Лабораторная работа № 12

Протокол передачи файлов FTP.

Краткие теоретические сведения

1. Протокол FTP.

FTP является стандартом Internet для передачи файлов. Необходимо различать передачу файлов, именно то, что предоставляет FTP, и доступ к файлам, что предоставляется такими приложениями как NFS. Передача файлов заключается в копировании целого файла из одной системы в другую. Чтобы использовать FTP, необходимо иметь учетную запись на сервере, или можно воспользоваться так называемым анонимным FTP (anonymous FTP).

Как и Telnet, FTP был создан для того, чтобы работать между узлами работающими под управлением различных операционных систем, использующих различные структуры файлов и, возможно, различные наборы символов. Telnet, однако, обеспечивает связь между разнородными системами, заставляя каждого участника соединения работать с одним и тем же стандартом: NVT, использующий 7-битный ASCII. FTP сглаживает различия между системами с использованием другого подхода. FTP поддерживает ограниченное количество типов файлов (ASCII, двоичный и так далее) и структуру файлов (поток байтов или ориентированный на запись).

RFC 959 [Postel and Reynolds 1985] является официальной спецификацией FTP. Этот RFC описывает историю и развитие передачи файлов в течение времени.

FTP отличается от других приложений тем, что он использует два TCP соединения для передачи файла.

1.Управляющее соединение устанавливается как обычное соединение клиент-сервер. Сервер осуществляет пассивное открытие на заранее известный порт FTP (21) и ожидает запроса на соединение от клиента. Клиент осуществляет активное открытие на TCP порт 21, чтобы установить управляющее соединение. Управляющее соединение существует все время, пока клиент общается с сервером. Это соединение используется для передачи команд от клиента к серверу и для передачи откликов от сервера. Тип IP сервиса для управляющего соединения устанавливается для получения «минимальной задержки».

2.Соединение данных открывается каждый раз, когда осуществляется передача файла между клиентом и сервером (Оно также открывается и в другие моменты). Тип сервиса IP для соединения данных должен быть «максимальная пропускная способность», так как это соединение используется для передачи файлов.

На рисунке 1 показано общение клиента и сервера по двум соединениям.

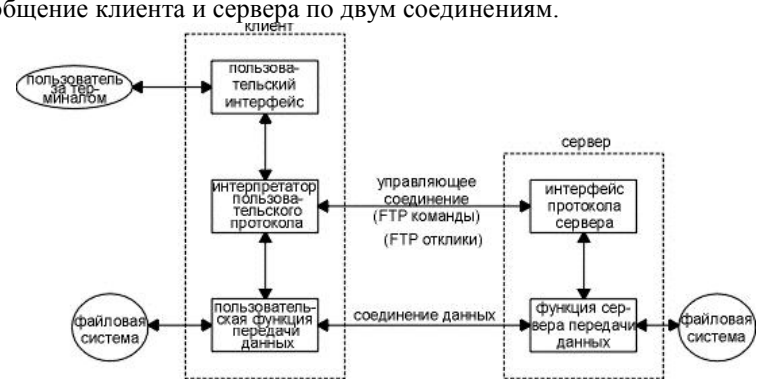

Рисунок 1. Процессы, участвующие в передаче файлов.

Из рисунка видно, что пользователь обычно не видит команды и отклики, которые передаются по управляющему соединению. Эти детали оставлены двум интерпретаторам протокола. Квадратик, помеченный как «пользовательский интерфейс», это именно то, что видит пользователь (полноэкранный интерфейс, основанный на меню, командные строки и так далее). Интерфейс конвертирует ввод пользователя в FTP команды, которые отправляются по управляющему соединению. Отклики, возвращаемые сервером по управляющему соединению, конвертируются в формат, удобный для пользователя.

Алгоритм работы протокола FTP состоит в следующем:

1. Сервер FTP использует в качестве управляющего соединение на TCP порт 21, который всегда находится в состоянии ожидания соединения со стороны пользователя FTP.

2. После того как устанавливается управляющее соединение модуля «Интерпретатор протокола пользователя» с модулем сервера – «Интерпретатор протокола сервера», пользователь (клиент) может отправлять на сервер команды. FTP-команды определяют параметры соединения передачи данных: роль участников соединения (активный или пассивный), порт соединения (как для модуля «Программа передачи данных пользователя», так и для модуля «Программа передачи данных сервера»), тип передачи, тип передаваемых данных, структуру данных и управляющие

директивы, обозначающие действия, которые пользователь хочет совершить (например, сохранить, считать, добавить или удалить данные или файл).

3. После того как согласованы все параметры канала передачи данных, один из участников соединения, который является пассивным (например, «Программа передачи данных пользователя»), становится в режим ожидания открытия соединения на заданный для передачи данных порт. После этого активный модуль (например, «Программа передачи данных сервера») открывает соединение и начинает передачу данных.

4. После окончания передачи данных, соединение между «Программой передачи данных сервера» и «Программой передачи данных пользователя» закрывается, но управляющее соединение «Интерпретатора протокола сервера» и «Интерпретатора протокола пользователя» остается открытым. Пользователь, не закрывая сессии FTP, может еще раз открыть канал передачи данных.

Возможна ситуация, когда данные могут передаваться на третью машину. В этом случае пользователь организует канал управления с двумя серверами и прямой канал данных между ними. Команды управления идут через пользователя, а данные – напрямую между серверами. Канал управления должен быть открыт при передаче данных между машинами. Иначе, в случае его закрытия передача данных прекращается. Соединение с двумя серверами показано на рисунке 2.

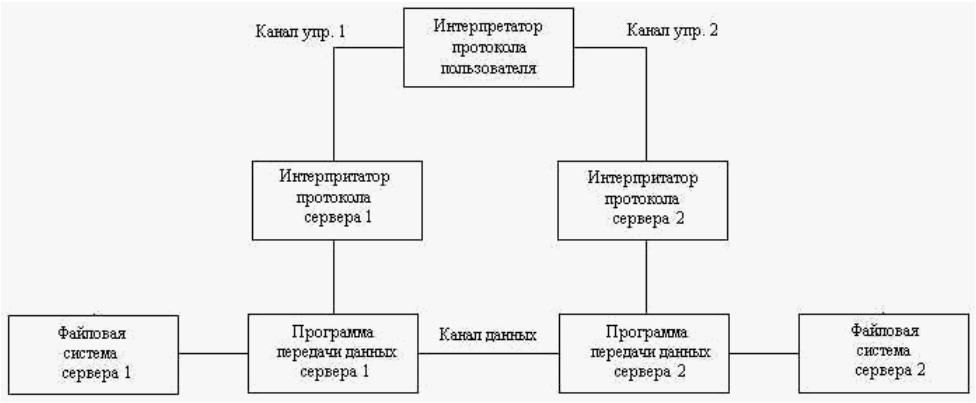

Рисунок 2.

Алгоритм работы при соединение двух FTP-серверов, ни один из которых не расположен на локальном узле пользователя:

1. Модуль «Интерпретатор протокола пользователя» указал модулю сервера «Интерпретатор протокола сервера 1» работать в пассивном режиме, после чего модуль «Интерпретатор протокола сервера 1» отправил пользователю адрес и номер порта (N), который он будет слушать.

2. Модуль «Интерпретатор протокола пользователя» назначил модуль сервера 2 «Интерпретатор протокола сервера 2» в качестве активного участника соединения и указал ему передавать данные на узел «Интерпретатор протокола сервера 1» на порт (N).

3. «Интерпретатор протокола пользователя» подал «Интерпретатору протокола сервера 1» команду «сохранить поступившие данные в таком-то файле», а «Интерпретатор протокола сервера 2» – «передать содержимое такого-то файла».

4. Между модулями «Интерпретатор протокола сервера 1» и «Интерпретатор протокола сервера 2» образуется поток данных, который управляется клиентским узлом.

Ниже приведена схема организации передачи данных между двумя серверами FTP, соответствующая рисунку 2. Здесь использованы следующие обозначения: User PI – интерпретатор протокола пользователя; Server1(2) – интерпретатор протокола сервера1 (сервера2).

User PI (U) ⇔ Server1 (S1)	User PI (U) ⇔ Server2 (S2)
U ⇒ S1: Connect U ⇒ S1: PASV U ⇐ S1: 227 Entering Passive Mode. A1, A2, A3, A4, a1, a2	U ⇒ S2 Connect U ⇒ S2: PORT A1, A2, A3, A4, a1, a2
	U ⇐ S2: 200 Okay
U ⇒ S1: STOR ...	U ⇒ S2: RETR ...
S1 ⇒ S2: Connect ...	

Основу передачи данных FTP составляет механизм установления соединения между соответствующими портами и выбора параметров передачи. Каждый участник FTP-соединения должен поддерживать порт передачи данных по

умолчанию. По умолчанию «Программа передачи данных пользователя» использует тот же порт, что и для передачи команд (обозначим его «U»), а «Программа передачи данных сервера» использует порт L-1, где «L» – управляющий порт. Однако, участниками соединения используются порты передачи данных, выбранные для них «Интерпретатором протокола пользователя», поскольку из управляющих процессов участвующих в соединении, только «Интерпретатор протокола пользователя» может изменить порты передачи данных как у «Программы передачи данных пользователя», так и у «Программы передачи данных сервера».

Пассивная сторона соединения должна до того, как будет подана команда «начать передачу», «слушать» свой порт передачи данных. Активная сторона, подающая команду к началу передачи данных, определяет направление перемещения данных.

2. Представление данных.

Протокол FTP предоставляет различные способы управления передачей и хранения файлов. Необходимо сделать выбор по четырем пунктам.

2.1. Тип файла.

(a)

ASCII файлы.

(По умолчанию) Текстовый файл передается по соединению данных как NVT ASCII. При этом требуется, чтобы отправитель конвертировал локальный текстовый файл в NVT ASCII, а получатель конвертировал NVT ASCII в текстовый файл. Конец каждой строки передается в виде NVT ASCII символа возврата каретки, после чего следует перевод строки. Это означает, что получатель должен просматривать каждый байт в поисках пары символов CR, LF.

(b)

EBCDIC файлы.

Альтернативный способ передачи текстовых файлов, когда на обоих концах системы EBCDIC.

(c)

Двоичные или бинарные файлы. (Image.)

Данные передаются как непрерывный поток битов.

(d)

Локальный тип файлов.

Способ передачи бинарных файлов между узлами, которые имеют различный размер байта. Количество битов в байте определяется отправителем. Для систем, которые используют 8-битные байты, локальный тип файла с размером байта равным 8 эквивалентен бинарному типу файла.

2.2. Управление форматом. Применяется только для ASCII и EBCDIC файлов.

(a)

Nonprint. (По умолчанию)

Файл не содержит информацию вертикального формата.

(b)

Telnet format control.

Файл содержит управляющие символы вертикального формата Telnet, которые интерпретируются принтером.

(c)

Fortran carriage control.

Первый символ каждой строки – это Fortran символ управления формата.

2.3. Структура.

(a)

Структура файла.

(По умолчанию) Файл воспринимается в виде непрерывного потока байтов. Файл не имеет внутренней структуры.

(b)

Структура записи.

Эта структура используется только в случае текстовых файлов (ASCII или EBCDIC).

(c)

Структура страницы.

Каждая страница передается с номером страницы, что позволяет получателю хранить страницы в случайном порядке. Предоставляется операционной системой TOPS-20. (Требование к узлам Host Requirements RFC не рекомендует использовать эту структуру.)

2.4. Режим передачи. Указывает на то, как файл передается по соединению данных.

(a)

Режим потока.

(По умолчанию) Файл передается как поток байтов. Для файловой структуры конец файла указывает на то, что отправитель закрывает соединение данных. Для структуры записи специальная 2-байтовая последовательность обозначает конец записи и конец файла.

(b)
Режим блоков.
Файл передается как последовательность блоков, перед каждым из них стоит один или несколько байт заголовков.
(c)
Сжатый режим.
Простое кодирование неоднократно встречающихся повторяющихся байт. В текстовых файлах обычно сжимаются пустые строки или строки из пробелов, а в бинарных строки из нулевых байт. (Этот режим поддерживается редко. Существуют более оптимальные способы сжатия файлов для FTP).

Если посчитать количество комбинаций из приведенных вариантов, то получится 72 способа передачи и хранения файла. Однако, можно игнорировать многие из этих опций, потому что они не поддерживаются в большинстве реализаций.

Самые распространенные Unix реализации FTP клиента и сервера предоставляют следующий выбор:

- Тип: ASCII или двоичный.
- Управление форматом: только nonprint.
- Структура: только файловая структура.
- Режим передачи: только потоковый режим.

3. Команды и отклики FTP.

Команды и отклики передаются по управляющему соединению между клиентом и сервером в формате NVT ASCII. В конце каждой строки команды или отклика присутствует пара CR, LF.

Единственные команды Telnet (начинающиеся с IAC), которые могут быть отправлены клиентом серверу – это команда прерывания процесса (<IAC, IP>) и Telnet сигнал синхронизации (<IAC, DM> в режиме срочности). Эти две команды Telnet используются для прекращения передачи файла или для того, чтобы отправить серверу запрос в процессе передачи.

Команды состоят из 3 или 4 байт, а именно из заглавных ASCII символов, некоторые с необязательными аргументами. Клиент может отправить серверу более чем 30 различных FTP команд. На рисунке 3 показаны некоторые наиболее широко используемые команды.

Команда	Описание
ABOR	прервать предыдущую команду FTP и любую передачу данных
LIST список файлов	список файлов или директорий
PASS пароль	пароль на сервере
PORT n1,n2,n3,n4,n5,n6	IP адрес клиента (n1.n2.n3.n4) и порт (n5 x 256 + n6)
QUIT	закрыть сеанс на сервере
RETR имя файла	получить (get) файл
STOR имя файла	положить (put) файл
SYST	сервер возвращает тип системы
TYPE тип	указать тип файла: A для ASCII, I для двоичного
USER имя пользователя	имя пользователя на сервере

Рисунок 3. Распространенные FTP команды.

В примерах некоторые команды полностью совпадают с тем, что вводит пользователь в качестве FTP команд. В этом случае они передаются по управляющему соединению, однако некоторые вводимые пользователем команды генерируют несколько FTP команд, которые, в свою очередь, передаются по управляющему соединению.

FTP отклики

Отклики состоят из 3-циферных значений в формате ASCII, и необязательных сообщений, которые следуют за числами. Подобное представление откликов объясняется тем, что программному обеспечению необходимо посмотреть только цифровые значения, чтобы понять, что ответил процесс, а дополнительную строку может прочитать человек. Поэтому пользователю достаточно просто прочитать сообщение (причем нет необходимости запоминать все цифровые коды откликов).

Каждая из трех цифр в коде отклика имеет собственный смысл. На рисунке 4 показаны значения первых и вторых цифр в коде отклика.

Отклик	Описание
1yz	Положительный предварительный отклик. Действие началось, однако необходимо дождаться еще одного отклика перед отправкой следующей команды.
2yz	Положительный отклик о завершении. Может быть отправлена новая команда.
3yz	Положительный промежуточный отклик. Команда принята, однако необходимо отправить еще одну команду.
4yz	Временный отрицательный отклик о завершении. Требуемое действие не произошло, однако ошибка временная, поэтому команду необходимо повторить позже.
5yz	Постоянный отрицательный отклик о завершении. Команда не была воспринята и повторять ее не стоит.
x0z	Синтаксическая ошибка.
x1z	Информация.
x2z	Соединения. Отклики имеют отношение либо к управляющему, либо к соединению данных.
x3z	Аутентификация. Отклик имеет отношение к логированию или командам, связанным с учетной записью.
x4z	Не определено.
x5z	Состояние файловой системы.

Рисунок 4. Значения первой и второй цифр в 3-циферном коде отклика.

Третья цифра дает дополнительное объяснение сообщению об ошибке. Ниже приведены некоторые типичные отклики с возможными объясняющими строками.

- 125 Соединение данных уже открыто; начало передачи.
- 200 Команда исполнена.
- 214 Сообщение о помощи (для пользователя).
- 331 Имя пользователя принято, требуется пароль.
- 425 Невозможно открыть соединение данных.
- 452 Ошибка записи файла.
- 500 Синтаксическая ошибка (неизвестная команда).
- 501 Синтаксическая ошибка (неверные аргументы).
- 502 Нереализованный тип MODE.

Обычно каждая FTP команда генерируют отклик в одну строку. Например, команда QUIT сгенерирует следующий отклик:

```
221 Goodbye.
```

Если необходим отклик в несколько строк, первая строка содержит дефис вместо пробела после 3-циферного кода отклика, а последняя строка содержит тот же самый 3-циферный код отклика, за которым следует пробел. Например, команда HELP сгенерирует следующий отклик:

```
214- The following commands are recognized (* =>'s unimplemented).
   USER    PORT    STOR    MSAM*     RNTO    NLST    MKD      CDUP
   PASS    PASV    APPE    MRSQ*     ABOR    SITE    XMKD     XCUP
   ACCT*   TYPE    MLFL*   MRCP*     DELE    SYST    RMD      STOU
   SMNT*   STRU    MAIL*   ALLO      CWD     STAT    XRMD     SIZE
   REIN*   MODE    MSND*   REST      XCWD    HELP    PWD      MDTM
   QUIT    RETR    MSOM*   RNFR      LIST    NOOP    XPWD
214 Direct comments to ftp-bugs@bsdi.tuc.noao.edu.
```

4. Управление соединением.

Использовать соединение данных можно тремя способами.

1.Отправка файлов от клиента к серверу.

2.Отправка файлов от сервера к клиенту.

3.Отправка списка файлов или директорий от сервера к клиенту.

FTP сервер посылает список файлов по соединению данных, вместо того чтобы посылать многострочные отклики по управляющему соединению. При этом появляется возможность избежать любых ограничений в строках, накладывающихся на размер списка директории, и позволяет просто сохранить список директории в файле, вместо того чтобы выдавать список на терминал.

Управляющее соединение остается в активизированном состоянии все время, пока установлено соединение клиент-сервер, однако соединение данных может выключаться и включаться по необходимости. Как выбираются номера портов для соединения данных, и кто осуществляет активное открытие, а кто пассивное открытие?

Во-первых, как было сказано ранее, распространенный режим передачи (в случае Unix это единственный режим передачи) – это потоковый режим. В этом режиме конец файла обозначает закрытие соединения данных. Из этого следует, что для передачи каждого файла или списка директории требуется новое соединение данных. Обычная процедура выглядит следующим образом:

1. Создание соединения данных осуществляется клиентом, потому что именно клиент выдает команды, которые требуют передать данные (получить файл, передать файл или список директории).
2. Клиент обычно выбирает динамически назначаемый номер порта на узле клиента для своего конца соединения данных. Клиент осуществляет пассивное открытие с этого порта.
3. Клиент посылает этот номер порта на сервер по управляющему соединению с использованием команды PORT.
4. Сервер принимает номер порта с управляющего соединения и осуществляет активное открытие на этот порт узла клиента. Сервер всегда использует порт 20 для соединения данных.

На рисунке 5 показано состояние соединений, пока осуществляется шаг номер 3. Предполагается, что динамически назначаемый порт клиента для управляющего соединения имеет номер 1173, а динамически назначаемый порт клиента для соединения данных имеет номер 1174. Команда, посылаемая клиентом – PORT, а ее аргументы это шесть десятичных цифр в формате ASCII, разделенные запятыми. Четыре первых числа – это IP адрес клиента, на который сервер должен осуществить активное открытие (140.252.13.34 в данном примере), а следующие два – это 16-битный номер порта. Так как 16-битный номер порта формируется из двух цифр, его значение в этом примере будет 4 x 256 + 150 = 1174.

На рисунке 6 показано состояние соединений, когда сервер осуществляет активное открытие на конец клиента соединения данных. Конечная точка сервера – это порт 20.

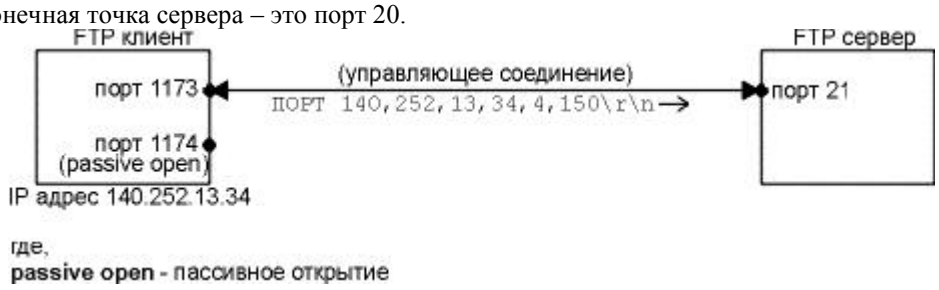

Рисунок 5. Команда PORT, передаваемая по управляющему соединению FTP.

Рисунок 6. FTP сервер осуществляет активное открытие соединения данных.

Сервер всегда осуществляет активное открытие соединения данных. Обычно сервер также осуществляет активное закрытие соединения данных, за исключением тех случаев, когда клиент отправляет файл на сервер в потоковом режиме, который требует, чтобы клиент закрыл соединение (что делается с помощью уведомления сервера о конце файла).

Если клиент не выдает команду PORT, сервер осуществляет активное открытие на тот же самый номер порта, который использовался клиентом для управляющего соединения (1173 в данном примере). В этом случае все работает корректно, так как номера порта сервера для двух соединений различны: один 20, другой 21. Тем не менее современные реализации не поступают таким образом.

5. Активный и пассивный режимы работы FTP сервера

5.1. Активный режим.
5.1.1. Клиент устанавливает соединение на 21 порт сервера с порта N (N>1024).
5.1.2. Сервер посылает ответ на порт N (N>1024) клиента.
5.1.3 Сервер устанавливает соединение для передачи данных с порта 20 на порт клиента N+1

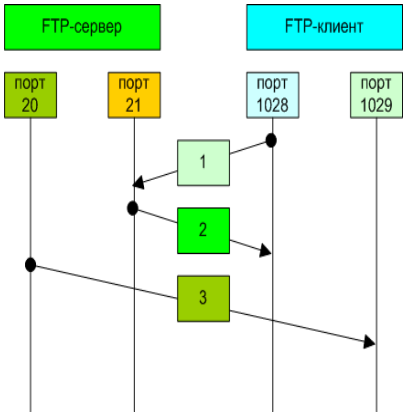

Рисунок 7. Активный режим.

5.2. Пассивный режим.

5.2.1. Клиент устанавливает соединение на 21 порт сервера с порта N (N>1024) и просит сервер перейти в пассивный режим.

5.2.2. Сервер посылает ответ и сообщает номер порта для канала данных P (P>1024) на порт N (N>1024) клиента.

5.2.3. Клиент устанавливает соединение для передачи данных с порта N+1 на порт сервера P (P>1024).

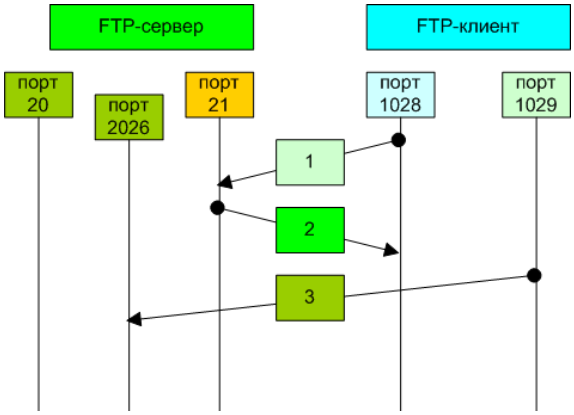

Рисунок 8. Пассивный режим

Реализация FTP (vsftpd) сервера на базе ОС FreeBSD UNIX

1. Загружаем дистрибутив `vsftpd-2.1.0.tar.gz` в каталог `/usr/local/src`.
2. Распаковываем архив:

```
# tar -xvf vsftpd-2.1.0.tar.gz
```

3. Переходим в созданный каталог:

```
# cd vsftpd-2.1.0
```

4. Собираем приложение:

```
# make
```

5. При необходимости добавляем пользователя `nobody`:

```
# adduser
Username: nobody
Full name: nobody
Uid (Leave empty for default): 65534
Login group [nobody]:
Login group is nobody. Invite nobody into other groups? []:
Login class [default]:
Shell (sh csh tcsh nologin) [sh]: nologin
Home directory [/home/nobody]: /nonexistent
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password:
Enter password again:
Lock out the account after creation? [no]:
```

```
Username   : nobody
Password   : *****
Full Name  : nobody
Uid        : 65534
Class      :
Groups     : nobody
Home       : /nonexistent
Shell      : /usr/sbin/nologin
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (nobody) to the user database.
Add another user? (yes/no): no
Goodbye!
```

6. Создаем пустой каталог /usr/share/empty/

```
# mkdir /usr/share/empty/
```

7. Для анонимного доступа к FTP серверу необходимо создать пользователя `ftp` с домашним каталогом, владельцем которого не является пользователь `ftp`, также пользователь `ftp` не должен иметь права записи в этот каталог.

```
# mkdir /var/ftp/
# adduser
Username: ftp
Full name: ftp
Uid (Leave empty for default):
Login group [ftp]:
Login group is ftp. Invite ftp into other groups? []:
Login class [default]:
Shell (sh csh tcsh nologin) [sh]: nologin
Home directory [/home/ftp]: /var/ftp
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password:
Enter password again:
Lock out the account after creation? [no]:
Username   : ftp
Password   : *****
Full Name  : ftp
Uid        : 1002
Class      :
Groups     : ftp
Home       : /var/ftp
Shell      : /usr/sbin/nologin
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (ftp) to the user database.
Add another user? (yes/no): no
Goodbye!
# chown root /var/ftp
# chgrp wheel /var/ftp/
# chmod og-w /var/ftp
```

8. Устанавливаем приложение

```
# make install
```

9. Копируем конфигурационный файл:

```
# cp vsftpd.conf /etc
```

10. Редактируем конфигурационный файл `/etc/vsftpd.conf`

11. Запускаем сервис в режиме `standalone`

```
# /usr/local/sbin/vsftpd &
```

Задание на работу

1. Установить и настроить сервер FTP на базе vsftpd (proftpd, ftpd).

2. Обеспечить возможность подключения согласно варианту.

2.1. Анонимного пользователя. В домашнем каталоге анонимного пользователя создать 2 каталога: `pub` и `incoming`. Каталог `pub` доступен только для чтения, каталог `incoming` доступен для чтения и записи.

2.2. Авторизованного пользователя с возможностью доступа ко всему дереву каталогов.

2.3. Авторизованного пользователя с возможностью доступа только к содержимому личного каталога.

3. Проверить работу сервера FTP в активном и пассивном режимах.

Варианты заданий.

№ варианта	Пользователи	
	Имя	**Доступ**
1	mercury	только к содержимому личного каталога
	venus	только к содержимому личного каталога
	earth	ко всему дереву каталогов
	saturn	ко всему дереву каталогов
	jupiter	только к содержимому личного каталога
2	tiger	ко всему дереву каталогов
	lion	только к содержимому личного каталога
	lynx	только к содержимому личного каталога
	leopard	ко всему дереву каталогов
	jaguar	ко всему дереву каталогов
3	alpha	ко всему дереву каталогов
	beta	ко всему дереву каталогов
	gamma	ко всему дереву каталогов
	delta	только к содержимому личного каталога
	omega	ко всему дереву каталогов
4	mercury	только к содержимому личного каталога
	venus	ко всему дереву каталогов
	earth	только к содержимому личного каталога
	saturn	ко всему дереву каталогов
	jupiter	только к содержимому личного каталога
5	tiger	только к содержимому личного каталога
	lion	ко всему дереву каталогов
	lynx	ко всему дереву каталогов
	leopard	ко всему дереву каталогов
	jaguar	только к содержимому личного каталога
6	rose	ко всему дереву каталогов
	tulip	только к содержимому личного каталога
	narcissus	ко всему дереву каталогов
	aster	ко всему дереву каталогов
	peony	только к содержимому личного каталога
7	alpha	только к содержимому личного каталога
	beta	ко всему дереву каталогов
	gamma	только к содержимому личного каталога
	delta	ко всему дереву каталогов
	omega	ко всему дереву каталогов
8	mercury	ко всему дереву каталогов
	venus	только к содержимому личного каталога
	earth	только к содержимому личного каталога
	saturn	только к содержимому личного каталога
	jupiter	ко всему дереву каталогов
9	tiger	ко всему дереву каталогов
	lion	только к содержимому личного каталога
	lynx	только к содержимому личного каталога
	leopard	ко всему дереву каталогов
	jaguar	ко всему дереву каталогов
10	rose	только к содержимому личного каталога
	tulip	ко всему дереву каталогов
	narcissus	только к содержимому личного каталога
	aster	ко всему дереву каталогов
	peony	только к содержимому личного каталога

№ варианта	Пользователи	
	Имя	Доступ
11	alpha	ко всему дереву каталогов
	beta	только к содержимому личного каталога
	gamma	только к содержимому личного каталога
	delta	ко всему дереву каталогов
	omega	ко всему дереву каталогов
12	mercury	ко всему дереву каталогов
	venus	ко всему дереву каталогов
	earth	только к содержимому личного каталога
	saturn	только к содержимому личного каталога
	jupiter	только к содержимому личного каталога
13	tiger	только к содержимому личного каталога
	lion	только к содержимому личного каталога
	lynx	ко всему дереву каталогов
	leopard	ко всему дереву каталогов
	jaguar	только к содержимому личного каталога
14	rose	только к содержимому личного каталога
	tulip	ко всему дереву каталогов
	narcissus	только к содержимому личного каталога
	aster	ко всему дереву каталогов
	peony	ко всему дереву каталогов
15	alpha	только к содержимому личного каталога
	beta	ко всему дереву каталогов
	gamma	только к содержимому личного каталога
	delta	ко всему дереву каталогов
	omega	только к содержимому личного каталога

Литература

1. Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е изд. – СПб.: Питер, 2010. – 944 с.

2. Стивенс У.Р. Протоколы TCP/IP. Практическое руководство/ Пер. с англ. и коммент. А.Ю. Глебовского. – СПб.: «Невский диалект» – «БХВ–Петербург», 2003. – 672 с.: ил.

3. Семенов Ю.А. Телекоммуникационные технологии v3.0, 17 августа 2007 года.