How can I interrupt or debounce an inotifywait loop?

Asked 7 years, 11 months ago Modified 7 months ago Viewed 2k times

12

I have a little script that watches files for changes using `inotifywait` . When something changes, a batch of files are sent through a process (compiled, compressed, reorganised, etc) that takes about ten seconds to run.

Consider the following example:

```
touch oli-test
inotifywait -mq oli-test | while read EV; do sleep 5; echo "$EV"; done
```

If you run `touch oli-test` in another terminal a few times, you'll see that each loop completes before it moves on. That scenario is very real to me. If I forget to save a file while it's already processing, or notice a mistake, the events stack up and I'm waiting minutes.

It strikes me that there are two techniques that would make this workflow objectively better. I'm not sure what is easiest or best, so I'm presenting both:

1. **Interrupt** previous run-throughs, and restart immediately. The scripted process is currently just an inline set of commands. I could break them out to Bash functions, I'm not wild about breaking them further out than that.
2. **Debounce** the list of things waiting to be processed so that if five events happen at once (or while it's already processing), it only runs once more.

(Or both... because I'm certain there are cases where both would be useful)

I am also open to approaches that are different from `inotifywait` but they need to give me the same outcome and work on Ubuntu.

bash interrupt inotify debouncing inotifywait

Share Improve this question Follow

edited Apr 13, 2017 at 12:22
Community Bot
1 1

asked Jan 28, 2015 at 15:18
Oli
232k 62 217 295

Sorted by:

```
inotifywait -mq oli-test | while read EV; do
    jobs -p | xargs kill -9
    (
        # do expensive things here
        sleep 5  # a placeholder for compiling
        echo "$EV"
    ) &
done
```

Share Improve this answer Follow

edited Jan 28, 2015 at 16:14

answered Jan 28, 2015 at 15:53

Oli
232k 62 217 295

This looks like a good solution for what you are trying to do. – hek2mgl Jan 28, 2015 at 16:15

Anyway to silence the output of the pipe to `kill` ? Ive tried redirecting to /dev/null and no matter what type of redirect i've tried, I still see output. – surgiie Feb 25 at 6:23

@surgiie Untested but kill might be outputting to STDERR (fd/2). That's usually the case when you can't redirect something. Try subbing that line for `jobs -p | xargs kill -9 2>/dev/null` – Oli Feb 25 at 9:00

@Oli I tried this as well but was still seeing the output for kill : `./myscript: line 12:  542 Killed ( mycommand)` – surgiie Feb 25 at 21:40

2 @Marcel Ish. If there are other background jobs in *this bash session* then yes, they'll all get thrown on the bonfire. But try it out, `sleep 2000 &` in one terminal, `jobs -p` in another. It won't nuke other sessions' jobs. – Oli Mar 31 at 10:28

Here is a compact solution:

6

```
inotifywait -q -m -e modify -e create -e close_write --format "%w%f"
/etc/nginx/ |\
while read -r path; do
    echo $path changed
    echo "Skipping $(timeout 3 cat | wc -l) further changes"
    service nginx reload
done
```

The first `read` waits for a line of data, so this won't eat your CPU. The `timeout 3 cat` reads any further change notifications that come in over the next 3 seconds. Only then does `nginx` get reloaded.

Share Improve this answer Follow

answered Nov 12, 2021 at 16:08

Max Murphy
1,543 1 17 27

A version with pure bash (no external commands except inotifywait), if we really want millisecond precision. (We can do even nano second precision, but will limit maximum debounce period)

```
#!/bin/bash

getTime() {
   # EPOCHREALTIME has the EPOCH time in nanoseconds
   # (i.e: 1653121901,339206 )
   # We extract the milliseconds and seconds up to 9
   # digits precision.
   # Using the value above, this is "121901339"
   # (Those values fit neatly into a bash integer)

   [[ $EPOCHREALTIME =~ ([^,].....),(...).*$ ]] && \
       echo "${BASH_REMATCH[1]}${BASH_REMATCH[2]}"

   # (This way of pattern extraction avoids the use of sed on this cases.
   # Totally recommended)
}

lastRunTime=$(getTime)

# We avoid the pipe, so the main program always
# runs on the same process.
while read path event file; do

    currentTime=$(getTime)
    delta=$(( $currentTime - $lastRunTime ))

    if [[ "${delta}" -gt 1000 ]] ; then
        echo "run"
        lastRunTime=$(getTime)
    fi
done < <(inotifywait -mr ./web -e create -e delete -e modify)
# This way the extra process is the generator command, which is generally what
you
# want.
```

That said, if you are interested on the events you receive, this treats all events as if they were the same. You don't know which events happened during the debouncing period.

This can be solved by using an associative array to store all the different events between debounce periods.

```
#!/bin/bash

getTime() {
   [[ $EPOCHREALTIME =~ ([^,].....),(...).*$ ]] && echo
"${BASH_REMATCH[1]}${BASH_REMATCH[2]}"
}

declare -A Changes
lastRunTime=0
```



```
for change in "${!Changes[@]}" ; do
```

```
            echo "run $change";
        done
        Changes=()
        lastRunTime=$currentTime
    fi
done < <(inotifywait -mr ./web/  -e create -e delete -e modify)
```

Share Improve this answer Follow

edited May 21 at 10:23

answered May 21 at 9:16

FrameGrace

565 4 4

Perhaps change the regex to `([^,].....)[^[:digit:]](...).*$` to accommodate a decimal separator that isn't COMMA. – Robin A. Meade Oct 24 at 21:16

0

I write this solution, working only if bc (calculator) application is installed, because bash can't handle float numbers.

```
lastRunTime=$(date +'%H%M%S.%N')

inotifywait -mr ./web -e create -e delete -e modify | while read file event tm;
do

    currentTime=$(date +'%H%M%S.%N')
    delta=$(bc <<< "$lastRunTime - $currentTime")
    echo "$currentTime, $lastRunTime, $delta"

    if (( $(echo "$delta < -1.0" | bc -l) )); then
        echo "run"
        lastRunTime=$(date +'%H%M%S.%N')
    fi

done
```

Explain: here we set last run datetime as NOW, and next run allowed only if last run datetime < delta (in my case is -1.0).

Example:

```
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
120845.009293691, 120842.581019388, -2.428274303
run
120845.018643243, 120845.017585539, -.001057704
120845.026360234, 120845.017585539, -.008774695
```