
Batsh

Notes from maintainers

This repo was transferred from the original author @BYVoid, and has been upgraded to build on more recent OCaml versions (buildable on at least OCaml 4.08.1). You can see the discussion thread which spawned this fork effort [here](#).

Note that this project is currently in minimum maintenance mode. Issues, PR may not be actively dealt with.

The following sections may contain out of date info as we are still in the process of going through the repo

Project description

Batsh is a simple programming language that compiles to Bash and Windows Batch. It enables you to write your script once runs on all platforms without **any** additional dependency.

Both Bash and Batch are messy to read and tricky to write due to historical reasons. You have to spend a lot of time learning either of them and write platform-dependent code for each operating system. I have wasted lots of time in my life struggling with bizarre syntaxes and unreasonable behaviors of them, and do not want to waste any more.

If you happen to be a maintainer of a cross-platform tool which relies on Bash on Linux/Mac and Batch on Windows as “glue code”, and found it painful to “synchronize” between them, you would definitely like to try Batsh.

How to get it

The easiest way

Try it online: <http://batsh.org>

Install from OPAM

Batsh is implemented in OCaml and managed by OPAM.

1. Install OPAM. See instructions.
2. Switch to the latest version (or at least 4.00.1) of OCaml by running `opam switch`.
3. Install Batsh: `opam install batsh`

Build from source

You have to install OCaml (version 4.00.1 or higher) development environment before compiling Batsh from source code, and follow steps below:

1. Download source code of Batsh from releases or clone with git.
2. Uncompress source code tarball.
3. `make`
4. `make install`
5. Run: `batsh`

Dependencies If there is any missing dependency, you can install them by running `opam install dune core_kernel ounit dlist cmdliner`

- dune: Build framework.
- core_kernel: Core_kernel is the system-independent part of Core, Jane Street's industrial-strength alternative to the OCaml standard library.
- ounit: Unit test framework.
- dlist: A purely functional list-like data structure supporting O(1) concatenation.
- cmdliner: Command line interfaces parser.

Syntax

The syntax of Batsh is C-based (derived from C programming language). If you have learned C, Java, C++ or JavaScript, Batsh is quite easy for you.

Assignment

```
1 a = 1;  
2 b = "string";  
3 c = [1, 2, "str", true, false];
```

Expression

```
1 a = 1 + 2;  
2 b = a * 7;  
3 c = "Con" ++ "cat";  
4 d = c ++ b;
```

Command

```
1 // On UNIX
2 output = ls();
3 // On Windows
4 output = dir();
5 // Platform independent
6 output = readdir();
7
8 // Test existence
9 ex = exists("file.txt");
```

If condition

```
1 a = 3;
2 if (a > 2) {
3     println("Yes");
4 } else {
5     println("No");
6 }
```

Loop

```
1 // Fibonacci
2 n = 0;
3 i = 0;
4 j = 1;
5 while (n < 60) {
6     k = i + j;
7     i = j;
8     j = k;
9     n = n + 1;
10    println(k);
11 }
```

Function

```
1 v1 = "Global V1";
2 v2 = "Global V2";
3 function func(p) {
4     v1 = "Local " ++ p;
5     global v2;
6     v2 = "V2 Modified.";
7 }
```

```
8 func("Var");
```

Recursion

```
1 function fibonacci(num) {
2   if (num == 0) {
3     return 0;
4   } else if (num == 1) {
5     return 1;
6   } else {
7     return (fibonacci(num - 2) + fibonacci(num - 1));
8   }
9 }
10 println(fibonacci(8));
```

More examples

Syntax Highlighting

Vim

- vim-Batsh

Emacs

- batsh-mode

Built-in functions

In order to make script cross-platform, Batsh provided some “built-in” functions that will compile to platform-dependent code. It is assumed that Bash script runs on Linux or Mac OS and Batch script runs on Windows (XP or higher), which means Cygwin or wine are not supported.

print(text, ...)

Prints a text string to console without a newline.

println(text, ...)

Prints a text string to console with a new line (LF for bash, CRLF for batch).

call(path, arg, ...)

Runs command from path through shell.

bash(rawStatement)

Put `rawStatement` into compiled code for Bash. Ignore for Windows Batch.

batch(rawStatement)

Put `rawStatement` into compiled code for Windows Batch. Ignore for Bash.

readdir(path)

Equals to `ls` and `dir /w`.

exists(path)

Test existence of given path.

Command Line Usage

```
1 NAME
2     batsh - A language that compiles to Bash and Windows Batch
3
4 SYNOPSIS
5     batsh COMMAND ...
6
7 COMMANDS
8     bash
9         Compile to Bash script.
10
11     batsh
12         Format source file.
13
14     winbat
15         Compile to Windows Batch script.
16
17 OPTIONS
18     --help[=FMT] (default=pager)
19         Show this help in format FMT (pager, plain or groff).
20
```

```
21      --version
22      Show version information.
```

Why not Python/Ruby/Node.js/Lua

Yes you can use any of them as platform-independent glue code. But there are several disadvantages:

1. None of them is **preinstalled on all platforms** (including Windows).
2. Functionalities like process piping are not convenient to use.
3. Hard to integrate with existing code written in Bash or Batch.

Those reasons are why I developed Batsh.

License

MIT

Contributors

- Carbo Kuo
- Song Zhang
- Anthony Chan
- jeb-de
- Al Ramirez
- Nixola
- Darren Ldl
- Anton Kochkov