filefilter — Convert ASCII or binary patterns in a file

Description

`filefilter` reads an input file, searching for *oldpattern*. Whenever a matching pattern is found, it is replaced with *newpattern*. All resulting data, whether matching or nonmatching, are then written to the new file.

Because of the buffering design of `filefilter`, arbitrarily large files can be converted quickly. `filefilter` is also useful when traditional editors cannot edit a file, such as when unprintable ASCII characters are involved. In fact, converting end-of-line characters between Macintosh, Windows, and Unix is convenient with the EOL codes.

Unicode is not directly supported, but UTF-8 encoded files can be operated on by using byte-sequence methods in some cases.

Although it is not mandatory, you may want to use quotes to delimit a pattern, protecting the pattern from Stata's parsing routines. A pattern that contains blanks must be in quotes.

Quick start

Create `newfile.txt` from `oldfile.txt` by replacing all tabs with semicolons

```
filefilter oldfile.txt newfile.txt, from(\t) to(";")
```

Create `newfile.txt` from `oldfile.txt` by replacing all instances of "The" with "the"

```
filefilter oldfile.txt newfile.txt, from("The") to("the")
```

Syntax

`filefilter` *oldfile newfile* ,

{ `from(`*oldpattern*`)` `to(`*newpattern*`)` | `ascii2ebcdic` | `ebcdic2ascii` } [*options*]

where *oldpattern* and *newpattern* for ASCII characters are

"*string*" or *string*

string := [*char*[*char*[*char*[...]]]]

char := *regchar* | *code*

regchar := ASCII 32–91, 93–127, or extended ASCII 128, 161–255; excludes '\'

code :=		
	`\BS`	backslash
	`\r`	carriage return
	`\n`	newline
	`\t`	tab
	`\M`	Classic Mac EOL, or \r
	`\W`	Windows EOL, or \r\n
	`\U`	Unix or Mac EOL, or \n
	`\LQ`	left single quote, ‘
	`\RQ`	right single quote, ’
	`\Q`	double quote, ”
	`\$`	dollar sign, $
	`\###d`	3-digit [0–9] decimal ASCII
	`\##h`	2-digit [0–9, A–F] hexadecimal ASCII

Options

Description

* `from(`*oldpattern*`)` to be replaced
* `to(`*newpattern*`)` to replace occurrences of `from()`
* `ascii2ebcdic` from ASCII to EBCDIC
* `ebcdic2ascii` from EBCDIC to ASCII
* `replace` *newfile* if it already exists

* Both `from(`*oldpattern*`)` and `to(`*newpattern*`)` are required, or `ascii2ebcdic` or `ebcdic2ascii` is required. `collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

`from(`*oldpattern*`)` specifies the pattern to be found and replaced. It is required unless `ascii2ebcdic` or `ebcdic2ascii` is specified.

`to(`*newpattern*`)` specifies the pattern used to replace occurrences of `from()`. It is required unless `ascii2ebcdic` or `ebcdic2ascii` is specified.

`ascii2ebcdic` specifies that characters in the file be converted from ASCII coding to EBCDIC coding. `from()`, `to()`, and `ebcdic2ascii` are not allowed with `ascii2ebcdic`.

`ebcdic2ascii` specifies that characters in the file be converted from EBCDIC coding to ASCII coding. `from()`, `to()`, and `ascii2ebcdic` are not allowed with `ebcdic2ascii`.

`replace` specifies that *newfile* be replaced if it already exists.

Remarks and examples

Convert Classic Mac-style EOL characters to Windows-style

```
. filefilter macfile.txt winfile.txt, from(\M) to(\W) replace
```

Convert left quote (‘) characters to the string “left quote”

```
. filefilter auto1.csv auto2.csv, from(\LQ) to("left quote")
```

Convert the character with hexidecimal code 60 to the string “left quote”

```
. filefilter auto1.csv auto2.csv, from(\60h) to("left quote")
```

Convert the character with decimal code 96 to the string “left quote”

```
. filefilter auto1.csv auto2.csv, from(\096d) to("left quote")
```

Convert strings beginning with hexidecimal code 6B followed by “Text” followed by decimal character 100 followed by “Text” to an empty string (remove them from the file)

```
. filefilter file1.txt file2.txt, from("\6BhText\100dText") to("")
```

Convert file from EBCDIC to ASCII encoding

```
. filefilter ebcdicfile.txt asciifile.txt, ebcdic2ascii
```

❑ Technical note

Unicode is not directly supported, but you can try to operate on a UTF-8 encoded Unicode file by working on the byte sequence representation of the UTF-8 encoded Unicode character. For example, the Unicode character é, the Latin small letter “e” with an acute accent (Unicode code point \u00e9), has the byte sequence representation (195,169). You can obtain the byte sequence by using `tobytes("é")`. Although you may use 195 and 169 in *regchar* and *code*, they will be treated as two separate bytes instead of one character é (195 followed by 169). In short, this goes beyond the original design of the command and is technically unsupported. If you try to use `filefilter` in this way, you might encounter problems. ❑

Stored results

`filefilter` stores the following in `r()`:

Scalars

`r(occurrences)`	number of *oldpattern* found
`r(bytes_from)`	# of bytes represented by *oldpattern*
`r(bytes_to)`	# of bytes represented by *newpattern*

Reference

Riley, A. R. 2008. Stata tip 60: Making fast and easy changes to files with filefilter. *Stata Journal* 8: 290–292.

Also see

[P] **file** — Read and write text and binary files

[D] **changeeol** — Convert end-of-line characters of text file

[D] **hexdump** — Display hexadecimal report on file