
grunt-shell

Run shell commands

A good way to interact with other CLI tools. For example, get the current Git branch with `git branch`.

Install

```
1 npm install --save-dev grunt-shell
```

Usage

```
1 require('load-grunt-tasks')(grunt);
2
3 grunt.initConfig({
4   shell: {
5     options: {
6       stderr: false
7     },
8     target: {
9       command: 'ls'
10    },
11    another: 'ls ./src' // Shorthand
12  }
13 });
14
15 grunt.registerTask('default', ['shell']);
```

Examples

Run command

Create a folder named `test`.

```
1 grunt.initConfig({
2   shell: {
3     makeDir: {
4       command: 'mkdir test'
5     }
6   }
7 });
```

The `command` property supports templates:

```
1 grunt.initConfig({
2   testDir: 'test',
3   shell: {
4     makeDir: {
5       command: 'mkdir <%= testDir %>'
6     }
7   }
8 });
```

You can also supply a function that returns the command:

```
1 grunt.initConfig({
2   shell: {
3     hello: {
4       command: () => 'echo hello'
5     }
6   }
7 });
```

Which can also take arguments:

```
1 module.exports = grunt => {
2   grunt.loadNpmTasks('grunt-shell');
3   grunt.initConfig({
4     shell: {
5       greet: {
6         command: greeting => `echo ${greeting}`
7       }
8     }
9   });
10  grunt.registerTask('default', ['shell:greet:hello']);
11 }
```

Run command and display the output

Output a directory listing in your Terminal.

```
1 grunt.initConfig({
2   shell: {
3     dirListing: {
4       command: 'ls'
5     }
6   }
7 });
```

Custom callback

Do whatever you want with the output.

```
1 function log(error, stdout, stderr, callback) {
2     if (error) {
3         callback(error);
4         return;
5     }
6
7     console.log(stdout);
8     callback();
9 }
10
11 grunt.initConfig({
12     shell: {
13         dirListing: {
14             command: 'ls',
15             options: {
16                 callback: log
17             }
18         }
19     }
20 });
```

Option passed to the .exec() method

Run a command in another directory. In this example, we run it in a subfolder using the `cwd` (current working directory) option.

```
1 grunt.initConfig({
2     shell: {
3         subfolderLs: {
4             command: 'ls',
5             options: {
6                 stderr: false,
7                 execOptions: {
8                     cwd: 'tasks'
9                 }
10             }
11         }
12     }
13 });
```

Multiple commands

Run multiple commands by placing them in an array which is joined using `&&` or `;`. `&&` means run this only if the previous command succeeded. You can also use `&` to have the commands run concurrently (by executing all commands except the last one in a subshell).

```
1 grunt.initConfig({
2   shell: {
3     multiple: {
4       command: [
5         'mkdir test',
6         'cd test',
7         'ls'
8       ].join('&&')
9     }
10  }
11 });
```

Config

command

Required

Type: `string` | `Function`

Command to run or a function which returns the command. Supports underscore templates.

Command can be omitted by directly setting the target with the command.

cwd

Type: `string`

Shortcut. Same as `options.execOptions.cwd` (see below).

Options

stdout

Type: `boolean`

Default: `true`

Show stdout in the terminal.

stderr

Type: **boolean**

Default: **true**

Show stderr in the terminal.

stdin

Type: **boolean**

Default: **true**

Forward the terminal's stdin to the command.

failOnError

Type: **boolean**

Default: **true**

Fail task if it encounters an error. Does not apply if you specify a **callback**.

stdinRawMode

Type: **boolean**

Default: **false**

Set **stdin** to act as a raw device.

callback(error, stdout, stderr, callback)

Type: **Function**

Lets you override the default callback with your own.

Make sure to call the **callback method when you're done.** Supply an error as the first argument to **callback** to print a warning and cause the task to fail.

preferLocal

Type: **boolean**

Default: **true**

Execute local binaries by name like `$ npm run-script`.

execOptions

Type: **object**

Specify some options to be passed to the `.exec()` method:

- **cwd** string *Current working directory of the child process*
- **env** Object *Environment key-value pairs*
- **setsid** boolean
- **encoding** string (Default: `'utf8'`)
- **timeout** number (Default: `0`)
- **maxBuffer** number (Default: `1000 * 1000 * 10` → 10 MB)
- **killSignal** string (Default: `'SIGTERM'`)