Роман Химов

InitNG — начни сначала

С InitNG мне довелось познакомиться еще в версии 0.06 — одной из первых публично доступных. Тогда новость о нем облетела все ленты. И я не удержался от того, чтобы не поэкспериментировать. И не зря: тогда я впервые увидел, что замученный жизнью Celeron 480 со 128 Мбайт памяти, оснащенный к тому же множеством сервисов, может запускать пингвина в полет быстрее, чем это делает стоящий рядышком Athlon 1700+ с 256 Мбайт.

Правда, с той версией пришлось изрядно повозиться, поскольку стабильно она работать никак не хотела и периодически норовила отказаться то от загрузки kdm, а то и других сервисов. Тем не менее старый добрый шаманский бубен меня выручил, и эта версия простояла на том самом компьютере очень долго, периодически выдавая сбои, но в целом обеспечивая достаточный уровень надежности работы.

Да, InitNG все еще является экспериментальным софтом, поэтому накладки в процессе его использовании весьма вероятны. Позже я заменил старую версию на InitNG 0.3.x, поборолся с новыми глюками и отметил чуть более стабильное поведение системы. Однако по-настоящему беспроблемной стала только лишь последняя ветка InitNG версий 0.4.x.

Init

Что же такое InitNG и что он призван заменить? InitNG является альтернативой Init, который однозначно наличествует в вашем дистрибутиве, каким бы он ни был. Init — процесс номер один в системе, именно ему ядро передает управление после инициализации, и именно он запускает все, что вы от него хотите на этапе загрузки. К тому же Init является предком любого процесса, запущенного в вашей системе.

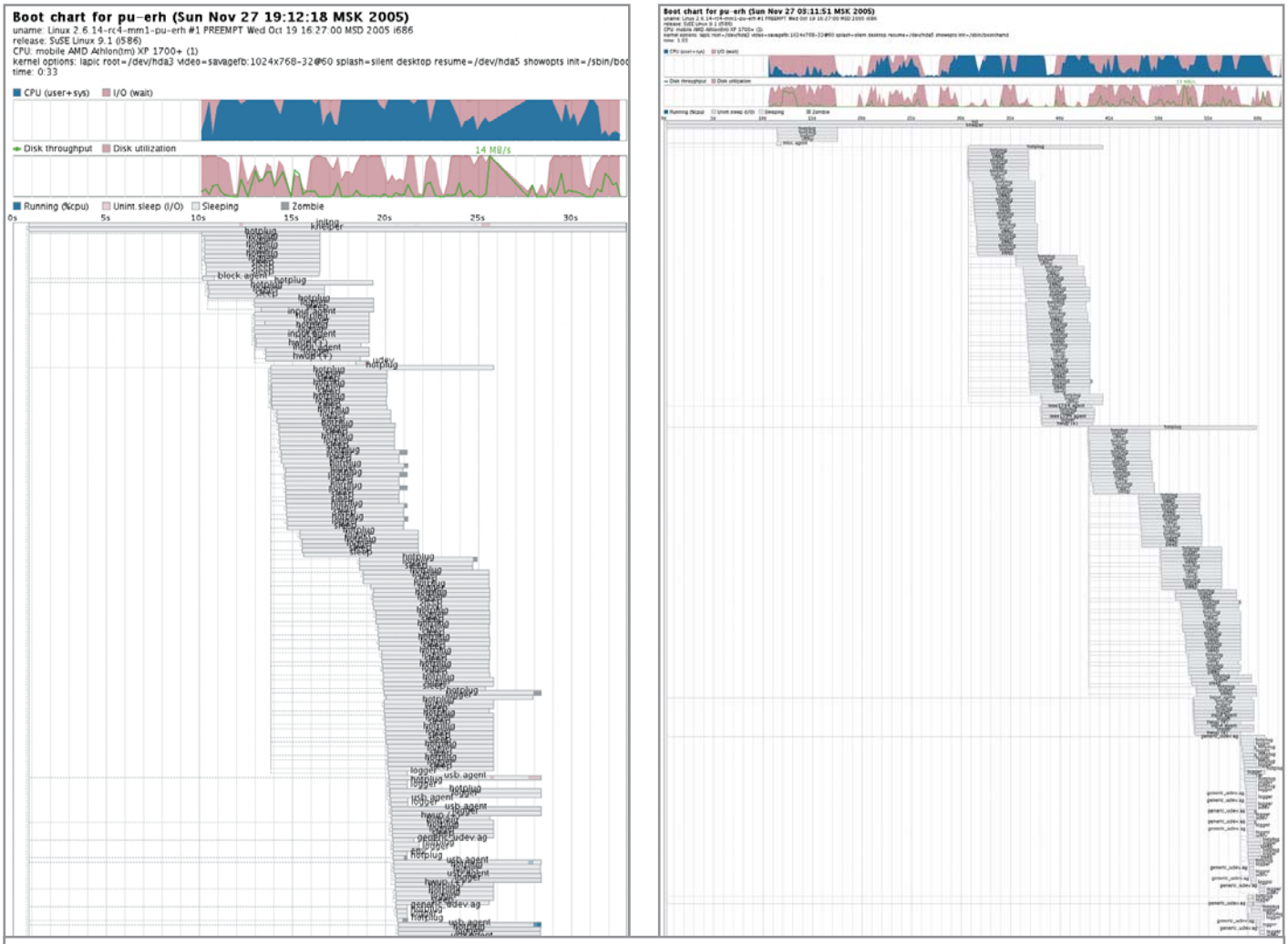
Так сложилось, что эта область свободных систем практически не развивалась. В GNU использовался аналог — SysVinit, и всем жилось, в общем-то, хорошо. Ровно до тех пор, пока не обнаружилось, что Windows XP загружается быстрее многочисленных пользовательских дистрибутивов GNU/Linux. Время загрузки явилось основным катализатором исследований в этой области, а InitNG стал одним из результатов этого трудоемкого процесса.

Естественно, проект не представляет интереса для серверов, поскольку в этой области машины не должны выключаться и перезагружаться вообще, а уж если и придется, то минута-другая погоды не делает. InitNG, равно как и многие из последних довольно значительных изменений в GNU/Linux, предназначен для настольных машин простых пользователей, которые не любят долго сидеть перед монитором в ожидании окончания процесса загрузки. Правда, пока он не используется как стандарт ни в одном дистрибутиве. Пользователям придется разминать пальчики в консоли и познавать таинства конфигурации своих систем.

Проблемы Init

В чем же проблема старого Init, почему ему потребовалась замена? Он плохо дружит с параллельным запуском сервисов, это факт. Весь процесс загрузки должен быть расписан в скриптах строго последовательно. Это плохо, так как большинство сервисов на самом деле не зависят друг от друга, но при этом они не могут запускаться параллельно из-за ограничений Init.

Как правило, все запускаемые сервисы что-то считывают с диска, а это всегда происходит очень медленно. Во время та-



Графики загрузки, полученные в bootchartd. InitNG обогнал Init почти в два раза

кого ввода/вывода процессор мог бы уделить внимание другому сервису, но старый Init этому не способствует. Аналогично происходит и с coldplug, который задает работу ядру по общению с устройствами, а это тоже весьма длительная операция, и хорошо бы в эти миллисекунды простоя занять процессор чем-нибудь другим, не менее полезным.

Новации InitNG

InitNG решает описанную проблему путем параллельного запуска всех не зависимых друг от друга сервисов, что способствует равномерной загрузке процессора и периферии, в результате все сервисы вместе успевают загрузиться значительно быстрее.

InitNG ладно скроен и имеет модульную архитектуру, хотя это больше интересно программистам, которые захотят его расширить. Параллельность же запуска обеспечивается правильной системой скриптов, в которой порядок их выполнения рассчитывается на основе реальных зависимостей сервисов.

Надо отметить, что существующая сегодня система загрузки в Gentoo имеет схожую структуру (основывается на зависимостях) и даже позволяет параллельно запускать сервисы, однако старый Init ограничивает эти возможности, и реальный прирост скорости загрузки от этого весьма невелик. Новый InitNG полностью лишен подобных недостатков.

Установка и дистрибутивы

Надо признать, что наиболее комфортно InitNG чувствует себя в той среде, где он и вырос, а именно в Gentoo. Еще в очень ранних версиях для него был доступен ebuild, который теперь также входит в основной репозиторий Gentoo. Поэтому все, что от вас здесь потребуется, — это ввести emerge initng, а portage сделает все самостоятельно. Хотя минимальная конфигурация, наверное, все-таки потребуется.

Обладателям других дистрибутивов придется засучить рукава. Пользователям Debian и его многочисленных отпрысков (в том числе и Ubuntu) здесь повезло несколько больше, поскольку для этих систем существует специальный пакет Debian (http://alioth.debian.org/projects/pkg-initng), который упрощает процесс установки.

Остальным же придется производить сборку своими руками из исходников (http://initng.thinktux.net), что на самом деле не так уж и плохо. InitNG имеет достаточно умный Makefile, который пытается догадаться, какой дистрибутив у вас установлен (соответственно подстраивая скрипты запуска сервисов) и что за сервисы стартуют по умолчанию (соответственно составляя список запуска у себя).

Отмечу, что все тесты и примеры конфигураций основаны на InitNG версии 0.4.4. Проект очень бурно развивается, по-

этому, возможно, к моменту выхода журнала некоторая информация будет уже не совсем точной.

После установки необходимо добавить еще один пункт в меню загрузки вашей системы и в нем прописать ядру параметр init=/sbin/initng. В GRUB он напрямую помещается в конец строки «kernel (..)/...», а в LILO это означает добавление строчки 'append = «init=/sbin/initng»' в желаемый пункт. Не стоит пока что делать эту конфигурацию загружаемой по умолчанию.

Далее необходимо попробовать перезагрузиться с InitNG и приготовиться к работе в чистом консольном режиме — ставлю пять к одному, что с первого раза X и [kgx]dm у вас не запустятся. Равно как и многие другие сервисы. Не нужно бояться этих проблем — хотя вывод об ошибках может быть и пугающим, навредить системе вы вряд ли сможете, незапускающийся сервис не означает разрушенной конфигурации, достаточно перезагрузиться со старым Init — и все будет в порядке.

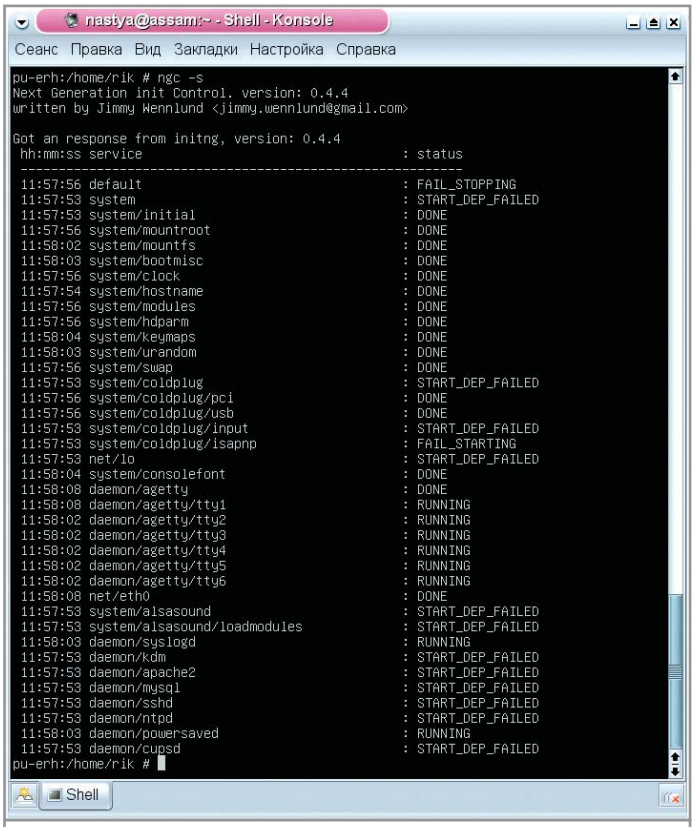
Притирка и доводка

К огромному сожалению, каждый дистрибутив считает чуть ли ни своим долгом использовать собственные загрузочные скрипты, а также расположение и формат конфигурационных файлов, скрипты-помощники и прочее. Но к этому можно отнестись с большим энтузиазмом — настраивая InitNG, вы лучше узнаете свою систему, а любители LFS (Linux from Scratch) уже могут примерять InitNG к своим системам в качестве единственного Init. В конечном счете на это вряд ли уйдет так много времени (все зависит от ваших навыков работы с консолью).

Итак, пробуем перезагрузиться с InitNG. Если у вас все заработало, знайте — вы редкий везунчик. Скорее же всего, вы увидите, что система загрузилась не полностью, и вывод InitNG запнулся на уровне 70–80% загрузки, при этом первая консоль совершенно непригодна к использованию, а «иксы»... о них на время вообще лучше забыть.

Вспоминаем о существовании других терминалов и смело нажимаем «Alt+F2». По умолчанию InitNG запускает agetty на шести терминалах, но первый из них запускается только в случае абсолютно успешной загрузки стандартного уровня исполнения. Заходите сразу как root и не стесняйтесь.

Основная задача на данном этапе — выявить сервис или демон, который помешал полноценной загрузке. Для этого



Неудачное расположение сервисов. Многие из них ждут разрешения конфликтов

придется познакомиться с инструментом ngc (New Generation Init Control), который входит в состав InitNG. Он позволяет полностью управлять процессом загрузки с InitNG. В нашем случае потребуется ngc -s, который покажет статус загруженных сервисов/демонов.

Скорее всего, в выводе будет обнаружено несколько сервисов, ожидающих разрешения зависимостей (START_DEP_FAILED), а также один или два, которые не сумели запуститься (FAIL_STARTING). Ваша задача — избавиться от «плохих» сервисов и демонов.

Ненужные сервисы можно просто удалить из рассмотрения с помощью ng-update del service/badservice, а проще, на мой взгляд, сразу перейти к прямой правке конфигурационных файлов InitNG. Загрузочные скрипты и настройки InitNG располагаются в /etc/initng. Там же вы обнаружите несколько файлов *.runlevel, *.i и каталогов с файлами *.i.

Файлы *.runlevel описывают уровни исполнения InitNG, вы можете загрузить любой из них, передав ядру в качестве параметра необходимый уровень в виде runlevel=level. Файлы *.i описывают конкретные сервисы и демоны, запускаемые InitNG, а в каталоги они просто логически группируются для удобства.

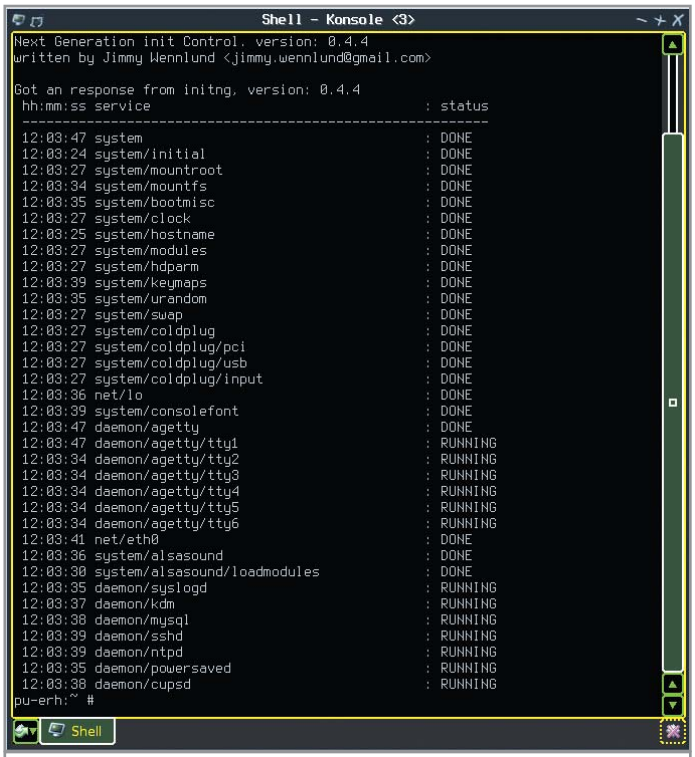
Таким образом, вы сразу можете перейти к правке файлов system.runlevel или default.runlevel (нетрудно догадаться, что именно этот уровень запускается по умолчанию, однако он зависит от параметра system) с целью удаления сервисов и демонов, которые дали сбои. В этих файлах используется простейший список, одна строка — один сервис.

Необходимая настройка

Новые демоны

К сожалению или к счастью, но в составе InitNG может и не оказаться скрипта запуска используемого вами демона. Однако теперь вы уже гуру в области его конфигурации, поэтому, думаю, у вас не возникнет проблем с добавлением новых скриптов. Вот в качестве примера самописный скрипт запуска pure-ftpd:

```
daemon daemon/pure-ftpd {
    need = system/initial system/mountfs system/hostname net/lo
    require_network
    daemon = /usr/sbin/pure-ftpd
}
```



Правильно подобранная цепочка сервисов — все запустилось

Однако какая же польза от InitNG, если мы вырежем все запускаемые сервисы и демоны? Они нам все-таки нужны, поэтому пора познакомиться с файлами *.i, описывающими все, что стартует при помощи InitNG. Стоит отметить, что сегодня также имеется возможность использовать XML для описания сервисов, но это сложнее и не так наглядно, как использование формата *.i. Оставим XML заботливым дистрибьюторам, которые добавят к ним красивые графические настройки; кто-нибудь из них, возможно, решится в следующем году сделать InitNG основным для своего дистрибутива.

Сервисы

Чем отличаются сервисы от демонов? Все очень просто, демоны — стандартные и привычные всем sshd или mysqld — запускаются и функционируют в течение всего времени работы системы. Сервисы же, в терминологии InitNG, являются сущностью, запускаемой единожды на этапе загрузки или выгрузки системы. Примеры — coldplug или настройка часов; все это выполняется один раз при загрузке, настраивает то, что надо, и завершается.

Примеры очень простых сервисов можно также увидеть в system/local.i, system/ntpdate.i и system/swap.i. Рассмотрим синтаксис на примере system/swap.i:

```
service system/swap {
    need = system/initial system/mountroot
    start = /sbin/swapon
    start_args = -a
#   start {
#   #!/bin/bash
#   /sbin/swapon -a
#   exit $?
#   }
}
```

Каждый сервис объявляется как «service имя_сервиса», и все его описание заключается в фигурные скобки. Оно состоит из ключевых слов и их параметров. Самое главное, что должно быть у сервиса, — это start и/или stop.

Посмотрите на различия между закомментированным форматом (а комментарии по традиции начинаются с «#») и рабочим (аналогично отличаются system/local.i и system/ntpdate.i): в них представлены два разных способа задания действий.

Первый использует ключевые слова start и start_args. Команда, выполняемая InitNG при запуске такого сервиса, формируется напрямую из них и тут же вызывается. В данном примере это /sbin/swapon -a.

Второй способ задания действий значительно более гибкий, поскольку все, что заключается в блоке start{} (или stop{}), является обычным shell-скриптом — со всеми вытекающими последствиями. Но такой метод работает медленнее (и именно поэтому в system/swap был закомментирован), так как для его запуска InitNG приходится разворачивать отдельный процесс интерпретатора командной строки и отдавать ему скрипт. Для простых сервисов, потребности которых можно описать в первом формате, он явно менее предпочтителен.

Простейший сервис может состоять только из одного блока start и будет успешно работать, однако сервисы могут зависеть от других сервисов, и мы переходим к наиболее интересной части InitNG — работе с зависимостями.

Зависимости

В простейшем виде используется ключевое слово need, в котором через пробелы вписывается все, что требуется запустить, до того как будет запущен описываемый сервис. В примере с system/swap это system/initial и system/mountroot; действительно, все сервисы зависят от успешного завершения работы скрипта начальной инициализации системы, и

Дополнительная информация

Полезные возможности ngc

После того как вы, надеюсь, разрешите все проблемы, связанные с настройкой InitNG, стоит поближе познакомиться с инструментом ngc. Продолжив это знакомство, например, с ngc -O. Эта нехитрая команда расскажет вам все об опциях, которые можно использовать в описаниях сервисов и демонов, поскольку на самом деле их несколько больше, чем было упомянуто ранее. Более того, они постоянно развиваются, расширяются и изменяются, поскольку InitNG — все еще очень молодой продукт. Также сразу же стоит запомнить ngc -u и ngc -d, которые соответственно запускают и останавливают сервисы и демоны, передаваемые следующим параметром. Полезен и ngc -r, позволяющий сервисы перезапускать. Остальное уже не так интересно, но при желании доступно для изучения по ngc -h.

почти всем необходимо иметь возможность записывать информацию в корневой раздел.

Важно, что описываемые в need зависимости являются критическими для сервиса, и без удовлетворения всех зависимостей InitNG даже не будет пытаться запускать сервис.

Несколько другой подход представляет использование ключевого слова use. Описанные с его помощью зависимости позволяют задавать порядок запуска сервисов/демонов, если они описаны в заданном уровне запуска. То есть, в отличие от need, если вы не собираетесь запускать нечто из списка use, все будет в порядке.

Например, в daemon/apache2 среди use числятся daemon/sshd и daemon/mysql. Это значит, что если вы запускаете на своей машине sshd, то apache2 будет запущен после него. Однако если вы не запускаете sshd, то apache2 будет успешно запущен и без него. Аналогично и с демоном mysql.

Дополнительные возможности

Одной из интересных зависимостей, вынесенной даже в отдельное ключевое слово, является также require_network. Как можно догадаться из названия, она используется теми сервисами и демонами, которые нуждаются в сетевом подключении. Локальное сетевое подключение lo эту зависимость не удовлетворяет.

Помимо этого также доступен env, в формате env = VAR=something можно передавать переменные окружения вызываемым сервисам и демонам. Таких определений может быть несколько для одного сервиса (соответственно, будут заданы несколько переменных окружения). Это и отличает env от других описателей.

Также можно задавать nice, suid и sgid — соответственно приоритет, пользователь и группа, от имени которых будет запущен процесс. На закуску имеются stdout и stderr, с помощью которых можно перенаправлять стандартные потоки ввода и вывода ошибок.

Демоны

Что касается различий сервисов и демонов, последние имеют дополнительные возможности и несколько иной формат задания процессов. С одной стороны, их можно описывать с помощью start{} и stop{} (в качестве примера смотрите daemon/ openvpn), но InitNG имеет гораздо более интересные возможности.

Существует ключевое слово daemon — опять-таки в разных форматах. Пример использования короткого формата можно подсмотреть в daemon/apache2: как видите, существует и daemon_args для передачи параметров демону. Пример формата daemon{} — в daemon/acpid.

Необходимо пояснить, что InitNG не просто запускает демоны, он за ними еще и приглядывает. По умолчанию считается, что если демон возвращает управление (стандартное поведение, например, sshd), значит он так или иначе был убит, а стало быть, InitNG попытается перезапустить его. Именно поэтому в скриптах для запуска демонов стоит использовать exec (как в примере с acpid).

Но можно поступить иначе. Для sshd, например, можно использовать параметр -D, и тогда все будет в порядке. А можно найти и альтернативные способы. Пускай демон возвращает управление InitNG, но мы можем отслеживать его состояние через файл pid. Для этого и предназначена опция pid_file. Пример ее использования есть в daemon/vixie-cron; он возвращает управление в любом случае, но благодаря pid-файлу InitNG может убедиться в том, что он жив-здоров.

Все это позволяет обеспечить надежность функционирования демонов. В случае их внезапной смерти по тем или иным причинам они будут перезапущены (без необходимости применения дополнительных решений вроде daemontools). Однако, чтобы обеспечить подобную функциональность, стоит прописать в параметрах демона «respawn=yes», так как стандартное поведение от версии к версии здесь менялось (в версии 0.4.4 по умолчанию перезапуск отключен).

Прочее

Осталось только упомянуть возможность использования шаблонов имен, а также то, что имена файлов не обязаны совпадать с именами сервисов (хотя это удобно). Прекрасным примером последнего является файл net/net.i, где описаны net/lo и net/*, имеющие разные зависимости и последствия (network_provider в net/* как раз и удовлетворяет зависимость require_network). Шаблоны здесь позволяют единообразно работать с сетевыми подключениями любого вида; запускаемое в данный момент подключение передается в переменной ${NAME}.

Настройка

Всей этой информации должно с лихвой хватить для осознанного изменения настроечных файлов InitNG, что в конечном счете приведет вас к желаемому результату.

В деле исправления настроек в первую очередь стоит обратить внимание на параметры, с которыми запускаются те или иные демоны/сервисы — убедиться в том, что запускаются именно те бинарники и что они лежат там, где предполагает это InitNG. Он действительно здорово подстраивается под окружающую обстановку (если не верите, сравните содержимое файлов *.ii и *.i каталога initfiles в дереве исходников InitNG после сборки), но всему есть свои пределы, и иногда он может ошибаться.

Например, созданный на SUSE 9.1, использованной в качестве тестовой площадки, скрипт daemon/apache2 планировал вызвать /usr/sbin/apache2 -D PHP5 -k start, в то время как PHP5 на этой системе не стоит, а файл /usr/sbin/apache2 вообще отсутствует. Вместо этого в SuSE 9.1 есть /usr/sbin/apache2ctl, которому необходимо передать единственный параметр start. Однако и этого маловато — pid-файл для Apache 2 в SUSE 9.1 виден как /var/run/httpd2.pid, а не /var/run/apache2.pid. После этих тривиальных изменений Apache стал запускаться без каких-либо проблем.

А вот на примере mysqld уже можно рассмотреть твикинг скриптов InitNG. Скрипт по умолчанию запускал /usr/bin/mysqld_safe и зависел от сетевого подключения (require_network), однако mysqld_safe всего лишь обеспечивает поддержку mysqld в запущенном состоянии и логиро-

вание некоторой информации. Последнее вам может быть просто не нужно, а первое вполне может сделать сам InitNG. Поэтому можно смело вписать «daemon=/usr /sbin/mysqld», а в daemon_args прописать все желаемые параметры, после чего добавить respawn=yes и убрать require_network, если вы используете этот сервер только с локальной машины. MySQL после этого прекрасно запускается и работает, а killall mysqld продемонстрирует живучесть сервера — InitNG его моментально перезапустит. Замечу, что в отличие от mysqld_safe эта связка значительно надежнее, поскольку первый можно просто убить, а вот если убить Init, будет слишком заметно.

По поводу «иксов» надо заметить, что основные проблемы с ними сводятся к загрузке модулей. Необходимо убедиться в том, что к моменту запуска [gkx]dm у вас будет загружено все, что касается устройств ввода/вывода. Например, драйвер графического планшета вместе с evdev или, не дай бог, проприетарный драйвер NVIDIA. Если с этим возникают какие-то проблемы, можно прописать принудительную последовательную загрузку необходимых модулей в отдельный (или существующий, хотя это не лучший вариант) сервис, от которого будет зависеть запуск [gkx]dm.

Все это потребует от вас определенных знаний о своей системе и ее настройках, используемых демонах, утилитах, скриптах и, соответственно, их настройках. Отличный подсказчик — /etc/init.d, где расположены «эталонные» скрипты запуска всего, что нужно вашей системе (вы же работали на ней до этого, не так ли?). Пользуйтесь ими как ценными источниками информации.

Хочется отметить, что на сегодня у InitNG есть еще одно огромнейшее преимущество перед стандартным init с любым стилем загрузки — он очень прост, и у вас есть прекрасная возможность проделать персональную конфигурацию с минимумом усилий, которая будет очень быстро запускаться и поддерживать работу необходимых вам демонов.

Как видите, ничего сверхсложного.

И о результатах

Так ради чего же боролись? Предлагаю взглянуть на красивейшие графики загрузки, полученные с помощью bootchartd (www.bootchart.org). Результат очевиден — время загрузки уменьшилось практически вдвое. При этом еще остался резерв для оптимизации, поскольку из скриптов можно убрать как минимум все специфичные для Gentoo проверки и настройки (если, конечно, вы не используете именно Gentoo), а также вообще обойтись без использования некоторых конфигурационных файлов, вписав желаемое прямо в скрипты.

Полезно будет прогнать на своей машине bootchartd и выявить узкие места, которые тормозят загрузку системы, после чего заняться их устранением. Как говаривал Ричард Столлмен, «счастливого хака!»

Самостоятельная работа

Еще быстрее?

Чтобы добиться еще более высокой скорости загрузки, можно переписать скрипты InitNG для своей конфигурации. Если это дополнить перекомпиляцией ядра и отказом от coldplug (все ведь уже в ядре?), то можно сократить время загрузки еще в полтора-два раза — взгляните на еще одну картину загрузки системы. Машина по-прежнему та же, сервисы по-прежнему те же, однако время — уже всего лишь 18 секунд!

Надо отметить, что за чертой двадцати секунд результаты загрузок довольно сильно варьируются (минимум, которого удавалось достичь на тестовой машине — 17 секунд). Например, задержавшийся ответ сервера на DHCP-запрос может запросто изменить время загрузки на несколько секунд, причем еще вопрос — в какую сторону, поскольку bootchart считает окончанием загрузки полноценный графический логин. Кстати, bootchart в такой конфигурации уже тоже начинает мешать сервисам, и сам успевает запустить мониторинг процессора/диска только тогда, когда половина дела уже сделана.

Также становится очевидным то, что дальнейшее сокращение времени зависит в первую очередь от винчестера и файловой системы, поскольку узким местом становится ввод/вывод — он нужен всем стартующим демонам, и параллельный запуск здесь может даже мешать.

С чего начать хакинг скриптов InitNG? В первую очередь об этом расскажет bootchartd, но вот некоторые общие рекомендации:

- **Начните с оптимизации самых первых скриптов, запускающихся при старте, их завершения ждут все.**
- **Трезво оценивайте, действительно ли сервис не может жить без сети.**
- **Разберитесь, нужно ли ему дожидаться, когда смонтируются все файловые системы кроме корневой.**
- **Перейдите на журналируемые ФС и избавьте себя от mountroot.**
- **Не стесняйтесь изучать скрипты-обертки для запуска разных сервисов и скрипты настройки сети вашего дистрибутива. Затем подумайте, нужно ли вам все это, и не проще ли написать их самим для своей системы и конфигурации?**

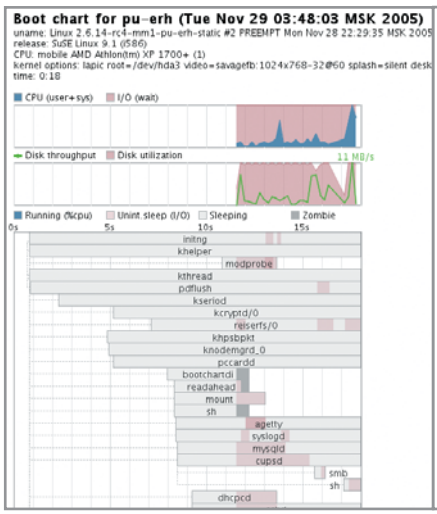


График оптимизированной загрузки