qКурсовая работа

Общие замечания

- Работа ведётся в парах через кафедральную систему управления версиями и ведения проекта.
- Каждый студент получает индивидуальное задание, решение которого описывается в его РПЗ.
- Поскольку задания образуют пары (прием почты-передача почты), то на пару студентов выдается одна система контроля версий, для хранения кодовой базы обоих проектов.
- Примерная процедура приёма: git clone && cd server (cd client) && make && make tests && make report, размышления над результатом.

Место работы в курсе

В ходе выполнения работы студент должен получить или закрепить следующиенавыки:

- проектирования реализации сетевого протокола по имеющейся спецификации;
- реализации сетевых приложений на Си;
- созданием реализацией сетевой службы без создания нити на каждое соединение;
- автоматизированного системного тестирования ПО сетевой службы;
- тестирования утечек памяти;
- создания сценариев сборки ПО;
- групповой работы с использованием VCS и CI;

Исходные данные и цель работы

Исходные данные.

- Спецификация протокола SMTP (RFC 5321) и смежные RFC.
- Средства разработки GNU, POSIX-совместимая система, компиляторы GCC и CLang.
- Дополнительные библиотеки:
  - cunit;
  - autogen;
  - firedns;
  - libconfig (1.3.2).
- Python 2.6, Ruby 1.8 или except для огранизации системного тестирования.

Содержание работы. Проектирование, реализация и тестирование почтового сервера (MTA). В каждом варианте указано, следует ли реализовывать в нём SMTP-сервер (для приёма почты) или SMTP-клиент (для удаленной доставки).

- Сервер должен подерживать команды HELO и EHLO, MAIL, RCPT, DATA, RSET, QUIT, VERIFY протокола SMTP.
- Серверу следует реализовать только указанные команды. VERIFY должен при этом выдавать всегда ошибку.
- Отправитель должен поддерживать набор команд, достаточный для отправки почты как минимум одной крупной почтовой публичной службе с веб-интерфейсом (по выбору студента).

Решаемые задачи

В скобках указаны используемые граф. нотации для отражения решения указанных задач в РПЗ.

- Аналитические задачи.
  - Плюсы и минусы указанного в задании способа решение проблемы «развязывания» потоков выполнения сервера и соединений.
  - Выделение сущностей предметной области: письмо, копия, получатель-ящик, получатель-хост (он же MX), поля To:, etc. (ER-диаграмма).
  - Выбор способа разделения нагрузки между рабочими потоками/процессами (если в задании из более одного).
  - Для SMTP-отправителя: выбор способа хранения писем в очередях.
- Конструкторские задачи.
  - Описание конечного автомата состояний протокола на основе его спецификации с учётом отслеживания таймаутов (dot из кода), как для задания клиента так и для задания сервера.
  - Для SMTP-отправителя: описание конечного автомата жизненного цикла сообщения (dot из кода).
  - Описание синтаксиса команд протокола в виде регулярных выражений.
  - Выбор и описание основных используемые структуры данных: логическое (ER-диаграмма) и физическое («стрелочки-квадратики», graphviz пойдёт) представления. Выбор структур, локальных для рабочих потоков, выбор механизма синхронизации доступа к общим структурам.
  - Проектирование взаимодействия подсистем (UML-sequences или dot в стиле диаграммы коммуникаций).
  - Проектирование алгоритмов обработки соединений в одном потоке выполнения (псевдокод).
  - Описание связей потоков/ процессов по IPC (нотация dot), распределение потоков по процессам.
  - Описание механизма хранения почты в очередях и алгоритмов работы с нею.
- Технологические задачи.
  - Выбор средств синхронизации доступа к данным в памяти и на диске.
  - Описание процесса генерации (кода и РПЗ) и сборки.
  - Написание исходного кода.
  - Создания сценария сборки ПО, создания записки, проведения тестов.
  - Описания файла конфигурации / параметров командной строки.
  - Создание системы модульного и системного тестирования, проверку утечек памяти.
  - Создание подраздела по итогам тестирования.

Требования к содержимому репозитория

- Исходный код должен храниться в кафедральной VCS (git), которая должна использоваться постоянно в ходе работы (commit & push).
- Клиент должен лежать в каталоге client, сервер — в каталоге server, общие части должны лежать директории common.
- Зарещается класть в репозитории backup-файлы, объектные, библиотечные и исполнимый файлы.
- Весь процесс компиляции, тестирования, создания отчета задаётся make-файлом. В состав сервера и клиента должен входить Makefile, имеющий цели для следующих

задач: сборки системы, выполнения тестирования (проверка утечек памяти, стиля, модульное тестирование, тестирование протокола), создание pdf с РПЗ. Названия целей:
    - по-умолчанию (сборка исполняемого файла, без РПЗ);
    - test_units, test_memory, test_system, test_style, tests (все четыре);
    - report (сборка РПЗ).
- В директории common должен быть Makefile с целями tests, test_units, test_memory, test_style.
- Просьба не менять рецепты тестирования Bitten, мы попробуем работать с одинаковыми (предполагается соблюдение соглашений о расположении каталогов и названиях целей).

Язык програмирования и сборка

- В качестве языка программирования используется язык Си стандарта C99 и компиляторы gcc 4.x и clang.
- Созданные полностью вручную исходные файлы **нужно** компилировать с -Wall -Werror.
- Для сборки используется GNU make. **Не надо** использовать cmake/autotools/qmake etc.

Средства подготовки отчёта

- Автомат протокола должен описываться с помощью библиотек cfsm или autofsm. Из его описания в ходе сборки должен генерироваться машинный код и рисунок в отчет (через преобразование в dot, опционное наведение лоска с помощью sed, и вставку в документ через dot2tex).
- Регулярные выражения, описывающие команды и ответы протокола, попадают в РПЗ из кода.
- Сценарии и результаты тестирования вставляются в отчет непосредственно в ходе сборки по результатам проведения тестирования.

Требования к программной реализации

Структуры данных

- Для работы с регулярными выражениями следует использовать Perl-compatible regular expressions и соответствующую библиотеку (pcre).
- Для организации списков и очередей следует использовать queue.h, для ассоциативных массивов — tree.h.
- Корректно и эффективно разделяйте доступ к данным, разделяемым несколькими потоками. Не синхронизируйте доступ к структурам, локальным (по логике работы) для потока.
- Не допускайте утечек памяти (в том числе логических) и «распухания» структур очередей на диске.
- Для хранения доставленной почты следует использовать формат maildir. Почта хранится как /каталог_почты/пользователь/Maildir.
- Для передачи писем от сервера клиенту (внутри MTA) используется единственный maildir.
- Формат очередей отсылаемых сообщений не регламентируется.

Комментарии и стиль

- Все исходные тексты должны использовать кодировку UTF-8 и \n как символ новой строки.
- Исходный код на Си (не считая автосгенерённого) должен отвечать некоторому стилю кодирования. Следует иметь в сценарии цель test_style для его проверки.
- Стиль кодирования: отсуп в 4 пробела, разделение_подчёркиванием, остальное в целом по вкусу.

Параметры / файл конфигурации

- Для разбора параметров командной строки рекомендуется (но не обязательно) использовать autoopts. Файл конфигурации программы не обязателен, но возможен (используйте libconfig). Минимальный список поддерживаемых параметров командной строки / файла конфигурации — ниже.
    - Для программы получения почты: порт, корневой каталог для почты (например, /var/mail, или /home/student/test_mail), корневой каталог для очередей сообщений, пользователь и группа для понижения привелегий, имя файла журнала (лог), сеть (сети), для который разрешен релей почты. сетевой адрес привязки, максимальное число рабочих потоков / процессов (см. задание).
    - Для программы передачи почты: корневой каталог для очередей сообщений, имя файла журнала (лог), общее время на попытки отправить письмо, минимальное время между попытками повтора, максимальное число рабочих потоков / процессов (см. задание).

Прочее

- Для хранения оригинального получателя письма следует использовать заголовок «X-Original-To:».
- ~~Для тестирования отсылки по MX-записи выделено специальное доменное имя local.iu7.bmstu.ru и имена c l01.iu7.bmstu.ru по l12.iu7.bmstu.ru.~~
- Идентификация пользователя не нужна.
- Система должна реализовать журналирование. Допустимо выводить сообщения в stdout, ошибки --- в stderr. Журналирование следует делать как указанно в варианте, для связи (если она требуется) нужно использовать Posix MQ или SysV MQ (последнее добавлено по просьбе маководов, где как минимум год назад не было Posix MQ).
- Из-за проблемы *well-known ports* программа должна реализовать понижение привилегий (если указаны соответствующие параметры запуска).
- В исходных текстах должна быть предусмотрена обработка всех возможных ошибок и корректное освобождение занятых ресурсов в случае ошибок.
- Программа должна работать с IPv4 И IPv6.
- Программа должна отслеживать таймауты для разрыва соединения с клиентом.
- В силу непереносимости вызова sendfile() в этом сезоне его использовать запрещено.
- Процессы создаются только через fork(). Никакого clone(), он непереносим (а его разрешение дало странный результат в прошлом сезоне).
- Увеличение размера буфера через realloc() / strndup() вообще плохо, а с малым шагом — особенно.
- При системном тестировании из скрипта нужно повторять connect() до успеха, а не спать.

Тестирование

- Для модульного тестирования нужно использовать cunit.
- Для тестирования реализации протокола в целом могут использоваться сценарии программы expect и/или собственные программа на языке Python 2.6 или Ruby 1.8, а так же утилиты netcat, netcat6.
- Для тестирования утечек памяти следует использовать программу Valgrind.
- Тестирование утечек памяти следует реализовывать как модульное тестирование под Valgrind + системное тестирование под Valgrind
- ~~На кафедре преподавателем будут развёрнуты билд-машины на debian 6.0 для x86 и amd64 для CI.~~

Примерное содержание РПЗ

РПЗ должно отражать решение всех стоящих в КР задач.

Введение

- Задание, на основе конкретного варианта.
- Цель работы и решаемые вами укрупнённые основные задачи.

Аналитический раздел

Описание решения всех аналитических задач.

Конструкторский раздел

Описание решения всех конструкторских задач.

Технологический раздел

- Описание решения всех технологических задач.
- Платформы и компиляторы, на которых ПО проверенно на работоспособность.
- Графическое описание процесса сборки. Строится из Makefile прилагаемой утилитой.
- Графы вызова основных функций программы. Они строятся с помощью GNU cflow. Делайте графы читаемыми, разбивая ПО по подсистемам.
- Описание и результаты модульного тестирования.
- Описание и результаты системного тестирования.
- Описание и результаты тестирования утечек памяти.

Заключение

Варианты заданий

- Варианты имеют следующие различия:
    - какой подход используется для опроса сокетов;
    - как реализовано журналирование;
    - нужно ли использовать свой (т.е. не полагающийся на конкурентное ожидание) способ выбора «работника», если указан;
    - нужно ли проверять DNS для внешних (т.е. не попадающих в сети релея) отправителей почты, если указан.

- Конкретный формат очереди(ей) сообщений, куда информация записывается SMTP-сервером и откуда считывается SMTP-клиентом может быть свой у каждой пары заданий, но он (очевидно) должен совпадать у обоих работы в паре.

- Создание SMTP-сервера, обеспечивающего локальную доставку и добавление в очередь удаленной доставки.
    - Вариант 1. Используется вызов poll и рабочие потоки. Журналирование в отдельном потоке. Нужно проверять обратную зону днс.
    - Вариант 2. Используется вызов select и рабочие процессы. Журналирование в отдельном процессе. Нужно проверять обратную зону днс.
    - Вариант 3. Используется вызов pselect и рабочие потоки. Журналирование в отдельном процессы. Нужно проверять обратную зону днс.
    - Вариант 4. Используется вызов poll и рабочие процессы. Журналирование в отдельном процессы. Нужно проверять обратную зону днс.
    - Вариант 5. Используется вызов poll и рабочие потоки. Журналирование в отдельном потоке.
    - Вариант 6. Используется вызов select и рабочие процессы. Журналирование в отдельном процессе.
    - Вариант 7. Используется вызов pselect и рабочие потоки. Журналирование в отдельном процессы.
    - Вариант 8. Используется вызов poll и рабочие процессы. Журналирование в отдельном процессы.
    - Вариант 9. Используется вызов poll и единственный рабочий поток. Журналирование в отдельном потоке. Нужно проверять обратную зону днс.
    - Вариант 10. Используется вызов pselect и единственный рабочий поток. Журналирование в отдельном процессе. Нужно проверять обратную зону днс.
    - Вариант 11. Используется вызов select и единственный рабочий поток. Журналирование в отдельном потоке.
    - Вариант 12. Используется вызов poll и единственный рабочий поток. Журналирование в отдельном процесс.
    - Вариант 13. Используется вызов select и рабочие процессы. Журналирование в отдельном потоке основного процесса.
- Создание SMTP-клиента (как части MTA), обеспечивающего удаленную доставку и поддерживающего очереди сообщений.
- Все варианты предполагают обработку нескольких исходящих соединений в одном потоке выполнения (т.е., одном процесс или одном потоке).
- На один удалённый MX надо создавать не более одного сокета (допустимый вариант — на один удалённый IP не более одного сокета).
- Следует использовать отдельную очередь собщений для каждого MX.
    - Вариант 21. Используется вызов poll и рабочие процессы. Журналирование в отдельном процессе. Пытаться отправлять все сообщения для одного MX за одну сессию.
    - Вариант 22. Используется вызов pselect и рабочие потоки. Журналирование в отдельном потоки. Пытаться отправлять все сообщения для одного MX за одну сессию.
    - Вариант 23. Используется вызов select и рабочие процессы. Журналирование в отдельном процессе. Пытаться отправлять все сообщения для одного MX за одну сессию.
    - Вариант 24. Используется вызов poll и рабочие потоки. Журналирование в отдельном процессе. Пытаться отправлять все сообщения для одного MX за одну сессию.

- Вариант 25. Используется вызов poll и рабочие процессы. Журналирование в отдельном процессе. Не обязательно пытаться отправлять все сообщения для одного MX за одну сессию.
- Вариант 26. Используется вызов pselect и рабочие потоки. Журналирование в отдельном потоки. Не обязательно пытаться отправлять все сообщения для одного MX за одну сессию.
- Вариант 27. Используется вызов select и рабочие процессы. Журналирование в отдельном процессе. Не обязательно пытаться отправлять все сообщения для одного MX за одну сессию.
- Вариант 28. Используется вызов poll и рабочие потоки. Журналирование в отдельном процессе. Не обязательно пытаться отправлять все сообщения для одного MX за одну сессию.
- Вариант 29. Используется вызов poll и единственный рабочий поток (или процесс). Журналирование в отдельном потоке. Пытаться отправлять все сообщения для одного MX за одну сессию.
- Вариант 30. Используется вызов poll и единственный рабочий поток (или процесс). Журналирование в отдельном процессе. Пытаться отправлять все сообщения для одного MX за одну сессию.
- Вариант 31. Используется вызов select и единственный рабочий поток (или процесс). Журналирование в отдельном процессе. Пытаться отправлять все сообщения для одного MX за одну сессию.
- Вариант 32. Используется вызов pselect и единственный рабочий поток (или процесс). Журналирование в отдельном потоке. Пытаться отправлять все сообщения для одного MX за одну сессию.

Консультациии по курсовому проектированию

- После лекции по ПВС по суббтам.

Распределение заданий

- При выполнении задания вне пары вся внешняя почта просто помещается в выделенный для этого maildir.