ed

ED, the classic text editor, is now available. SD-8516 KERNAL ROM v0.7.3 contains an implementation of the full 1969 spec for ed. You can start it by typing ed at the ready prompt.

History

In August 1968 at Bell Labs, Ken Thompson was on the verge of something huge. Working on a humble PDP-7 computer, he began building Unix; a brand-new operating system that would eventually change the world. To bootstrap Unix from nothing, Thompson realized he needed exactly three critical tools, like the essential components to trigger a stable resonance cascade in a complex experiment involving lasers and crystals from an alien planet in an alternate dimension. These were:

- a shell to communicate with the system and enable basic operations,
- an assembler to write new programs and develop the system,
- and a text editor to load, edit and store the files for the assembler (and more; config files, etc.)

That editor was ed, and it became one of the foundational pillars of the Unix-verse.

ed was no ordinary program. Because early computers used slow, printing teletypes instead of screens, Thompson designed ed as a line-oriented editor, suitable for using a line printer instead of a terminal.

Programmers could address specific lines, make precise changes, and use something called regular expressions – pattern-matching algorithms – to search and transform text with incredible precision. As one of the three pillars of userland UNIX alongside the assembler and shell, ed gave Thompson the ability to write, fix, and improve the system right on the machine. It was elegant, minimal, and unbelievably effective.

Even today, ed remains quietly powerful. It is still required by the official POSIX standard and its DNA lives on in tools like vi, sed, and grep. What started as a simple idea in 1969 became one of computing's most important building blocks: ed, the first editor. The program that launched the operating system revolution.

Commands

q, Q

- Requires version 0.7.1 and above

```
: q
```

The q command quits the program. If you have unsaved work you must save it before quitting with q.

```
: Q
```

The Q command insta-quits. You will lose any unsaved work.

#,

- Requires version 0.7.2 and above

```
3
```

If you enter a number the editor will set the current line. For example, if there are five lines in your document and you type 3, you will move to the third line.

```
=
```

The = command shows you what line number you're on. For example, if there are 10 lines in your document and your current line-pointer is three, it will display a 3.

This is the line you will append after if you type a (append line), or the line you will push down if you type i (insert line).

a, c, i

- Requires version 0.7.2 and above

```
: a
```

This command appends to the document, adding a new line after the current line.

```
: a
hello world
this is line two
testing ed
.
```

This will allow you to type text into the file. If you are appending to an existing line, it will insert the text after the line you are on. For example, if you are on line 3, it will push the old line 4 down and create a new line 4. This will continue every line you type until you enter a . on a line by itself.

```
: i
```

This is the insert command. If you have five lines in your document and you are on line 3, the insert command will create a new line 3 and move all the other lines down (the old line 3 becomes the new line 4, etc). This will continue every line you type until you enter a . on a line by itself.

```
: c
```

Finally, this line replaces the line you're on with a new line, and you can continue to add additional lines until you enter a . on a line by itself.

l, n, p

- Requires version 0.7.2 and above

```
: l
```

This command print the current line with all non-printable characters shown as an escape code.

```
: n
```

This command prints the current line with line numbers in front for easy reference.

```
: p
```

This simply prints the current line's text as-is in human-readable format.

#l, #n, #p

- Requires version 0.7.2 and above

```
: 1l
```

This command prints the first line with all non-printable characters shown as an escape code.

```
: 2n
```

This command prints the second line with a line number in front for easy reference.

```
: 3p
```

This simply prints line 3 in normal, human-readable format.

#,#n, #,#bl, #,#p

- Requires version 0.7.2 and above.

```
: 1,3l
```

This command prints lines 1 to 3 with all non-printable characters shown as an escape code.

```
: 2,5n
```

This command prints the second line through to the fifth line with line numbers in front for easy reference.

```
: 3,1p
```

This is the same as 1,3p – print the lines from 1 to 3.

```
: 1,$p
```

$ means the last line, so 1,$p prints the entire document.

d

- Requires version 0.7.2 and above

This deletes the line you are currently on. There is no undo in this version, so be careful!

f

- Requires version 0.7.2 and above.

```
f
```

This command shows the `filename`.

```
f file.txt
```

This command sets `filename` to file.txt. You can choose your own `filename` or use file.txt if you are bored.

w

- Requires version 0.7.2 and above.

```
w
```

This command writes the buffer to `filename` (see `f` above).

Always remember to save your work before quitting.

e

```
e
```

e with no argument reloads the current `filename` as set by `f` or a previous e.

```
e myfile.txt
```

This clears the entire buffer, loads the file into the buffer, sets the `filename`. Any unsaved changes are lost.

If the buffer is dirty, e warns with ? (like q).

From:
https://www.appledog.ca/wiki/ - **Appledog**

Permanent link:
https://www.appledog.ca/wiki/doku.php?id=sd:ed

Last update: **2026/04/14 06:09**