Найти

Последнее изменение: Пт Ноя 27 09:44:56 2020

Проверьте мою страницу Unix и мой блог

Содержание

Оригинальная версия написана в 1995 году и опубликована в Sun Observer

> Полное руководство по поиску. Как искать файл и что-то с ним делать. Как запретить поиск в определенных каталогах.

Введение в поиск

Одна из команд, которую должен освоить каждый, - это команда **find** . Первое и наиболее очевидное применение - это способность **find** находить старые, большие или неиспользуемые файлы или файлы, которые вы забыли, где они находятся.

Другой важной характеристикой является способность **find** перемещаться по подкаталогам. Если вам нужен рекурсивный список каталогов, а в **ls** нет этой опции, используйте **find**.

Проблемы с другими методами

Обычно командная оболочка предоставляет список аргументов команде. То есть программам Unix часто присваиваются имена файлов, а не имена каталогов. Только нескольким программам можно присвоить имя каталога и пройти по каталогу в поисках подкаталогов. Это делают программы **find**, **tar**, **du** и **diff**. Команды **chmod**, **chgrp**, **rm** и **cp** будут, но только если указана опция **-**r **или -**R. В SunOS 4.1 **ls** поддерживает рекурсивную опцию **-R.**

В общем, большинство команд не понимают структуры каталогов и полагаются на оболочку для расширения метасимволов до имен каталогов. То есть, если вы хотите удалить все объектные файлы, заканчивающиеся на ".o" в группе каталогов, вы можете ввести

```
rm *.o */*.o */*/*.o
```

Это не только утомительно для ввода, но и может не найти все файлы, которые вы ищете. Оболочка имеет определенные мертвые зоны. Он не будет соответствовать файлам в каталогах, начинающихся с точки. И, если какие-либо файлы соответствуют ***/*/*/*.o**, они не будут удалены.

Другая проблема, хотя и менее частая в SunOS 4.0 и выше, заключается в вводе приведенного выше и получении ошибки "Слишком длинные аргументы". Это означает, что количество аргументов в списке было слишком большим для оболочки.

Ответ на эти проблемы заключается в использовании **find** .

Самый простой пример

Простой пример поиска - использовать его для печати имен всех файлов в каталоге и всех подкаталогов. Это делается с помощью простой команды

```
Найти. -печать
```

Первый аргумент для **поиска** - это имя каталога или файла. Аргументы после имен путей всегда начинаются со знака минус и сообщают find, что делать, как только он найдет файл. Это параметры поиска. В этом случае печатается имя файла. Можно указать любое количество каталогов. Вы можете использовать символ тильды, поддерживаемый оболочкой C, а также определенные пути.

```
найти ~ ~barnett /usr/local -печать
```

И если у вас очень медленный день, вы можете ввести

```
найти / -печать
```

в котором будут перечислены все файлы в системе.

Прошу прощения. Я только что сказал неправду. Каждый каталог всегда содержит файлы **.**и **... Поиск** мог бы сообщить об этих очевидных файлах, но, к счастью, этого не делает.

Использование поиска с другими командами

Find отправляет свои выходные данные в стандартный вывод. Один из способов использовать это - подстановка команды обратной кавычки:

```
ls -ld `найти . -печать`
```

Выполняется команда **find**, и ее вывод заменяет строку, заключенную в обратные кавычки. **Ls** видит результат **find** и даже не знает, что использовался **find** .

Использование xargs с помощью find

Альтернативный метод использует команду **xargs**. **Xargs** и **find** прекрасно работают вместе. **Xargs** выполняет свои аргументы как команды и считывает стандартный ввод, чтобы указать аргументы для этой команды. **Xargs** знает максимальное количество аргументов, которые может обрабатывать каждая командная строка, и не превышает этот предел. В то время как команда

```
ls -ld `найти / -распечатать`
```

может генерировать ошибку, когда командная строка слишком велика, эквивалентная команда, использующая **xargs**, никогда не будет генерировать эту ошибку:

```
найти / -print | xargs ls -ld
```

Поиск файлов с определенными именами

Вы можете искать определенные файлы, используя регулярное выражение с метасимволами в качестве аргумента для параметра **-name** . Оболочка также понимает эти мета-символы, поскольку она также понимает регулярные выражения. Необходимо заключить эти специальные символы в кавычки, чтобы они передавались в **find** без изменений. Можно использовать либо обратную кавычку, либо одинарные кавычки:

```
Найти. -имя *.o -напечатать
найти . -имя '*.o' -печать
поиска . -имя '[a-zA-Z]*.o' -печать
```

Путь к файлу не сопоставляется с параметром **-name**, просто имя в каталоге без пути, ведущего к файлу.

Поиск файлов по типу

Если вас интересуют только файлы определенного типа, используйте аргумент **-type**, за которым следует один из следующих символов:

```
+-------------------------------------------------+
 |Значение символа |
 +-------------------------------------------------+
 |b Специальный файл блока (см. mknode(8)) |
 специальный файл с символом |c (см. mknode(8)) |
 |общий каталог |
 |f Обычный файл |
 |p именованный файл канала |
 |l Символическая ссылка |
 |s сокет |
 +-------------------------------------------------+
```

Если вы не системный администратор, важными типами являются каталоги, обычные файлы или символические ссылки (например, типы **d**, **f** или **l**).

Используя параметр **-type**, другим способом рекурсивного перечисления файлов является:

```
Найти. -введите f -print | xargs ls -l
```

Может быть сложно отслеживать все символические ссылки в каталоге. Следующая команда найдет все символические ссылки в вашем домашнем каталоге и распечатает файлы, на которые указывают ваши символические ссылки.

```
Найти. -введите l -print | xargs ls -ld | awk '{напечатать $10}'
```

Поиск файлов по размерам

Поиск имеет несколько вариантов, которые принимают десятичное целое число. Одним из таких аргументов является **-размер**. Число после этого аргумента - это размер файлов в дисковых блоках. К сожалению, это очень расплывчатое число. Более ранние версии Unix использовали дисковые блоки по 512 байт. Более новые версии допускают большие размеры блоков, поэтому "блок" в 512 байт вводит в заблуждение.

Эта путаница усугубляется, когда используется команда **ls -s** . Параметр **-s** отображает размер файла в блоках. Если команда "/usr/bin/ls", размер блока составляет 1024 байта. Если команда "/usr/5bin/ls", размер блока составляет 512 байт.

Позвольте мне запутать вас еще немного. Ниже показаны две версии ls в ОС Solaris:

```
% /usr/bin/ls -sl файл
14 -rwxr-xr-x 1 barnett 13443 25 июля 23:27 файл
% /usr/5bin/ls -sl файл
28 -rwxr-xr-x 1 barnett staff 13443 25 июля 23:27 файл
```

Можете ли вы угадать, какой размер блока должен быть указан, чтобы **find** напечатал этот файл? Правильная команда:

```
найти . -размер 27 -печать
```

потому что фактический размер составляет от 26 до 16 блоков по 512 байт каждый. Как вы можете видеть, "ls -s" не является точным числом для **поиска**. Вы можете поставить **c** после числа и указать размер в байтах,

Для поиска файлов с использованием диапазона размеров файлов перед числом можно указать знак минус или плюс. Знак минус означает "меньше, чем", а знак плюс означает "больше, чем". В следующем примере перечислены все файлы, размер которых превышает 10 000 байт, но не превышает 32 000 байт:

```
найти . -размер +10000 c -размер -32000c -печать
```

Когда задано более одного определителя, оба должны быть истинными.

Поиск старых файлов

Если вы хотите найти файл, которому 7 дней, используйте параметр **-mtime**:

```
Найти. -mtime 7 -печать
```

Альтернативный способ - указать диапазон времени:

```
Найти. -mtime +6 -mtime -8 -печать
```

Mtime - это время последнего изменения файла. Вы также можете думать об этом как о времени создания файла, поскольку Unix не различает создание и модификацию. Если вы хотите найти файлы, которые не использовались, проверьте время доступа с помощью аргумента **-atime** . Команда для перечисления всех файлов, которые не были прочитаны в течение тридцати или более дней,

```
Найти. -введите f -atime +30 -печать
```

Трудно найти каталоги, к которым не было доступа, потому что команда **find** изменяет время доступа к каталогу.

С каждым файлом связано другое время, называемое **ctime**, доступ к которому осуществляется с помощью параметра **-ctime**. Это значение будет иметь более свежее значение, если владелец, группа, разрешение или количество ссылок изменены, а сам файл - нет. Если вы хотите выполнить поиск файлов с определенным количеством ссылок, используйте опцию **-ссылки**.

Поиск файлов по разрешению

Поиск может искать файлы с определенным разрешением. Для этих разрешений используется восьмеричное число. Строка **rw-rw-r--** указывает, что у вас и членов вашей группы есть права на чтение и запись, в то время как у всего мира есть права только на чтение. Те же права доступа, выраженные в виде восьмеричного числа, равны **664**. Чтобы найти все файлы "*.o" с указанным выше разрешением, используйте:

```
Найти. -имя *.o -пермь 664 -печать
```

Если вы хотите узнать, есть ли у вас какие-либо каталоги с мировым разрешением на запись, используйте:

```
Найти. -тип d -пермь 777 -печать
```

Это соответствует только точной комбинации разрешений. Если вы хотите найти все каталоги с групповым разрешением на запись, существует несколько комбинаций, которые могут совпадать. Вы могли бы перечислить каждую комбинацию, но **поиск** позволяет указать шаблон, который может быть побитовым **и** редактироваться с правами доступа к файлу. Просто поставьте знак минус перед восьмеричным значением. Бит разрешения групповой записи равен восьмеричной цифре 20, поэтому следующее отрицательное значение:

```
Найти. -пермь -20 -печать
```

будет соответствовать следующим общим разрешениям:

```
+-------------------------+
|Разрешение Восьмеричное значение |
+-------------------------+
|rwxrwxrwx 777 |
|rwxrwxr-x 775 |
|rw-rw-rw- 666 |
|rw-rw-r-- 664 |
|rw-rw---- 660 |
+-------------------------+
```

Если вы хотите найти файлы, которые вы можете выполнять (например, сценарии оболочки или программы), вы хотите сопоставить шаблон "--x--x--x", введя:

```
Найти. -пермь -100 -печать
```

Если аргумент **-perm** имеет знак минус, проверяются все биты разрешений, включая установленные биты идентификатора пользователя.

Владельцев и группы

Часто вам нужно искать файл, который имеет определенные разрешения и принадлежит определенному пользователю или группе. Это делается с помощью параметров поиска **-user** и **-group**. Чтобы найти все файлы, для идентификатора пользователя которых задано значение root, используйте:

```
Найти. -корень пользователя -пермь -4000 -печать
```

Чтобы найти все файлы, для идентификатора группы которых задано значение staff, используйте:

```
Найти. -персонал группы -пермь -2000 -печать
```

Вместо того, чтобы использовать имя или группу в **/etc/passwd** или **/etc/group**, вы можете использовать номер:

```
Найти. -пользователь 0 -пермь -4000 -распечатать
поиск . -группа 10 -пермь -2000 -печать
```

Часто, когда пользователь покидает сайт, его учетная запись удаляется, но его файлы остаются на компьютере. Системный менеджер может использовать **-nouser** или **-nogroup** для поиска файлов с неизвестным идентификатором пользователя или группы.

Поиск и команды

До сих пор, после того, как **find** нашел файл, все, что он сделал, это напечатал имя файла. Он может сделать гораздо больше, чем это, но синтаксис может стать сложным. Использование **xargs** экономит ваши умственные усилия, но это не всегда лучшее решение.

Если вам нужен рекурсивный список, выходные данные **find** могут быть переданы в **| xargs ls -l**, но более эффективно использовать встроенную опцию **-ls**:

```
Найти. -ls
```

Это похоже на команду:

```
Найти. -печать | xargs ls -gilds
```

Вы также можете использовать команду **ls -R**, но это было бы слишком просто.

Найти и Cpio

Find также понимает **cpio** и поддерживает команды **-cpio** и **-ncpio**:

```
найти . -depth -cpio >/dev/rmt0
найти . -depth -ncpio >/dev/rmt0
```

которые делают то же самое, что и

```
найти . -глубина печати | cpio -oB >/dev/rmt0
найти . -глубина печати | cpio -ocB >/dev/rmt0
```

Использование поиска для выполнения команд

Я уже обсуждал, как использовать **xargs** для выполнения команд. **Find** может выполнять команды напрямую. Синтаксис своеобразен, что является одной из причин, по которой я рекомендую **xargs**. Синтаксис

опции **-exec** позволяет выполнить любую команду, включая другую команду find. Если вы подумаете об этом на мгновение, вы поймете, что **find** нужен какой-то способ отличить команду, которую она выполняет, от ее собственных аргументов. Очевидный выбор - использовать тот же символ конца команды, что и в командной строке (т.Е. Точка с запятой). Поскольку оболочка обычно использует саму точку с запятой, необходимо "экранировать" символ обратной косой чертой или кавычками. Существует еще один специальный аргумент, который **find** трактует по-разному: **{}** . Эти два символа используются в качестве переменной, именем которой является найденный файл **find**. Не утруждайте себя перечитыванием последней строки. Пример пояснит использование. Следующий случай является тривиальным и использует параметр **-exec** для имитации параметра **"-print"**.

Найти. -exec echo {} ;

Оболочка C использует символы **{** и **}**, но не изменяет **{}**, поэтому нет необходимости заключать эти символы в кавычки. Однако точка с запятой должна быть заключена в кавычки. Вместо обратной косой черты можно использовать кавычки:

Найти. -exec echo {} ';'

поскольку оба передадут точку с запятой мимо оболочки в команду **find**. Как я уже говорил, **find** может даже вызывать **find** . Если вы хотите перечислить каждую символическую ссылку в каждом каталоге, принадлежащем группе "персонал", вы можете выполнить:

найти `pwd` -персонал группы -exec find {} -введите l -печать ;

Для поиска всех файлов с групповым разрешением на запись и удаления разрешения можно использовать

Найти. -пермь -20 -exec chmod g-w {} ;

или

Найти. -пермь -20 -печать | xargs chmod g-w

Разница между **-exec** и **xargs** неуловима. Первый запускает программу один раз для каждого файла, в то время как **xargs** может обрабатывать несколько файлов в каждом процессе. Однако у **xargs** могут возникнуть проблемы с файлами, содержащими встроенные пробелы.

Иногда люди создают странный файл, который они не могут удалить. Это может быть вызвано случайным созданием файла с пробелом или некоторым управляющим символом в имени. **Найти** и **-exec** может удалить этот файл, в то время как **xargs** не смог. В этом случае

используйте **ls -il** для перечисления файлов и i-узлов и используйте параметр **-inum** с **-exec** для удаления файла:

```
Найти. -inum 31246 -exec rm [] ';'
```

Если вы хотите, вы можете использовать **-ok**, который выполняет то же самое, что и **-exec** , за исключением того, что программа сначала попросит вас подтвердить действие перед выполнением команды. Рекомендуется соблюдать осторожность при использовании **find**, потому что ошибка программы может привести к катастрофе. Если вы сомневаетесь, используйте **echo** в качестве команды. Или отправьте выходные данные в файл и проверьте файл, прежде чем использовать файл в качестве входных данных для **xargs**. Так я обнаружил, что **find** может использовать только один **{}** в аргументах **-exec** . Я хотел переименовать некоторые файлы, используя "-exec mv {} {}.orig", но я узнал, что мне нужно написать сценарий оболочки, который я сказал **find** для выполнения.

Сравнение файлов

Всякий раз, когда я обновлялся до новой версии Unix, одной из распространенных проблем было сохранение всех изменений, внесенных в стандартную версию Unix. Ранее я делал **ls -lt** в каждом каталоге, а затем проверял дату изменения. Файлы, которые были изменены, имеют явно более новую дату, чем исходные программы. Тем не менее, поиск каждого изменения был утомительным, так как нужно было искать в десятках каталогов.

Лучшим решением является создание файла в качестве первого шага при обновлении. Я обычно вызываю этот **FirstFile** . **Find** имеет опцию **-newer**, которая проверяет каждый файл и сравнивает дату изменения с более новым файлом. Если затем вы хотите составить список всех файлов в **/usr**, которые необходимо сохранить при обновлении операционной системы, используйте:

```
найти /usr -более новый /usr/FirstFile -печать
```

Затем это можно использовать для создания файла **tar** или **cpio**, который будет восстановлен после обновления.

Выражения

Поиск позволяет использовать сложные выражения. Чтобы отменить тест, поставьте **!**перед опцией. Поскольку оболочка C интерпретирует эту команду, она должна быть экранирована. Чтобы найти все файлы того же возраста или старше, чем "FirstFile", используйте

найти /usr ! -более новый /Первый файл -печать

Функция "и" выполняется параметром **-a**. Это не требуется, так как два или более параметров отображаются **И** редактируются автоматически. Функция "или" выполняется с помощью опции **-o**. Скобки можно использовать для добавления приоритета. Они также должны быть экранированы. Если вы хотите распечатать объектные файлы и файлы "a.out", которые старше 7 дней, используйте:

найти . (-name a.out -o -name *.o) -mtime 7 -печать

Удержание поиска от того, чтобы зайти слишком далеко

Наиболее болезненным аспектом большой среды NFS является предотвращение доступа к файлам на неработающих серверах NFS. **Поиск** особенно чувствителен к этому, потому что очень легко получить доступ к десяткам машин с помощью одной команды. Если **find** попытается исследовать файловый сервер, который не работает, время ожидания истечет. Важно понимать, как не допустить, чтобы **поиск** зашел слишком далеко.

Важным вариантом в этом случае является **-чернослив**. Этот вариант сбивает людей с толку, потому что он всегда верен. У него есть важный побочный эффект. Если просматриваемый файл является каталогом, он не будет перемещаться по каталогу. Вот пример, в котором перечислены все файлы в каталоге, но не просматриваются файлы в подкаталогах верхнего уровня:

найти * -тип f -печать -o -тип d -обрезка

Это выведет все обычные файлы и сократит поиск по всем каталогам. Для печати файлов, за исключением файлов, находящихся в директориях управления исходным кодом, используйте:

Найти. -print -o -name SCCS -чернослив

Если опция **-o** исключена, каталог SCCS будет напечатан вместе с другими файлами.

Другой полезной комбинацией является использование **-prune** с **-fstype** или **-xdev**. **Fstype** проверяет типы файловых систем и ожидает аргумент типа **nfs** или **4.2**. Последнее относится к файловой системе, представленной в выпуске 4.2 дистрибутива Berkeley Software. Чтобы ограничить **поиск** файлами только на локальном диске или дисках, используйте предложение **-fstype 4.2 -prune** или **-o -fstype nfs -prune**. Если вам нужно ограничить поиск одним конкретным разделом диска,

используйте **-xdev**, более поздний вариант очень полезен, если вы хотите помочь перегруженному разделу диска и хотите искать все файлы размером более 40 блоков на текущем разделе диска;

```
найти . -размер -40 -xdev -print
```

Быстрый поиск

В **find** появилась новая функция, которая значительно улучшает поиск файлов. Обычно это работает только в том случае, если вы ищете файл на локальном диске. Системный менеджер должен запускать скрипт с именем **updatedb** один раз в ночь, используя **cron**. Это создает базу данных с именем **/usr/lib/find/find.codes**.

Когда **find** выполняется с одним аргументом, он ищет файл, используя базу данных, и сообщает, где находится файл. Вы можете создать эту базу данных самостоятельно, если хотите, так как **find** будет искать базу данных в любом месте, определенном переменной **среды** FCODES.

Спасибо

Благодаря следующему:
Феликс Гартсман

Этот документ был переведен troff2html версии 0.21 22 сентября 2001 года, а затем отредактирован вручную, чтобы сделать его совместимым с: