# Contents

1  What is this?  3

2  Install  5

3  Setup  7

4  Usage  9
   4.1  Searching history – rash search  9
   4.2  Showing detailed information – rash show  9
   4.3  Interactive search – rash isearch  10

5  Dependency  11
   5.1  Platforms  11
   5.2  Shells  11

6  Design principle  13

7  License  15

8  More resources  17
   8.1  RASH command line interface  17
   8.2  RASH configuration  25
   8.3  Tips  27

Python Module Index  29
Links:

- Documentation (at Read the Docs)
  - Commands
  - Configuration
  - Tips
- Repository (at GitHub)
- Issue tracker (at GitHub)
- PyPI
- Travis CI
What is this?

Shell history is useful. But it can be more useful if it logs more data points. For example, if you forget which `make` target to run for certain project, you’d want to search shell commands that are run in particular directory. Wouldn’t it be nice if you can do this?:

```
rash search --cwd . "make*
```

RASH records many data points and they are stored in SQLite database. Here is a list of recorded information ¹.

2. Exit code (`$?`).
3. Exit code of pipes (`$PIPESTATUS / $pipestatus`)
4. The time command is started and terminated.
5. Environment variable (`$PATH, $SHELL, $TERM, $HOST, etc.`)
6. Real terminal. `$TERM` is used to fake programs. RASH can detect if you are in tmux, byobu, screen, gnome-terminal, etc.
7. Session information. If you go back and forth in some terminals, RASH does not loose in which sequence you ran the commands in which terminal.

RASH also has interactive search interface. You can see the search result as you type. If you are using zsh, you can execute the result instantaneously.

¹ If you are curious, checkout `rash record --help`.

---

**CHAPTER 1**

---
RASH is written in Python. The easiest way to install is to use `pip` (or `easy_install`, if you wish). You may need `sudo` for installing it in a system directory:

```bash
pip install rash
pip install percol  # if you want interactive search feature
```

If you use virtualenv to install RASH, you may have trouble when switching environment. In that case, it is safe to make an alias to full path of the rash executable:

```bash
alias rash="PATH/TO/VIRTUALENV/bin/rash"
```

If you want to use developmental version, just clone the git repository and add the following in your RC file:

```bash
alias rash="PATH/TO/RASH/rash_cli.py"
```
Add this to your `.zshrc` or `.bashrc`. That’s all:

```bash
eval "$(rash init)"
```

For more information, see `rash init --help`. 
4.1 Searching history – rash search

After your shell history is accumulated by RASH, it’s the time to make use of the history! See rash search --help for detailed information. Here is some examples.

Forget how to run automated test for the current project?:

```
rash search --cwd . --include-pattern "*test*" --include-pattern "tox*"
```

All git commands you ran in one week.:  
```
rash search --time-after "1 week ago" "git*"
```

Some intensive task you ran in the current project that succeeded and took longer than 30 minutes.:  
```
rash search --cwd-under . --include-exit-code 0 --duration-longer-than 30m
```

What did I do after cd-ing to some directory?:
```
rash search --after-context 5 "cd SOME-DIRECTORY"
```

All failed commands you ran at this directory.:  
```
rash search --cwd . --exclude-exit-code 0
```

Count number of commands you ran in one day:  
```
rash search --limit -1 --no-unique --time-after "1 day ago" | wc -l
```

4.2 Showing detailed information – rash show

If you give --with-command-id to rash search command, it prints out ID number for each command history.:  
```
% rash search --with-command-id --limit 5 "*git*"
359  git log
1253 git help clone
1677 git help diff
1678 git diff --word-diff
1780 git merge
```

You can see all information associated with a command with rash show command:
4.3 Interactive search – rash isearch

Searching history using command line is not fast. You can use rash isearch command to interactively search history and see the result immediately as you type.

You need percol to use this feature.

Zsh user can setup a keybind like this:

```
# Type `Ctrl-x r` to start isearch
bindkey "^Xr" rash-zle-isearch
```

Defining this function in your rc file can be handy and it is usable for bash users:

```
rash-isearch(){
    eval "$(rash isearch)"
}
```
Chapter 5

Dependency

RASH tested against Python 2.6, 2.7 and 3.2. However, as some dependencies are not Python 3 compatible, some functionality is missing when used with Python 3.

Python modules:
- watchdog
- parsedatetime

5.1 Platforms

UNIX-like systems  RASH is tested in Linux and I am using in Linux. It should work in other UNIX-like systems like BSD.

Mac OS  I guess it works. Not tested.

MS Windows  Probably no one wants to use a shell tool in windows, but I try to avoid stuff that is platform specific. Only the daemon launcher will not work on Windows but there is several ways to avoid using it. See rash init --help.

5.2 Shells

RASH currently supports zsh and bash.

---

1 These modules do not support Python 3. They are not installed in if you use Python 3 and related functionality is disabled.
RASH’s design is focused on sparseness. There are several stages of data transformation until you see the search result, and they are done by separated processes.

First, rash record command dumps shell history in raw JSON record. This part of program does not touches to DB to make process very fast. As there is no complex transformation in this command, probably in the future version it is better to rewrite it entirely in shell function.

Second, rash daemon runs in background and watches the directory to store JSON record. When JSON record arrives, it insert the data into database.

rash record and rash daemon are setup by simple shell snippet eval $(rash init).

Finally, you can search through command history using search interface such as rash search. This search is very fast as you don’t read all JSON records in separated files.

<table>
<thead>
<tr>
<th>Shell</th>
<th>Raw</th>
<th>SQLite</th>
<th>Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>hooks</td>
<td>JSON</td>
<td>DB</td>
<td>result</td>
</tr>
<tr>
<td>+------</td>
<td>record</td>
<td>+-------</td>
<td>+-------</td>
</tr>
<tr>
<td><code>rash record</code></td>
<td><code>rash daemon</code></td>
<td><code>rash search</code></td>
<td><code>rash show</code></td>
</tr>
<tr>
<td>---------------------/</td>
<td>-------------/</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>rash init</code> setups them</td>
<td>search interface</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
License

RASH is licensed under GPL v3. See COPYING for details.
8.1 RASH command line interface

8.1.1 Search interface

rash search

usage: rash search [-h] [--match-pattern GLOB] [--include-pattern GLOB]
                 [--exclude-pattern GLOB] [--match-regexp REGEXP]
                 [--include-regexp REGEXP] [--exclude-regexp REGEXP]
                 [--cwd DIR] [--cwd-glob GLOB] [--cwd-under DIR]
                 [--time-after TIME] [--time-before TIME]
                 [--duration-longer-than DURATION]
                 [--duration-less-than DURATION] [--include-exit-code CODE]
                 [--exclude-exit-code CODE] [--include-session ID]
                 [--exclude-session ID] [--match-environ-pattern ENV ENV]
                 [--include-environ-pattern ENV ENV]
                 [--exclude-environ-pattern ENV ENV]
                 [--match-environ-regexp ENV ENV]
                 [--exclude-environ-regexp ENV ENV] [--limit NUM]
                 [--no-unique] [--ignore-case] [--reverse]
                 [--sort-by {code,count,program_count,start,stop,success_count,success_ratio,time}]
                 [--sort-by-cwd-distance DIR] [--after-context NUM]
                 [--before-context NUM] [--context NUM]
                 [--context-type {time,session}] [--with-command-id]
                 [--with-session-id] [--format FORMAT] [-f]
                 [-o OUTPUT]
                 [pattern [pattern ...]]

Search command history.

optional arguments:
  -h, --help           show this help message and exit

Filter:
  pattern

Glob pattern to match substring of command. It is as same as
--match-pattern/-m except that the pattern is going to be wrapped by `*`'s. If you want to use strict
glob pattern that matches to entire command, use
--match-pattern/-m. (default: None)
--match-pattern GLOB, -m GLOB
Only commands that match to this glob pattern are
listed. Unlike --include-pattern/-g, applying this
option multiple times does AND match. (default: [])

--include-pattern GLOB, -g GLOB
glob patterns that matches to commands to include.
(default: [])

--exclude-pattern GLOB, -G GLOB
glob patterns that matches to commands to exclude.
(default: [])

--match-regexp REGEXP, -M REGEXP
Only commands that matches to this grep pattern are
listed. Unlike --include-regexp/-e, applying this
option multiple times does AND match. (default: [])

--include-regexp REGEXP, -e REGEXP
Regular expression patterns that matches to commands
to include. (default: [])

--exclude-regexp REGEXP, -E REGEXP
Regular expression patterns that matches to commands
to exclude. (default: [])

--cwd DIR, -d DIR
The working directory at the time when the command was
run. When given several times, items that match to one
of the directory are included in the result. (default: [])

--cwd-glob GLOB, -D GLOB
Same as --cwd but it accepts glob expression.
(default: [])

--cwd-under DIR, -u DIR
Same as --cwd but include all subdirectories.
(default: [])

--time-after TIME, -t TIME
commands run after the given time (default: None)

--time-before TIME, -T TIME
commands run before the given time (default: None)

--duration-longer-than DURATION, -S DURATION
commands that takes longer than the given time
(default: None)

--duration-less-than DURATION, -s DURATION
commands that takes less than the given time (default: None)

--include-exit-code CODE, -x CODE
include command which finished with given exit code.
(default: [])

--exclude-exit-code CODE, -X CODE
exclude command which finished with given exit code.
(default: [])

--include-session ID, -n ID
include command which is issued in given session.
(default: [])

--exclude-session ID, -N ID
exclude command which is issued in given session.
(default: [])

--match-environ-pattern ENV ENV
select command which associated with environment
variable that matches to given glob pattern. (default: [])

--include-environ-pattern ENV ENV, -v ENV ENV
include command which associated with environment
variable that matches to given glob pattern. (default: [])

--exclude-environ-pattern ENV ENV, -V ENV ENV
exclude command which associated with environment variable that matches to given glob pattern. (default: [])

--match-environ-regexp ENV ENV
select command which associated with environment variable that matches to given glob pattern. (default: [])

--include-environ-regexp ENV ENV, -W ENV ENV
include command which associated with environment variable that matches to given glob pattern. (default: [])

--exclude-environ-regexp ENV ENV
exclude command which associated with environment variable that matches to given glob pattern. (default: [])

--limit NUM, -l NUM maximum number of history to show. -l means no limit. (default: 10)

--no-unique, -a Include all duplicates. (default: True)

--ignore-case, -i Do case insensitive search. (default: False)

Sorter:
--reverse, -r Reverse order of the result. By default, most recent commands are shown. (default: False)

--sort-by {code,count,program_count,start,stop,success_count,success_ratio,time}
Sort keys `count`: number of the time command is executed; `success_count`: number of the time command is succeeded; `program_count`: number of the time *program* is used; `start` (=`time`): the time command is executed; `stop`: the time command is finished; `code`: exit code of the command; Note that --sort-by=count cannot be used with --no-unique. If you don’t give anything, it defaults to `count`. However, if you give this option at least once, the default is ignored (i.e., the result is *not* sorted by `count` unless you give it explicitly.). (default: [])

--sort-by-cwd-distance DIR, -y DIR
Sort by distance of recorded cwd from DIR. Commands run at DIR are listed first, then commands run at one level down or one level up directories, and then two level down/up, and so on. (default: None)

Modifier:
--after-context NUM, -A NUM
Print NUM commands executed after matching commands. See also --context option. (default: None)

--before-context NUM, -B NUM
Print NUM commands executed before matching commands. See also --context option. (default: None)

--context NUM, -C NUM
Print NUM commands executed before and after matching commands. When this option is given --no-unique is implied and --sort-by is ignored. (default: None)

--context-type {time,session}
`session`: commands executed in the same shell session; `time`: commands executed around the same
time; (default: time)

Formatter:

--with-command-id Print command ID number. When this is set, --format option has no effect. If --with-session-id is also specified, session ID comes at the first column then command ID comes the next column. (default: False)

--with-session-id Print session ID number. When this is set, --format option has no effect. See also: --with-command-id (default: False)

--format FORMAT Python string formatter. Available keys: command, exit_code, pipestatus (a list), start, stop, cwd, command_history_id, session_history_id. See also: http://docs.python.org/library/string.html#format-string-syntax (default: {command}

-f Set formatting detail. This can be given multiple times to make more detailed output. For example, giving it once equivalent to passing --with-command-id and one more -f means adding --with-session-id. (default: 0)

Misc:

--output OUTPUT Output file to write the results in. Default is stdout. (default: -)

rash show

usage: rash show [-h] command_history_id [command_history_id ...]

Show detailed command history by its ID.

positional arguments:
  command_history_id Integer ID of command history.

optional arguments:
  -h, --help show this help message and exit

rash isearch


Interactive history search that updated as you type.

The query for this program is the same as the one for `rash search` command.

You need percol_ to use this command.

_ percol: https://github.com/mooz/percol

If you use zsh, you can setup a keybind like this to quickly launch iserch and execute the result:::
# Type `Ctrl-x r` to start isearch
bindkey "^Xr" rash-zle-isearch

If you like command or you are not using zsh, you can add something like the following in your rc file to start and execute the chosen command.

```bash
rash-isearch(){
  eval "$(rash isearch)"
}
```

To pass long and complex query, give them after "--", like this:

```bash
rash isearch --
  --cwd . 
  --exclude-pattern "*rash *" 
  --include-pattern "*test*" 
  --include-pattern "tox*" 
  --include-pattern "make *test*"
```

**Positional arguments:**
- **base_query** The part of query that is not shown in UI and is impossible to rewrite in this session. Useful for putting long and complex query. (default: None)

**Optional arguments:**
- `-h`, `--help` show this help message and exit
- `--query QUERY`, `-q QUERY` default query (default: None)
- `--query-template QUERY_TEMPLATE` Transform default query using Python string format. (default: None)
- `--caret CARET` caret position (default: None)

### 8.1.2 System setup interface

**rash init**

```bash
usage: rash init [-h] [--shell SHELL] [--no-daemon]
                  [--daemon-opt DAEMON_OPTIONS]
                  [--daemon-outfile DAEMON_OUTFILE]
```

Configure your shell.

Add the following line in your shell RC file and then you are ready to go:

```bash
eval $(rash init)
```

To check if your shell is supported, simply run:

```bash
rash init --no-daemon
```

If you want to specify shell other than `$SHELL`, you can give `--shell option`:
eval $(rash init --shell zsh)

By default, this command also starts daemon in background to automatically index shell history records. To not start daemon, use --no-daemon option like this::

eval $(rash init --no-daemon)

To see the other methods to launch the daemon process, see `rash daemon --help`.

optional arguments:
- `-h`, `--help` show this help message and exit
- `--shell SHELL` name of shell you are using. directory before the last `/` is discarded. It defaults to $SHELL. (default: /bin/bash)
- `--no-daemon` Do not start daemon. By default, daemon is started if there is no already running daemon. (default: False)
- `--daemon-opt DAEMON_OPTIONS` Add options given to daemon. See "rash daemon --help" for available options. It can be specified many times. Note that --no-error is always passed to the daemon command. (default: [])
- `--daemon-outfile DAEMON_OUTFILE` Path to redirect STDOUT and STDERR of daemon process. This is mostly for debugging. (default: /dev/null)

rash daemon


Run RASH index daemon.

This daemon watches the directory `~/.config/rash/data/record` and translate the JSON files dumped by `record` command into sqlite3 DB at `~/.config/rash/data/db.sqlite`.

`rash init` will start RASH automatically by default. But there are alternative ways to start daemon.

If you want to organize background process in one place such as supervisord, it is good to add `--restart` option to force stop other daemon process if you accidentally started it in other place. Here is an example of supervisord setup::

```
[program:rash-daemon]
command=rash daemon --restart
```

Alternatively, you can call `rash index` in cron job to avoid using daemon. It is useful if you want to use RASH on NFS, as it looks like watchdog does not work on NFS::
# Refresh RASH DB every 10 minutes
*/10 * * * * rash index

optional arguments:
- `--help`    show this help message and exit
- `--no-error` Do nothing if a daemon is already running. (default: False)
- `--restart` Kill already running daemon process if exist. (default: False)
- `--record-path RECORD_PATH` specify the directory that has JSON records. (default: None)
- `--keep-json` Do not remove old JSON files. It turns on --check-duplicate. (default: False)
- `--check-duplicate` do not store already existing history in DB. (default: False)
- `--use-polling` Use polling instead of system specific notification. This is useful, for example, when your $HOME is on NFS where inotify does not work. (default: False)
- `--log-level` {CRITICAL,ERROR,WARNING,INFO,DEBUG} logging level. (default: None)

rash locate

usage: rash locate [-h] [--no-newline] [--output OUTPUT] {base,config,db,daemon_pid,daemon_log}
Print location of RASH related file.

positional arguments:
  {base,config,db,daemon_pid,daemon_log} Name of file to show the path (e.g., config).

optional arguments:
  -h, --help    show this help message and exit
  --no-newline, -n do not output the trailing newline. (default: False)
  --output OUTPUT Output file to write the results in. Default is stdout. (default: -)

rash version

usage: rash version [-h]
Print version number.

optional arguments:
  -h, --help show this help message and exit
8.1.3 Low level commands

**rash record**

```bash
```

Record shell history.

Optional arguments:
- `-h, --help` show this help message and exit
- `--record-type {command,init,exit}` type of record to store. (default: command)
- `--command COMMAND` command that was ran. (default: None)
- `--cwd CWD` Like $PWD, but callee can set it to consider command that changes directory (e.g., cd). (default: None)
- `--exit-code EXIT_CODE` exit code $? of the command. (default: None)
- `--pipestatus PIPESTATUS [PIPESTATUS ...]` $pipestatus (zsh) / $PIPESTATUS (bash) (default: None)
- `--start START` the time COMMAND is started. (default: None)
- `--stop STOP` the time COMMAND is finished. (default: None)
- `--session-id SESSION_ID` RASH session ID generated by --print-session-id. This option should be used with `command` or `exit` RECORD_TYPE. (default: None)
- `--print-session-id` print generated session ID to stdout. This option should be used with `init` RECORD_TYPE. (default: False)

**rash index**

```bash
usage: rash index [-h] [--keep-json] [--check-duplicate] [record_path]
```

Convert raw JSON records into sqlite3 DB.

Normally RASH launches a daemon that takes care of indexing. See `rash daemon --help`.

Positional arguments:
- `record_path` specify the directory that has JSON records. (default: None)

Optional arguments:
- `-h, --help` show this help message and exit
- `--keep-json` Do not remove old JSON files. It turns on --check-duplicate. (default: False)
- `--check-duplicate` do not store already existing history in DB. (default: False)
8.1.4 ZSH functions

**rash-zle-isearch**

To setup Ctrl-x \texttt{r} to start \texttt{rash isearch}, add this to your \texttt{.zshrc}:

```
bindkey "^Xr" rash-zle-isearch
```

8.2 RASH configuration

```python
class rash.config.Configuration
RASH configuration interface.

If you define an object named \texttt{config} in the \texttt{configuration file}, it is going to be loaded by RASH. \texttt{config} must be an instance of \texttt{Configuration}.

\textbf{configuration file} In unix-like systems, it's \texttt{~/.config/rash/config.py} or different place if you set \texttt{XDG_CONFIG_HOME}. In Mac OS, it's \texttt{~/Library/Application Support/RASH/config.py}. Use \texttt{rash locate config} to locate the exact place.
```

Example:

```python
>>> from rash.config import Configuration
>>> config = Configuration()
>>> config.isearch.query = '-u .'
```

Here is a list of configuration variables you can set:

<table>
<thead>
<tr>
<th>Configuration variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{config.record.environ}</td>
<td>Environment variables to record.</td>
</tr>
<tr>
<td>\texttt{config.search.alias}</td>
<td>Search query alias.</td>
</tr>
<tr>
<td>\texttt{config.search.kwds_adapter}</td>
<td>Transform keyword arguments.</td>
</tr>
<tr>
<td>\texttt{config.isearch.query}</td>
<td>Default isearch query.</td>
</tr>
<tr>
<td>\texttt{config.isearch.query_template}</td>
<td>Transform default query.</td>
</tr>
<tr>
<td>\texttt{config.isearch.base_query}</td>
<td>Default isearch base query.</td>
</tr>
</tbody>
</table>

```python
class rash.config.RecordConfig
Recording configuration.
```

```python
\textbf{environ} = None
Environment variables to record.

Each key (str) represent record type (init/exit/command). Each value (list of str) is a list of environment variables to record.

Example usage:

```python
>>> config = Configuration()
>>> config.record.environ['command'] += ['VIRTUAL_ENV', 'PYTHONPATH']
```

```python
class rash.config.SearchConfig
Search configuration.
```

```python
\textbf{alias} = None
Search query alias.

It must be a dict-like object that maps a str to a list of str when "expanding" search query.

Example:
```
```python
>>> config = Configuration()
>>> config.search.alias['test'] = "--exclude-pattern", "--include-pattern", "test*

then:
```
rash search test
```
is equivalent to:
```
rash search --exclude-pattern "*rash *" --include-pattern "*test*"
```

**kwd_adapter = None**

A function to transform keyword arguments.

This function takes a dictionary from command line argument parser and can modify the dictionary to do whatever you want to do with it. It is much more lower-level and powerful than alias. This function must return the modified, or possibly new dictionary.

Example definition that does the same effect as the example in alias:
```
>>> def adapter(kwargs):
...    if 'test' in kwargs.get('pattern', []):
...        kwargs['pattern'] = [p for p in kwargs['pattern']
...            if p != 'test']
...        kwargs['exclude_pattern'].append("*rash *")
...        kwargs['include_pattern'].append("*test")
...    return kwargs

>>> config = Configuration()
>>> config.search.kwd_adapter = adapter
```

**class rash.config.ISearchConfig**

Configure how rash isetach is started.

See also SearchConfig. Once isearch UI is started, SearchConfig controls how search query is interpreted. For example, aliases defined in SearchConfig can be used in isearch.

**query = None**

Set default value (str) for --query option.

If you want to start isearch with the query -d . (only list the command executed at this directory), use the following configuration:
```
>>> config = Configuration()
>>> config.iseach.query = '-d .'
```

As rash-zle-iseach passes the current line content to --query which override this setting, you need to use query_template instead if you want to configure the default query.

**query_template = None**

Transform default query using Python string format.

The string format should have only one field `{0}`. The query given by --query or the one specified by query fills that fielded. Default value is do-nothing template `{0}`.

```
>>> config = Configuration()
>>> config.iseach.query_template = '-d . {0}'
```

**base_query = None**

Set default value (list of str) for --base-query option.
8.3 Tips

8.3.1 Define Zsh ZLE widget

You can use the ZLE widget `rash-zle-isearch` loaded by `rash init` to define your own modified widget. It takes arguments and passes them to `rash isearch` directly. Here is a recipe for “Do What I Mean” search:

```bash
rash-zle-dwim(){
    rash-zle-isearch --query-template "-x 0 -d . @ {0} "
}
zle -N rash-zle-dwim
bindkey "^Xs" rash-zle-dwim
```

In the `configuration file`, you should define an alias called `@` like this (see also `config.search.alias`):

```bash
config.search.alias['@'] = [...] # some complex query
```

8.3.2 Using RASH in old version of zsh

RASH depends on `precmd_functions`/`preexec_functions` hooks in zsh. In old version zsh doesn’t have it. However, you can use RASH by adding this in your `.zshrc`.

```bash
precmd(){
    for f in $precmd_functions
    do
        "$f"
    done
}
preexec(){
    for f in $preexec_functions
    do
        "$f"
    done
}
```

- genindex
- modindex
- search
r

rash.1
rash.config, 25
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
</tbody>
</table>
| C     | Configuration (class in rash.config), 25  
       | configuration file, 25 |
| E     | environ (rash.config.RecordConfig attribute), 25  
       | environment variable  
       | XDG_CONFIG_HOME, 25 |
| I     | ISearchConfig (class in rash.config), 26 |
| K     | kwds_adapter (rash.config.SearchConfig attribute), 26 |
| Q     | query (rash.config.ISearchConfig attribute), 26  
       | query_template (rash.config.ISearchConfig attribute), 26 |
| R     | rash (module), 1  
       | rash.config (module), 25  
       | RecordConfig (class in rash.config), 25 |
| S     | SearchConfig (class in rash.config), 25 |
| X     | XDG_CONFIG_HOME, 25 |