

Задание

HTTP сервер

Задание: разработать веб-сервер частично реализующий протокол HTTP, архитектуру выбрать самостоятельно. 1. Разрешается использовать библиотеки помогающие реализовать асинхронную обработку соединений, запрещается использовать библиотеки реализующие какую-либо часть обработки HTTP. Работать с сокетами и всем прочим нужно самостоятельно. 2. Провести нагрузочное тестирование, проверку стабильности и корректности работы. 3. Если сервер асинхронный, то обязательно использовать `epoll` (<https://github.com/m13253/python-asyncore-epoll>)

Подсказка: некоторые фичи (например, `SO_REUSEPORT`) могут некорректно работать на Mac и прочих неUnix системах. Лучше экспериментировать в контейнере с CentOS 7 или тому подобным.

Веб-сервер должен уметь:

- Отвечать 200, 403 или 404 на GET-запросы и HEAD-запросы
- Отвечать 405 на прочие запросы
- Возвращать файлы по произвольному пути в `DOCUMENT_ROOT`.
- Вызов `/file.html` должен возвращать содержимое `DOCUMENT_ROOT/file.html`
- `DOCUMENT_ROOT` задается аргументом командной строки `-r`
- Возвращать `index.html` как индекс директории
- Вызов `/directory/` должен возвращать `DOCUMENT_ROOT/directory/index.html`
- Отвечать следующими заголовками для успешных GET-запросов: `Date`, `Server`, `Content-Length`, `Content-Type`, `Connection`
- Корректный `Content-Type` для: `.html`, `.css`, `.js`, `.jpg`, `.jpeg`, `.png`, `.gif`, `.swf`
- Понимать пробелы и `%XX` в именах файлов
- (задание со "звездочкой") Масштабироваться на несколько worker'ов. Числов worker'ов задается аргументом командной строки `-w`

Что проверять:

- Проходят тесты <https://github.com/s-stupnikov/http-test-suite>
- http://localhost/httptest/wikipedia_russia.html корректно показывается в браузере

- Нагрузочное тестирование: запускаем `ab -n 50000 -c 100 -r http://localhost:8080/` и смотрим результат
– Опционально: вместо `ab` воспользоваться `wrk`

Что на выходе:

- сам сервер в `httpd.py`. Это точка входа (т.е. этот файл обязательно должен быть), можно разбить на модули.
- `README.md` с описанием использованной архитектуры (в двух словах: `asynchronous/thread pool/prefork/...`) и результатами нагрузочного тестирования

Цель задания: разобраться в различных аспектах сетевого взаимодействия. В результате улучшится понимание того как работают веб-сервера, будет получен навык написания сетевых приложений.

Критерии успеха: задание **обязательно** кроме частей со “звездочкой”, которые являются расширением основного функционала и выполняются по возможности. Критерием успеха является работающий согласно заданию код, для которого проверено соответствие пер8, написана минимальная документация с примерами запуска, в `README`, например. Далее успешность определяется `code review`.

Deadline

Задание нужно сдать через неделю. То есть ДЗ, выданное в понедельник, нужно сдать до следующего занятия в понедельник. Код, отправленный на ревью в это время, рассматривается в первом приоритете. Нарушение дедлайна (пока) не карается, пытаться сдать ДЗ можно до конца курсы. Но код, отправленный с опозданием, когда по плану предполагается работа над более актуальным ДЗ, будет рассматриваться в более низком приоритете без гарантий по высокой скорости проверки

Обратная связь

Студент коммитит все необходимое в свой `github/gitlab` репозиторий. Далее необходимо зайти в ЛК, найти занятие, ДЗ по которому выполнялось, нажать “Чат с преподавателем” и отправить ссылку. После этого ревью и общение на тему ДЗ будет происходить в рамках этого чата.