

NAME

script – make or replay a typescript of a terminal session

SYNOPSIS

```
script [options] > typescript++
script [options] < typescript++
script [options] [typescript] [-t 2>timing]
script [options] -t [typescript] < timing
script [options] typescript [timing] [speed]
```

DESCRIPTION

Script writes a copy of everything printed on your terminal. It is useful for students who need a hardcopy record of an interactive session as proof of an assignment. With some post processing, the typescript file can be printed out later with `lpr(1)`. **Script** is also useful for keeping records of actions on production systems, and can prove useful in the debugging of pathological terminal applications. A script typescript can also provide an interactive demonstration of something claimed (e.g. your program crashes on my system, all your base are us, etc).

Script recording

The simplest way to record a terminal session with **script** is to redirect standard output to a file.

```
script > typescript++
```

This typescript file contains the terminal output that is displayed during your session.

When redirection is not used, then if the file name argument *typescript* is given, **script** saves all dialogue in that file. If no file name is given, the typescript is saved in a file named `typescript`. If this file exists in the current directory, it must not be a symbolic link. In this form, timing information is not saved unless the **-t** option is supplied, and then it is written as the standard error output.

Script replay

To review your session, supply the typescript file as standard input to **script**.

```
script < typescript++
```

The session will be replayed using the timing information in the script file.

The interactive delays during your session are represented a CSI escape sequences which is not part of the standard `console_codes(4)`.

```
ESC [ 42 ; seconds ; milliseconds ]
```

Like earlier versions of `script`, this version of **script** can also replay a raw dump file using timing information which was stored in a separate timing file. This duplicates the functionality of `scriptreplay(1)`. If **script** is invoked in this way, it replays the supplied script

```
script [options] [timing] [typescript [speed]]
```

For symmetry with the recording command, the following form is also supported for replay of a session:

```
script [options] -t [typescript] [-x speed] < timing
```

The script ends when the forked shell exits (a *control-D* will exit the Bourne shell (`sh(1)`), and *exit*, *logout* or *control-D* (if *ignoreeof* is not set) for the C-shell, `csh(1)`).

Most interactive commands, such as `vi(1)`, create garbage in the typescript file by sending terminal escape sequences. **Script** works best with non-interactive commands that do not manipulate the screen, the results are meant to emulate a hardcopy terminal.

Options

- a** Append the output of the session to the `typescript`, file after the prior contents.
- f** Flush output after each write. This is nice for telecooperation: One person does ‘`mkfifo foo; script -f foo`’ and another can supervise real-time what is being done using ‘`tail -f foo`’
- q** Be quiet. This suppresses the start-up line in the typescript.
- c** *command*
Run the *command* rather than an interactive shell. With this option you can capture the output of a command other than the shell. This is useful for programs which misbehave only when their output is a terminal.
- t** Output timing data to standard error. This data contains two fields, separated by a space. The first field indicates how much time elapsed since the previous output. The second field indicates how many characters were output this time. This information can be used to replay typescripts with realistic typing and output delays.
- s** *command*
Set the screensaver command. In recording mode, if no input or output has occurred for the configured idle time, the screensaver command is run. If a screensaver is specified with no idle time, the idle time is set to 10 minutes. The `cmatrix(1)` application can be launched in this way.
- i** *seconds*
Set the idle time for script. In recording mode, when no input or output has occurred for *seconds* seconds, then the screensaver will run. If an idle time is configured without specifying a screensaver, **script** will close the session. In replay mode, the idle time is the maximum delay (in seconds, or fractions of a second) that the replay will pause for. To replay without any delay, use **-i 0**.
- x** *n* In replay mode, replay the output *n* times faster. The interactive delay is divided by this number.
- k** *file*
Log input to the named file. With this argument, script will act as a key logger. If the **-a** argument is used, this file will also be opened in **append** mode.

EXAMPLES

Record a session and play it ack:

```
script > mysession
pwd
date
cal
exit
script < mysession
```

To play it back a little faster - 2.5 times the speed and no more than a 5 second wait at a time:

```
script < mysession -x 2.5 -i 5
```

To play it back (in a terminal) with no delays at all:

```
cat mysession
```

To see the timing escape sequences inserted into the script file, use `less(1)`

```
less mysession
```

You can remove the fake timing escape sequences from a typescript file with the following `sed(1)` filter:

```
sed 's/.\[42;[0-9;]*\]//g' < mysession > stripped
```

Alternatively, you can do the same thing with **script** by setting the replay delay to 0

```
script -i 0 < mysession > output
```

Here's how you can record the progress of all of your interactive shell sessions. Put the following snippet in your *.bashrc* and *.profile* (although, start with *.bashrc* in case you break it).

```
if [ ! "$LOGFILE" ] && tty -s ; then
    export LOGFILE=$HOME/log/$( date +%Y%m%d-%H%M%S )
    exec $HOME/bin/script > $HOME/log/$STAMP
fi
declare +x LOGFILE
```

Note that this is a poor security measure by itself, since a user can trivially modify or remove an incriminating script file. Piping the output to a root process is left as an exercise to the reader.

When you join your local domain, you may like to impress on your system administrator that they should trust neither you nor your computer:

```
script -k UrPWND
clear
net join -U administrator
```

Run a console screensaver after 5 minutes of idle time.

```
script -i 300 -s 'cmatrix -s'
```

ENVIRONMENT

The following environment variable is utilized by **script**:

SHELL If the variable **SHELL** exists, the shell forked by **script** will be that shell. If **SHELL** is not set, and no **-c** parameter is supplied, the Bourne shell is assumed. (Most shells set this variable automatically.)

SEE ALSO

scriptreplay(1) (the way we used to play it back with timings), **console_codes(4)** (why you will have a hard time printing your typescript), **screen(1)** (another way to share a view of a terminal), **csh(1)** (for the *history* mechanism),

HISTORY

The **script** command appeared in 3.0BSD.

BUGS

Script places **everything** in the log file, including linefeeds, backspaces and escape codes. This is not what the naive user expects. One of the more annoying escape codes is "cursor position report". When we replay that code to the terminal, it "types" the cursor position, which appears as a dropping when script exits. The escape code for delay is non-standard. Should code 42 acquire a meaning, this will be a problem. This version of **script** attempts to be backwards compatible with old versions, which means it doesn't write timing information with the typescript unless you redirect standard output. If you want to replay a typescript you appended to another file with detached timings, you are on your own.