ed und sed

David Frey

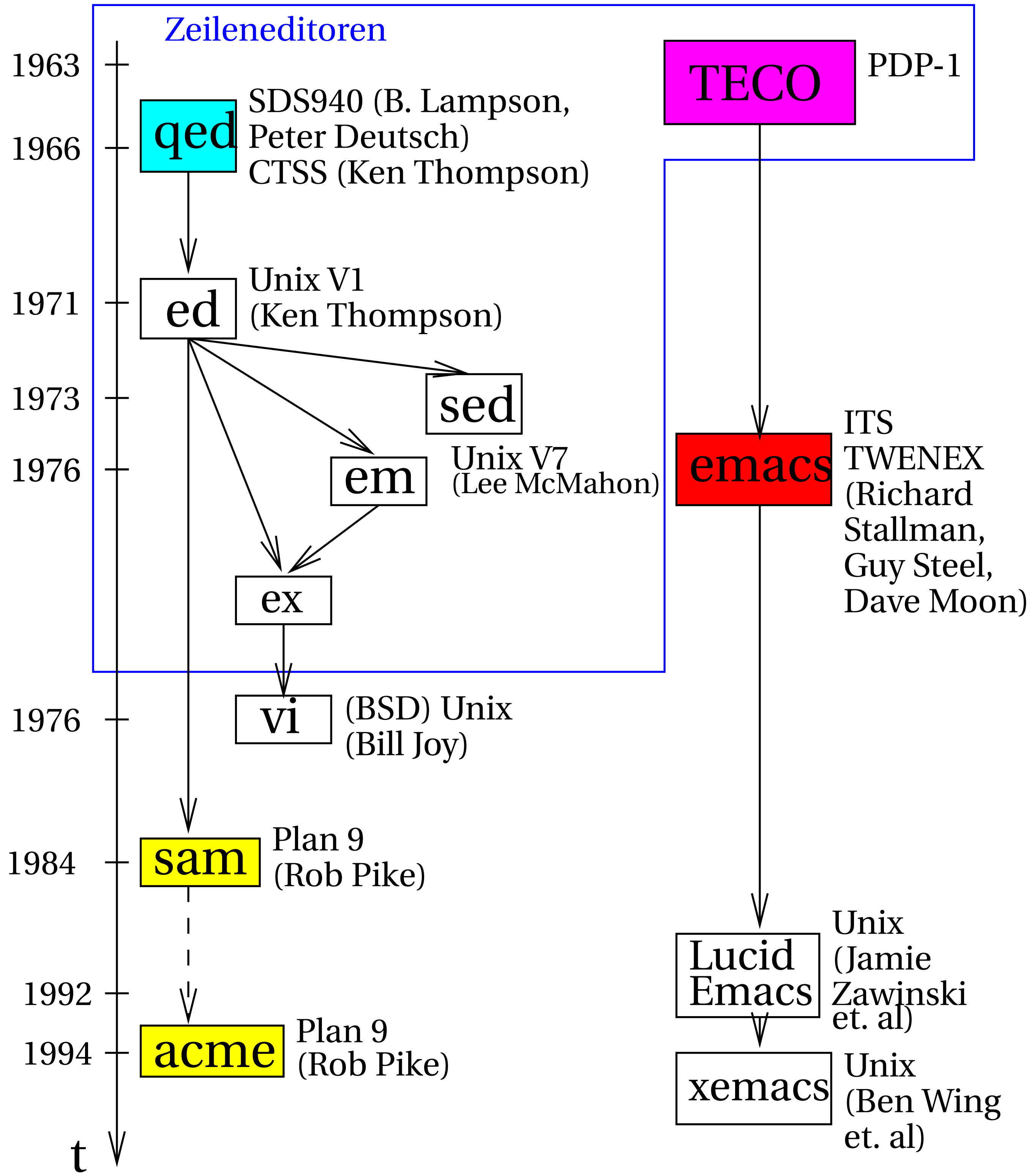
ed: Ueberblick

> The cycle of editing and testing is very convenient in Visual Basic, though the facilities for viewing and editing text are so primitive that one yearns for a powerful editor like ed.
>
> Brian W. Kernighan

`ed` ist der Unix-Standard Editor; es ist eine vereinfachte Version des `qed`-Editors [1]. `qed` konnte z.B. mehrere Buffer, erweiterte Regular Expressions, rechnen und war mit seinen eigenen Befehlen programmierbar.

Alle diese Editoren sind *Line-Editoren*, d. h. man editiert Linien resp. Linienbereiche.

Sie sind **modal** d. h. die Editoren unterscheiden zwischen *Input* und *Command*-Mode.


Zeileneditoren
1963
1966
1971
1973
1976
1976
1984
1992
1994
t
qed
SDS940 (B. Lampson,
Peter Deutsch)
CTSS (Ken Thompson)
TECO
PDP-1
ed
Unix V1
(Ken Thompson)
sed
em
Unix V7
(Lee McMahon)
ex
vi
(BSD) Unix
(Bill Joy)
emacs
ITS
TWENEX
(Richard
Stallman,
Guy Steel,
Dave Moon)
sam
Plan 9
(Rob Pike)
acme
Plan 9
(Rob Pike)
Lucid
Emacs
Unix
(Jamie
Zawinski
et. al)
xemacs
Unix
(Ben Wing
et. al)

ed: Syntax und Addressen

Syntax eines Kommandos:

`[`adr_1`[,`adr_2`]]`*cmd*`[`*params*`]`

eine Adresse:

`.`	momentane Zeile
`$`	letzte Zeile
n	$n$te Zeile
$-n$,^n	n-Zeilen zurück
$+n$	n-Zeilen vorwärts
`,`,%	≡`1,$`
`;`	≡`.,$`
`/`*RE*`/`	regular expression vorwärts
`?`*RE*`?`	regular expression rückwärts
`'`*m*	mit *m* markierte Zeile

Ein Range besteht aus zwei Adressen:

a`,`*b*	Zeilen $[a,b]$
a`;`*b*	Zeilen, setze zuerst *a*, und berechne dann *b*

ed: Befehle (1) [2]

`a`	append
`c`	change (replace) line
`d`	delete
`e` *fn*,`e !`*cmd*	edit file, stdout of *cmd*
`E` *fn*	edit (unconditionally)
`f`	set default file name
`g/`*RE*`/`*cmds*	apply *cmds* on lines with matching REs.
`G/`*RE*`/`	interactively edit addressed lines with RE.
`h`,`H`	print help on last error, turn off explanations
`i`	insert text
`j`	join lines
`k`*m*	mark line with letter *m*
`l`	print lines
`m`	move lines

ed: Befehle (2) [2]

`n`	number lines
`p`	print lines
`P`	toggle command prompt
`q`,`Q`	quit (unconditionally)
`r` *fn*,`r` `!`*cmd*	read *fn*, stdout of *cmd*
`s/`*RE*`/`*rep*`/`[g|*n*]	search & replace first/all (g)/*n*th match(es)
`t`*addr*	copy (transfer) after *addr*.
`u`	undo last command
`v/`*RE*`/`,/V*RE*`/`	apply commands *not* matched by *RE*
`w`[`q`] *fn*,`w` `!`*cmd*	write *fn*, stdout of *cmd*
`W` *fn*	append to *fn*
`x`	put cut buffer after addressed line
`y`	yank lines to the cut buffer
`z`*n*	scroll *n* lines.

ed: Befehle (3) [2]

! *cmd*	execute *cmd*
#	comment
=	print line number
newline	print addressed line and set address

sed

`sed` steht für *Stream Editor*, die gestreamte Variante des Standard Editors, `ed`.

Somit fallen alle Befehle, die eine Position im Stream benötigen, weg.

Alle interaktiven Befehle fallen auch weg (`hHPtxyuz`).

sed: Befehle (1) [3]

![a]	`a`	append
-	`c`	change (replace) line
	`d`	delete
-	`e,E`	edit file
-	`f`	set default file name
!	`g/G`	copy/append hold space to pattern space
!	`h/H`	copy/append pattern space to hold space
![a]	`i`	insert text
-	`j`	join lines
-	`k`*m*	mark line with letter *m*
	`l`	print lines unambigously
	`m`	move lines

[a] ‘\’ als Fortsetzungszeichen anstelle von ‘.’ am Ende.

sed: Befehle (2) [3]

!	`n`,`N`	read/append next line of input into pattern space
	`p`	print lines
!	`P`	print up to first embedded newline
	`q`,`Q`	immediately quit (q: print line if auto-print)
!	`r`*fn*,`R`*fn*	append text/a line read from *fn*
	`s/`*RE*`/`*rep*`/`[`g`|*n*]	search & replace first/all (g)/*n*th match(es)
!	`t`*label*	branch to label on successful substitution.
!	`T`*label*	branch to label on unsuccessful substitution.
-	`u`	undo last command
-	`v/`*RE*`/`,`/V`*RE*`/`	apply commands *not* matched by *RE*

sed: Befehle (3) [3]

!	`w`*fn*	write current pattern space to *fn*
!	`W`*fn*	write the first line of the current pattern space to *fn*
-	`x`	put cut buffer after addressed line
!	`y/`*src*`/`*dst*`/`	transliterate characters in pattern space
-	`z`*n*	scroll *n* lines.
+	`:`	label
-	`!`*cmd*	execute *cmd*
	`#`	comment
	`=`	print line number
!	`;`	command separator
!	*newline*	white space
+	`{,}`	opening, closing block

Aufrufsyntax

Standard `sed`-Aufruf:

`sed [-n] -e`*`script`*`|-f`*`script-file`*

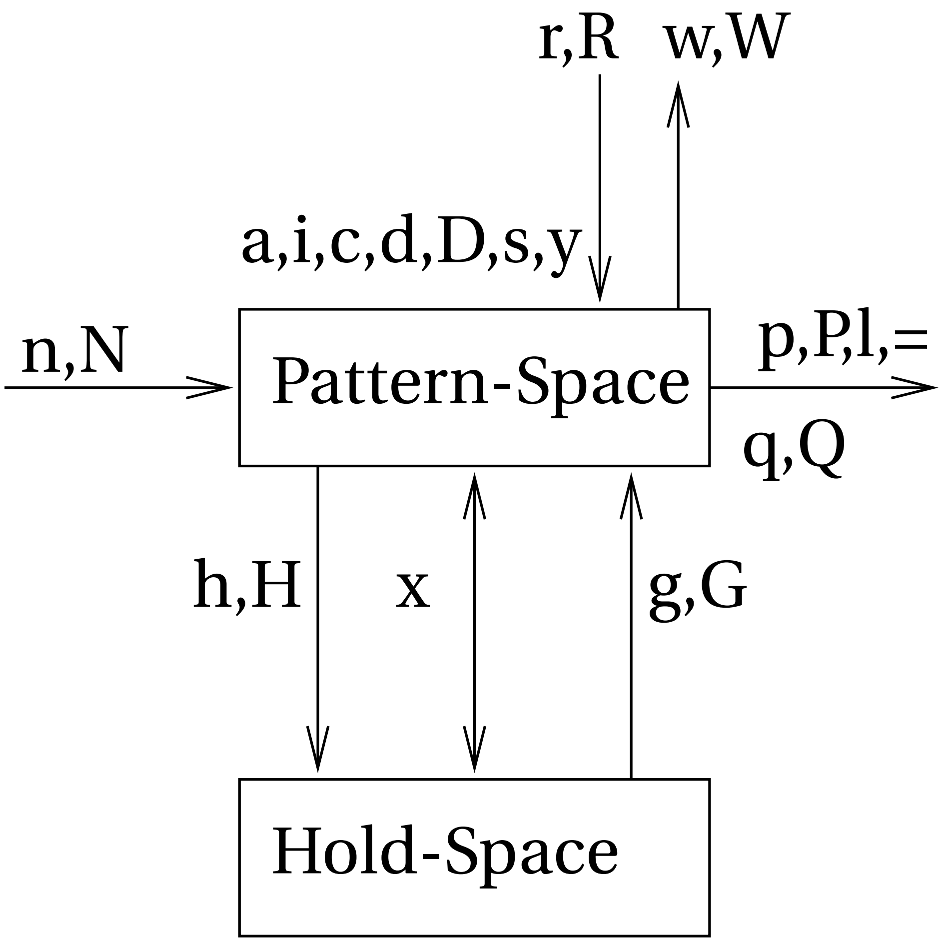
GNU-`sed` kann wie üblich ein paar Sachen mehr:

`sed [-nrsu][-i[`*`suffix`*`]][-l`*`N`*`]`
`-e`*`script`*`|-f`*`script-file`*

Aus der Manpage [3]:

- -n Pattern-Space nicht automatisch ausgeben
- -e Script folgt auf der Command-Line
- -f Script folgt im File
- -i File in-place (evtl mit Backup) editieren
- -l *N* Line-Wrap-Länge auf *N* setzen
- -r Perl Regular Expressions (ERE) verwenden
- -s separate Files
- -u unbuffered

Hold/Match-Space



h,g,n	copy (h,g), read (n)
H,G,N	append
#	comment
:,b,t,T	labels, branch
{,}	block

I/O-Paradigma

Zeilen werden prinzipiell sequentiell gelesen und bearbeitet.

Flow-Control

label

:	define label

go-to

b	branch unconditionally

if

t	branch on successful substitution
T	branch on unsuccessful substitution

N.B: sed ist *turing-complete*.

Beispiele (1) [z.T. aus [4]]

cat

```
sed -n p /etc/motd
```

cat -n resp. nl -ba

Einfache Lösung:[a]

```
sed -n '=;p' /etc/motd|
sed -e 'N; s/\([[:digit:]]\+\)\n/\1 /'
```

cat -vET

```
sed -n l /etc/motd
```

cat -s

```
:x
/^\n*$/{
  N
  bx
}
s/^\(\n\)*/\1/
```

[a] mit nur einem Prozess geht es auch [4]

Beispiele (2) [z.T. aus [4]]

`tac`

`sed -ne '2,$G;$p;h' /etc/motd`

original: `sed -ne '1!G;$p;h' /etc/motd`

`head`

`sed -n 1,10p /etc/motd`

`tail`

~~`sed -n '$-10,$p' /etc/motd`~~

Es geht, ist aber kein Einzeiler (→ [4]):

`wc -l`

`sed -n '$=' /etc/motd`

`wc -c`

Es geht, ist aber kein Einzeiler (→ [4]):

Beispiele (3) [z.T. aus [4]]

rot13

leicht exotisches Beispiel:[a]

```
sed -e 'y/\
ABCDEFGHIJKLMNOPQRSTUVWXYZ\
abcdefghijklmnopqrstuvwxyz/
NOPQRSTUVWXYZABCDEFGHIJKLM\
nopqrstuvwxyzabcdefghijklm/' /etc/motd
```

[a] rot13, caesar 13, tr 'A-MN-Za-mn-z' 'N-ZA-Mn-za-m'

komplexes Beispiel (Input)

```
Title

Header 1
========

Text *bold* _underlined_ /italic/
URL <http://www.lugs.ch>

Header 2
--------

Text *bold* _underlined_ /italic/

        o bullet 1
        o bullet 2
                - bullet bullet 1
                - bullet bullet 2
        o bullet 3

```

komplexes Beispiel (Script)

```
#!/bin/sed -f

s,<\([A-Za-z.:/]\+\)>,<a href="\1">\1</a>,
t skip

s,/\([^*]*\)/,<em>\1</em>,g;
s,\*\([^*]*\)\*,<b>\1</b>,g;
s,_\([^_]*\)_,<u>\1</u>,g

:skip

1i \
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">\
<html>
1{
s,^,<head><title>,; s,$,</title></head>,;
a \
<body>
}
$a \
</body>\
</html>

/^[         ]*[o.-]/{
  x
  /^$/{
    s/^/UL/
  i \
<ul>
  }
  x
```

```
  s/^[         ]*[o.-]/<li>/
}

/^$/{
  x
  /^UL$/{
    s/^UL//
  i \
</ul>
  }
  x
}


N
/^.*\n *====* *$/{
s/\n *====* *//;
s,^,<h1>,; s,$,</h1>,
}


/^.*\n *----* *$/{
s/\n *----* *//;
s,^,<h2>,; s,$,</h2>,
}

P
D
```

komplexes Beispiel (Output)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<body>
<head><title>Title</title></head>

<h1>Header 1</h1>

Text <b>bold</b> <u>underlined</u> <em>italic</em>
URL <a href="http://www.lugs.ch">http://www.lugs.ch</a>

<h2>Header 2</h2>

Text <b>bold</b> <u>underlined</u> <em>italic</em>

<ul>
<li> bullet 1
<li> bullet 2
<li> bullet bullet 1
<li> bullet bullet 2
<li> bullet 3
</ul>

</body>
</html>
```

Vorteile

`sed` ist für das Editieren von Streams ausgelegt; daher ist der Aufwand um etwas zu Editieren gering.

Nachteile

- Keine Variablen (⇒ `sh`,`awk`,`m4`,`perl`)
- Kann nicht rechnen (⇒ `sh`,`awk`,`m4`,`perl`)
- (Standard `sed`) kann nicht *in-place* editieren (⇒ `ed`[,`perl`])

Zusammenfassung

`sed`-Programme sind fast ausschliesslich Einzeiler und werden in Pipelines in Shell-Scripts verwendet.

Ab ≥5…10-Zeilen wird es (langsam) kompliziert.

Kuriosa oder sed-Stresstests

Folgende sed-Scripts wurde von Leuten geschrieben, die Langeweile hatten und/oder mal ein bisschen sed-Programmierung üben wollten [5]:

- dc in sed: `/usr/share/doc/sed/examples/dc.sed`
- Infix-Mathematik in sed: `/var/local/ftp/pub/copied/source-cd-1993/bsd_srcs/usr.bin/sed/math.sed`
- Towers of Hanoi in sed: `/var/local/ftp/pub/copied/source-cd-1993/bsd_srcs/usr.bin/sed/hanoi.sed`
- 99-Bottles of Beer in sed: `http://www.99-bottles-of-beer.net/s.html#sed`
- Sokoban in sed: `http://http://aurelio.net/sed/`

Literatur

[1] Dennis M. Ritchie. An incomplete history of the QED Text Editor. `http://cm.bell-labs.com/cm/cs/who/dmr/qed.html`.

[2] Free Software Foundation, Inc. *ed manual*, November 1994. ed(1).

[3] Free Software Foundation, Inc., 59 Temple Place – Suite 330, Boston, MA 02111-1307, USA. *sed manual*, 4.1.2 edition, August 2004. sed(1).

[4] Free Software Foundation, Inc., 59 Temple Place – Suite 330, Boston, MA 02111-1307, USA. *sed info pages*, 4.1.2 edition, August 2004. `info sed`.

[5] various. `sed` FAQ. `http://www.faqs.org/faqs/editor-faq/sed/`.

[6] Brian W. Kernighan and Rob Pike. *The UNIX*

Programming Environment. Prentice–Hall International, Inc., Eaglewood Cliffs, New Jersey, NJ 07632, USA, 1984.
Excellent introduction into Shell Programming, sed, and awk. Discusses programming style, i.e. how to build a filter etc.
ISBN 0-13-937699-2 (hardcover), 0-13-937681-X (paperback).

[7] Dale Dougherty and Arnold D. Robbins. *sed & awk.* O'Reilly & Associates, Inc., 103 Morris Street, Suite A Sebastopol, CA 95472, USA, second edition, March 1997.
ISBN 1-56592-225-5.

[8] Arnold D. Robbins. *sed & awk Pocket Reference.* O'Reilly & Associates, Inc., 103 Morris Street, Suite A Sebastopol, CA 95472, USA, First edition, January 2000.
ISBN 1-56592-729-X.

[9] Brian W. Kernighan. *A Tutorial Introduction to the UNIX Text Editor*, Seventh edition, ca. 1977.
7th edition Unix manual.

`~ftp/pub/doc/OS/Unix/7thEdMan/vol2/edtut.{ps,txt}.gz`.

[10] Lee E. McMahon. *SED – A Non-interactive Text Editor*, Seventh edition, August 1978.
7th edition Unix manual.
`~ftp/pub/doc/OS/Unix/7thEdMan/vol2/sed.{ps,txt}.gz`.