MNorin.com

Блог про Linux, Bash и другие информационные технологии

Inotify в bash: ловим изменения файловой системы

Inotify – это подсистема ядра Linux, которая позволяет отслеживать изменения файловых систем. Использование этой подсистемы позволяет выполнять определенные действия в том случае если вы создали файл, что-то в него записали, открыли, закрыли, удалили и так далее. Использование этого механизма позволяет избавиться от необходимости вставлять в скрипты проверки с таймаутами при помощи команды sleep. Это упрощает логику скрипта, поскольку нам нужно просто ждать информацию об изменениях файловой системы, а при получении информации о том, что что-то изменилось, выполнить необходимые действия. Давайте рассмотрим пример скрипта, использующего inotify.

Программы inotifywait и inotifywatch

Эти программы как раз и помогут нам обеспечить необходимый функционал отслеживания изменений файловой системы. В Debian и Ubuntu они входят в пакет inotify-tools. Установка пакета:

```
apt-get install inotify-tools
```

Как работает inotifywatch

Программа inotifywatch собирает статистику доступа к файловой системе при помощи inotify. По окончании работы программа выводит статистическую информацию в виде таблицы, включающей информацию об общем количестве событий, о количестве событий каждого типа и файле, для которого произошло событие. У неё есть ряд параметров, которые можно использовать для изменения параметров работы программы.

-h, –help	Вывод информации по использованию
-v, –verbose	Выводить дополнительную информацию на стандартный поток ошибок во время работы
-r, –recursive	Отслеживать изменения рекурсивно для директории, переданной в качестве аргумента командной строки. Если внутри отслеживаемой директории будут созданы поддиректории во время работы программы, они автоматически будут отслеживаться. Если для отслеживания выбрана корневая директория или директория с большим количеством поддиректорий и файлов, установка отслеживания всех элементов файловой системы может занять некоторое время, в течение которого события получены не будут. Кроме того, возможно достижение максимального количества объектов файловой системы, разрешенных для отслеживания каждому пользователю. Значение по умолчанию 8192, и оно может быть увеличено записью нового значения в /proc/sys/fs/inotify/max_user_watches: echo «51200» > /proc/sys/fs/inotify/max_user_watches
–exclude <шаблон>	Исключить из отслеживания файлы с именами, совпадающими с шаблоном. Для шаблона используются расширенные регулярные выражения POSIX. Регистр при использовании данного параметра **учитывается**.
–excludei <шаблон>	Исключить из отслеживания файлы с именами, совпадающими с шаблоном. Для шаблона используются расширенные регулярные выражения POSIX. Регистр при использовании данного параметра **игнорируется**.
@<файл>	Во время рекурсивного слежения за директорией исключить указанный файл из отслеживания. Если для файла указаны одновременно включение в список для отслеживания и исключение из отслеживания, он будет отслеживаться. Для файла можно использовать как относительный путь, так и абсолютный. Если имя файла включает символ «@», используйте абсолютный путь.
–fromfile <файл>	Читать список файлов, которые будут отслеживаться и игнорироваться, из файла, каждое имя файла должно начинаться с новой строки. Если имя файла начинается с символа «@», отслеживание для него будет отключено. Если в качестве имени файла указать символ «-» (минус), то список файлов будет считываться со стандартного потока ввода. Этот параметр имеет смысл использовать, когда нужно отслеживать большое количество файлов, и их список неудобно передавать как аргументы командной строки.
-z, –zero	Выводить колонки и строки таблицы даже в том случае, когда они пусты. По умолчанию пустые строки и колонки не выводятся.
-t <секунды>, –timeout <секунды>	Работать только в течение указанного количества секунд. Если этот параметр не указан, программа будет работать до получения сигнала на прерывание работы, например, по нажатию клавиш Ctrl+C
-e <событие>, –event <событие>	Отслеживать только события указанных типов. Этот параметр может быть указан более одного раза. Если он не указан, будут отслеживаться события всех типов

-a <событие>, –ascending <событие>	Сортировать вывод по возрастанию количества событий для указанного типа событий. Типы событий, по которым можно осуществлять сортировку, включают «total» и события, которые перечислены ниже, за исключением «move» и «close» (вместо них нужно указывать «moved_to», «moved_from», «close_write» и «close_nowrite»). По умолчанию сортировка производится по убыванию по полю «total».
-d <событие>, –descending <событие>	Сортировать вывод по уменьшению количества событий для указанного типа событий. Типы событий, по которым можно осуществлять сортировку, включают «total» и события, которые перечислены ниже, за исключением «move» и «close» (вместо них нужно указывать «moved_to», «moved_from», «close_write» и «close_nowrite»). По умолчанию сортировка производится по убыванию по полю «total».

Типы событий для отслеживания:

access	Отслеживаемый файл или файл в отслеживаемой директории был прочитан
modify	Отслеживаемый файл или файл в отслеживаемой директории был записан
attrib	Метаданные отслеживаемого файла или файла в отслеживаемой директории были изменены. Сюда включаются изменения времени доступа и изменения, права доступа, расширенные атрибуты и так далее.
close_write	Отслеживаемый файл или файл в отслеживаемой директории был закрыт после открытия его в режиме записи. Это не говорит о том, что в файл были записаны какие-то данные.
close_nowrite	Отслеживаемый файл или файл в отслеживаемой директории был закрыт после открытия в режиме только для чтения
close	Отслеживаемый файл или файл в отслеживаемой директории был закрыт, при этом неважно, в каком режиме он был открыт. Имейте в виду, что это событие рализовано просто прослушиванием обоих событий close_write и close_nowrite, поэтому все события будут выведены именно как одно из них, а не как CLOSE. Этот тип скорее как сокращение для двух других типов событий
open	Отслеживаемый файл или файл в отслежваемой директории был открыт
moved_to	Файл или директория были перемещены в отслеживаемую директорию. Это событие возникает даже в том случае, когда файл был перемещен в пределах одной и той же директории
moved_from	Файл или директория были перемещены из отслеживаемой директории. Это событие возникает даже в том случае, когда файл был перемещен в пределах одной и той же директории
move	Файл или директория были перемещены из отслеживаемой директории или в неё. Это событие реализовано аналогично событию CLOSE, то есть, скорее как псевдоним для двух типов событий, moved_to и moved_from,поэтому все события будут выведены именно как одно из них, а не как MOVE
move_self	Отслеживаемый файл или директорию были перемещены. После возникновения этого события файл или директория больше не отслеживается.
create	Файл или директория были созданы в отслеживаемой директории

delete	Файл или директория были удалены из отслеживаемой директории
delete_self	Отслеживаемый файл или директория были удалены. После этого события файл или директория больше не отслеживается. Обратите внимание, что это событие может произойти, даже если оно явно не слушалось.
unmount	Файловая система, накоторой находится отслеживаемый файл или директория, была отмонтирована. Поле возникновения этого события файл или директория перестает отслеживаться.Обратите внимание, что это событие может произойти, даже если оно явно не слушалось.

Пример работы программы inotifywatch:

```
$ inotifywatch -r -t 300 /var/cache
Establishing watches...
Finished establishing watches, now collecting statistics.
total  attrib  open  moved_to  filename
182    1       1     180       /var/cache/apt/archives/
1      0       1     0         /var/cache/apt/
```

Как работает inotifywait

Программа inotifywait работает несколько иначе. Она ждет возникновения события и выводит информацию об этом событии сразу, не дожидаясь сигнала прерывания.

Параметры inotifywait очень похожи на параметры inotifywatch:

-h, –help	Вывод информации по использованию
-r, –recursive	Отслеживать изменения рекурсивно для директории, переданной в качестве аргумента командной строки. Если внутри отслеживаемой директории будут созданы поддиректории во время работы программы, они автоматически будут отслеживаться. Если для отслеживания выбрана корневая директория или директория с большим количеством поддиректорий и файлов, установка отслеживания всех элементов файловой системы может занять некоторое время, в течение которого события получены не будут. Кроме того, возможно достижение максимального количества объектов файловой системы, разрешенных для отслеживания каждому пользователю. Значение по умолчанию 8192, и оно может быть увеличено записью нового значения в /proc/sys/fs/inotify/max_user_watches: echo «51200» > /proc/sys/fs/inotify/max_user_watches

–exclude <шаблон>	Исключить из отслеживания файлы с именами, совпадающими с шаблоном. Для шаблона используются расширенные регулярные выражения POSIX. Регистр при использовании данного параметра **учитывается**.
–excludei <шаблон>	Исключить из отслеживания файлы с именами, совпадающими с шаблоном. Для шаблона используются расширенные регулярные выражения POSIX. Регистр при использовании данного параметра **игнорируется**.
@<файл>	Во время рекурсивного слежения за директорией исключить указанный файл из отслеживания. Если для файла указаны одновременно включение в список для отслеживания и исключение из отслеживания, он будет отслеживаться. Для файла можно использовать как относительный путь, так и абсолютный. Если имя файла включает символ «@», используйте абсолютный путь.
–fromfile <файл>	Читать список файлов, которые будут отслеживаться и игнорироваться, из файла, каждое имя файла должно начинаться с новой строки. Если имя файла начинается с символа «@», отслеживание для него будет отключено. Если в качестве имени файла указать символ «-» (минус), то список файлов будет считываться со стандартного потока ввода. Этот параметр имеет смысл использовать, когда нужно отслеживать большое количество файлов, и их список неудобно передавать как аргументы командной строки.
-t <секунды>, –timeout <секунды>	Работать только в течение указанного количества секунд. Если этот параметр не указан, программа будет работать до получения сигнала на прерывание работы, например, по нажатию клавиш Ctrl+C
-e <событие>, –event <событие>	Отслеживать только события указанных типов. Этот параметр может быть указан более одного раза. Если он не указан, будут отслеживаться события всех типов
-m, –monitor	Выполняться постоянно. По умолчанию программа завершает свою работу после появления первого события
-o, –output <файл>	Выводить события в файл вместо стандартного потока вывода
-s, –syslog	Выводить ошибки в системный модуль syslog вместо вывода в стандартный поток ошибок
-d, –daemon	То же самое, что и –monitor, но запись событий производится в файл, указанный в параметре –output. Вывод ошибок производится так же, как при использовании параметра –syslog
-q, –quiet	Если этот параметр указан один раз, программа будет выводить меньше информации. В частности, не будет выводиться сообщение о том, что все соединения для отслеживания установлены. Если этот параметр указан дважды, программа не будет выводить никакой информации вообще, за исключением случаев возникновения фатальной ошибки.

-c, –csv	Вывод в формате CSV (comma-separated values). Такой формат вывода удобно использовать в тех случаях, когда имена файлов могут содержать пробелы, потому что в таком случае разбивать информацию на поля по пробелам небезопасно.
–format <формат>	Вывод в формате, определенном пользователем, с использованием синтаксиса, похожего на тот, который использует printf. Длина строки вывода информации о событии ограничена примерно 4000 символов и строки будут укорочены до этой длины. Поддерживаются следующие шаблоны: 1) %w – Этот шаблон будет заменен именем отслеживаемого файла, для которого произошло событие 2) %f – Когда событие возникает при отслеживании директории, этот шаблон будет заменен именем файла, для которого произошло событие. В противном случае он будет заменен пустой строкой. 3) %e – Этот шаблон заменяется событием или событиями, разделенными запятыми, которые произошли. 4) %Xe – Заменяется событиями, которые возникли, разделенными символом, который в данном шаблоне будет стоять на месте «X». 5) %T – Заменяется текущим временем в формате, определенным параметром –timefmt, который должен быть задан в видее, подходящем для передачи функции strftime.
–timefmt <формат>	Установить формат вывода даты и времени в виде, подходящем для передачи функции strftime для использования совместно с шаблоном «%T» параметра «–format». Более подробно формат строки шаблона даты и времени можно посмотреть командой «man 3 strftime»

События, которые отслеживаются, точно такие же, как для программы inotifywatch.

Пример работы программы inotifywait:

```
$ inotifywait /var/cache
Setting up watches.
Watches established.
/var/cache/ OPEN,ISDIR apt
```

Пишем скрипт, который использует inotify

Прежде всего определимся с условиями задачи, которую нам предстоит решить. Есть некоторые директории, куда пользователи загружают файлы (пусть это будут домашние директории пользователей, /home/ИМЯ-ПОЛЬЗОВАТЕЛЯ). Иногда пользователи эти файлы удаляют, поэтому нам нужно отслеживать

изменения и при появлении новых файлов (или новых версий уже существующих файлов) копировать их в директорию /backup/ИМЯ-ПОЛЬЗОВАТЕЛЯ.

Для реализации нам нужно отслеживать изменения файловой системы, на которой находится директория /home, определять, что произошло событие закрытия файла после записи в него, получать директорию, в которой произошло закрытие файла, имя самого файла, определять имя директории, в которую планируется копировать файл, создавать её, если она еще не существует, и, собственно, копировать этот файл в эту директорию.

```
#!/bin/bash

SRC_DIR="/home"
DST_DIR="/backup"

# Функция, которая будет выполнять необходимые действия
# В нашем случае копировать в другую директорию
make_action(){
    # Получаем директорию назначения
    DIR_TO_COPY_TO=${1/${SRC_DIR}/${DST_DIR}}
    # Создаем ее, если ее еще не существует
    mkdir -p $DIR_TO_COPY_TO
    # Копируем файл
    cp $1$2 $DIR_TO_COPY_TO
}

IFS='
'
# Отслеживаем закрытие файлов после записи
# Получаем вывод в нужном нам формате
inotifywait -e close_write --format '%w %f' -m -r $SRC_DIR |\
(
while read
do
    # Получаем имя директории
    DIR=$(echo $REPLY | cut -f 1 -d' ')
    # Получаем имя файла
    FILE=$(echo $REPLY | cut -f 2 -d' ')
    # Передаем имена директории и файла в функцию
    make_action $DIR $FILE
done
)
```

Как видите, всё достаточно просто. Если необходимо работать с файлами, имена которых содержат пробелы, вы можете использовать параметр «–csv» и разделять поля по запятой, а не по пробелу, как в примере, или добавить какой-то другой символ в строку шаблона вывода. Основной принцип работы при этом не изменится.

Похожие посты:

- Параллельное выполнение в bash
- Скрипт, который работает и в Linux, и в Windows
- Проверка сертификата сервера из bash
- Объединение нескольких файловых систем без создания массива
- Система мониторинга на bash'е
- Bash. Как обойтись без goto
- Использование образов дисков VDI в Linux
- Linux: управление квотами дискового пространства
- etckeeper: сохранение системных настроек
- Сохраняем входящую и исходящую почту в postfix

Запись опубликована 18.12.2016 [https://mnorin.com/inotify-v-bash.html] в рубрике Статьи с метками bash, debian, inotify, inotifywait, inotifywatch, linux, ubuntu.

Inotify в bash: ловим изменения файловой системы: 4 комментария

A
22.06.2018 в 10:02

А как мониторить /proc ?

2 Автор записи 10:36

Тут зависит от того, что именно и зачем. Можно побольше информации?

foxiys
23.01.2020 в 02:22

Можно так:

```
#!/bin/bash
watchdir=/proc
logfile=/tmp/change_proc.log
while : ; do
inotifywait $watchdir|while read path action file; do
ts=$(date +»%C%y%m%d%H%M%S»)
echo «$ts :: file: $file :: $action :: $path»>>$logfile
done
done
exit 0
```

0 Автор записи 10:20

И как оно, работает?

Обсуждение закрыто.