NAME

reform – reformat text file

SYNOPSIS

reform [*tabspec1* [*tabspec2*]] [**+b***n*] [**+e***n*] [**+f**] [**+i***n*] [**+m***n*] [**+p***n*] [**+s**] [**+t***n*]

DESCRIPTION

Reform reads each line of the standard input file, reformats it, and then writes it to the standard output. Various combinations of reformatting operations can be selected, of which the most common involve rearrangement of tab characters.

Reform first scans any arguments, which may be given in any order. It then processes its input file, performing the following actions upon each line, in the order given.

- A line is read from the standard input.
- If **+s** is given, the first 10 characters of the line are stripped off. Characters 1–4 (SCCS Release) and 6–9 (SCCS Level) are saved for later addition to the end of the line.
- The line is expanded into a tabless form, by replacing tabs with blanks according to the input tab specification *tabspec1*.
- If **+p***n* is given, *n* blanks are prepended to the line.
- If **+t***n* is given, the line is truncated to a length of *n* characters.
- All trailing blanks are now removed.
- If **+e***n* is included, the line is extended out with blanks to the length of *n* characters.
- If **+s** is given, the previously saved SCCS Release and Level are added to the end of the line.
- If **+b***n* is given, the *n* characters at the beginning of the line are converted to blanks, if and only if all of them are either digits or blanks.
- If **+m***n* is included, the line is moved left, i.e., *n* characters are removed from the beginning of the line.
- The line is now contracted by replacing some blanks with tab characters according to the list of tabs indicated by the output tab specification *tabspec2*, and is written to the standard output file. Option **+i** controls the method of contraction (see below).

The various arguments accepted by *reform* are as follows:

tabspec1 describes the tab stops assumed for the input file. This tab specification may take on any of the forms described below. In addition, the operand –– indicates that the tab specification is to be found in the first line read from the standard input. If no legal tab specification is found there, **–8** is assumed. If *tabspec1* is omitted entirely, –– is assumed.

tabspec2 describes the tabs assumed for the output file. It is interpreted the same as *tabspec1*, except that omission of *tabspec2* causes the value of *tabspec1* to be used for *tabspec2*.

The remaining arguments are all optional and may be used in any combination, although only a few combinations make much sense. Specifying an argument causes an action to be performed, as opposed to a usual default of not performing the action. Some options include numeric values, which also have default values. Option actions are applied to each line in the order described above. Any line length mentioned applies to the length of a line just previous to the execution of the option described, and the terminating newline is never counted in the line

length.

+b*n* causes the first *n* characters of a line to be converted to blanks, if and only if those characters include only blanks and digits. If *n* is omitted, the default value is 6, which is useful in deleting sequence numbers from COBOL programs.

+e*n* causes each line shorter than *n* characters to be extended out with blanks to that length. Omitting *n* implies a default value of 72. This option is useful for those rare cases in which some sequence numbers need to be added to an existing unnumbered file, e.g., the use of $ in editor regular expressions is more convenient if all lines have equal length. I.e., the user intends to issue editor commands like:

s/$/00001000/

+f causes a format line to be written to the standard output, preceding any other lines written. The format line is taken from *tabspec2*, i.e., the line normally appears as follows:

<:t−*tabspec2* **d:>**

+i*n* controls the technique used to compress interior blanks into tabs. Unless this option is specified, any sequence of 1 or more blanks may be converted to a single tab character if it occurs in an appropriate place. This causes no problems for blanks occurring before the first nonblank character in a line, and it is always possible to convert the result back to an equivalent tabless form. However, occasionally an interior blank (one occurring after the first nonblank) is converted to a tab when this is not intended. For instance, this might occur in any program written in a language utilizing blanks as delimiters. Any single blank might be converted to a tab if it occurs just before a tab stop. Insertion or deletion of characters preceding such a tab may cause it to be interpreted in an unexpected way at a later time. If the **+i** option is used, no string of blanks may be converted to a tab unless there are *n* or more contiguous blanks. The default value is 2. Note that leading blanks are always converted to tabs when possible.

+m*n* causes each line to be moved left *n* characters, with a default value of 6. This can be useful for crunching COBOL programs.

+p*n* causes *n* blanks to be prepended (default of 6 if omitted). This option is effectively the inverse of **+m***n*, and is often useful for adjusting the position of *nroff(1)* output for terminals lacking both forms tractor positioning and settable left margin.

+s is used with the **−m** option of *get(1)*. The **−m** option causes *get* to prepend SCCS Release and Level numbers to each generated line. The **+s** option causes these numbers to be removed from the front of each line, saved, and then later added to the end of the line. Because **+e72** is implied by this option, the effect is to produce 80-character card images with SCCS Release and Level in columns 73−80.

+t*n* causes any line longer than *n* characters to be truncated to that length. If *n* is omitted, the length defaults to 72. Sequence numbers can thus be removed, also permitting additional preceding blanks to be deleted.

Three types of tab specification are accepted for *tabspec:* "canned", repetitive, and arbitrary.

−*code* gives the name of one of a set of "canned" tabs. The legal codes and their meanings are as follows:

−a 1,10,16,36,72
Assembler, IBM S/370, first format

−a2 1,10,16,40,72
Assembler, IBM S/370, second format

−c 1,8,12,16,20,55
COBOL, normal format

−c2 1,6,10,14,49
COBOL compact format (columns 1−6 omitted).

−cx 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
COBOL compact format (columns 1−6 omitted), with more tabs than **−c2.**

−f 1,7,11,15,19,23
FORTRAN

−p 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
PL/I

−s 1,10,55
SNOBOL

−u 1,12,20,44
UNIVAC 1100 Assembler

In addition to these "canned" formats, three other types exist:

−n a repetitive specification requests tabs at columns 1+*n*, 1+2**n*, etc.

n1,n2,... the arbitrary format permits the user to type any chosen set of numbers, separated by commas, in ascending order. Up to 20 numbers are allowed. The maximum tab value accepted is 158. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Thus, the tab lists 1,10,20,30 and 1,10,+10,+10 are considered identical.

DIAGNOSTICS

All diagnostics are fatal, and the offending line is displayed following the message.
"line too long" if any line exceeds 512 characters (in tabless form).
"not SCCS -m" if first 10 characters of line are not in proper format when **+s** flag used.

EXIT CODES

0 − normal
1 − any error

SEE ALSO

get(I), nroff(I)

BUGS

Reform is aware of the meanings of Backspace and Escape sequences, so that it can be used as a postprocessor for *nroff.* However, be warned that the **+e, +m, +t** options only count characters, not positions. Anyone using these options on output containing backspaces or halfline motions will probably obtain unexpected and useless results.