Getting started with ed

T-DOSE, Geldrop, 23-04-2023

Matto Fransen
https://box.matto.nl
gopher://box.matto.nl

ed is the standard editor

In 1969, while is wife and son were away on vacation, Ken Thompson allocated three weeks to create the basic components of a new system.

One week for the assembler, one week the shell, and one week for the editor.
This is how Unix was born.

That editor was ed.

It was written in assembler, later followed a rewrite in C by Dennis Ritchie.

ed became the standard editor for Unix.

ed is a line editor

Lines are the smallest addressable unit.

All operations are done on lines.

ed and vi

ed evolved into ex (“extended”), written by Chuck Haley and Bill Joy

Bill Joy also created the first BSD Unix release, in 1978, which included ex 1.1.

Bill Joy created vi, as a visual mode for ex

In the BSD version of 1979, vi was a hard link to ex, making them the same executable

One could say that vi is a direct descendant of ed

Why ed

ed is simple

ed gives you the power of regular expressions

ed is very resource-efficient and works fine on low capacity hardware

ed is fast, works efficient and is great for small edits

ed is useful in scripts

ed doesn’t clobber your screen

ed (and not vi) is part of the rescue image of OpenBSD

Resources

man ed

info ed, or see https://www.gnu.org/software/ed/manual/ed_manual.html *

A Tutorial Introduction to the UNIX Text Editor by Brian W. Kernighan (PDF) *

Advanced Editing on UNIX by Brian W. Kernighan (PDF) *

* Links on https://box.matto.nl/links.html

Invoking ed

Start ed with a filename to open or create that file:

```
ed ˜/.ssh/config
ed new-file
```

There are some switches, some depending on which ed you use (GNU, BSD, etc)

```
ed -p “myprompt> ”
ed -s file
```

rlwrap

You can make life easier, by using rlwrap with ed:

```
rlwrap ed <myfile>
```

This way, you can use command history

Buffer

In ed, you work in a buffer.

When you open an existing file, it will be read into the buffer.

When you create a new file, you start with an empty buffer.

There is only one buffer, you can only work on one file at a time.

Only when you issue a write command, the file on disk will be touched.

When you close ed without writing your buffer to the file first, all your changes will be lost.

When you have touched the buffer, ed issues a warning before closing.

Addressing lines

Address	**Meaning**
12	Line number 12
12,14	Lines 12, 13, 14
. (dot)	The current line
1	First line of the file
$	Last line of the file
1,$	From first to last line (whole file) - 1 and $ are default values:
,4	From first line to line 4
12,	From line 12 to last line
, (comma)	From first to last line (whole file)

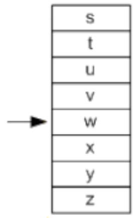
Relative addresses

Address	**Meaning**
-	The line above the current line
-4	4 lines above the current line
+	The line below the current line
+5	5 lines below the current line
-4,+6	from 4 lines above the current line to 6 line below the current line

The current line (.)

See the current line (.) as a pointer to some line in the buffer.

What the current line points at, changes all the time.



Almost every action changes the current line.

The current line points to the last line (highest line number) that was part of the latest action.

Prepare a file to play with

Prepare a text file, with length of 3-4 screen-heights

Open this file in ed

E.g.:

```
man man > man.txt
ed man.txt
```

Making ed more talkative

Command	Meaning
H	Toggles printing of error explanation
h	Prints explanation of last error
P (Uppercase p)	Toggles printing of a prompt

Tip:

In the beginning, right after opening ed, issue the commands H and P.

Later, when you have become more comfortable with ed, you can ommit this.

Displaying content

Command	**Meaning**
p	Print (normal display of line)
l (lowercase L)	List (shows tabs like , end of line like $)
n	Line numer + print line

Examples

3,9p	Print lines 3 - 9
12,l	List lines 12 - end of file
,n	Show line number + print line of all lines
-4,+5n	From 4 lines above current line to 5 below the current line show line nummer and print the line

Demo: line 96 from utf8-demo.txt

Handy commands

Command	Meaning
.	Print the current line
.=	Show line number of the current line
n	Show current line number and print the current line
30	Make line 30 the current line and print it.
=	Show line number of the last line (doesn’t change what the current line is)

Scrolling

z	Scrolls (show lines until end of screen) Sets the current line to the last shown line.

Examples

Command	**Meaning**
1zn	Show linenumber + print line from line 1 until end of screen
zn	Start at current line, and scroll until end of screen
28zn	Start at line 28, and scroll until end of screen

Tip:

Start scrolling with <line number>zn, and follow up with zn.

Searching

Command	Meaning
/this/	Search for the first occurrence of “this” “this” can be a regular expression
?this?	Search backwards for the first occurrence of “this” “this” can be a regular expression
//	Repeat last search

Tip:

/this and ?this will work too.

Undo

Command	**Meaning**
u	Undo

you can only undo the last operation.

Adding content

Command	Meaning
a	**A**ppend: add one or more new lines below the designated address
i	**I**nsert: add one or more new lines above the designated address
c	**C**hange: remove the lines at the designated address and append one or more new lines

Enter a line with a single dot (.) to close the append/input mode and revert to command mode

Tip:

Watch for the prompt: when there is no prompt, you are still in append, insert or change mode.

Adding content examples

Command	Meaning
a	append new line(s) immidiately below the current line
90a	append new line(s) immediately below line 90
1i	insert new lines at the start of the file (above the first line)
12,16c	change the lines 12-16: remove the lines and replace by one or more new lines

Tip:

When you create a new file, the buffer will be empty. Start with a (append) to add content.

When you open an existing file, the current line will be the last line.
Start with a (append) to add lines below the last line.

Delete, move or copy content

Command	**Meaning**
d	**D**elete: delete the lines at the designated address.
m	**M**ove: delete the lines at the designated address and append those right below the appointed address
t	Copy **t**o: copy the lines at the designated address and append right below the appointed address

Delete, move or copy examples

Command	**Meaning**
d	Delete the current line
10,20d	Delete lines 10-20
22,25m49	Delete lines 22-25 and paste those right below line 49
4,7t$	Copy lines 4-7 to the end of the file (just below the last line)
-3,.m12	From 3 lines above the current line to the current line, delete the lines and paste those below line 12
t123	Copy current line to just below line 123

Tip:

Deleting, moving and copying change the line numbers.

Check the line numbers between two consecutive actions.

Subsititute command

Command	**Meaning**
s/this/that/	Substitute in the current line the first occurence of “this” with “that” This also works with regular expression
s/this/that/g	Substitute in the current line every occurence of “this” with “that”
s/this/that/n	Substitute in the current line the nth occurence of “this” with “that”
s/ˆ/foo /	Add “foo ” at the start of the current line (ˆ stands for start of the line)
s/sister/&s/n	The & stands for the matched text, so you don’t have to type that again
s/$/bar/n	Append “bar” at the end of the current line ($ stands for end of the line)
12,20s/foo/bar/	In lines 12 - 20, subsitute “foo” with “bar”.

The slashes can be replaced by another character, providing you do this for all three:

```
,sX/usr/shareX/usr/local/shareX  or  ,s:/home/someone:/home/someoneelse:
```

Global commands

Command	**Meaning**
g/re/<command>	Select all lines that conform to the regular expression “re”, and perform <command>
g/this/s/foo/bar/	Select all lines containing “this” and perform substitution of first “foo” with “bar”
g/this/s/foo/&bar/3	Select all lines containing “this” and on each line substitute the third occurence of “foo” with “foobar”
g/re/p	Print all lines that conform to the regular expression “re” Fun fact: This is where the name of “grep” comes from
v/re/<command>	Select all lines that **do not** conform to the regular expression “re”, and perform <command> (just like “grep -v ...”)
v/this/s/foo/bar/	Select all lines **not** containing “this” and perform substitution “foo” with “bar”

Substitution with global command examples

Command	Meaning
g/ˆ#/s/ˆ/! /	Prepend “! ” to all lines starting with ’#’
g/ˆ#/d	Delete all lines that start with “#”
v/foo/m$	Move all lines that don’t contain “foo” to the end of the file
s/png$/gif/	On the current line, replace “png” at the end of the line with “gif”
10,12s/png/gif/2	On line 10-12, replace the second occurence of “png” with “gif”
s/png/gif/g	On the current line, replace all occurences of “png” with “gif”
sX/libX/usr/libX	On the current line, replace the first occurence of “/lib” with “/usr/lib”

Tip:

Substitution and global commands are incredible powerful. Dive into this when you have mastered the basics!

Working with files

Command	Meaning
w (lowercase w)	Write current buffer to disk (using the current file name)
w that-file	Write current buffer to the file that-file (overwrite if exists)
12,30w some-file	Write the lines 12-30 to the file “some-file”
8,12W another-file (uppercase w)	Append the lines 8-12 to the end of file “another-file”
12r that-file	Read the contents of the file “that-file” and insert starting right below line 12.
e this-file	Discard the contents of the buffer and start editing file “this-file”

Quiting ed

wq	Write current buffer to disk and close ed
q	Close ed. When the buffer is touched, ed issues a warning. Repeat q to ignore the warning and close ed.

ed with here documents

```
#!/bin/sh

ls > myfilelist

ed myfilelist <<\END
H
v/jpg/d
g/jpg/s/\(^.*$\)/convert \1 \1/
,s/jpg$/png/
wq
END
```

join and split lines

Command	Meaning
j	join: merge next line at end of this one (think: remove newline at end of current line)
s/this that/this\	
	Split: subsitute with a backslash and <enter>

Tip:

Before joining lines, first add a space add the end of the first, or the beginning of the second line.

Mark lines

k<character>	Mark address with <character>
’<character>	Address marked with <character>
/broom/kc	Search for the first line containing “broom” and set mark ’c’ there
?ˆzoo?kx	Search for the previous line starting with ’zoo’ and set mark ’x’ there
’xa	Start appending below the line marked with ’x’
’e,’fd	delete region from line marked with ’e’ to line marked with ’f’

Tip:

In scripts, when you want to use searching to get the address of a line, you can use marks. E.g.:

1	
/foo/ka	Search for foo and mark with ’a’
1,’ad	delete from top to line with foo

Shell command

Command	Meaning
!date	Run shell command “date”
9r !date	Read the output of shell command “date” and insert right below line 9.
!cp this that	Run shell command “copy this that”

Diff to create ed edits

Command	Meaning
diff --ed fileA fileB	Create a diff in the form of an ed script

```
diff --ed fileA fileB > mypatch
echo "wq" >> mypatch
cat mypatch | ed fileA
```

Thanks !

Thanks for your attention !

Happy ed-ing :)