Linux Fu: Easier File Watching

Originalartikel

Backup

<html> In an earlier installment of Linux Fu, I mentioned how you can use

```
inotifywait
```

to efficiently watch for file system changes. The comments had a lot of alternative ways to do the same job, which is great. But there was one very easy-to-use tool that didn’t show up, so I wanted to talk about it. That tool is

```
entr
```

. It isn’t as versatile, but it is easy to use and covers a lot of common use cases where you want some action to occur when a file changes. The program is dead simple. It reads a list of file names on its standard input. It will then run a command and repeat it any time the input files change. There are a handful of options we’ll talk about in a bit, but it is really that simple. For example, try this after you install

```
entr
```

with your package manager. Open two shell windows In one window, open your favorite editor to create an empty file named /tmp/foo and save it In the second window issue the command:

```
echo "/tmp/foo" | entr wc /tmp/foo
```

 Back in the first window (or your GUI editor) make some changes to the file and save it while observing the second window If you can’t find

```
entr
```

, you can download it from the website. Frequently, you’ll feed the output from

```
find
```

or a similar command to

```
entr
```

. <h2>What It Isn’t</h2> I had mentioned

```
incron
```

before as a way to attach actions to file changes. It also makes things easier, although perhaps not as easy as

```
entr
```

. As the name suggests,

```
incron
```

does for filesystem changes what

```
cron
```

does for time. That is, it causes some action to occur any time the specified file system changes happen. That survives a reboot or anything else short of you canceling it. This is very different from

```
entr
```

. With

```
entr
```

, the command runs like a normal application. As long as it is running, changes in the target files will trigger the specified action. When you stop the program, that’s the end of it. Usually, you’ll stop with a Control+C or a “q” character. Speaking of the keyboard, you can also press space to trigger the action manually, as though a file changed. So, unlike

```
incron
```

,

```
entr
```

is an interactive tool. You’ll want it running in the foreground. <h2>Options</h2> There are several command line options: -c – Clear screen before executing command -d – Track directories that do not start with “.” -p – Do not execute the command until an event occurs -r – Kill previously run command before executing command -s – Use shell on first argument The /_ placeholder gets the name of the first file that caused a trigger, although that doesn’t seem to work properly with -s. For example: <pre>find /tmp/t/data -name '*.txt' | entr cp /_ /tmp/archive</pre> When one of the .txt files changes, it will copy to

```
/tmp/archive
```

. The -d option has a peculiarity. With it you can use a directory name or a file name and the program will watch that directory along with any files. The file changes behave as normal. However, any new files in the directory will cause

```
entr
```

to stop. This lets you write things in a loop like this: <pre>while true; do

```
ls -d src/*.[ch] | entr -d make
```

done</pre> The loop ensures that

```
entr
```

is always looking at the right list of files. This will also cause an error exit if you delete one of the files. The

```
ls
```

command provides all the .c and .h files in the src directory. The command is smart enough to infer the directory, so you don’t need to set it explicitly. <h2>Missing Changes</h2> The -r option is good if you are running a program that persists — for example, you might use

```
kdiff3
```

to show the differences between a recently changed file and an original copy. This option causes

```
entr
```

to kill the program before starting a new one. Without this flag, it is possible, too, for

```
entr
```

to miss a file change. For example, make a file called foo and try this: <pre>echo foo | entr -ps 'echo change; sleep 20'</pre> In another shell, change

```
foo
```

. You’ll see the change message print on the original shell. If you wait 20 seconds you’ll see something like “bash returned exit code 0.” Now change

```
foo
```

again, but before you see the bash message, change it again. Once the 20 second timer expires,

```
entr
```

will go back to waiting and the new change will not cause a trigger! Exit

```
entr
```

 and start it again with the options -psr instead of -ps. Now do the same test again.

You’ll see that the change registers and the original bash script never completes. However, if there are no changes for 20 seconds, the last script will exit normally. <h2>Examples</h2> There are plenty of examples in the tool’s man page. For example: <pre>find src/ | entr -s 'make | head -n 20'</pre> This gets a list of all files in the source tree (including subdirectories) and when any change, you run

```
make
```

. The

```
head
```

command only shows the top 20 lines of output. A lot of editors make automatic backup of files, but if yours doesn’t it would be pretty simple to make an auto archive with

```
entr
```

, although, honestly, use

```
git
```

or something if you want real version control: <pre>echo testfile.txt | entr -s 'AFN=/tmp/testfile-$(date).txt; cp testfile.txt „$AFN“; zip -j archive.zip „$AFN“'</pre> This stores each version of testfile.txt in archive.zip along with a timestamp on the file name:
 It was tempting to use /_ in this script, but it doesn’t seem to work with the -s option very well. <h2>Embrace Change</h2> I doubt I’ll use

```
entr
```

as much as I use

```
incron
```

. However, for little one-off projects, it is pretty handy and I could see making some use of in that case. However, as usual, for any given task there are usually many ways to accomplish it. Having

```
entr
```

in your toolbox can’t hurt. There are still other ways. For example, last time, someone mentioned in the comments that

```
systemd
```

can take a “path unit” that can trigger when a file or directory appears or changes. This still uses

```
inotify
```

internally, so it is really just another wrapper. Still, if you like

```
systemd
```

, it is a consistent way to set up something similar to

```
incron
```

but under the tentacles of

```
systemd
```

. </html>