NAME

script - make typescript of terminal session

SYNOPSIS

script [**-aeFfkqr**] [**-t** *time*] [*file* [*command ...*]]
script -p [**-deq**] [**-T** *fmt*] [*file*]

DESCRIPTION

The **script** utility makes a typescript 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, as the typescript file can be printed out later with lpr(1).

If the argument *file* is given, **script** saves all dialogue in *file*. If no file name is given, the typescript is saved in the file *typescript*.

If the argument *command* is given, **script** will run the specified command with an optional argument vector instead of an interactive shell.

The following options are available:

-a Append the output to *file* or *typescript*, retaining the prior contents.

-d When playing back a session with the **-p** flag, do not sleep between records when playing back a timestamped session.

-e Accepted for compatibility with *util-linux* **script**. The child command exit status is always the exit status of **script**.

-F Immediately flush output after each write. This will allow a user to create a named pipe using mkfifo(1) and another user may watch the live session using a utility like cat(1).

-f Create *file.filemon* or *typescript.filemon* using filemon(4).

-k Log keys sent to the program as well as output.

-p Play back a session recorded with the **-r** flag in real time.

-q Run in quiet mode, omit the start, stop and command status messages.

-r Record a session with input, output, and timestamping.

-t *time* Specify the interval at which the script output file will be flushed to disk, in seconds. A value of 0 causes **script** to flush after every character I/O event. The default interval is 30 seconds.

-T *fmt* Implies **-p**, but just reports the time-stamp of each output. This is very useful for assessing the timing of events.

If *fmt* does not contain any '%' characters, it indicates the default format: '%n@ %s [%Y-%m-%d %T]%n', which is useful for both tools and humans to read, should be used. Note that time-stamps will only be output when different from the previous one.

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

Certain interactive commands, such as vi(1), create garbage in the typescript file. The **script** utility works best with commands that do not manipulate the screen. The results are meant to emulate a hardcopy terminal, not an addressable one.

ENVIRONMENT

The following environment variables are utilized by **script**:

SCRIPT The SCRIPT environment variable is added to the sub-shell. If SCRIPT already existed in the users environment, its value is overwritten within the sub-shell. The value of SCRIPT is the name of the *typescript* file.

SHELL If the variable SHELL exists, the shell forked by **script** will be that shell. If SHELL is not set, the Bourne shell is assumed. (Most shells set this variable automatically).

EXAMPLES

Record a simple csh(1) session with no additional details like input, output, and timestamping:

```
$ SHELL=/bin/csh script
Script started, output file is typescript
% date
Tue Jan  5 15:08:10 UTC 2021
% exit
exit

Script done, output file is typescript
```

Now, replay the session recorded in the previous example:

```
$ cat ./typescript
Script started on Tue Jan  5 15:08:08 2021
% date
Tue Jan  5 15:08:10 UTC 2021
% exit
exit

Script done on Tue Jan  5 15:08:13 2021
```

Record a csh(1) session, but this time with additional details like timestamping:

```
$ SHELL=/bin/csh script -r
Script started, output file is typescript
% date
Tue Jan  5 15:17:11 UTC 2021
% exit
exit

Script done, output file is typescript
```

In order to replay a sessions recorded with the **-r** flag, it is necessary to specify **-p** (cat(1) will not work because of all the aditional information stored in the session file). Also, let us use **-d** to print the whole session at once:

```
$ script -dp ./typescript
Script started on Tue Jan  5 15:17:09 2021
% date
Tue Jan  5 15:17:11 UTC 2021
% exit
exit

Script done on Tue Jan  5 15:17:14 2021
```

SEE ALSO

csh(1) (for the *history* mechanism), filemon(4)

HISTORY

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

The **-d**, **-p** and **-r** options first appeared in NetBSD 2.0 and were ported to FreeBSD 9.2.

BUGS

The **script** utility places **everything** in the log file, including linefeeds and backspaces. This is not what the naive user expects.

It is not possible to specify a command without also naming the script file because of argument parsing compatibility issues.

When running in **-k** mode, echo cancelling is far from ideal. The slave terminal mode is checked for ECHO mode to check when to avoid manual echo logging. This does not work when the terminal is in a raw mode where the program being run is doing manual echo.

If **script** reads zero bytes from the terminal, it switches to a mode when it only attempts to read once a second until there is data to read. This prevents **script** from spinning on zero-byte reads, but might cause a 1-second delay in processing of user input.