Environment Variables and Starting Other Programs

This lesson explains what environment variables are and how they can be used. In the later part of the lesson, we see how a program can start another program.

WE'LL COVER THE FOLLOWING

Environment variables

The environment that a program is started in generally provides some variables that the program can use. The environment variables can be accessed through the associative array interface of `std.process.environment`. For example, the following program displays the `PATH` environment variable:

```
import std.stdio;
import std.process;

void main() {
    writeln(environment["PATH"]);
}
```

Program to print PATH environment variable

`std.process.environment` provides access to the environment variables through the associative array syntax. However, the environment itself is not an associative array. When needed, the environment variables can be converted to an associative array by using `toAA()`:

```
import std.stdio;
```

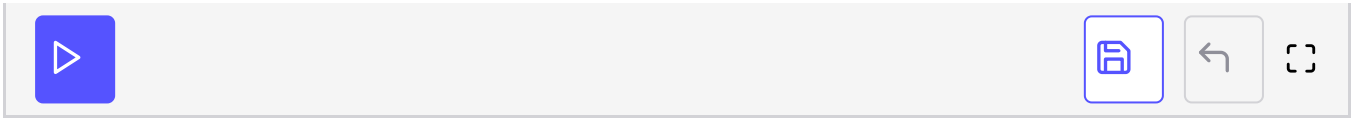
```
import std.process;

void main() {


    string[string] envVars = environment.toAA();

    foreach(i,ele;envVars) {
        writeln(i,' ',ele);
    }

}
```



Starting other programs

Programs may start other programs and become the environment for those programs. A function that can execute other programs is `executeShell`, from the `std.process` module.

`executeShell` executes its parameter as if the command was typed at the terminal. It then returns both the return code and the output of that command as a tuple. Tuples are array-like structures.

```
import std.stdio;
import std.process;
void main() {
    const result = executeShell("ls -l deneme");
    const returnCode = result[0];
    const output = result[1];
    writefln("ls returned %s.", returnCode);
    writefln("Its output:\n%s", output);
}
```

executeShell( ) function

The output is:

```
# ./deneme
ls returned 0.
Its output:
-rwxrwxr-x. 1 acehreli acehreli 1359178 Apr 21 15:01 deneme
```

Chapter summary

- Even when it is defined with a return type of `void`, `main()` automatically

returns zero for success and nonzero for failure.

- `stderr` is suitable to print error messages.
- `main` can take parameters as string[].
- `std.getopt` helps with parsing command-line options.
- `std.process` helps with accessing environment variables and starting other programs.

In the next lesson, you will find a quiz based on the concepts covered in this chapter.