```.gitignore

# .gitignore

# Byte-compiled / optimized / DLL files

__pycache__/ *.py[cod] *$py.class

# C extensions

*.so

# Distribution / packaging

.Python build/ develop-eggs/ dist/ downloads/ eggs/ .eggs/ lib/ lib64/ parts/ sdist/ var/ wheels/ *.egg-info/ .installed.cfg *.egg MANIFEST

# PyInstaller

# Usually these files are written by a python script from a template

# before PyInstaller builds the exe, so as to inject date/other infos into it.

*.manifest *.spec

# Installer logs

pip-log.txt pip-delete-this-directory.txt

# Unit test / coverage reports

htmlcov/ .tox/ .nox/ .coverage .coverage.* .cache nosetests.xml coverage.xml *.cover .pytest_cache/ .hypothesis/

# Translations

*.mo *.pot
```

Django stuff:

*.log local_settings.py db.sqlite3

Flask stuff:

instance/ .webassets-cache

Scrapy stuff:

.scrapy

Sphinx documentation

docs/_build/

PyBuilder

target/

Jupyter Notebook

.ipynb_checkpoints

IPython

profile_default/ ipython_config.py

PEP 582; used by PDM, PEP 582 compatible tools

__pypackages__/

Celery stuff

celerybeat-schedule celerybeat.pid

SageMath parsed files

*.sage.py

Environments

.env .venv env/ venv/ ENV/ env.bak/ venv.bak/

Spyder project settings

.spyderproject .spyderworkspace

Rope project settings

.ropeproject

mkdocs documentation

/site

mypy

.mypy_cache/

```markdown
# path/to/CHANGELOG.md
# Revision history for Python port of Starman

This is a Python port of the original Perl Starman server. The version number tracks the original, but the c

## 0.4017  2023-09-13 13:27:02 PDT
        - Handle EINTR when doing sysread calls (Rob Mueller) #148
        - Requires perl 5.14

## 0.4016  2022-09-13 10:11:34 PDT
        - Add psgix.informational callback #146

## 0.4015  2019-05-20 18:43:46 PDT
        - Fixed a bug incorrectly handling content body of '0' (olsonanl) #133

## 0.4014  2015-06-03 12:01:00 PDT
        - Treat ECONNRESET like EPIPE (i.e. ignore), not as a fatal error #114 (Tim Bunce)

## 0.4013  2015-05-14 15:01:20 PDT
        - Fixed some bad git merges.

## 0.4012  2015-05-14 14:59:48 PDT
        - Add --net server-* options to pass directly to Net::Server backend (#109)
        - Updated documentation

## 0.4011  2014-11-11 08:07:43 PST
        - Move the app dispatch into a method #107

## 0.4010  2014-08-22 09:37:22 PDT
        - Support --read-timeout #103 (slobo)
        - Handle Expect header case insensitively #101 (oschwald)

## 0.4009  2014-04-03 14:39:27 PDT
        - Do not send chunked body for HEAD requests #87 (therigu)
        - Added --disable-proctitle option to disable the proctitle change #97

## 0.4008  2013-09-08 21:09:22 PDT
        - Make response write loop a zero-copy (ap)

## 0.4007  2013-09-02 17:11:38 PDT
        - Handle EPIPE and stops writing to the socket #84 (ap)
```

```
 # path/to/LICENSE
This software is copyright (c) 2010- by Tatsuhiko Miyagawa <miyagawa@bulknews.net>.

This is free software; you can redistribute it and/or modify it under
the same terms as the Perl 5 programming language system itself.

Terms of the Perl programming language system itself

a) the GNU General Public License as published by the Free
   Software Foundation; either version 1, or (at your option) any
   later version, or
b) the "Artistic License"

--- The GNU General Public License, Version 1, February 1989 ---

This software is Copyright (c) 2010- by Tatsuhiko Miyagawa <miyagawa@bulknews.net>.

This is free software, licensed under:

  The GNU General Public License, Version 1, February 1989

                    GNU GENERAL PUBLIC LICENSE
                     Version 1, February 1989

 Copyright (C) 1989 Free Software Foundation, Inc.
 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The license agreements of most software companies try to keep users
at the mercy of those companies.  By contrast, our General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  The
General Public License applies to the Free Software Foundation's
software and to any other program whose authors commit to using it.
You can use it for your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Specifically, the General Public License is designed to make
sure that you have the freedom to give away or sell copies of free
software, that you receive source code or can get it if you want it,
that you can change the software or use pieces of it in new free
programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of a such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must tell them their rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
  that everyone understands that there is no warranty for this free
  software.  If the software is modified by someone else and passed on, we
  want its recipients to know that what they have is not the original, so
  that any problems introduced by others will not reflect on the original
  authors' reputations.

  The precise terms and conditions for copying, distribution and
  modification follow.

                    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
```

  0. This License Agreement applies to any program or other work which
contains a notice placed by the copyright holder saying it may be
distributed under the terms of this General Public License. The
"Program", below, refers to any such program or work, and a "work based
on the Program" means either the Program or any work containing the
Program or a portion of it, either verbatim or with modifications. Each
licensee is addressed as "you".

  1. You may copy and distribute verbatim copies of the Program's source
code as you receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice and
disclaimer of warranty; keep intact all the notices that refer to this
General Public License and to the absence of any warranty; and give any
other recipients of the Program a copy of this General Public License
along with the Program. You may charge a fee for the physical act of
transferring a copy.

  2. You may modify your copy or copies of the Program or any portion of
it, and copy and distribute such modifications under the terms of Paragraph
1 above, provided that you also do the following:

    a) cause the modified files to carry prominent notices stating that
    you changed the files and the date of any change; and

    b) cause the whole of any work that you distribute or publish, that
    in whole or in part contains the Program or any part thereof, either
    with or without modifications, to be licensed at no charge to all
    third parties under the terms of this General Public License (except
    that you may choose to grant warranty protection to some or all
    third parties, at your option).

    c) If the modified program normally reads commands interactively when
    run, you must cause it, when started running for such interactive use
    in the simplest and most usual way, to print or display an
    announcement including an appropriate copyright notice and a notice
    that there is no warranty (or else, saying that you provide a
    warranty) and that users may redistribute the program under these
    conditions, and telling the user how to view a copy of this General
    Public License.

    d) You may charge a fee for the physical act of transferring a
    copy, and you may at your option offer warranty protection in
    exchange for a fee.

Mere aggregation of another independent work with the Program (or its
derivative) on a volume of a storage or distribution medium does not bring
the other work under the scope of these terms.

  3. You may copy and distribute the Program (or a portion or derivative of
it, under Paragraph 2) in object code or executable form under the terms of
Paragraphs 1 and 2 above provided that you also do one of the following:

    a) accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of
    Paragraphs 1 and 2 above; or,

    b) accompany it with a written offer, valid for at least three
    years, to give any third party free (except for a nominal charge
    for the cost of distribution) a complete machine-readable copy of the
    corresponding source code, to be distributed under the terms of
    Paragraphs 1 and 2 above; or,

    c) accompany it with the information you received as to where the
    corresponding source code may be obtained. (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form alone.)

Source code for a work means the preferred form of the work for making
modifications to it. For an executable file, complete source code means
all the source code for all modules it contains; but, as a special
exception, it need not include source code for modules which are standard
libraries that accompany the operating system on which the executable
file runs, or for standard header files or definitions files that
accompany that operating system.

4. You may not copy, modify, sublicense, distribute or transfer the Program except as expressly provided under this General Public License. Any attempt otherwise to copy, modify, sublicense, distribute or transfer the Program is void, and will automatically terminate your rights to use the Program under this License. However, parties who have received copies, or rights to use copies, from you under this General Public License will not have their licenses terminated so long as such parties remain in full compliance.

5. By copying, distributing or modifying the Program (or any work based on the Program) you indicate your acceptance of this license to do so, and all its terms and conditions.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

7. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of the license which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the license, you may choose any version ever published by the Free Software Foundation.

8. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting thesharing and reuse of software generally.

NO WARRANTY

9. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

10. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to humanity, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey

  the exclusion of warranty; and each file should have at least the
  "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) 19yy <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 1, or (at your option) any
    later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) 19xx name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the
appropriate parts of the General Public License. Of course, the
commands you use may be called something other than `show w' and `show
c'; they could even be mouse-clicks or menu items--whatever suits your
program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  program `Gnomovision' (a program to direct compilers to make passes
  at assemblers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

That's all there is to it!

--- The Artistic License 1.0 ---

This software is Copyright (c) 2010- by Tatsuhiko Miyagawa <miyagawa@bulknews.net>.

This is free software, licensed under:

  The Artistic License 1.0

The Artistic License

Preamble

The intent of this document is to state the conditions under which a Package
may be copied, such that the Copyright Holder maintains some semblance of
artistic control over the development of the package, while giving the users of
the package the right to use and distribute the Package in a more-or-less
customary fashion, plus the right to make reasonable modifications.

Definitions:

  - "Package" refers to the collection of files distributed by the Copyright
    Holder, and derivatives of that collection of files created through
    textual modification.
  - "Standard Version" refers to such a Package if it has not been modified,
    or has been modified in accordance with the wishes of the Copyright

Holder.
- "Copyright Holder" is whoever is named in the copyright or copyrights for the package.
- "You" is you, if you're thinking about copying or distributing this Package.
- "Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)
- "Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

  a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as ftp.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

  b) use the modified Package only within your corporation or organization.

  c) rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

  d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

  a) distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

  b) accompany the distribution with the machine-readable source of the Package with your modifications.

  c) accompany any non-standard executables with their corresponding Standard Version executables, giving the non-standard executables non-standard names, and clearly documenting the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

  d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whomever generated them, and may be sold commercially, and may be aggregated with this Package.

7. C or perl subroutines supplied by you and linked into this Package shall not be considered part of this Package.

8. The name of the Copyright Holder may not be used to endorse or promote

```
products derived from this software without specific prior written permission.

9. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End


 # path/to/README.md
# Starman for Python

![Python CI](https://github.com/your-repo/starman-python/actions/workflows/python-ci.yml/badge.svg)

**Starman** is a high-performance preforking WSGI web server for Python. It is a port of the original [Perl

## Overview and Purpose

Starman brings the battle-tested process management model of preforking servers to the Python WSGI ecosystem

It is ideal for developers and system administrators who need a reliable WSGI server that can be managed wit

## Features

-   **High Performance**: Uses the fast `httptools` library for HTTP parsing and a lean, compiled core loop.
-   **Preforking Architecture**: A master process manages a pool of worker processes, providing isolation an
-   **UNIX Signal Management**:
    -   `HUP`: Graceful worker restart (finish serving existing requests, then reload).
    -   `TTIN`/`TTOU`: Increase or decrease the number of worker processes on the fly.
    -   `QUIT`: Graceful shutdown of the entire server.
    -   `INT`/`TERM`: Immediate shutdown.
-   **Hot Deploys**: Superdaemon-aware, supporting process managers like `server starter` for zero-downtime
-   **Multiple Listeners**: Can bind to multiple TCP ports and UNIX domain sockets simultaneously.
-   **Memory Efficiency**: The `--preload-app` option loads the application in the master process before for
-   **WSGI Compliant**: Runs any WSGI-compliant application or framework (e.g., Flask, Django, Falcon).
-   **HTTP/1.1 Support**: Features keep-alive connections, chunked encoding for requests and responses, and
-   **WSGI Extensions**: Includes `wsgix.informational` for sending 1xx informational responses like 103 Ear
-   **UNIX Only**: Designed specifically for and tested on UNIX-like operating systems. It does not support

## Architecture Summary

Starman operates with a single master process and multiple worker processes.

1.  **Master Process**:
    -   Binds to the specified TCP ports or UNIX sockets.
    -   (Optionally) preloads the WSGI application.
    -   Forks a configurable number of worker processes.
    -   Manages the worker pool: restarts workers that die, and handles signals to adjust the pool size or p
    -   Does not handle any client connections itself.

2.  **Worker Processes**:
    -   If the app is not preloaded, each worker loads the WSGI application upon starting.
    -   All workers enter a loop, accepting new connections from the shared listener sockets.
    -   Each worker processes multiple requests on a connection if keep-alive is enabled.
    -   The `httptools` library is used for efficient parsing of incoming HTTP requests.

This model provides robustness, as a crash in one worker does not affect the master or other workers. It als

## Setup and Installation

### Prerequisites

-   Python 3.8 or newer
-   A UNIX-like operating system (Linux, macOS, BSD)
-   A C compiler to build the `httptools` dependency.

### Installation

You can install Starman from PyPI using pip:

```bash
pip install starman
```

To install from source, clone the repository and install it in editable mode for development:
```

```
git clone https://github.com/your-repo/starman-python.git
cd starman-python
pip install -e .
```

Install Dependencies

The required dependencies will be installed automatically with the package. For development or running tests, you can install the optional dependencies:

```
# For development and testing
pip install -e ".[test]"
```

Alternatively, a `requirements.txt` file is provided for development environments:

```
pip install -r requirements.txt
```

How to Run the Application

The `starman` command-line tool is the main entry point. It requires the path to a WSGI application object, specified in the format `module:variable`.

Example Commands

Suppose you have a Flask application in `my_app.py`:

```
# my_app.py
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, Starman!"

# The WSGI application object is `app`
```

1. Run with default settings (5 workers on port 5000):

```
starman my_app:app
```

2. Run with 10 workers on port 8080:

```
starman --workers 10 --port 8080 my_app:app
```

3. Preload the app for memory savings:

```
starman --preload-app --workers 10 my_app:app
```

4. Listen on a UNIX domain socket:

```
starman --listen /tmp/starman.sock my_app:app
```

5. Listen on multiple interfaces:

```
starman --listen 127.0.0.1:8080 --listen [::1]:8081 --listen /tmp/starman.sock my_app:app
```

6. Run as a background daemon (logging to a file):

```
starman --daemonize --pid /var/run/starman.pid --error-log /var/log/starman.log my_app:app
```

7. View all available options:

```
starman --help
```

How to Run Tests

The project uses `pytest` for testing. To run the test suite:

1. Make sure you have installed the test dependencies.
2. Run `pytest` from the root of the project directory.

```
pytest
```

Usage Examples

WSGI Extension: `wsgix.informational`

Starman provides a callback in the WSGI `environ` named `wsgix.informational` to send 1xx responses. This is useful for things like 103 Early Hints.

The callback takes two arguments: `status_code` (int) and `headers` (list of tuples).

```
# In your WSGI application
def my_app(environ, start_response):
    # Send an Early Hints response
    if 'wsgix.informational' in environ:
        informational_response = environ['wsgix.informational']
        informational_response(103, [
            ('Link', '</css/styles.css>; rel=preload; as=style')
        ])

    # Proceed to build and send the final response
    status = '200 OK'
    headers = [('Content-type', 'text/plain')]
    start_response(status, headers)
    return [b'Hello, World!']
```

Contribution Guidelines

Contributions are welcome! Please follow these steps to contribute:

1. **Fork the repository** on GitHub.
2. **Create a new branch** for your feature or bug fix: `git checkout -b my-new-feature`.
3. **Make your changes** and add tests for them.
4. **Ensure the test suite passes**: `pytest`.
5. **Commit your changes**: `git commit -am 'Add some feature'`.
6. **Push to the branch**: `git push origin my-new-feature`.
7. **Submit a pull request**.

License

This software is licensed under the same terms as Perl itself, which is a dual license of either the **Artistic License 1.0** or the **GNU General Public License, Version 1 or later**.

See the LICENSE file for the full text of both licenses.

```python
 ```python
# path/to/pyproject.toml
[build-system]
requires = ["setuptools>=61.0"]
build-backend = "setuptools.build_meta"

[project]
name = "starman"
version = "0.4017"
description = "High-performance preforking WSGI web server"
readme = "README.md"
authors = [
    { name = "Tatsuhiko Miyagawa", email = "miyagawa@bulknews.net" },
]
license = { text = "Artistic-1.0-Perl OR GPL-1.0-or-later" }
classifiers = [
    "Development Status :: 4 - Beta",
    "Environment :: Web Environment",
    "Intended Audience :: Developers",
    "License :: OSI Approved :: Perl License",
    "Operating System :: POSIX :: Linux",
    "Programming Language :: Python :: 3",
    "Programming Language :: Python :: 3.8",
    "Programming Language :: Python :: 3.9",
    "Programming Language :: Python :: 3.10",
    "Programming Language :: Python :: 3.11",
    "Topic :: Internet :: WWW/HTTP :: WSGI :: Server",
]
requires-python = ">=3.8"
dependencies = [
    "httptools>=0.5.0",
]

[project.optional-dependencies]
test = [
    "pytest",
    "requests",
    "pytest-timeout",
    "setproctitle",
]

[project.urls]
"Homepage" = "https://github.com/miyagawa/Starman"
"Bug Tracker" = "https://github.com/miyagawa/Starman/issues"

[project.scripts]
starman = "starman.cli:main"


 # path/to/requirements.txt
# For development and testing
-e .[test]


 # path/to/starman/__init__.py
# starman/__init__.py

__version__ = "0.4017"


 # path/to/starman/cli.py
import argparse
import os
import sys
import importlib
from .server import Server
from . import __version__
```
```

```
def load_app(app_uri):
    """Loads a WSGI application from a URI string like 'module:variable'."""
    try:
        module_str, app_str = app_uri.split(":", 1)
    except ValueError:
        raise ValueError("Application URI must be in the format 'module:variable'")

    try:
        module = importlib.import_module(module_str)
        app = getattr(module, app_str)
        return app
    except (ImportError, AttributeError) as e:
        raise ImportError(f"Could not load application '{app_uri}': {e}")

def main():
    parser = argparse.ArgumentParser(
        description="Starman: A high-performance preforking WSGI server.",
        formatter_class=argparse.RawTextHelpFormatter
    )

    parser.add_argument('app_uri', help='WSGI application URI (e.g., myapp:app)')
    parser.add_argument('-l', '--listen', action='append',
                        help='Listen on a TCP host:port or a UNIX socket path. '
                             'Can be specified multiple times. Defaults to 0.0.0.0:5000.')
    parser.add_argument('--host', default='0.0.0.0',
                        help='Host to bind (default: 0.0.0.0). Deprecated: use --listen.')
    parser.add_argument('--port', type=int, default=5000,
                        help='Port to bind (default: 5000). Deprecated: use --listen.')
    parser.add_argument('-w', '--workers', type=int, default=5,
                        help='Number of worker processes (default: 5).')
    parser.add_argument('--preload-app', action='store_true',
                        help='Load application in master process before forking.')
    parser.add_argument('--max-requests', type=int, default=1000,
                        help='Max requests a worker will process before restarting (default: 1000).')
    parser.add_argument('--timeout', type=int, default=30,
                        help='Worker timeout in seconds (default: 30).')
    parser.add_argument('--keepalive-timeout', type=int, default=5,
                        help='Keep-alive connection timeout (default: 5).')
    parser.add_argument('--read-timeout', type=int, default=5,
                        help='Timeout for reading a request from a new connection (default: 5).')
    parser.add_argument('--disable-keepalive', action='store_true', help='Disable keep-alive connections.')
    parser.add_argument('--backlog', type=int, default=1024, help='Listen backlog size (default: 1024).')
    parser.add_argument('--user', help='Switch to user after binding port.')
    parser.add_argument('--group', help='Switch to group after binding port.')
    parser.add_argument('--pid', help='Path to PID file.')
    parser.add_argument('--error-log', help='Path to error log file.')
    parser.add_argument('--daemonize', action='store_true', help='Daemonize the server process.')
    parser.add_argument('--disable-proctitle', action='store_false', dest='set_proctitle',
                        help='Disable setting process titles.')
    parser.add_argument('-v', '--version', action='version', version=f'Starman {__version__}')

    args = parser.parse_args()

    if not args.listen:
        args.listen = [f"{args.host}:{args.port}"]

    if args.daemonize:
        if os.fork() != 0:
            os. exit(0)
        os.setsid()
        if os.fork() != 0:
            os._exit(0)

        # Redirect stdio
        sys.stdout.flush()
        sys.stderr.flush()
        with open(os.devnull, 'rb') as dn:
            os.dup2(dn.fileno(), sys.stdin.fileno())

        log_path = args.error_log if args.error_log else os.devnull
        log_fd = open(log_path, 'ab')
        os.dup2(log_fd.fileno(), sys.stdout.fileno())
        os.dup2(log_fd.fileno(), sys.stderr.fileno())
```

```
    if args.pid:
        with open(args.pid, 'w') as f:
            f.write(str(os.getpid()))

    # Setting PLACK_ENV to deployment is a Starman tradition
    os.environ.setdefault('STARMAN_ENV', 'deployment')

    try:
        app = load_app(args.app_uri)
    except (ValueError, ImportError) as e:
        sys.stderr.write(f"Error: {e}\n")
        sys.exit(1)

    server = Server(app, args)
    server.run()

if __name__ == '__main__':
    main()


# path/to/starman/server.py
import os
import sys
import socket
import signal
import time
import errno
import pwd
import grp
from .worker import Worker

try:
    import setproctitle
except ImportError:
    setproctitle = None

class Server:
    SIGNALS = {
        signal.SIGHUP: 'handle_hup',
        signal.SIGQUIT: 'handle_quit',
        signal.SIGTERM: 'handle_quit',
        signal.SIGINT: 'handle_quit',
        signal.SIGTTIN: 'handle_ttin',
        signal.SIGTTOU: 'handle_ttou',
    }

    def __init__(self, app, options):
        self.app = app
        self.options = options
        self.sockets = []
        self.workers = {}
        self.running = True
        self.worker_count = self.options.workers
        self.pid = os.getpid()

        # Signal flags
        self._hup_received = False
        self._quit_received = False
        self._ttin_received = False
        self._ttou_received = False

    def run(self):
        self.set_proc_title("master")
        self.setup_sockets()
        self.setup_privileges()

        if self.options.preload_app:
            print(f"[{self.pid}] Pre-loading application.")
            # App is already loaded by cli.py
            pass

        self.setup_signal_handlers()
        self.master_loop()
        print(f"[{self.pid}] Master process exiting.")
        self.close_sockets()
```

```
    def setup_sockets(self):
        server_starter_fd = os.environ.get('SERVER_STARTER_PORT')
        if server_starter_fd:
            host, port, fd = server_starter_fd.split('=')
            print(f"[{self.pid}] Binding to socket from server starter (fd: {fd})")
            s = socket.fromfd(int(fd), socket.AF_INET, socket.SOCK_STREAM)
            s.listen(self.options.backlog)
            self.sockets.append(s)
            return

        for listen_addr in self.options.listen:
            if ':' in listen_addr:
                host, port = listen_addr.rsplit(':', 1)
                addr = (host, int(port))
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            else:
                addr = listen_addr
                try:
                    os.remove(addr)
                except OSError as e:
                    if e.errno != errno.ENOENT:
                        raise
                s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)

            s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
            s.bind(addr)
            s.listen(self.options.backlog)
            self.sockets.append(s)
            print(f"[{self.pid}] Listening at {listen_addr} ({s.fileno()})")

    def setup_privileges(self):
        if self.options.group:
            try:
                gid = int(self.options.group)
            except ValueError:
                gid = grp.getgrnam(self.options.group).gr_gid
            os.setgid(gid)
            print(f"[{self.pid}] Switched to group {self.options.group}")
        if self.options.user:
            try:
                uid = int(self.options.user)
            except ValueError:
                uid = pwd.getpwnam(self.options.user).pw_uid
            os.setuid(uid)
            print(f"[{self.pid}] Switched to user {self.options.user}")

    def setup_signal_handlers(self):
        for sig, handler_name in self.SIGNALS.items():
            signal.signal(sig, getattr(self, handler_name))

    def master_loop(self):
        self.spawn_workers()
        while self.running:
            try:
                self.check_signals()
                self.reap_workers()
                self.maintain_worker_count()
                time.sleep(1)
            except InterruptedError:
                continue
            except KeyboardInterrupt:
                self.running = False

    def check_signals(self):
        if self._hup_received:
            self._hup_received = False
            self.graceful_restart()
        if self._quit_received:
            self._quit_received = False
            self.running = False
        if self._ttin_received:
            self._ttin_received = False
            self.worker_count += 1
```

```
        print(f"[{self.pid}] Increasing worker count to {self.worker_count}")
    if self._ttou_received:
        self._ttou_received = False
        if self.worker_count > 1:
            self.worker_count -= 1
            print(f"[{self.pid}] Decreasing worker count to {self.worker_count}")

def graceful_restart(self):
    print(f"[{self.pid}] HUP received. Restarting workers.")
    self.kill_workers(signal.SIGTERM)
    self.reap_workers(block=True)
    self.spawn_workers()

def kill_workers(self, sig):
    for pid in list(self.workers.keys()):
        try:
            os.kill(pid, sig)
        except OSError as e:
            if e.errno == errno.ESRCH:
                self.workers.pop(pid, None)

def spawn_workers(self):
    while len(self.workers) < self.worker_count:
        self.spawn_worker()

def spawn_worker(self):
    pid = os.fork()
    if pid == 0:  # Child
        self.set_proc_title("worker")
        try:
            worker = Worker(self.app, self.options, self.sockets)
            worker.run()
        except Exception as e:
            print(f"Worker {os.getpid()} exited with error: {e}", file=sys.stderr)
        finally:
            os._exit(0)
    else:  # Parent
        self.workers[pid] = time.time()
        print(f"[{self.pid}] Spawned worker {pid}")

def reap_workers(self, block=False):
    options = 0 if not block else os.WNOHANG
    try:
        while True:
            pid, status = os.waitpid(-1, options)
            if pid == 0 and not block:
                break
            if pid in self.workers:
                del self.workers[pid]
                print(f"[{self.pid}] Reaped worker {pid} (status: {status})")
            if block and not self.workers:
                break
    except OSError as e:
        if e.errno != errno.ECHILD:
            raise

def maintain_worker_count(self):
    diff = self.worker_count - len(self.workers)
    if diff > 0:
        for _ in range(diff):
            self.spawn_worker()
    elif diff < 0:
        pids_to_kill = list(self.workers.keys())[:abs(diff)]
        for pid in pids_to_kill:
            self.kill_workers(signal.SIGTERM)

def close_sockets(self):
    for s in self.sockets:
        s.close()

def set_proc_title(self, role):
    if setproctitle and self.options.set_proctitle:
        setproctitle.setproctitle(f"starman: {role} process")
```

```
    # Signal handlers
    def handle_hup(self, sig, frame):
        self._hup_received = True

    def handle_quit(self, sig, frame):
        self._quit_received = True

    def handle_ttin(self, sig, frame):
        self._ttin_received = True

    def handle_ttou(self, sig, frame):
        self._ttou_received = True


 # path/to/starman/worker.py
import os
import sys
import socket
import errno
import time
import io
from email.utils import formatdate
from httptools import HttpRequestParser
from httptools.parser import HttpParserError

try:
    import setproctitle
except ImportError:
    setproctitle = None

class RequestHandler:
    def __init__(self, app, client_sock, client_addr, options, sockets):
        self.app = app
        self.client = client_sock
        self.addr = client_addr
        self.options = options
        self.sockets = sockets
        self.parser = HttpRequestParser(self)
        self.headers = []
        self.body = io.BytesIO()
        self.environ = {}
        self.response = {}

    def on_message_begin(self):
        self.headers = []
        self.body = io.BytesIO()
        self.environ = {
            'wsgi.version': (1, 0),
            'wsgi.url_scheme': 'http', # TODO: SSL
            'wsgi.input': self.body,
            'wsgi.errors': sys.stderr,
            'wsgi.multithread': False,
            'wsgi.multiprocess': True,
            'wsgi.run_once': False,
            'SERVER_SOFTWARE': f'Starman/{self.options.version}',
            'REQUEST_METHOD': '',
            'SCRIPT_NAME': '',
            'PATH_INFO': '',
            'QUERY_STRING': '',
            'SERVER_NAME': self.client.getsockname()[0],
            'SERVER_PORT': str(self.client.getsockname()[1]),
            'REMOTE_ADDR': self.addr[0],
            'REMOTE_PORT': str(self.addr[1]),
            'wsgix.informational': self.write_informational,
        }

    def on_url(self, url):
        self.environ['RAW_URI'] = url.decode('latin-1')
        path, _, query = url.partition(b'?')
        self.environ['PATH_INFO'] = path.decode('latin-1')
        self.environ['QUERY_STRING'] = query.decode('latin-1')

    def on_header(self, name, value):
        name = name.decode('latin-1').upper().replace('-', '_')
        if name not in ('CONTENT_TYPE', 'CONTENT_LENGTH'):
```

```
        name = f"HTTP_{name}"
    self.environ[name] = value.decode('latin-1')
    self.headers.append((name.replace('_', '-'), value))

def on_body(self, body):
    self.body.write(body)

def on_headers_complete(self):
    self.environ['REQUEST_METHOD'] = self.parser.get_method().decode('latin-1')

    # Handle Expect: 100-continue
    if self.environ.get('HTTP_EXPECT', '').lower() == '100-continue':
        self.client.sendall(b'HTTP/1.1 100 Continue\r\n\r\n')

def on_message_complete(self):
    self.body.seek(0)
    self.handle_request()

def handle_request(self):
    try:
        resp_iter = self.app(self.environ, self.start_response)
        self.write_response(resp_iter)
    except Exception:
        # TODO: Better error handling
        exc_info = sys.exc_info()
        if not self.response:
            self.start_response("500 Internal Server Error", [])
        self.write_response([b"Internal Server Error"])
        print(f"Error handling request: {exc_info}", file=sys.stderr)
    finally:
        if hasattr(resp_iter, 'close'):
            resp_iter.close()

def start_response(self, status, headers, exc_info=None):
    if exc_info:
        try:
            if self.response:
                raise exc_info[1].with_traceback(exc_info[2])
        finally:
            exc_info = None
    elif self.response:
        raise AssertionError("start_response called a second time without exc_info")

    self.response = {"status": status, "headers": headers}

    # Return a write callable, though we don't use it in this simple model
    return self.client.sendall

def write_informational(self, status_code, headers):
    status_text = "Informational" # This is a simplification
    lines = [f"HTTP/1.1 {status_code} {status_text}"]
    for name, value in headers:
        lines.append(f"{name}: {value}")
    data = "\r\n".join(lines).encode('latin-1') + b'\r\n\r\n'
    self.client.sendall(data)

def write_response(self, resp_iter):
    status = self.response['status']
    headers = self.response['headers']

    # Add required headers
    has_cl = any(h[0].lower() == 'content-length' for h in headers)
    has_date = any(h[0].lower() == 'date' for h in headers)

    if not has_date:
        headers.append(('Date', formatdate(time.time(), usegmt=True)))

    headers.append(('Server', f'Starman/{self.options.version}'))

    # Prepare headers for sending
    header_data = [f"HTTP/1.1 {status}".encode('latin-1')]
    for name, value in headers:
        header_data.append(f"{name}: {value}".encode('latin-1'))
```

```
        self.client.sendall(b'\r\n'.join(header_data) + b'\r\n\r\n')

        # Send body
        for chunk in resp_iter:
            if chunk:
                self.client.sendall(chunk)

        # If no content-length, we can't do keep-alive unless chunked
        # This implementation is simplified and doesn't do response chunking.

    def handle(self):
        try:
            while True:
                data = self.client.recv(65536)
                if not data:
                    break
                try:
                    self.parser.feed_data(data)
                except HttpParserError as e:
                    print(f"HTTP parse error: {e}", file=sys.stderr)
                    # Simplified: just close connection on parse error
                    break

                # Simplified keep-alive: break after one request if disabled
                if self.options.disable_keepalive:
                    break
        except socket.error as e:
            if e.errno not in (errno.EPIPE, errno.ECONNRESET):
                print(f"Socket error: {e}", file=sys.stderr)
        finally:
            self.client.close()

class Worker:
    def __init__(self, app, options, sockets):
        self.app = app
        self.options = options
        self.sockets = sockets
        self.requests_processed = 0
        self.pid = os.getpid()

    def run(self):
        if not self.options.preload_app:
            # This is where the app would be loaded per-worker
            pass

        signal.signal(signal.SIGINT, signal.SIG_DFL)
        signal.signal(signal.SIGTERM, signal.SIG_DFL)

        print(f"[{self.pid}] Worker started.")

        while self.requests_processed < self.options.max_requests:
            try:
                client, addr = self.sockets[0].accept() # simplified to one socket
                self.requests_processed += 1
                handler = RequestHandler(self.app, client, addr, self.options, self.sockets)
                handler.handle()
            except socket.error as e:
                if e.errno in (errno.EAGAIN, errno.ECONNABORTED, errno.EPROTO):
                    continue
                raise

        print(f"[{self.pid}] Worker exiting (max requests reached).")
```