%writefile trip.c
#include "stdio.h"
#include "unistd.h"
int main() {
    printf("start trip\n");
    if( access("grocery_store", F_OK ) != -1 ) {
        printf("I saw it.\n");
    } else {
        printf("what?\n");
    }
    printf("thanks for using our service\n");
}
```

```
Overwriting trip.c
```

```
!gcc trip.c -o trip
```

This time `gcc` didn't say anything. It's because `gcc` follows the good old Unix philosophy – no new is good news, Gancarz (2003).

The above command compiled the C code into a binary file `trip`, which can be executed directly on the machine. If it's on a Microsoft Windows system, the name of the output binary file might be something like `trip.exe`. We can execute the executable binary file directly. Like:

```
!./trip
```

```
start trip
I saw it.
thanks for using our service
```

3 The mix of compiled and interpreted langauges

So, which way of making the trip to your friend's place do you like? Probably, you will do both. You might try your best to plan the direction before you start the journey, but you would also like to get ready to adjust for the situation you'll meet on the go.

Programming languages are the same. Nowadays, most languages have a little bit of both. **Java** is mostly considered a compiled language, but it compiles only to the bytecode that the JVM (Java Virtual Machine) could interpret. **Python** is considered an interpreted language. But it also compiles the source code to its own bytecode before execution. And interpreted programming languages like **JavaScript** has the technology called **Just-In-Time** (Brookshear 2011, JIT, p.271) compilation. JIT translates the script into machine code at run-time. That's why JavaScript is so fast these days. On the other hand, the traditional low-level programming language like C are now using technology called LLVM, which is originally known as the Low Level Virtual Machine. So it might not always compiles to machine code, Lattner (2008).

4 Conclusion

It has been a while that new languages are mostly interpreted language, like Python, Ruby, JavaScript. But the most recent noticeable new language are compiled languages again, like the Go language by Google and the Swift language by Apple. That means the two kinds might just co-exist. And we want to take advantage from both of them.

* I'm using OS X Yosemite. The GCC compiler I used is actually Clang.

References

Brookshear, J. G. (2011), *Computer science: an overview*, Paul Muljadi.

Gancarz, M. (2003), *Linux and the Unix philosophy*, Digital Press.

Lattner, C. (2008), Llvm and clang: Next generation compiler technology, *in* 'The BSD Conference', pp. 1–2.

Wikipedia (2014*a*), 'Compiled language — wikipedia, the free encyclopedia'. [Online; accessed 30-October-2014].
URL: *http://en.wikipedia.org/w/index.php?title=Compiled_language&oldid=619022934*

Wikipedia (2014*b*), 'Interpreted language — wikipedia, the free encyclopedia'. [Online; accessed 30-October-2014].
URL: *http://en.wikipedia.org/w/index.php?title=Interpreted_language&oldid=628934282*

Wikipedia (2014*c*), 'Unix shell — wikipedia, the free encyclopedia'. [Online; accessed 30-October-2014].
URL: *http://en.wikipedia.org/w/index.php?title=Unix_shell&oldid=629756474*