Консольный сервер

Gregory Bond <gnb@itga.com.au >
Андрей Захватов
Дмитрий Морозовский
Издание: 45050

FreeBSD это зарегистрированная торговая марка FreeBSD Foundation.

Cisco, Catalyst, и IOS это зарегистрированные торговые марки Cisco Systems, Inc. и/или ее компаньонов в Соединенных Штатах и других странах.

Intel, Celeron, EtherExpress, i386, i486, Itanium, Pentium и Xeon это торговые марки или зарегистрированные торговые марки Intel Corporation или ее дочерних компаний в Соединенных Штатах и других странах.

Lantronix и EasyIO это торговые марки Lantronix Corporation.

Microsoft, FrontPage, IntelliMouse, MS-DOS, Outlook, Windows, Windows Media и Windows NT это или зарегистрированные торговые марки или торговые марки Microsoft Corporation в Соединенных Штатах и/или других странах.

Motif, OSF/1 и UNIX это зарегистрированные торговые марки, а IT DialTone и The Open Group это торговые марки Open Group в Соединенных Штатах и других странах.

Sun, Sun Microsystems, Java, Java Virtual Machine, JDK, JRE, JSP, JVM, Netra, Solaris, StarOffice, SunOS это торговые марки или зарегистрированные торговые марки Sun Microsystems, Inc. в Соединенных Штатах и других странах.

Многие из обозначений, используемые производителями и продавцами для обозначения своих продуктов, заявляются в качестве торговых марок. Когда такие обозначения появляются в этом документе, и Проекту FreeBSD известно о торговой марке, к обозначению добавляется знак «TM» или «(R)».

2014-06-13 taras.

Аннотация

В этом документе описывается, как можно использовать FreeBSD, аппаратное и программное обеспечение, работающее с FreeBSD, для построения «консольного сервера». Консольным сервером обычно называют машину, которую можно использовать для отслеживания консолей многих других машин вместо использования многих последовательных терминалов.

Содержание

1. Проблема 2
2. Возможные решения 3
3. Наше решение 5
4. Настройка сервера 7
5. Подключение кабелей 13
6. Про системы Sun и сигнал Break 19
7. Использование последовательной консоли в FreeBSD 20
8. Соображения безопасности 21
9. Различные версии Conserver 22
10. Ссылки 22
11. Справочные страницы 23
Предметный указатель 23

1. Проблема

У вас есть компьютерный зал с множеством UNIX®-серверов и коммуникационным оборудованием. Каждой этой машине необходима последовательная консоль. Однако последовательные терминалы трудно найти и они достаточно дороги (особенно по сравнению с ПК, обладающими гораздо большими возможностями). И всё это в компьютерном зале занимает много места.

Вам необходим доступ к консоли, потому что когда что-то не работает, сообщения об ошибках направляются туда. И некоторые работы выполняются с консоли (к примеру, при возникновении проблем с загрузкой или при установке или обновлении ОС). Некоторые UNIX®-системы позволяют переходить с консоли в режим монитора ПЗУ, который иногда является единственным способом заставить функционировать неработающую машину. Часто это осуществляется посылкой `LINE BREAK` на последовательный порт консоли.

Если мы собираемся поработать с консолями, то было бы великолепно осуществить ещё несколько вещей:

- Удалённый доступ. Даже в одном помещении было бы неплохо иметь доступ ко всем консолям с вашего рабочего места без необходимости передвигаться по компьютерному залу. А иногда машины расположены где-то далеко, может быть, даже в другой стране.

- Протоколирование. Если что-то идёт не так, вам не помешает возможность посмотреть предыдущую выдачу на консоль, чтобы понять происходящее. Обычные консольные экраны дают вам последние 25 строк. Чем таких строк будет больше, тем лучше.

- Независимость от сети. Решение должно функционировать даже при неработающей сети. В конце концов, больше всего консоли вам нужны именно при отключении сети! Ещё лучше добиться независимости от сети с возможностью удалённого доступа.

- Отсутствие одной точки, критичной для работы. Консольная система, которая приводит к неработоспособности всех машин при сбое, не нужна. Это особенно важно при использовании с UNIX®-хостами Sun, так как они будут воспринимать выключение терминала как `BREAK` и будут переходить в режим ROM-монитора.

- Интерфейс с пейджинговым или другим подобным устройством подачи предупреждающих сообщений.

- Возможность удалённого выключения и повторного включения машин.

- Не *слишком* высокая стоимость. Ещё лучше, если система будет бесплатной!

2. Возможные решения

Если для ваших серверов вы используете ПК-оборудование, то одним из возможных решений является «KVM-переключатель». Такой KVM-переключатель позволяет использовать одну клавиатуру, видеомонитор и мышь с несколькими системными блоками. Это снижает остроту проблемы с физическим пространством, однако работает только с ПК-оборудованием (а не с любыми типами имеющихся коммуникационных устройств), и не обеспечивает доступ вне компьютерного зала. При этом даже отсутствует прокрутка истории или протоколирование, и вам нужно организовывать оповещение каким-то другим способом. Большим минусом является то, что это не работает с устройствами только с последовательным интерфейсом, таким, как коммуникационное оборудование. Это означает, что даже в зале, заполненном ПК-серверами, вам может оказаться нужным доступ к последовательной консоли.

Примечание

На самом деле, Doug Schache указал, что вы *можете* найти KVM-переключатели с поддержкой последовательных консолей или совместимые как с Sun, так и с ПК, но они дороги. Посмотрите, например, на сайте Avocent.)

Вы можете попытаться обойтись без консольного терминала, однако когда творятся странные вещи, вам *действительно* нужно видеть происходящее на консоли. И вам нужно использовать консоль для загрузки и выполнения таких действий, как установка или обновление ОС.

Вы можете попытаться выделить один консольный терминал и переключаться при необходимости между серверами, либо при помощи последовательного переключателя, либо просто подключая его к нужной машине. Последовательные переключатели также трудно найти и они не дёшевы, к тому же могут иметь проблемы с посылкой сигнала `BREAK` при переключении. И (если ваш компьютерный зал похож на наш) у вас никогда не будет совпадать комбинация кабелей для подключения к нужной машине, и даже если все кабели на месте, вы никогда с точностью не будете знать, какая именно комбинация окончаний `DTE/DCE` ведёт к конкретному оборудованию. Так что первые 10 минут вы потратите на возню с исходящими и входящими окончаниями, тогда как сервер не работает, а пользователи уже кричат. Конечно, это не удовлетворяет требованиям удалённого подключения к системе и протоколирования. И неизбежно консоль окажется не подключенной к нужной машине, так что вы потеряете все консольные сообщения, могущие рассказать вам о происходящем.

Одним из распространённых решений является использование аппаратного терминального сервера. Обычно последовательные порты подключены к консолям различных машин, и настроены на «обратный telnet-доступ». Это значит, что пользователь может подключиться по протоколу telnet к определённому IP/порту и оказаться подключенным к соответствующей консоли. Это может быть очень эффективным с точки зрения стоимости, так как подходящие старые терминальные серверы можно найти по достаточно низкой цене (полагаем, что пары таких у вас ещё нет). И, конечно, они доступны по сети, что подходит для организации управления из сети. Однако у них есть один большой недостаток: если сеть не работает, то вы *теряете* доступ к любой консоли, даже если находитесь прямо перед машиной. (Это может быть несколько не так, если у вас есть подходящий терминал, подключенный к одному из портов терминального сервера, с которого можно выполнять подключения, но программное обеспечение терминального сервера может этого и не поддерживать.) К тому же при этом отсутствует протоколирование и повтор кон-

сольных сообщений. Однако приложив некоторые усилия и используя определённое программное обеспечение, такое, как conserver (описано далее), эту систему можно заставить хорошо работать.

Ещё один подход, предложенный Броном Гондваной (Bron Gondwana), похож на описанное выше решение. Если ваши серверы имеют несколько последовательных портов, вы можете подключить каждый свободный последовательный порт к консольному порту «ближайшего» сервера, создав тем самым кольцо консольных соединений (в некотором порядке). Это может достаточно хорошо работать вместе с программным обеспечением conserver, однако, с другой стороны, может несколько запутывать (в смысле необходимости запоминания, какой порт к какой консоли подключен). И этого не получится, если вам нужно использовать последовательные порты в других целях (таких, как подключение модемов) либо на ваших машинах нет свободных портов.

Либо, если ваш бюджет превышает необходимость в хакерских решениях, вы можете приобрести одно из готовых решений. Они различаются по стоимости и своим возможностям. Посмотрите, к примеру, Lightwave, Perle, Avocent или Black Box. Эти решения могут оказаться достаточно дорогими - обычно от 100 до 400 долл. США за порт.

3. Наше решение

В свете требований выше мы выбрали решение на основе выделенного ПК под управлением UNIX® с многопортовым последовательным адаптером и определённым программным обеспечением, предназначенным для работы с последовательными консолями.

Оно состоит из следующих элементов:

- Подержанный ПК. Мы использовали Pentium® 166 с шиной PCI, 2-гигабайтным жёстким диском и 64 мегабайтами ОЗУ. Это превышает требования выполняемой задачи, более чем достаточным будет P-100, 500 Мб, 32 Мб.

- UNIX®-система для ПК. Мы использовали FreeBSD 4.3, так как в нашем офисе она использовалась и для других задач.

- Многопортовый последовательный адаптер. Мы выбрали 8-портовый адаптер EasyIO™ PCI компании Stallion Technologies. Это стоило нам порядка $AUD740, меньше чем $100 за порт, заплаченных Harris Technologies (у них есть много всего, но это не обязательно самое дешёвое место - поищите поблизости, вы можете найти место гораздо дешевле). Адаптер имеет сзади большой разъём DB80 и подключаемый кабель, имеющий блок из 8 гнёзд RJ-45. (Мы выбрали вариант с RJ-45, так как наша кабельная система полностью построена на RJ-45. Это позволяет нам переключать соединения от нужного блока к консольному серверу без

дополнительных кабелей.) Это единственная вещь, которую нам пришлось приобрести, чтобы всё заработало.

Примечание

В России, возможно, будет проще найти карты Omega PCI компании КБ "Кроникс" / Cronyx Engineering, менее $40 за порт. [*прим. перев.*].

- Мы построили два сервера, по одному для каждого машинного зала, с 8 портами в одном и 16 портами (двумя адаптерами EasyIO™ PCI) в другом. Если бы нам нужно было более 16 портов, то по стоимости более эффективным было бы использование других адаптеров Stallion. Теоретически мы можем поддерживать 128 портов на каждом сервере (при помощи 2 хост-адаптеров EasyConnect 8/64 и 8 16-портовых модулей RJ-45) общей стоимостью $AUD12,000.

- Модем для удалённого доступа к хосту консольного сервере при отсутствии сети. Мы ещё этого не делали, так как компьютерный зал находится рядом, но когда мы перенесем сервер в Сидней, мы добавим модем. Идея заключается в том, что при отсутствии сети вы можете позвонить, подключиться к серверной машине и запустить консольную программу локально. В целях безопасности мы, скорее всего, оставим модем выключенным, и попросим тамошних жителей Сиднея нажать хорошо видную кнопку при необходимости.

- Программа под названием conserver. Она выполняет всё, что требуется для включения удалённого доступа к консолям, обеспечивает повтор ввода, протоколирование и так далее. Она поставляется в виде двух блоков: сервер под именем conserver, работающий как даемон и подключающийся к последовательным портам, выполняющий ведение журналов и прочие действия, и клиентская программа под названием console, которая может подключаться к серверу, показывать консольные сообщения, посылать последовательности нажатий клавиш (и `BREAK`) и тому подобное.

Такая архитектура обеспечивает выполнение всех основных требований, кроме удалённого управления электропитанием:

- Удалённый доступ обеспечивается за счёт того, что клиентская программа console работает в сети.

- Протоколирование ведётся программой conserver.

- Если сеть не работает, то мы можем использовать консоль ПК для локального запуска клиента console. В случае географически удалённых мест мы можем доба-

вить модем для коммутируемого доступа к командной строке сервера для запуска клиента.

- Установив патчи на серверы Solaris™ (обратитесь к Раздел 6, «Про системы Sun и сигнал Break»), мы можем избежать неработоспособности всего компьютерного зала при сбое в консольном сервере на базе ПК (или при отключения электропитания, или по какой-то другой причине).

- У нас уже есть пейджинговое оповещение с другой установленной нами системы, однако на консольном сервере есть вся нужная информация журналов, так что при необходимости это может быть легко реализовано. И даже есть модем для звонка в пейджинговую компанию!

- На данный момент мы не поддерживаем удалённое управление электропитанием. Некоторые версии программы conserver это поддерживают, но это требует наличия специальных адаптеров, управляемых через последовательные соединения. У нас нет острой необходимости по удалённому выключению (у нас есть обслуживающий персонал в каждом удалённом офисе, который может это сделать под нашим руководством), так что это не большая проблема, и мы можем легко это добавить, если увидим в этом необходимость и получим соответствующее оборудование.

- Это решение было очень дешёвым. Общая стоимость 9-портового сервера составила $AUD750 за адаптеры ввода/вывода, так как мы использовали устаревший ПК и у нас имелось оборудование в виде специальных кабелей. Если бы мы всё покупали, то это обошлось бы всего лишь примерно в $AUD1500 за 8-портовый сервер.

4. Настройка сервера

4.1. Проверка драйвера Stallion

FreeBSD адекватно поддерживает адаптеры Stallion начиная с версии 4.4. Если ваша версия старше, вам потребуется обновить ее (это нужно сделать еще и для того, чтобы ваша система не была подвержена известным проблемам защиты). Обратитесь к описанию в файле `/usr/src/UPDATING` и Руководстве FreeBSD за подробной информацией об обновлении системы.

4.2. Конфигурация нового ядра

Драйвер Stallion не включён в используемое по умолчанию ядро `GENERIC`, так что вам нужно создать конфигурационный файл ядра с соответствующими записями. Обратитесь к справке по stl(4) и соответствующему разделу Руководства FreeBSD.

4.3. Создание устройств

Для адаптера Stallion вам нужно создать файлы устройств (которые по умолчанию не создаются). Во время выполнения описанной выше процедуры новая версия `/dev/MAKEDEV` с поддержкой Stallion будет создана утилитой mergemaster. Если у вас имеется адаптер Stallion с более чем 8 портами, то вам нужно отредактировать `/dev/MAKEDEV` и изменить определение `maxport` в районе строки 250. По умолчанию `MAKEDEV` создает файлы устройств для 8 портов, чтобы уменьшить размер каталога `/dev`.

Выполните примерно такую команду:

```
# cd /dev/ && sh MAKEDEV cuaE0
```

для создания устройств для исходящих звонков для первого адаптера Stallion. Для получения более полной информации обратитесь к разъяснениям в `MAKEDEV` и справочной странице stl(4).

4.4. Компиляция conserver

Примечание

Посмотрите раздел Раздел 9, «Различные версии Conserver» о версиях conserver; используемая мной версия находится в коллекции портов FreeBSD, однако, существуют и другие версии.

Имеется два способа установки conserver. Вы можете либо скомпилировать её из исходных текстов, либо воспользоваться механизмом портов FreeBSD.

4.4.1. Использование механизма портов

Использование портов является более ясным подходом, так как система пакетов может отслеживать установленное программное обеспечение и полностью удалять его, если оно не используется. Рекомендуем использовать порт comms/conserver-com. Перейдите в каталог этого порта и (работая как пользователь `root`) наберите:

```
# make DEFAULTHOST= consolehost  install
```

где *`consolehost`* является именем машины, на которой работает консольный сервер. Задание этого при компиляции бинарного файла избавляет от необходимости указывать его каждый раз при запуске программы либо поддерживать файлы `conserver.cf` для каждого хоста. Эта команда загрузит, установит патчи, сконфигурирует, скомпилирует и установит программу conserver.

После этого вы можете выполнить `make package` для создания бинарного пакета, который можно установить на остальных хостах FreeBSD по команде pkg_add(1). Для дополнительной гибкости вы можете создать две версии пакета: одну для машины с консольным сервером без параметра `DEFAULTHOST`, а вторую для всех остальных хостов с параметром `DEFAULTHOST`. Это значит, что клиентская программа консоли на машине с консольным сервером по умолчанию будет использовать `localhost`, что будет работать при отсутствии сервера имён, при сбоях в сети, а также позволит выполнять «доверяемые» (то есть беспарольные) подключения через IP-адрес localhost для пользователей, подключенных к машине с консольным сервером (либо с экрана консоли, либо с вспомогательного модема). Версия для остальных машин с аргументом `DEFAULTHOST` означает, что пользователи могут просто использовать клиента console без указания каждый раз имени хоста, и необходимости настраивать файл `conserver.cf` на каждой машине.

4.4.2. Из tar-архива исходных текстов

Если вы предпочитаете такой способ, то можете загрузить conserver и скомпилировать его самостоятельно. Вам может понадобиться сделать это, если вы хотите установить клиент консоли на не-FreeBSD системы. Мы используем клиент на наших машинах с Solaris™, и он без проблем взаимодействует с сервером на FreeBSD. Это позволяет каждому во всей компании (многие из которых имеют ПК без доступа к хосту с FreeBSD со своего рабочего места) обращаться к консольному серверу.

Загрузите файл с FTP-сайта conserver.com. Распакуйте его в любой каталог, затем сконфигурируйте, выполнив

```
% ./configure --with-master= consoleserver --with-port= 782
```

Параметр `--with-master` помогает избежать указания главного сервера каждый раз при удалённом запуске клиента (или постоянного обновления конфигурационных файлов на всех удалённых хостах). Параметр `--with-port` помогает избежать необходимости в обновлении файла `/etc/services` на всех машинах.

После этого наберите `make` и, работая как пользователь root, `make install`.

4.5. Конфигурация conserver

Программа conserver настраивается через файл с именем `conserver.cf`. Этот файл обычно находится в каталоге `/usr/local/etc` и он задокументирован на справочной странице conserver.cf(5).

Наш конфигурационный файл выглядит примерно так:

```
LOGDIR=/var/log/consoles
gallows:/dev/cuaE0:9600p:&:
roo:/dev/cuaE1:9600p:&:
kanga:/dev/cuaE2:9600p:&:
```

```
%%
allow:  itga.com.au
trusted: 127.0.0.1 buzz
```

Первая строка означает, что по умолчанию все файлы протоколов будут располагаться в каталоге `/var/log/consoles` . Символ «&» в каждой строке указывает на то, что файл журнала для этой машины будет называться `/var/log/consoles/`*`machine`*`.`

В следующих трёх строках показаны три машины, к которым нам нужно подключаться. Мы используем устройства `cuaE`*`x`* вместо `ttyE`*`x`*, потому что на консольных портах обычно отсутствует несущая. Это означает, что открытие `ttyE`*`x`* будет зависать, и conserver никогда не сможет осуществить подключение. Использование устройства `cuaE`*`x`* позволяет уйти от этой проблемы. Другим решением будет использование устройств `ttyE`*`x`* и разрешение использования на этим портах «программной несущей», возможно, путём установки этого при помощи устройства `ttyiE`*`x`* в файле `/etc/rc.serial` . Посмотрите комментарии в этом файле для выяснения всех деталей. Также посмотрите sio(4) для получения информации об устройствах с начальным состоянием и с блокированным состоянием. (Драйвер Stallion также поддерживает эти соглашения). И прочтите stty(1) для получения подробностей об установке режимов работы устройств.

В последнем разделе указано, что любой пользователь, зарегистрировавшийся на серверной машине, имеет доступ без пароля ко всем консолям. Мы делаем так, потому что на этой машине нет учётных записей пользователей, и она безопасно изолирована от внешнего мира межсетевым экраном. Строка разрешения позволяет всем на этой машине внутри нашей организации иметь доступ к консольному серверу, если он сообщит свой пароль, который записан в файле `conserver.passwd` (обратитесь к следующему разделу).

4.6. Задание паролей для conserver

Файл `conserver.passwd` содержит зашифрованную версию пароля каждого пользователя. Файл описан на справочной странице `conserver.cf(5)` .

Единственной хитростью является заполнение файла зашифрованными паролями. Во FreeBSD нет единого способа генерации зашифрованных паролей для включения в другой файл (однако смотрите ниже). Так что я наскоро создал хакерский perl-скрипт для этого:

```
@rands = ();
foreach (0..4) {
 push(@rands, rand 64);
}

$salt = join '', ('.', '/', 0..9, 'A'..'Z', 'a'..'z')[@rands];
```

```
$salt = '$1$' . $salt . '$';

print 'Enter password: ';
`stty -echo`;
$cleartext = <>;
`stty echo`;
chop($cleartext);
print crypt($cleartext, $salt), "\n";
```

Примечание

Он использует пароли FreeBSD с MD5-шифрованием. Запуск скрипта на других вариантах UNIX® или во FreeBSD с шифрованием паролей DES, скорее всего, потребует другой базы шифрования.

Недавно Kris Kennaway <kris@FreeBSD.org > показал, что вы можете достичь того же эффекта при помощи команды `openssl passwd` :

```
% openssl passwd -1
Password: password
$1$VTd27V2G$eFu23iHpLvCBM5nQtNlKj/
```

4.7. Запуск conserver во время загрузки системы

Существуют два способа это сделать. Во-первых, вы можете запускать conserver при помощи init, включив строчку в `/etc/ttys` , подобную следующей:

```
cuaE0 "/usr/local/sbin/conserver" unknown on insecure
```

Здесь есть два преимущества: init перезапустит главный консольный сервер, если по какой-то причине он аварийно завершит свою работу (но мы пока подобных случаев не наблюдали), и он обеспечивает то, что стандартная выдача процесса conserver будет направлена на указанный tty (в этом случае `cuaE0`). Это полезно, потому что вы можете подключить терминал к порту, а программа conserver выдаст всю консольную выдачу, не попавшую подключенному консольному клиенту. Такое использование полезно в качестве инструмента мониторинга общего характера, чтобы смотреть, что происходит. Мы сделали такой терминал в компьютерном зале видимым из основного офиса. Это очень удобная возможность. Минусом запуска conserver из файла ttys является невозможность его запуска в режиме даемона (либо init(8) будет постоянно его перезапускать). Это значит, что conserver не будет записывать PID-файл, что усложняет смену журнальных файлов.

Таким образом, мы запускаем conserver из rc.d-скрипта. Если вы устанавливали conserver как порт, то в каталоге `/usr/local/etc/rc.d` будет установлен файл

conserver.sh.sample .Скопируйте и/или переименуйте его в conserver.sh для того, чтобы заставить conserver запускаться в момент загрузки системы.

На самом деле мы используем модифицированную версию этого скрипта, которая также подключает conserver к терминалу посредством tty-устройства, так что мы можем отслеживать незамеченную консольную выдачу. Наш скрипт conserver.sh выглядит примерно так:

```
#!/bin/sh
#
# Startup for conserver
#

PATH=/usr/bin:/usr/local/bin

case "$1" in
 'start')
  TTY=/dev/cuaE7
  conserver -d > $TTY
  # get NL->CR+NL mapping so msgs look right
  stty < /dev/cuaE7 opost onlcr
  echo -n ' conserver'
  ;;

 'stop')
  kill `cat /var/run/conserver.pid` && echo -n ' conserver'
  ;;

 *)
  echo "Usage: $0 { start | stop }"
  ;;

esac
exit 0
```

Примечание

Отметьте использование устройства cuaE0 и необходимость задания tty-режимов для правильной обработки последовательностей NL-<CR).

4.8. Обрезание журнальных файлов

Во FreeBSD имеется программа под названием newsyslog, которая будет обслуживать усечение журнального файла в автоматическом режиме. Просто добавьте некоторые строки в конфигурационный файл /etc/newsyslog.conf для журналов консолей:

```
#
# The log files from conserver
/var/log/consoles/gallows 644  10    1000 *     Z /var/run/
conserver.pid
/var/log/consoles/kanga  644  10    1000 *     Z /var/run/conserver.↺
pid
/var/log/consoles/roo  644  10    1000 *     Z /var/run/conserver.pid
```

Здесь программе newsyslog (которая выполняется по таймеру один раз в каждый час) указывается, что файлы протоколов работы консолей должны архивироваться и сжиматься, как только они достигнут объёма в 1 Мбайт, что мы должны хранить 10 таких журналов, и что для подачи сигнала `SIGHUP` вы используете PID, записанный в файле `conserver.pid` . Это главный сервер, и он будет передавать сигнал всем дочерним процессам. Да, он будет посылать сигнал `HUP` всем клиентам, как только понадобится обновить единственный файл журнала, но это достаточно дёшево. Для выяснения всех подробностей обратитесь к newsyslog(8).

5. Подключение кабелей

Это всегда является самой сложной частью такого рода проблем. Для построения у нас имелось только около десятка кабелей/окончаний к ним, и ещё набор соответствующих инструментов и оборудования, так что мы сделали всё сами. Однако если вы к этому не готовы, либо вам нужно сделать большое количество кабелей, то вам можно приобрести их на заказ. Посмотрите справочники фирм, там найдётся на удивление много мест, где сделают всё нужное! Приобретение кабелей, сделанных на заказ, это хорошо, и вы получите более профессиональный результат, однако это может быть дороговато. К примеру, наборы переходников RJ-45 в DB-25, описываемые ниже, стоят около $10 каждый; кабели на заказ обойдутся примерно в два раза дороже (и будут доставлены через несколько недель). Подобным же образом изготовление переходника RJ-45 в RJ-45 обойдётся достаточно дёшево (скажем, по $5 каждый), но займёт много времени. Заказное гнездо RJ-45 с переходником RJ-45 стоит около $25 каждое.

Во всех случаях в офисе и компьютерном заде мы использовали кабель типа RJ-45 Cat-V. Сюда включается проброс монтажных кабелей между стойками в компьютерном зале. Для последовательных соединений мы используем подключаемые соединители, у которых на задней стенке есть гнёзда RJ-45. Это позволяет нам при необходимости организовывать соединения RJ-45-DB-25.

Которое также удобно, потому что есть множество неправильных способов организовать последовательные соединения на вилке RJ-45. Так что при пробросе кабелей нужно очень осторожно использовать правильное соответствие.

5.1. Цветовая разметка RJ-45

Кабели и вилки RJ-45 имеют 8 контактов/проводников. Они используются как 4 соответствующих пары. Имеется несколько соглашений о том, как пары соответствуют контактам, однако в 100baseT используется самый распространённый (известный как EIA 586B). Имеются три распространённых соглашения по цветовому обозначению для отдельных проводников в кабелях RJ-45. Вот они:

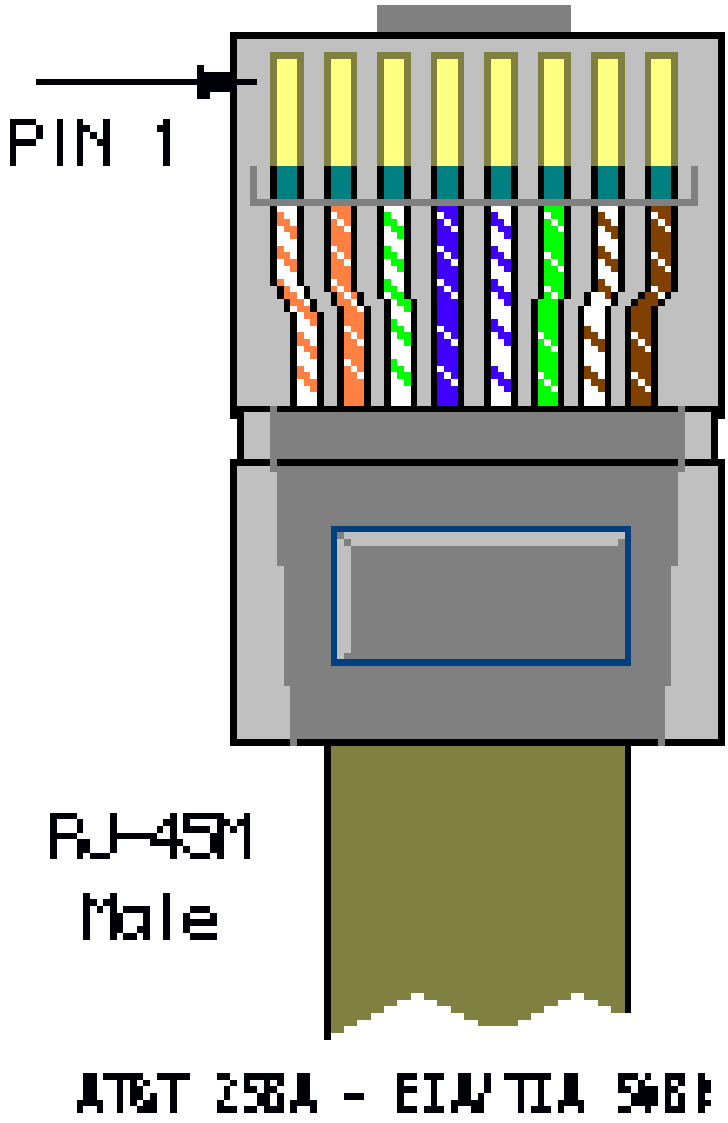
Таблица 1.

Контакт	Схема 1	Схема 2 (EIA 568B)	Схема 3 (EIA 568A)	Пара
1	Синий	Белый+Зелёный	Белый+Оранжевый	2+
2	Оранжевый	Зелёный	Оранжевый	2-
3	Чёрный	Белый+Оранжевый	Белый+Зелёный	3+
4	Красный	Синий	Синий	1+
5	Зелёный	Белый+Синий	Белый+Синий	1-
6	Жёлтый	Оранжевый	Зелёный	3-
7	Коричневый	Белый+Коричневый	Белый+Коричневый	4+
8	Белый или Серый	Коричневый	Коричневый	4-

Заметим, что стандарты EIA 468A and EIA 568B отличаются только цветом 2 и 3 пары.

Подробности можно прочитать на сайте технической поддержки Cabletron.

Контакты разъема RJ-45 нумеруются с 1 до 8. Первый контакт расположен слева, если держать обжатый кабель разъемом вверх и защелкой от себя. В розетке RJ-45, расположенной защелкой вверх, контакт 1 расположен справа. Вот иллюстрация (бесстыдно стянутая с сайта Cabletron), показывающая все это:



В нашем случае мы имели дело с четырьмя видами оборудования:

Сервера Sun
: Консоль сервера Sun работает в режиме DTE (т.е. посылает данные по линии TxD, принимает данные по RxD и активирует сигнал DTR) с разъемом DB-25 "мама". Для консольного сервера на базе Stallion нам потребовались переход-

ники, работающие как DCE и обладающие разъемом DB-25 "папа" (т.е. работающие одновременно как «нуль-модем» и как переходник RJ-45-DB-25. Мы использовали разборные переходники, содержащие розетку RJ-45, 8 коротких проводов, заканчивающихся контактами DB-25, которые могут произвольно коммутироваться в корпус разъема DB-25. Мы использовали несколько схем соединения, в частности, MOD-TAP part no. 06-9888-999-00 и FA730 series от компании Black Box.

Контакты переходников, которые попались нам, были маркированы так (контакты с 1 по 8): Синий, Оранжевый, Черный, Красный, Зеленый, Желтый, Коричневый, Белый (при взгляде со стороны розетки RJ-45, защелка сверху, контакт 1 справа). Они были скоммутированы с разъемом DB-25 вот так:

Таблица 2.

Контакт RJ-45 Stallion	Цвет	Сигнал	Контакт разъема DB-25 "папа" Sun	Сигнал RS232
1	Синий	DCD	20	DTR
2	Оранжевый	RTS	5	CTS
3	Черный	Заземление	1	Заземление
4	Красный	TxD	3	RxD
5	Зеленый	RxD	2	TxD
6	Желтый	Сигнальный ноль	7	Сигнальный ноль
7	Коричневый	CTS	4	RTS
8	Белый	RTS	8	DCD

Для ваших кабелей и переходников цвета могут отличаться. Например, 8 провод может быть серого, а не белого цвета.

Не забудьте *четко* пометить переходник, так чтобы пометка не стерлась и не отвалилась со временем!

Маршрутизаторы Cisco 16xx/26xx/36xx

Я полагаю, что все продукты Cisco, использующие разъемы RJ-45 для консоли и работающие под управлением IOS®, требуют одинаковых кабелей, но лучше будет дополнительно проверить. Мы работали только с маршрутизаторами серий 1600, 2600 и 3600.

И Stallion, и Cisco 2600 используют разъемы RJ-45, но они, разумеется, не совместимы, поэтому вам потребуется специально обжатый (и подключенный в правильной ориентации!) кабель RJ-45-RJ-45. Мы использовали стандартные про-

вода RJ45 от маршрутизаторов до патч-панелей и специально подготовленные от патч-панели до разъемов карты Stallion.

Пара специальных кабелей Stallion-Cisco была сделана путем разрезания пополам стандартного двухметрового патч-корда и набивания разъемов RJ-45 на получившиеся свободные концы. Изначальный разъем предназначается для стороны маршрутизатора Cisco, обжатый для Stallion. Цвета проводов (как и прежде, держа кабель разъемом вверх и защелкой от себя, слева направо) в нашем случае были такими: бело-зеленый, зеленый, бело-оранжевый, синий, бело-синий, оранжевый, бело-коричневый, коричневый. Для стороны Stallion следовало обрезать коричневую и зеленую пары. Затем, в уже описанной расстановке, оставшиеся провода коммутировались так: пусто, пусто, синий, оранжевый, бело-оранжевый, бело-синий, пусто, пусто, как показано ниже:

Таблица 3.

Контакт RJ-45 Cisco	Цвет	Сигнал Cisco	Контакт RJ-45 Stallion	Сигнал Stallion
1	бело-зеленый	RTS	N/C	
2	зеленый	DTR	N/C	
3	бело-оранжевый	TxD	5	RxD
4	синий	Gnd	3	Gnd
5	синий	Gnd	6	Gnd
6	оранжевый	RxD	4	TxD
7	бело-коричневый	DSR	N/C	
8	коричневый	CTS	N/C	

Вновь отметим, что цвета ваших кабелей и переходников могут отличаться.

Аккуратно пометьте каждый конец кабеля и тщательно протестируйте его. Тестирование может стать *по-настоящему* сложным, поскольку его нельзя произвести при помощи стандартного RJ-45 тестера!

Заметим еще раз: *убедитесь*, что вы пометили новый кабель так, чтобы его было легко сразу опознать как специальный и нельзя было бы перепутать с обычным патч-кордом. Несколько советов от Хью Ирвина (Hugh Irvine):

- Делайте их из кабеля другого цвета

- Лучший способ маркировки кабеля, который мне встречался: запаять напечатанную этикетку под прозрачный термоусадочный кембрик на конец кабеля перед разделкой разъема

- Можно использовать маркеры типа Panduit, прикрепляемые к кабелю стяжками, но на них со временем выцветают чернила.

Коммутаторы Cisco Catalyst®

Как ни странно, расположение сигналов на контактах консольного порта коммутаторов Catalyst® иногда *отличается* от используемого в маршрутизаторах Cisco. Я полагаю, что карта сигналов определяется используемым программным обеспечением. Если коммутатор работает под управлением IOS®, применяется раскладка, описанная выше. В противном случае, следует применить хитрость.

К счастью, вся разница в расположении сигналов заключается в том, что одна из раскладок является зеркальным отражением другой. Еще радостнее то, что в комплекте с оборудованием Cisco (как с коммутаторами Catalyst®, так и с 2600) поставляется специальный «перевернутый» («rollover») кабель; он-то нам и нужен. Мы использовали перевернутый кабель для связи консольного порта коммутаторов Catalyst® и патч-панели, а затем описанный выше для Cisco 2600 специальный кабель от патч-панели до карты Stallion. Все прекрасно работало.

Перевернутый кабель имеет на обоих концах разъемы RJ-45 и предназначен для использования вместе с переходниками RJ-45 - DB-25 и RJ-45 - DB-9 (также поставляемыми в комплекте; неразборными) для присоединения к консоли. В нашем случае кабель был плоским, длиной около 2 м, голубого или черного цвета. Попытки использовать его как обычный 100-Мбитный сетевой кабель окончатся неудачей! Определить такой кабель легко, если взять оба разъема (кабелем вниз, защелкой от себя) и сравнить цвета контактов. Разъемы должны выглядеть зеркально; в нашем случае это были серый-оранжевый-черный-красный-зеленый-желтый-синий-коричневый с одной стороны, и коричневый-синий-желтый-зеленый-красный-черный-оранжевый-серый с другой.

Если у вас нет под рукой перевернутого кабеля, вы можете использовать кабель для маршрутизатора 26xx, перевернув его: исходный разъем с 8 контактами в Stallion, новый с 4 контактами в коммутатор Catalyst®.

Сервера FreeBSD (или любые другие системы на базе ПК i386™, использующие последовательную консоль)

Мы используем FreeBSD 4 на паре ПК архитектуры i386™ для различных периферийных нужд. FreeBSD обычно использует экран и клавиатуру в качестве консоли, но может быть сконфигурирована в режим последовательной консоли (как правило, на первый последовательный порт, известный как `COM1` в DOS/Windows® или `ttyd0` в UNIX®).

Подключение таких консолей зависит от используемого оборудования. Старые ПК использовали разъем DB-25 "мама", и для них подходит описанный выше вариант для сервера Sun. Для современных ПК с разъемами DB-9 "папа" существуют два варианта: использовать переходник DB9 - DB-25 (не рекомендуется, поскольку со временем такие соединения разбалтываются и ведут к непредсказуемым потерям связи) или собрать кабель RJ-45 - DB-9:

Таблица 4.

Контакт RJ-45 Stallion	Цвет	Сигнал	Контакт DB-9 "мама"	Сигнал RS232
1	Синий	DCD	4	DTR
2	Оранжевый	RTS	8	CTS
3	Черный	Защитная земля	N/C	
4	Красный	TxD	2	RxD
5	Зеленый	RxD	3	TxD
6	Желтый	Сигнальный ноль	5	Сигнальный ноль
7	Коричневый	CTS	7	RTS
8	Белый	RTS	1	DCD

См. также раздел Раздел 7, «Использование последовательной консоли в FreeBSD». В нем вы найдете советы по конфигурированию последовательной консоли в FreeBSD.

6. Про системы Sun и сигнал Break

Всякий, кто хоть раз выключал терминал, используемый в качестве консоли для сервера Sun, знает, что происходит в результате и почему это является проблемой. Оборудование Sun считает сигнал BREAK на консоли командой остановить систему и вернуться в монитор загрузчика. Сигнал BREAK - специальный срочный сигнал последовательного порта, заключающийся в активизации (установки в уровень ниже -5 В) сигнала TxD на время большее чем требуется на передачу двух символов (около 2 мс для скорости 9600 bps). К сожалению, этот сигнал часто возникает при включении или выключении коммуникационного оборудования. В частности, карты Stallion также генерируют BREAK при отключении питания компьютера. Если не предпринимать специальных действий, это может привести к остановке всех серверов Sun, подключенных к консольному серверу, при его отключении (при отказе блока питания, или в результате неосторожных действий оператора, или по еще каким-либо причинам). Ясно, что такая ситуация неприемлема.

К счастью, у компании Sun есть набор исправлений. Для ОС Solaris™ версии 2.6 и более поздних, при помощи команды `kbd(1)` можно запретить переход в монитор загрузчика по сигналу BREAK. Это уже неплохо для начала, но лишает вас шансов восстановить повисшую машину, вернув ее в загрузчик.

Начиная с Solaris™ версии 8, команду `kbd` можно использовать для установки альтернативной последовательности прерывания: `kbd -a alternate` . После активации, для возврата в монитор загрузки необходимо в течение 5 секунд выдать последовательность: Return ~ Ctrl+B. Эта возможность может быть включена на постоянной основе путем редактирования файла `/etc/default/kbd` ; подробнее см. справочную страницу `kbd(1)` . Отметим, что альтернативная последовательность активируется после перехода ядра в многопользовательский режим и обработки файла начальных установок. В период начальной загрузки (включение питания и в процессе загрузки ядра) и в однопользовательском режиме для возврата в монитор загрузки нужно использовать сигнал `BREAK`. Из консольного клиента его можно активировать последовательностью Esc c l 1.

Если у вас есть сервисный контракт с компанией Sun, вы можете скачать патчи, реализующие альтернативную последовательность прерывания, для Solaris™ версий 2.6 и 2.7. Solaris™ 2.6 требует патча 105924-10 или выше; Solaris™ 2.7 - 107589-02 или выше.

Мы применили этот патч на всех наших серверах Solaris™ 2.6 и добавили его (вместе с установкой для файла /etc/default/kbd) в стартовую конфигурацию, так чтобы любой новый сервер автоматически был правильно сконфигурирован.

Наши тесты показали, что ни маршрутизаторы Cisco 16xx, 26xx, ни коммутаторы Catalyst® не подвержены проблеме сигнала `BREAK`, возникающего при потере питания картой Stallion. В настоящее время маршрутизаторы и коммутаторы Cisco реагируют на сигнал `BREAK` только в течение первых 30 секунд после включения питания или перезагрузки.

7. Использование последовательной консоли в FreeBSD

Подробно эта процедура описана в отдельной главе Руководства FreeBSD. Здесь мы приводим краткий перечень.

7.1. Проверьте конфигурацию ядра

Проверьте, что файл конфигурации ядра содержит `flags 0x10` в строке, описывающей устройство `sio0`. Этот флаг разрешает использование устройства (известного также как `COM1` в DOS/Windows® или как `/dev/ttyd0` в FreeBSD) в качестве консоли. Флаг установлен в обоих примерах стандартной конфигурации ядра (`GENERIC` и `LINT`), так что, скорее всего, он установлен и в вашем ядре.

7.2. Создайте файл `/boot.conf` file

Этот файл должен состоять из одной строки, содержащей только «`-h`» (без кавычек). Этот флаг указывает загрузочным блокам FreeBSD переключиться на последовательную консоль.

7.3. Отредактируйте файл `/etc/ttys`

Необходимы следующие изменения:

Если вы не собираетесь подключать клавиатуру и монитор к этому серверу, найдите все строки для устройств `ttyv`, таких как

```
ttyv1   "/usr/libexec/getty Pc"  cons25  on  secure
```

Замените `on` на `off`. Это запретит запуск утилит регистрации на ненужных более видео консолях.

Найдите строку, содержащую `ttyd0`. Измените ее с

```
ttyd0   "/usr/libexec/getty std.9600"   dialup  off secure
```

на

```
ttyd0   "/usr/libexec/getty std.9600"   vt100   on secure
```

(замените `vt100` на тип терминала вашей консоли. Хорошим выбором может быть `xterm`). Это позволит вам зарегистрироваться на консоли после того, как система перейдет в многопользовательский режим.

Перезагрузитесь - и все!

8. Соображения безопасности

Протокол "клиент-сервер" утилиты conserver требует от пользователя клиентской утилиты console ввода пароля. Этот пароль передается по сети в *открытом виде*! Как следствие, утилита conserver не особенно пригодна для использования в небезопасных сетях (в том числе в интернет). Использование уникальных паролей для conserver слегка смягчает проблему, однако любой, кто способен прослушать сетевой трафик между клиентом и сервером conserver, может легко получить консольный доступ, а с консоли использовать последовательность прерывания. Для удаленной работы используйте какие-либо защищенные протоколы, такие как SSH для регистрации на консольном сервере и запускайте консольный клиент непосредственно оттуда.

9. Различные версии Conserver

Программа conserver расслоилась на несколько независимых версий. Сайт, упоминаемый ниже, судя по всему, содержит последнюю и наиболее полную версию из доступных. На момент написания статьи (июль 2004 г.) это версия «8.1.9». Сайт поддерживается Брайаном Стэнселлом (Bryan Stansell), <bryan@conserver.com >, который свел воедино работу многих разработчиков (перечислены на его сайте).

Коллекция портов FreeBSD содержит conserver версии 8.5 в каталоге comms/conserver. Судя по всему, он более старый, и содержит меньше возможностей, чем 8.1.9 (в частности, не поддерживает консоли, соединенные с портами терминальных серверов, или файл паролей `conserver.passwd`), а также написан довольно своеобразно (используя препроцессор для генерации исходного текста на языке C). Версия 8.5 ведется Кевином Браунсдорфом (Kevin S. Braunsdorf) <ksb+conserver@sa.fedex.com >, в прошлом основным автором conserver; Брайан основывался на его разработках. Версия 8.5 поддерживает одну возможность, не поддерживаемую 8.1.9: управление питанием удаленных машин через специальный контроллер, управляемый по последовательному порту.

Начиная с декабря 2001 г., версия Брайана (в настоящее время 8.1.9) присутствует в дереве портов в каталоге comms/conserver-com. Мы рекомендуем использовать именно ее как более подходящую для построения консольного сервера.

10. Ссылки

`http://www.conserver.com/`
: Сайт последней версии программы conserver.

ftp://ftp.conserver.com/conserver/conserver-8.1.9.tar.gz
: Архив исходных текстов для версии 8.1.9 программы conserver.

`http://www.stallion.com/`
: Сайт компании Stallion Technologies.

`http://www.conserver.com/consoles/msock.html`
: Написанный Дэвидом Харрисом (Davis Harris) «Малый Свиток Знаний о Консолях», содержащий массу полезной информации о последовательных консолях и вообще о последовательных портах.

`http://www.conserver.com/consoles/`
: «Большой Свиток Знаний о Консолях» содержит еще более подробную информацию о соединении одних устройств с другими. О, эти Стандарты!

`http://www.eng.auburn.edu/users/doug/console.html`
: Дуг Хьюджес (Doug Hughes) создал схожий консольный сервер на основе утилиты screen и старой машины под управлением SunOS™.

`http://www.realweasel.com/`
: Компания Real Weasel производит видеокарты для шин ISA или PCI, в реальности производящие вывод в последовательный порт. Они могут использоваться для консолей ПК в тех операционных системах, которые не могут быть сконфигурированы в режим последовательной консоли достаточно рано в процессе загрузки.

11. Справочные страницы

- console(8)
- conserver(8)
- conserver.cf(5)

Предметный указатель

Символы

консольный сервер, 2