VADEMECUM INOTIFY

premessa: partiamo come base da una fedora core 15 correttamente installata, in modalita' minimale , abilitando i repository aggiuntivi in fase di installazione.
Premessa numero 2: non vi e' nessuna pretesa di completezza in questo how to, e' soltanto la somma dei mei appunti. Scrivo tutto questo solo per poter consultare qualcosa “in fretta” in un futuro.

a) installare inotify-tools

```
yum install inotify-tools
```

b) installare il servizio incron

```
yum install incron
```

c) creare il file /etc/incrond.allow , all'interno del quale inseriremo gli utenti abilitati all'utilizzo del servizio di notifica. I nominativi degli utenti devono essere uno per riga, senza spazi aggiuntivi oltre al nome. Ipotizziamo che sia l'utente " tux " ad utilizzare inotify. tux Avra' gia' un account disponibile.

d) abilitiamo il servizio incrond nei vari runlevels

```
chkconfig --level 123456 incrond on
```

e) login come utente tux

f) impostiamo i dettagli , in pratica utilizziamo il comando incrontab (ATTENZIONE!!! I CAMPI ALL'INTERNO DEL FILE CHE L'UTENTE tux STA EDITANDO, DEVONO ESSERE SEPARATI DA UNO SPAZIO E NON DA UN TAB. SE NON RISPETTIAMO QUESTA SEMPLICE DIRETTIVA IL TUTTO NON FUNZIONA)

```
incrontab -e
```

```
/home/tux IN_CREATE /bin/comando_personalizzato
```

Ogni volta che viene creata una directory oppure viene creato

un file nella home dell'utente tux, inotify manda in esecuzione lo script /bin/comando personalizzato. L'esempio spiega bene che dopo aver lanciato il comando " incrontab -e " occorre specificare tre campi separati da uno spazio vuoto (non tab) :

a) directory da monitorare
b) evento da controllare
c) comando da eseguire

Ecco un elenco parziale degli eventi che inotify puo' monitorare:

IN_ACCESS File was accessed (read)
IN_ATTRIB Metadata changed (permissions, timestamps, extended attributes, etc.)
IN_CLOSE_WRITE File opened for writing was closed
IN_CLOSE_NOWRITE File not opened for writing was closed
IN_CREATE File/directory created in watched directory
IN_DELETE File/directory deleted from watched directory
IN_DELETE_SELF Watched file/directory was itself deleted
IN_MODIFY File was modified
IN_MOVE_SELF Watched file/directory was itself moved
IN_MOVED_FROM File moved out of watched directory
IN_MOVED_TO File moved into watched directory
IN_OPEN File was opened
IN_ALL_EVENTS = vedi pagine man

Nel terzo campo possiamo usare della variabili, come da questo piccolo elenco

$@ – Il path monitorato

$# – Il file sul quale si è scatenato l'evento

$% – L'evento, in forma testuale, che si è verificato

$& – L'evento, in forma numerica, che si è verificato

$$ – Il carattere $

Ecco alcuni esempi possibili all'interno del file incrontab:

/tmp IN_ALL_EVENTS abcd $@/$# $%

/usr/bin IN_ACCESS,IN_NO_LOOP abcd $#

/home IN_CREATE /usr/local/bin/abcd $#

/var/log 12 abcd $@/$#

Il primo esempio controlla tutti gli eventi possibili sulla directory /tmp. Quando un evento si verifica viene avviata l'applicazione chiamata “abcd”, con il path del file monitorato come primo argomento, e come secondo argomento viene utilizzato l'evento flag in formato numerico.

Il secondo esempio controlla l'accesso in lettura sulla directory /usr/bin. L'applicazione “abcd” viene lanciata. Il nome del file (senza path) monitorato viene utilizzato come argomento.

Il terzo esempio e' usato per controllare la creazione di files o directory nella directory /home . Quando cio' accade viene lanciata l'applicazione “abcd” con il path assoluto del file creato (o della directory creata) come argomento.

L'ultimo esempio mostra l'utilizzo come indicare gli eventi da controllare in formato numerico anziche' in formato letterale. I valori “ 12 “ indicano rispettivamente IN_ATTRIB e IN_CLOSE_WRITE .

Seguono alcune pagine man non tradotte.

incrond(8) incron documentation

NAME
incrond - inotify cron (incron) daemon

SYNOPSIS
incrond [-f file] [-n | -k]

DESCRIPTION
The inotify cron daemon (incrond) is a daemon which monitors filesystem events and executes commands
defined in system and user tables. It's use is generally similar to cron(8).

incrond can be started from /etc/rc, /etc/rc.local and so on. It daemonizes itself (returns immediately)
and doesn't need to be started with & and through nohup(1). It can be run on foreground too.

incrond uses two categories of tables incrontab(5). System tables are usually located in /etc/incron.d
and are maintained outside of incron (e.g. by various applications). These tables work on root rights
level and thus any file may be watched and commands are executed with root privileges.

User tables are located in /var/spool/incron by default and have names based on user accounts. These
tables use users' access rights, thus only files which the user may access are watched. Commands are
executed with users' privileges.

If a table (incrontab) is changed incrond reacts immediately and reloads the table. Currently running
child processes (commands) are not affected.

There are two files determining whether an user is allowed to use incron. These files have very simple
syntax - one user name per line. If /etc/incron.allow exists the user must be noted there to be allowed
to use incron. Otherwise if /etc/incron.deny exists the user must not be noted there to use incron. If
none of these files exists there is no other restriction whether anybody may use incron. Location of
these files can be changed in the configuration.

The daemon itself is currently not protected against looping. If a command executed due to an event
causes the same event it leads to an infinite loop unless a flag mask containing IN_NO_LOOP is speciâoր
fied. Please beware of this and do not allow permission for use incron to unreliable users.

-n (or --foreground) option causes running on foreground. This is useful especially for testing, debugâoր
ging and optimization.

-k (or --kill) option terminates a running instance of incrond.

-f <FILE> (or --config=<FILE>) option specifies another location for the configuration file
(/etc/incron.conf is used by default).

Environment variables: For system tables, the default (the same as for incrond itself) environment variâo
able set is used. The same applies to root's table. For non-root user tables, the whole environment is
cleared and then only these variables are set: LOGNAME, USER, USERNAME, SHELL, HOME and PATH. The variâo
ables (except PATH) take values from the user database (e.g. /etc/passwd). The PATH variable is set to
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin.

SEE ALSO
incrontab(1), incrontab(5), incron.conf(5)

BUGS
incrond is currently not resistent against looping. Recursive monitoring (whole subtrees) has not been
implemented yet.

AUTHOR
Lukas Jelinek <lukas@aiken.cz> (please report bugs to http://bts.aiken.cz or <bugs@aiken.cz>).

COPYING
This program is free software. It can be used, redistributed and/or modified under the terms of the GNU
General Public License, version 2.

incrontab(1) incron documentation

NAME
incrontab - table manipulator for inotify cron (incron)

SYNOPSIS
incrontab [-u user] [-f config] file

incrontab [-u user] [-f config] [-l | -r | -e | -t | -d]

DESCRIPTION
incrontab is a table manipulator for the inotify cron (incron) system. It creates, removes, modifies and
lists user tables (incrontab(5)).

Each user (including system users even they haven't home directories) has an incron table which can't be
manipulated directly (only root can effectively change these tables and is NOT recommended to do so).

All informational messages of this program are printed to the standard error output (stderr).

If /etc/incron.allow exists only users listed here may use incron. Otherwise if /etc/incron.deny exists
only users NOT listed here may use incron. If none of these files exists everyone is allowed to use
incron. (Important note: This behavior is insecure and will be probably changed to be compatible with
the style used by ISC Cron.) Location of these files can be changed in the configuration.

The first form of this command imports a file, validates it and stores to the table. "-" can be used for
loading from the standard input.

-u (or --user) option overrides the current (real) user to the given one. This option is intended for
manipulation with system users' tables (such as apache, postfix, daemon etc.). It can be used only if
the current user has root's effective rights.

-l (or --list) option causes the current table is printed to the standard output.

-r (or --remove) option causes the current table (if any) is permanently remove without any warning or
confirmation. Use with caution!

-e (or --edit) option causes executing an editor for editting the user table (see below for the informaâo
tion about editor selection). You can edit your incron table now. If the table is changed it stores the
modified version.

-t (or --types) option causes the list of supported event types (delimited by commas) is printed to the

standard output. This feature is intended for front-end applications to find out which event types was
compiled in.

-d (or --reload) option causes reloading the current table by incrond(8). It is done through "touching"
the table (writing into it without modifying it). This feature is intended e.g. for creating watches on
newly created files (with already existing rules) or for rearming IN_ONESHOT watches.

-f <FILE> (or --config=<FILE>) option specifies another location for the configuration file
(/etc/incron.conf is used by default). This feature requires root privileges.

There is a few complex algorithm how to determine which editor will be user for editting. If any of the
following rule succeeds the appropriate editor is used:

1. EDITOR environment variable

2. VISUAL environment variable

3. configuration value

4. etc/alternatives/editor

5. hard-wired editor (vim by default)

It's not recommended to use graphical editors (such as gVim, KEdit etc.) due to possible problems with
connecting to the X server.

SEE ALSO
incrond(8), incrontab(5), incron.conf(5)

AUTHOR
Lukas Jelinek <lukas@aiken.cz> (please report bugs to http://bts.aiken.cz or <bugs@aiken.cz>).

COPYING
This program is free software. It can be used, redistributed and/or modified under the terms of the GNU
General Public License, version 2.

inotifywait(1)

NAME

inotifywait - wait for changes to files using inotify

SYNOPSIS

inotifywait [-hcmrq] [-e <event>] [-t <seconds>] [--format <fmt>] [--timefmt <fmt>] <file> [...]

DESCRIPTION

inotifywait efficiently waits for changes to files using Linux's inotify(7) interface. It is suitable for waiting for changes to files from shell scripts. It can either exit once an event occurs, or continually execute and output events as they occur.

OUTPUT

inotifywait will output diagnostic information on standard error and event information on standard output. The event output can be configured, but by default it consists of lines of the following form:

watched_filename EVENT_NAMES event_filename

watched_filename

is the name of the file on which the event occurred. If the file is a directory, a trailing slash is output.

EVENT_NAMES

are the names of the inotify events which occurred, separated by commas.

event_filename
is output only when the event occurred on a directory, and in this case the name of the file
within the directory which caused this event is output.

By default, any special characters in filenames are not escaped in any way. This can make the
output of inotifywait difficult to parse in awk scripts or similar. The --csv and --format
options will be helpful in this case.

OPTIONS
-h, --help
Output some helpful usage information.

@<file>
When watching a directory tree recursively, exclude the specified file from being watched. The
file must be specified with a relative or absolute path according to whether a relative or absoâ
lute path is given for watched directories. If a specific path is explicitly both included and
excluded, it will always be watched.

Note: If you need to watch a directory or file whose name starts with @, give the absolute path.

--fromfile <file>
Read filenames to watch or exclude from a file, one filename per line. If filenames begin with @
they are excluded as described above. If <file> is `-', filenames are read from standard input.
Use this option if you need to watch too many files to pass in as command line arguments.

-m, --monitor
Instead of exiting after receiving a single event, execute indefinitely. The default behaviour
is to exit after the first event occurs.

-d, --daemon
Same as --monitor, except run in the background logging events to a file that must be specified

by --outfile. Implies --syslog.

-o, --outfile <file>
Output events to <file> rather than stdout.

-s, --syslog
Output errors to syslog(3) system log module rather than stderr.

-r, --recursive
Watch all subdirectories of any directories passed as arguments. Watches will be set up recurâo
sively to an unlimited depth. Symbolic links are not traversed. Newly created subdirectories
will also be watched.

Warning: If you use this option while watching the root directory of a large tree, it may take
quite a while until all inotify watches are established, and events will not be received in this
time. Also, since one inotify watch will be established per subdirectory, it is possible that
the maximum amount of inotify watches per user will be reached. The default maximum is 8192; it
can be increased by writing to /proc/sys/fs/inotify/max_user_watches.

-q, --quiet
If specified once, the program will be less verbose. Specifically, it will not state when it has
completed establishing all inotify watches.

If specified twice, the program will output nothing at all, except in the case of fatal errors.

--exclude <pattern>
Do not process any events whose filename matches the specified POSIX extended regular expression,
case sensitive.

--excludei <pattern>
Do not process any events whose filename matches the specified POSIX extended regular expression,

case insensitive.

-t <seconds>, --timeout <seconds>
Exit if an appropriate event has not occurred within <seconds> seconds. If <seconds> is zero (the
default), wait indefinitely for an event.

-e <event>, --event <event>
Listen for specific event(s) only. The events which can be listened for are listed in the EVENTS
section. This option can be specified more than once. If omitted, all events are listened for.

-c, --csv
Output in CSV (comma-separated values) format. This is useful when filenames may contain spaces,
since in this case it is not safe to simply split the output at each space character.

--timefmt <fmt>
Set a time format string as accepted by strftime(3) for use with the `%T' conversion in the
--format option.

--format <fmt>
Output in a user-specified format, using printf-like syntax. The event strings output are limâ○ர
ited to around 4000 characters and will be truncated to this length. The following conversions
are supported:

%w This will be replaced with the name of the Watched file on which an event occurred.

%f When an event occurs within a directory, this will be replaced with the name of the File which
caused the event to occur. Otherwise, this will be replaced with an empty string.

%e Replaced with the Event(s) which occurred, comma-separated.

%Xe Replaced with the Event(s) which occurred, separated by whichever character is in the place of
`X'.

%T Replaced with the current Time in the format specified by the --timefmt option, which should be a
format string suitable for passing to strftime(3).

EXIT STATUS

0 The program executed successfully, and an event occurred which was being listened for.

1 An error occurred in execution of the program, or an event occurred which was not being listened
for. The latter generally occurs if something happens which forcibly removes the inotify watch,
such as a watched file being deleted or the filesystem containing a watched file being unmounted.

2 The -t option was used and an event did not occur in the specified interval of time.

EVENTS

The following events are valid for use with the -e option:

access A watched file or a file within a watched directory was read from.

modify A watched file or a file within a watched directory was written to.

attrib The metadata of a watched file or a file within a watched directory was modified. This includes
timestamps, file permissions, extended attributes etc.

close_write
A watched file or a file within a watched directory was closed, after being opened in writeable
mode. This does not necessarily imply the file was written to.

close_nowrite
A watched file or a file within a watched directory was closed, after being opened in read-only
mode.

close A watched file or a file within a watched directory was closed, regardless of how it was opened.
Note that this is actually implemented simply by listening for both close_write and
close_nowrite, hence all close events received will be output as one of these, not CLOSE.

open A watched file or a file within a watched directory was opened.

moved_to
A file or directory was moved into a watched directory. This event occurs even if the file is
simply moved from and to the same directory.

moved_from
A file or directory was moved from a watched directory. This event occurs even if the file is
simply moved from and to the same directory.

move A file or directory was moved from or to a watched directory. Note that this is actually impleâo
mented simply by listening for both moved_to and moved_from, hence all close events received will
be output as one or both of these, not MOVE.

move_self

A watched file or directory was moved. After this event, the file or directory is no longer being
watched.

create A file or directory was created within a watched directory.

delete A file or directory within a watched directory was deleted.

delete_self
A watched file or directory was deleted. After this event the file or directory is no longer
being watched. Note that this event can occur even if it is not explicitly being listened for.

unmount
The filesystem on which a watched file or directory resides was unmounted. After this event the
file or directory is no longer being watched. Note that this event can occur even if it is not
explicitly being listened to.

EXAMPLES

Example 1

Running inotifywait at the command-line to wait for any file in the `test' directory to be accessed.

After running inotifywait, `cat test/foo' is run in a separate console.

```
% inotifywait test
Setting up watches.
Watches established.
test/ ACCESS foo
```

Example 2

A short shell script to efficiently wait for httpd-related log messages and do something appropriate.

```
#!/bin/sh
while inotifywait -e modify /var/log/messages; do
```

```
  if tail -n1 /var/log/messages | grep httpd; then
    kdialog --msgbox "Apache needs love!"
  fi
done
```

Example 3

A custom output format is used to watch `~/test'. Meanwhile, someone runs `touch ~/test/badfile; touch ~/test/goodfile; rm ~/test/badfile' in another console.

```
% inotifywait -m -r --format '%:e %f' ~/test
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
CREATE badfile
OPEN badfile
ATTRIB badfile
CLOSE_WRITE:CLOSE badfile
CREATE goodfile
OPEN goodfile
ATTRIB goodfile
CLOSE_WRITE:CLOSE goodfile
DELETE badfile
```

BUGS

There are race conditions in the recursive directory watching code which can cause events to be missed if they occur in a directory immediately after that directory is created. This is probably not fixable.

It is assumed the inotify event queue will never overflow.

AUTHORS

inotifywait is written and maintained by Rohan McGovern <rohan@mcgovern.id.au>.

inotifywait is part of inotify-tools. The inotify-tools website is located at: http://inotify-tools.sourceforge.net/

SEE ALSO

inotifywatch(1), strftime(3), inotify(7)

inotifywatch(1)

NAME

inotifywatch - gather filesystem access statistics using inotify

SYNOPSIS

inotifywatch [-hvzrqf] [-e <event>] [-t <seconds>] [-a <event>] [-d <event>] <file> [...]

DESCRIPTION

inotifywatch listens for filesystem events using Linux's inotify(7) interface, then outputs a summary
count of the events received on each file or directory.

OUTPUT

inotifywatch will output a table on standard out with one column for each type of event and one row for
each watched file or directory. The table will show the amount of times each event occurred for each
watched file or directory. Output can be sorted by a particular event using the -a or -d options.

Some diagnostic information will be output on standard error.

OPTIONS

-h, --help

Output some helpful usage information.

-v, --verbose

Output some extra information on standard error during execution.

@<file>

When watching a directory tree recursively, exclude the specified file from being watched. The
file must be specified with a relative or absolute path according to whether a relative or absoâ
lute path is given for watched directories. If a specific path is explicitly both included and
excluded, it will always be watched.

Note: If you need to watch a directory or file whose name starts with @, give the absolute path.

--fromfile <file>
Read filenames to watch or exclude from a file, one filename per line. If filenames begin with @
they are excluded as described above. If <file> is `-', filenames are read from standard input.
Use this option if you need to watch too many files to pass in as command line arguments.

-z, --zero
Output table rows and columns even if all elements are zero. By default, rows and columns are
only output if they contain non-zero elements. Using this option when watching for every event
on a lot of files can result in a lot of output!

--exclude <pattern>
Do not process any events whose filename matches the specified POSIX extended regular expression,
case sensitive.

--excludei <pattern>
Do not process any events whose filename matches the specified POSIX extended regular expression,
case insensitive.

-r, --recursive
Watch all subdirectories of any directories passed as arguments. Watches will be set up recurâ
sively to an unlimited depth. Symbolic links are not traversed. If new directories are created
within watched directories they will automatically be watched.

Warning: If you use this option while watching the root directory of a large tree, it may take
quite a while until all inotify watches are established, and events will not be received in this
time. Also, since one inotify watch will be established per subdirectory, it is possible that
the maximum amount of inotify watches per user will be reached. The default maximum is 8192; it
can be increased by writing to /proc/sys/fs/inotify/max_user_watches.

-t <seconds>, --timeout <seconds>
Listen only for the specified amount of seconds. If not specified, inotifywatch will gather staâo
tistics until receiving an interrupt signal by (for example) pressing CONTROL-C at the console.

-e <event>, --event <event>
Listen for specific event(s) only. The events which can be listened for are listed in the EVENTS
section. This option can be specified more than once. If omitted, all events are listened for.

-a <event>, --ascending <event>
Sort output ascending by event counts for the specified event. Sortable events include `total'
and all the events listed in the EVENTS section except `move' and `close' (you must use
`moved_to', `moved_from', `close_write' or `close_nowrite' instead). The default is to sort
descending by `total'.

-d <event>, --descending <event>
Sort output descending by event counts for the specified event. Sortable events include `total'
and all the events listed in the EVENTS section except `move' and `close' (you must use
`moved_to', `moved_from', `close_write' or `close_nowrite' instead). The default is to sort
descending by `total'.

EXIT STATUS

0 The program executed successfully.

1 An error occurred in execution of the program.

EVENTS

The following events are valid for use with the -e option:

access A watched file or a file within a watched directory was read from.

modify A watched file or a file within a watched directory was written to.

attrib The metadata of a watched file or a file within a watched directory was modified. This includes timestamps, file permissions, extended attributes etc.

close_write

A watched file or a file within a watched directory was closed, after being opened in writeable

mode. This does not necessarily imply the file was written to.

close_nowrite

A watched file or a file within a watched directory was closed, after being opened in read-only

mode.

close A watched file or a file within a watched directory was closed, regardless of how it was opened.

Note that this is actually implemented simply by listening for both close_write and

close_nowrite, hence all close events received will be output as one of these, not CLOSE.

open A watched file or a file within a watched directory was opened.

moved_to
A file or directory was moved into a watched directory. This event occurs even if the file is simply moved from and to the same directory.

moved_from
A file or directory was moved from a watched directory. This event occurs even if the file is simply moved from and to the same directory.

move A file or directory was moved from or to a watched directory. Note that this is actually implemented simply by listening for both moved_to and moved_from, hence all close events received will be output as one or both of these, not MOVE.

move_self
A watched file or directory was moved. After this event, the file or directory is no longer being watched.

create A file or directory was created within a watched directory.

delete A file or directory within a watched directory was deleted.

delete_self
A watched file or directory was deleted. After this event the file or directory is no longer being watched. Note that this event can occur even if it is not explicitly being listened for.

unmount
The filesystem on which a watched file or directory resides was unmounted. After this event the file or directory is no longer being watched. Note that this event can occur even if it is not explicitly being listened to.

EXAMPLE
Watching the `~/.beagle' directory for 60 seconds:

% inotifywatch -v -e access -e modify -t 60 -r ~/.beagle
Establishing watches...
Setting up watch(es) on /home/rohan/.beagle
OK, /home/rohan/.beagle is now being watched.
Total of 302 watches.
Finished establishing watches, now collecting statistics.
Will listen for events for 60 seconds.
total access modify filename
1436 1074 362
/home/rohan/.beagle/Indexes/FileSystemIndex/PrimaryIndex/
1323 1053 270
/home/rohan/.beagle/Indexes/FileSystemIndex/SecondaryIndex/
303 116 187
/home/rohan/.beagle/Indexes/KMailIndex/PrimaryIndex/
261 74 187 /home/rohan/.beagle/TextCache/
206 0 206 /home/rohan/.beagle/Log/
42 0 42 /home/rohan/.beagle/Indexes/FileSystemIndex/Locks/
18 6 12 /home/rohan/.beagle/Indexes/FileSystemIndex/
12 0 12 /home/rohan/.beagle/Indexes/KMailIndex/Locks/
3 0 3 /home/rohan/.beagle/TextCache/54/
3 0 3 /home/rohan/.beagle/TextCache/bc/
3 0 3 /home/rohan/.beagle/TextCache/20/
3 0 3 /home/rohan/.beagle/TextCache/62/
2 2 0
/home/rohan/.beagle/Indexes/KMailIndex/SecondaryIndex/

BUGS

There are race conditions in the recursive directory watching code which can cause events to be missed if they occur in a directory immediately after that directory is created. This is probably not fixable.
It is assumed the inotify event queue will never overflow.

AUTHORS

inotifywatch is written by Rohan McGovern <rohan@mcgovern.id.au>.

inotifywatch is part of inotify-tools. The inotify-tools website is located at: http://inotify-tools.sourceforge.net/

SEE ALSO

inotifywait(1), inotify(7)