entr Run on file change

START HERE

`ls *.c | entr make`
Rebuild when a file changes

`ls -d . | entr -d <cmd>`
Also catch new files

`find src | entr -r <server>`
Restart a server on change

`Space / q`
Run now / quit

`ls | entr -c <cmd>`
Clear screen each run

`ls | entr -p <cmd>`
Wait for first change

`ls | entr -s '<a> && <b>'`
Run a shell string

`echo f | entr <cmd> /_`
Pass the changed file

File list ls, find, git ls-files → `stdin` → **entr** watches those files → `change` → **Utility** run or restart → `exit` → **Wait** for the next change

CONCEPTS

entr reads the file list from stdin, so it only watches what you pipe in. It never searches directories on its own.

Events that arrive while the command runs are merged into one by default, so saving twice mid-build gives one rerun. Add -a to react to all of them.

The list is read once at start, so new files are not noticed. With -d, entr exits with status 2 when a directory gains or loses an entry; wrap it in a loop to restart.

RUNNING

`<list> | entr <cmd> [args]`
Run a command whenever a listed file changes

`entr -r <cmd>`
Keep it running; restart on change
The child gets no stdin, so interactive programs do not work with -r

RUNNING (cont.)

`entr -c <cmd>`
Clear the screen before each run

`entr -p <cmd>`
Postpone: skip the first run

`entr -s '<shell string>'`
Run a string through $SHELL (pipes, &&)

`entr -cc <cmd>`
Clear screen and scrollback

`entr -z <cmd>`
Exit after the command finishes

`entr -rz <cmd>`
Restart on change, exit when it dies

`entr -n <cmd>`
Non-interactive: leave the TTY alone

`entr -a <cmd>`
React to events during the run too

`entr -x <cmd>`
Custom exit messages via an awk script

RUNNING (cont.)

`SHELL=<shell> entr -s ...`
Choose the shell for -s (default /bin/sh)

FEEDING THE LIST

`ls | entr <cmd>`
Watch the files in this directory

`ls *.py | entr <cmd>`
Watch files matching a glob

`find . -name '*.go' | entr <cmd>`
Watch a tree by pattern

`git ls-files | entr <cmd>`
Watch tracked files only

`fd -e rs | entr <cmd>`
Use fd for the list

`echo <file> | entr <cmd>`
Watch exactly one file

`find src tests | entr <cmd>`
Watch several directories

`git ls-files -co --exclude-standard | entr <cmd>`
Tracked and new unignored files

FEEDING THE LIST (cont.)

`find . -type f -not -path './.git/*' | entr <cmd>`
Skip a directory

`ls -d src src/*.rb | entr <cmd>`
List the directory too (for -d)

CHANGED FILE AND DIRECTORIES

`/_`
Placeholder: path of the file that changed
Only the first changed file; if several change at once you get one of them

`entr <cmd> /_`
Pass the changed file as an argument

`entr -s 'echo $0'`
In -s mode the changed file is $0

`entr -d <cmd>`
Watch parent dirs; exit 2 on new entry

CHANGED FILE AND DIRECTORIES (cont.)

`entr -dd <cmd>`
Also notice new directory entries
Names starting with a dot are ignored unless you list them

`while sleep 1; do <list> | entr -d <cmd>; done`
Rescan the list after a new file

KEYS AND ENVIRONMENT

Space
Run the command now
With -r this restarts the child

q / Ctrl+C
Quit entr

`ENTR_RESTART_SIGNAL=<sig>`
Signal for -r restart (HUP INT QUIT TERM USR1 USR2)

`EV_TRACE=1`
Print file system events for debugging

`ENTR_STATUS_SCRIPT=<path>`
awk script used with -x

KEYS AND ENVIRONMENT (cont.)

PAGER
Defaults to cat so output does not wait for keys

`echo $?`
0 after SIGINT, 1 on error, 2 on dir change

`entr -z <cmd>; echo $?`
Exit status: child's, or 128+signal

RECIPES

Rebuild and test, trimmed output

```
find src tests | entr -c -s
'make && make test | head -n
20'
```

Reload a server, pick up new files

```
while sleep 0.5; do
  git ls-files | entr -d -r ./
server
done
```

Rerun SQL on update

```
echo my.sql | entr -cp psql -
f /_
```

Test only the saved file

```
ls tests/*.py | entr -c
pytest /_
```