


 [sheffercool](#) / [babun](#)forked from [babun/babun](#)

Babun - a Windows shell you will love!

 0 stars  561 forks Star Watch ▼

Code

Pull requests

Actions


Projects

Wiki

Security


Insights

Settings

 master ▼

...

This branch is even with babun:master.

 Pull request  Compare jlupi ...on 11 Jun 2019 [View code](#)README.adoc 

Babun - a windows shell you will love

[PROJECT DISCONTINUED]

maintainers wanted

Would you like to use a linux-like console on a Windows host without a lot of fuzz? Try out babun!

Have a look at a 2 minutes long screencast by [@tombujok](#): <http://vimeo.com/95045348>

Installation

Just download the dist file from <http://babun.github.io>, unzip it and run the install.bat script. After a few minutes babun starts automatically. The application will be installed to the %USER_HOME%\babun directory. Use the '/target' option to install babun to a custom directory.

Note

There is no interference with existing Cygwin installation

Note

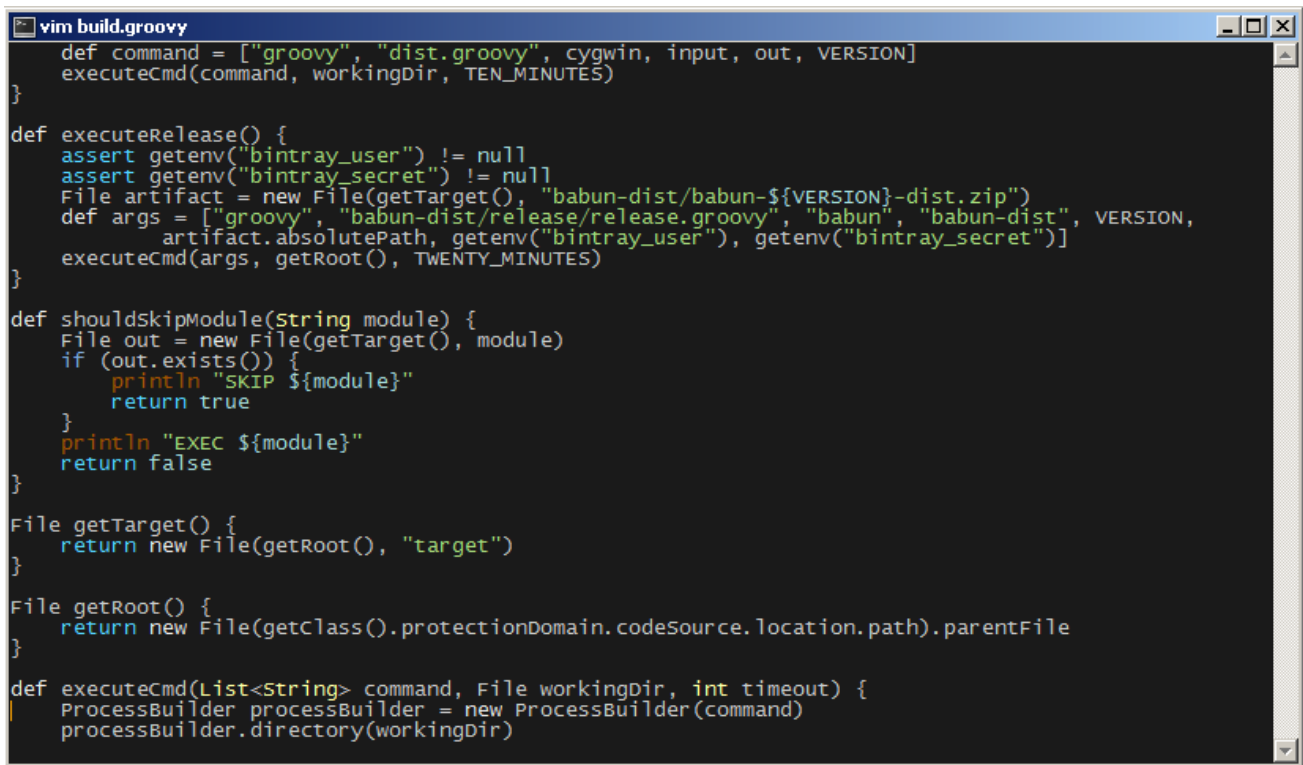
You may have "whitespace" chars in your username - it is not recommended by Cygwin though [FAQ](#)

Features in 10 seconds

Babun features the following:

- Pre-configured Cygwin with a lot of addons
- Silent command-line installer, no admin rights required
- `pact` - advanced package manager (like `apt-get` or `yum`)
- xTerm-256 compatible console
- HTTP(s) proxying support
- Plugin-oriented architecture
- Pre-configured git and shell
- Integrated oh-my-zsh
- Auto update feature
- "Open Babun Here" context menu entry

Have a look at a sample screenshot!



```
vim build.groovy
def command = ["groovy", "dist.groovy", cygwin, input, out, VERSION]
executeCmd(command, workingDir, TEN_MINUTES)
}

def executeRelease() {
    assert getenv("bintray_user") != null
    assert getenv("bintray_secret") != null
    File artifact = new File(getTarget(), "babun-dist/babun-${VERSION}-dist.zip")
    def args = ["groovy", "babun-dist/release/release.groovy", "babun", "babun-dist", VERSION,
        artifact.absolutePath, getenv("bintray_user"), getenv("bintray_secret")]
    executeCmd(args, getRoot(), TWENTY_MINUTES)
}

def shouldSkipModule(String module) {
    File out = new File(getTarget(), module)
    if (out.exists()) {
        println "SKIP ${module}"
        return true
    }
    println "EXEC ${module}"
    return false
}

File getTarget() {
    return new File(getRoot(), "target")
}

File getRoot() {
    return new File(getClass().protectionDomain.codeSource.location.path).parentFile
}

def executeCmd(List<String> command, File workingDir, int timeout) {
    ProcessBuilder processBuilder = new ProcessBuilder(command)
    processBuilder.directory(workingDir)
```

Do you like it? Follow babun on Twitter [@babunshell](#) or [@tombujok](#).

Features in 3 minutes

Cygwin

The core of Babun consists of a pre-configured Cygwin. Cygwin is a great tool, but there's a lot of quirks and tricks that makes you lose a lot of time to make it actually 'usable'. Not only does babun solve most of these problems, but also contains a lot of vital packages, so that you can be productive from the very first minute.

Package manager

Babun provides a package manager called `pact`. It is similar to 'apt-get' or 'yum'. Pact enables installing/searching/upgrading and deinstalling cygwin packages with no hassle at all. Just invoke `pact --help` to check how to use it.

Shell

Babun's shell is tweaked in order to provide the best possible user-experience. There are two shell types that are pre-configured and available right away - bash and zsh (zsh is the default one). Babun's shell features:

- syntax highlighting
- UNIX tools
- software development tools

- git-aware prompt
- custom scripts and aliases
- and much more!

Console

Mintty is the console used in babun. It features an `xterm-256` mode, nice fonts and simply looks great!

Proxying

Babun supports HTTP proxying out of the box. Just add the address and the credentials of your HTTP proxy server to the `.babunrc` file located in your home folder and execute `source .babunrc` to enable HTTP proxying. SOCKS proxies are not supported for now.

Developer tools

Babun provides many packages, convenience tools and scripts that make your life much easier. The long list of features includes:

- programming languages (Python, Perl, etc.)
- git (with a wide variety of aliases and tweaks)
- UNIX tools (grep, wget, curl, etc.)
- vcs (svn, git)
- oh-my-zsh
- custom scripts (pbcopy, pbpaste, babun, etc.)

Plugin architecture

Babun has a very small microkernel (cygwin, a couple of bash scripts and a bit of a convention) and a plugin architecture on the top of it. It means that almost everything is a plugin in the babun's world! Not only does it structure babun in a clean way, but also enables others to contribute small chunks of code. Currently, babun comprises the following plugins:

- cacert
- core
- git

- oh-my-zsh
- pact
- cygdrive
- dist
- shell

Auto-update

Self-update is at the very heart of babun! Many Cygwin tools are simple bash scripts - once you install them there is no chance of getting the newer version in a smooth way. You either delete the older version or overwrite it with the newest one losing all the changes you have made in between.

Babun contains an auto-update feature which enables updating both the microkernel, the plugins and even the underlying cygwin. Files located in your home folder will never be deleted nor overwritten which preserves your local config and customizations.

Installer

Babun features an silent command-line installation script that may be executed without admin rights on any Windows hosts.

Using babun

Setting up proxy

To setup proxy uncomment following lines in the `.babunrc` file
(`%USER_HOME%\babun\cygwin\home\USER\.babunrc`)

```
# Uncomment this lines to set up your proxy
# export http_proxy=http://user:password@server:port
# export https_proxy=$http_proxy
# export ftp_proxy=$http_proxy
# export no_proxy=localhost
```

Setting up git

Babun has a pre-configured git. The only thing you should do after the installation is to add your name and email to the git config:

```
git config --global user.name "your name"
git config --global user.email "your@email.com"
```

There's a lot of great git aliases provided by the git plugin:

```
gitalias['alias.cp']='cherry-pick'
gitalias['alias.st']='status -sb'
gitalias['alias.cl']='clone'
gitalias['alias.ci']='commit'
gitalias['alias.co']='checkout'
gitalias['alias.br']='branch'
gitalias['alias.dc']='diff --cached'
gitalias['alias.lg']="log --graph --pretty=format:%Cred%h%Creset -%C(yellow)%d%Crese
gitalias['alias.last']='log -1 --stat'
gitalias['alias.unstage']='reset HEAD --'
```

Installing and removing packages

Babun is shipped with `pact` - a Linux like package manager. It uses the cygwin repository for downloading packages:

```
{ ~ } » pact install arj
Working directory is /setup
Mirror is http://mirrors.kernel.org/sourceware/cygwin/
setup.ini taken from the cache

Installing arj
Found package arj
--2014-03-30 19:34:38-- http://mirrors.kernel.org/sourceware/cygwin/x86/release/arj
Resolving mirrors.kernel.org (mirrors.kernel.org)... 149.20.20.135, 149.20.4.71, 2001
Connecting to mirrors.kernel.org (mirrors.kernel.org)|149.20.20.135|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 189944 (185K) [application/x-bzip2]
Saving to: `arj-3.10.22-1.tar.bz2'

100%[=====>] 189,944      193K/s   in 1.0s

2014-03-30 19:34:39 (193 KB/s) - `arj-3.10.22-1.tar.bz2' saved [189944/189944]

Unpacking...
Package arj installed
```

Here's the list of all pact's features:

```
{ ~ } » pact --help
pact: Installs and removes Cygwin packages.
```

Usage:

```
"pact install <package names>" to install given packages
"pact remove <package names>" to remove given packages
"pact update <package names>" to update given packages
"pact show" to show installed packages
"pact find <patterns>" to find packages matching patterns
"pact describe <patterns>" to describe packages matching patterns
"pact packageof <commands or files>" to locate parent packages
"pact invalidate" to invalidate pact caches (setup.ini, etc.)
```

Options:

```
--mirror, -m <url> : set mirror
--invalidate, -i      : invalidates pact caches (setup.ini, etc.)
--force, -f : force the execution
--help
--version
```

Changing the default shell

The zsh (with .oh-my-zsh) is the default babun's shell.

Executing the following command will output your default shell:

```
{ ~ } » babun shell
/bin/zsh
```

In order to change your default shell execute:

```
{ ~ } » babun shell /bin/bash
/bin/zsh
/bin/bash
```

The output contains two lines: the previous default shell and the new default shell

Checking the configuration

Execute the following command to check the configuration:

```
{ ~ } » babun check
Executing babun check
Prompt speed      [OK]
Connection check  [OK]
Update check      [OK]
Cygwin check      [OK]
```

By executing this command you can also check whether there is a newer cygwin version available:

```
{ ~ } » babun check
Executing babun check
Prompt speed      [OK]
Connection check  [OK]
Update check      [OK]
Cygwin check      [OUTDATED]
Hint: the underlying Cygwin kernel is outdated. Execute 'babun update' and follow the
```

It will check if there are problems with the speed of the git prompt, if there's access to the Internet or finally if you are running the newest version of babun.

The command will output hints if problems occur:

```
{ ~ } » babun check
Executing babun check
Prompt speed      [SLOW]
Hint: your prompt is very slow. Check the installed 'BLODA' software.
Connection check  [OK]
Update check      [OK]
Cygwin check      [OK]
```

On each startup, but only every 24 hours, babun will execute this check automatically. You can disable the automatic check in the ~/.babunrc file.

Tweaking the configuration

You can tweak some config options in the ~/.babunrc file. Here's the full list of variables that may be modified:

```
# JVM options
export JAVA_OPTS="-Xms128m -Xmx256m"

# Modify these lines to set your locale
export LANG="en_US.UTF-8"
export LC_CTYPE="en_US.UTF-8"
export LC_ALL="en_US.UTF-8"

# Uncomment these lines to the set your machine's default locale (and comment out the
# export LANG=$(locale -uU)
# export LC_CTYPE=$(locale -uU)
# export LC_ALL=$(locale -uU)

# Uncomment this to disable daily auto-update & proxy checks on startup (not recommen
# export DISABLE_CHECK_ON_STARTUP="true"
```



```
# Uncomment to increase/decrease the check connection timeout
# export CHECK_TIMEOUT_IN_SECS=4

# Uncomment this lines to set up your proxy
# export http_proxy=http://user:password@server:port
# export https_proxy=$http_proxy
# export ftp_proxy=$http_proxy
# export no_proxy=localhost
```

Updating babun

To update babun to the newest version execute:

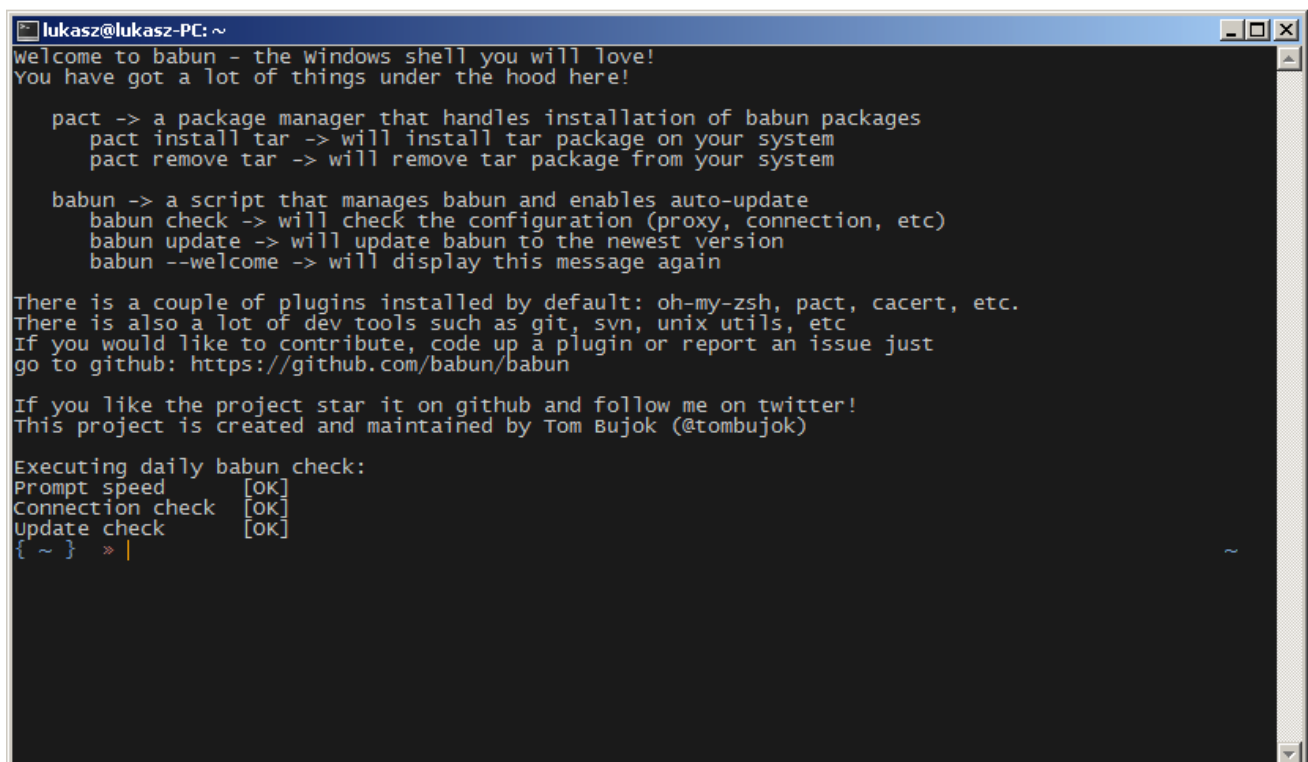
```
babun update
```

Please note that your local configuration files will not be overwritten.

The 'babun update' command will also update the underlying cygwin version if newer version is available. In such case babun will download the new cygwin installer, close itself and start the cygwin installation process. Once cygwin installation is completed babun will restart.

Screenshots

Startup screen

A screenshot of a terminal window titled 'lukasz@lukasz-PC: ~'. The terminal displays the Babun startup message: 'welcome to babun - the windows shell you will love! You have got a lot of things under the hood here!'. It then lists several commands and their functions: 'pact' for package management, 'babun' for configuration and updates, and 'oh-my-zsh' for plugins. It also mentions dev tools like git, svn, and unix utils. A daily check is performed, showing 'Prompt speed [OK]', 'Connection check [OK]', and 'Update check [OK]'. The prompt is '{ ~ } >> |'.

Pact - package installation

```

lukasz@lukasz-PC: ~/babun
{ babun } master » pact remove arj
Removing arj
Package arj removed
{ babun } master » pact install arj
working directory is /setup
Mirror is http://mirrors.kernel.org/sourceware/cygwin/
setup.ini taken from the cache

Installing arj
Found package arj
--2014-04-15 22:43:16-- http://mirrors.kernel.org/sourceware/cygwin//x86/release/arj/arj-3.10.22-1.
tar.bz2
Resolving mirrors.kernel.org (mirrors.kernel.org)... 149.20.4.71, 2001:4f8:1:10:0:1994:3:14
Connecting to mirrors.kernel.org (mirrors.kernel.org)[149.20.4.71]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 189944 (185K) [application/x-bzip2]
Saving to: `arj-3.10.22-1.tar.bz2'

100%[=====>] 189,944      193K/s   in 1.0s

2014-04-15 22:43:17 (193 KB/s) - `arj-3.10.22-1.tar.bz2' saved [189944/189944]

unpacking...
Package arj installed
{ babun } master » |

```

Pact - package installed

```

lukasz@lukasz-PC: ~/babun
{ babun } master » pact install arj
working directory is /setup
Mirror is http://mirrors.kernel.org/sourceware/cygwin/
--2014-04-15 22:38:58-- http://mirrors.kernel.org/sourceware/cygwin//x86/setup.bz2
Resolving mirrors.kernel.org (mirrors.kernel.org)... 149.20.4.71, 2001:4f8:1:10:0:1994:3:14
Connecting to mirrors.kernel.org (mirrors.kernel.org)[149.20.4.71]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 430762 (421K) [application/x-bzip2]
Saving to: `setup.bz2'

100%[=====>] 430,762      327K/s   in 1.3s

2014-04-15 22:39:00 (327 KB/s) - `setup.bz2' saved [430762/430762]

updated setup.ini
Package arj is already installed, skipping
{ babun } master » |

```

Babun oh-my-zsh - auto-update

```

ue63662@v10401: ~
Executing daily babun check:
Prompt speed [OK]
Connection check [OK]
Update check [OK]
[Oh My Zsh] would you like to check for updates?
Type Y to update oh-my-zsh: y
upgrading Oh My Zsh
remote: Counting objects: 84, done.
remote: Compressing objects: 100% (63/63), done.
remote: Total 84 (delta 36), reused 51 (delta 19)
Unpacking objects: 100% (84/84), done.
From https://github.com/robbyrussell/oh-my-zsh
* branch      master      -> FETCH_HEAD
lib/functions.zsh                2 +-
lib/grep.zsh                     23 +++-
plugins/docker/_docker          2 +-
plugins/git-prompt/gitstatus.py  2 +-
.../history-substring-search.zsh 9 +-
plugins/osx/osx.plugin.zsh      2 +-
plugins/per-directory-history/README.md 56 ++++++
.../per-directory-history.plugin.zsh 150 +-----
.../per-directory-history.zsh    149 ++++++
plugins/zsh_reload/zsh_reload.plugin.zsh 13 +-
themes/agnoster.zsh-theme       14 +-
tools/install.sh                2 +-
12 files changed, 253 insertions(+), 171 deletions(-)
create mode 100644 plugins/per-directory-history/README.md
mode change 100644 => 120000 plugins/per-directory-history/per-directory-history.plugin.zsh
create mode 100644 plugins/per-directory-history/per-directory-history.zsh
First, rewinding head to replay your work on top of it...
Fast-forwarded master to eafd5f325208421b82a770e57441dd1063eb5745.

oh my zsh

Hooray! Oh My Zsh has been updated and/or is at the current version.
To keep up on the latest, be sure to follow Oh My Zsh on twitter: http://twitter.com/ohmyzsh
{ ~ } >

```

VIM syntax highlighting

```

vim build.groovy
def command = ["groovy", "dist.groovy", cygwin, input, out, VERSION]
executeCmd(command, workingDir, TEN_MINUTES)
}

def executeRelease() {
    assert getenv("bintray_user") != null
    assert getenv("bintray_secret") != null
    File artifact = new File(getTarget(), "babun-dist/babun-${VERSION}-dist.zip")
    def args = ["groovy", "babun-dist/release/release.groovy", "babun", "babun-dist", VERSION,
        artifact.absolutePath, getenv("bintray_user"), getenv("bintray_secret")]
    executeCmd(args, getRoot(), TWENTY_MINUTES)
}

def shouldSkipModule(String module) {
    File out = new File(getTarget(), module)
    if (out.exists()) {
        println "SKIP ${module}"
        return true
    }
    println "EXEC ${module}"
    return false
}

File getTarget() {
    return new File(getRoot(), "target")
}

File getRoot() {
    return new File(getClass().protectionDomain.codeSource.location.path).parentFile
}

def executeCmd(List<String> command, File workingDir, int timeout) {
    ProcessBuilder processBuilder = new ProcessBuilder(command)
    processBuilder.directory(workingDir)
}

```

Nano syntax highlighting

```

nano babun-core/plugins/pact/src/pact
GNU nano 2.2.6 File: babun-core/plugins/pact/src/pact Modified
{
echo "pact version 1.0.0 (based on apt-cyg 0.57)"
echo "Tweaked and maintained by Tom Bujok (@tombujok)"
echo "Copyright (c) 2014. Released under the MIT."
}

function findworkspace()
{
# default working directory and mirror
mirror=$PACT_REPO
if [[ -z $mirror ]]; then
echo "WARNINIG! ~/.pact/pact.repo does not contains \$PACT_REPO mirror address. Using the default$
mirror=http://mirrors.kernel.org/sourceware/cygwin/
fi
cache=/setup

# work wherever setup worked last, if possible
mirrordir="`echo "$mirror" | sed -e "s/:/%3a/g" -e "s:/:%2f:g"`"

echo working directory is $cache
echo Mirror is $mirror
mkdir -p "$cache/$mirrordir"
cd "$cache/$mirrordir"
}

function getsetup()
{
^G Get Help      ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell

```

Git aliases - git lg

```

git lg
<Tom Bujok>
* da0fc7d - Fixing path in bash prompt (5 weeks ago) <Tom Bujok>
* 68ec85d - Removing unnecessary lines - cleaning up prompts (5 weeks ago) <Tom Bujok>
* a4bfc2f - Adding babun zsh-template installation - spelling fix (5 weeks ago) <Tom Bujok>
* 475ea15 - Adding babun zsh-template installation (5 weeks ago) <Tom Bujok>
* f45adca - work on prompt speed in progress (5 weeks ago) <Tom Bujok>
* 5c9e8c5 - Update README.md (5 weeks ago) <Tom Bujok>
* 92f9016 - Backup of mail notes (5 weeks ago) <Tom Bujok>
* 4b3ff9c - Other combinations of bash prompt - not ready (5 weeks ago) <Tom Bujok>
* 69ec8a0 - quick bash prompt (6 weeks ago) <Tom Bujok>
* 52eca72 - Fixing bash prompt (6 weeks ago) <Tom Bujok>
* 14f7a4d - zsh is the default shell again (6 weeks ago) <Tom Bujok>
* 53641b1 - Corrected shell function in babun (6 weeks ago) <Tom Bujok>
* eabbcca - zsh is not default shell anymore (6 weeks ago) <Tom Bujok>
* 7890599 - Adding the shell handling (6 weeks ago) <Tom Bujok>
* 2ebd475 - Adding the shell handling (6 weeks ago) <Tom Bujok>
* 6fe934e - Adding the shell handling (6 weeks ago) <Tom Bujok>
* 47c606c - Removing bash completion (6 weeks ago) <Tom Bujok>
* ffbdbde0 - Adding note to the shell files (6 weeks ago) <Tom Bujok>
* dd8ee11 - Adding user agent option (6 weeks ago) <Tom Bujok>
* 10ed92f - Merge (6 weeks ago) <Tom Bujok>
|/
* 4b1819f - Adding detect_bloda to the CYGWIN var (6 weeks ago) <Tom Bujok>
* e5b8235 - Fixing bash prompt (6 weeks ago) <Tom Bujok>
* ce504b5 - Removing font handling on startup (6 weeks ago) <Tom Bujok>
* b689359 - Fixing the home installer (6 weeks ago) <Tom Bujok>
* ae2a3d1 - Changing git prompt in bash (6 weeks ago) <Tom Bujok>
* d1b3af0 - Fixed #49 - shortcut script works again (6 weeks ago) <Tom Bujok>
* 0c0dcfa - Fixing bash supplementation (6 weeks ago) <Tom Bujok>
* 0b69600 - Adding git branch name to the bash prompt (6 weeks ago) <Tom Bujok>
* | d51947c - Adding git branch name in bash (6 weeks ago) <Tom Bujok>
|/
* baa7110 - Removing xorg package (6 weeks ago) <Tom Bujok>
* e6d3a68 - Removing old desktop link (6 weeks ago) <Tom Bujok>

```

Git aliases - git st

```

lukasz@lukasz-PC: ~/babun
{ babun } master » git st
## master
D README.adoc
A README.txt
M babun-core/plugins/pact/src/pact
M babun.version
{ babun } master » git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   README.txt
#      modified:   babun.version
#
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    README.adoc
#      modified:   babun-core/plugins/pact/src/pact
#
{ babun } master »

```

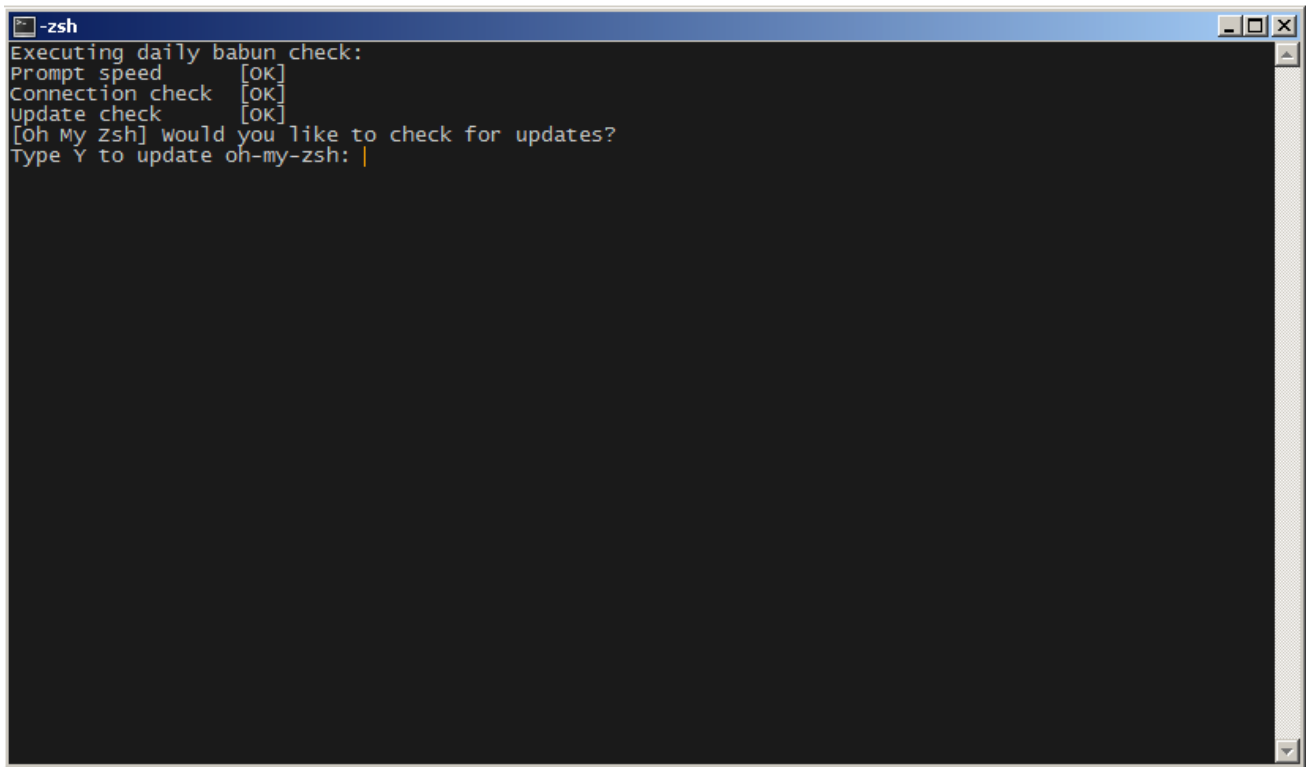
Shell prompt

```

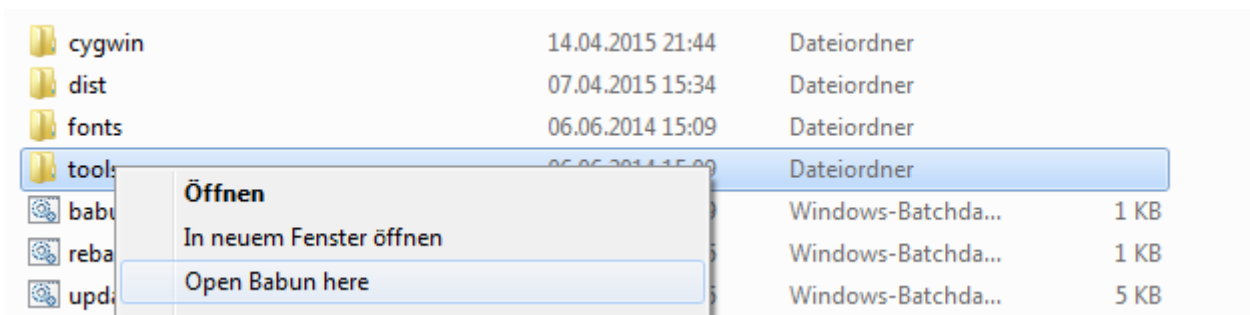
lukasz@lukasz-PC: /usr/local/etc/babun/source
{ source } master » babun check
Executing babun check
Prompt speed      [OK]
Connection check  [OK]
Update check      [OK]
Hint: your version is outdated. Execute 'babun update'
{ source } master » babun update
Executing babun update
  upstream [master]
  installed [0.0.0]
  newest    [1.0.0]
HEAD is now at 312de74 readme updated
Fetching origin
Fetching origin
Already up-to-date.
Already on 'master'
Fixing new line feeds
Making core scripts executable
core
  installed [1]
  newest    [1]
  action    [skip]
pact
  installed [1]
  newest    [1]
  action    [skip]
cacert
  installed [1]
  newest    [1]
  action    [skip]
shell
  installed [1]
  newest    [1]
  action    [skip]
oh-my-zsh

```

Babun update



Open Babun here - Context Menu



Development

Project structure

The project consists of five modules.

babun-packages

The main goal of the `babun-packages` module is to download the cygwin packages listed in the `conf/cygwin.x86.packages` file. The above mentioned packages will be downloaded together with the whole dependency tree. Repositories which the packages are downloaded from are listed in the `conf/cygwin.repositories` file. At the beginning the first repository is taken, if a package is not available in this repo the second repo is used, etc. The process continues until all packages have been downloaded.

All downloaded packages are stored in the `target/babun-packages` folder.

babun-cygwin

The main goal of the `babun-cygwin` module is to download and invoke the native `cygwin.exe` installer. The packages downloaded by the `babun-packages` module are used as the input - all of them will be installed in the offline `cygwin` installation.

It is not trivial to install and zip a local instance of `Cygwin` - there are problems with the symlinks as the `symlink-file-flags` are lost during the compression process. Babun can work it around though. At first, just after the installation, the `symlinks_find.sh` script is invoked in order to store the list of all `cygwin`'s symlinks. This file is delivered as a part of the the `babun`'s core. Then, after `babun` is installed from the zip file on the user's host the `symlinks_repair.sh` script is invoked - it will correct all the broken symlinks listed in the above mentioned file.

Preinstalled `cygwin` is located in the `target/babun-cygwin` folder.

babun-core

The main goal of the `babun-core` module is to install `babun`'s core along with all the plugins and tools. `install.sh` script is invoked during the creation of the distribution package in order to preinstall the plugins. Whenever `babun` is installed on the user's host the `install_home.sh` script is invoke in order to install the `babun`-related files to the `cygwin-user`'s home folder.

Preinstalled `cygwin` with installed `babun` is located in the `target/babun-cygwin` folder.

babun-dist

The main goal of the `babun-dist` module is to zip the ready-made instance of `babun`, copy some installation scripts and zip the distribution.

Distribution package is located in the `target/babun-dist` folder.

babun-doc

This module contains documentation written in `ASCIIDOC`.

Building from source

The project is regularly build on Jenkins, on a slave node featuring the Windows Server OS. The Windows OS is required to fully build the distribution package as one of the goals invokes the native `cygwin.exe` installer. The artifacts created by each module are cached/stored in the target folder after a successful build of each step. This mechanism is not intelligent enough to calculate the diffs so if you would like to fully rebuild the whole dist package make sure to invoke the `clean` goal before the `package` goal. For now it's not possible to invoke a build of a selective modules only.

In order to build the dist package invoke:

```
groovy build.groovy package
```

In order to clean the project target folder invoke:

```
groovy build.groovy clean
```

In order to publish the release version to bintray invoke:

```
groovy build.groovy release
```

The release goal expects the following environment variables: `bintray_user` and `bintray_secret`

Developing a plugin

Every plugin has to consist of three main files:

- `install.sh` - a file that will be executed during the creation of the babun's distribution
- `install_home.sh` - a file that will be executed during the installation of babun to the user's home folder
- `plugin.desc` - a plugin description that contains the `plugin_name` and `plugin_version` variables
- `start.sh` (optional) - a file that will be executed on babun startup
- `exec.sh` (optional) - a file that allows adding commands to babun script

Have a look at the pact plugin - it's a perfect example of a relatively small plugin using all the features.

install.sh

Its main responsibility is to install the plugin - for example to copy the plugin files to, e.g. `/usr/local/etc` or `/usr/local/bin` directories. `install.sh` script is also responsible for preparing the user's home folder template. The template files have to be copied to the `/usr/local/babun/home/<plugin_name>` folder.

`install.sh` will be invoked many times - on every plugin update if the plugin version is higher than the version of the installed plugin - thus it's logic has to work in an incremental way. This mechanism is invoked automatically though. The plugin does not have to contain the version check.

The script has to begin with the following statement:

```
#!/bin/bash
set -e -f -o pipefail
source "/usr/local/etc/babun/source/babun-core/tools/script.sh"
```

install_home.sh

Its main responsibility is to configure the user's home folder with the plugin related stuff, if necessary. For example, it may copy the files from the `/usr/local/babun/home/<plugin_name>` folder to the user's home folder. It is also responsible for any other things that may be necessary during the user's home configuration process.

`install_home.sh` will be invoked many times - on every plugin update if the plugin version is higher than the version of the installed plugin - thus it's logic has to work in an incremental way.

Both scripts (`install.sh` and `install_home.sh`) scripts have to begin with the following statement:

```
#!/bin/bash
set -e -f -o pipefail
source "/usr/local/etc/babun/source/babun-core/tools/script.sh"
```

uninstall.sh (optional)

Its responsibility is to cleanup all entries that a plugin may leave for example on the filesystem or in the windows registry.

plugin.desc

A plugin descriptor looks like this:

```
# plugin descriptor
plugin_name=pact
```

```
plugin_version=1
```

Every time the plugin is changed the version has to be incremented. Otherwise the newest version will not be installed.

start.sh (optional)

The start.sh is an optional script for plugins that require triggering certain actions on every babun start (for example update check).

exec.sh (optional)

If the plugin folder contains an exec.sh script, whenever `babun <plugin_name> xxx yyy` command is invoked, the execution is passed to `<plugin_name>/exec.sh` script with params `xxx yyy`. In this way a plugin may add some additional shell commands without implementing its own `/usr/local/bin/xxx` script.

Branches

The babun's repository contains three main branches:

- master - development branch
- candidate - release candidate branch, no direct commits, only fast forwards from the master/other branch
- release - release, no direct commits, only fast forwards from the candidate branch

In order to check babun update against other branch (for example during a development of a plugin), set the `babun_branch` variable to (master or candidate). External repo's are not supporter (this mechanism has to be extended to include user's repos).

Folder structure in Cygwin

An instance of babun installed in Cygwin is located in the `/usr/local/etc/babun` folder. The folder structure looks like this:

```
├─ babun
│   ├── external
│   │   └─ oh-my-zsh
│   ├── home
│   │   ├── core
│   │   ├── oh-my-zsh
│   │   ├── pact
│   │   └─ shell
│   └─ installed
```

```
|
|
|  └─ babun
|  └─ cacert
|  └─ core
|  └─ git
|  └─ oh-my-zsh
|  └─ pact
|  └─ shell
|
|  └─ source
|      └─ babun.version
|      └─ babun-core
|      └─ babun-cygwin
|      └─ babun-dist
|      └─ babun-doc
|      └─ babun-packages
|      └─ build.groovy
|      └─ README.adoc
|
|  └─ stamps
|      └─ check
|      └─ welcome
|
└─ babun.bash
└─ babun.instance
└─ babun.rc
└─ babun.start
└─ babun.zsh
```

16 directories, 17 files

source

The folder contains the sources of babun checkout from github.

stamps

The folder contains files which modification time indicates certain things to babun. For example `babun check` is executed automatically on babun's start up every 24 hours. Whenever it's invoked a file named `checked` is being modified (the content of the modification does not matter). Whenever the `mod_time` of this file is not within 24 hours and babun is being started a `babun check` will be invoked and the file `check` located in the `stamps` folder will be modified again.

installed

The folder contains files that indicated which versions of babun's plugins and babun itself are installed. Each file contains a number - for example: a file named `core` contains has one line with number `2` in its content. It means that the plugin `core` is installed and has version `2`

external

The folder contains external resources, like cloned repos of other projects (for example oh-my-zsh).

home

The folder contains folders named like plugins. If a plugin needs to install something to user's folder this content has to be copied to `home/<plugin_name>` folder. It's just a store of the user's home files, so that whenever a new user's account is created babun can install user's home related content to the user's home folder (it's the plugin `install_home.sh` script's responsibility, however, to copy this content to the actual user's home folder).

Licence

The source code located in the babun's repository is published under the Apache License, Version 2.0, January 2004 if not stated otherwise.

Since the distribution (zip) package contains the Cygwin's DLLs the distribution package is licensed under the GPLv3+ licence to satisfy the Cygwin's licensing terms (<http://cygwin.com/licensing.html>).

Supporters

Special thanks go to companies who provided free hosting!

XCLOUD

[XCLOUD.ME](#) provided a free hosted OS X instance (a free Xcloud Mini Server subscription). It works like a charm! Thank you!



XCLOUD

"Run, manage and scale your virtual dedicated OS X Server in the Cloud."

XCLOUD is a trademark of AG from Switzerland.

Windows Azure

[Windows Azure](#) provided a free Windows Hosting (a free, renewable MSDN subscription). Everything was organised by @bureado. Thank you!

Microsoft Azure

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States of America and other countries. Windows Azure is a trademark of Microsoft Corporation.

Contribute

Babun is open source and driven by the community. There are many ways to contribute:

- Use it and tell us what you think
- Recommend it to your friends
- Submit a [feature request](#) or a [bug report](#)
- Fork it on [github](#) and submit pull request
- Motivate the community, tweet about the project and star it on github :)

We are looking for new contributors, so if you fancy bash programming and if you would like to contribute a patch or a code up a new plugin give us a shout!

Visit the [development](#) section to find out how to create plugins and extensions.

Meet the team

[@tombujok](#)

[@lukaszpielak](#)

 [Bitdeli Badge](#)

Releases

 5 tags

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Languages

● Shell 60.3% ● Groovy 23.6% ● Batchfile 11.7% ● Vim script 2.6% ● Visual Basic .NET 1.8%