Documentation

About Sed

sed ("stream editor") is a Unix utility that parses and transforms text, using a simple, compact programming language.
Several version/implementations of Sed exist.GNU Sed is widely used, Super-sed is an extended version of sed that includes regular expressions compatible with Perl.
Another variant of sed is minised. minised was used by the GNU Project until the GNU Project wrote a new version of sed based on the new GNU regular expression library.
The current minised contains some extensions to BSD sed but is not as feature-rich as GNU sed.

GNU Sed, BSD Sed, MiniSed, Super Sed, all have different syntax, different extensions.
GNU Sed is considered as base for this implementation. Features that are obsolete or non-portable are not implemented.

Structure of Sed-script

each sed-script line is of this format:

> [selector][NEG][command][flags]

[selector] - Specifies address(es) of lines on which command should be executed
[command] - one of any command
[flags] - flags for command

Grammmar

Sed-script fits into the following grammar:
[file] → [lines]
where each line has a command with optional parameters except commands a, c, and i(these commands can span multiple lines)
lines can be seperated by a newline or semicolon.
[lines] → [lines] SEP [line] | [line]
[line] → [selector][NEG][command][flags] | [label]
selector, NEG(!), flags are optional.
Several commands can be combined using "{" and "}", with ";" seperating each line(commands)
lines/commands can also be grouped using "{" and "}", with ";" seperating each line(commands)
Also sed accepts "{}", ""(empty file), which creates ambiguous grammar.
For example:
Following sed commands are accepted by sed:

```
$sed ''
$sed '{}'
$sed '{p}'
$sed '{{}}'
$sed '{{p}}'
```

However, following commands are not accepted:

```
$sed '{p{}}'
$sed '{{}p}'
$sed '{{p}p}'
$sed '{p{p}}'
```

These conflicts are because of optional end of command ";" and optional selector
Also, several semicolons/newlines can be used to seperate commands/lines.
Following sed commands are accepted by sed:

```
$sed ';'
$sed '{;;};;;'
$sed 'p;;;p'
```

This implementation includes above abnormal positives and to cover all posible parse trees with no ambiguity.
We also added grammar rules (whenever possible, without ambiguity) to correct syntax error where two commands are not seperated by ; or newline but have atleast one space between them.

Commands Implemented

Command	Function
=	print current line number

Command	Function
a	append text after current line
b	branch to label
c	change current line
d	delete all of pattern space
D	delete first line of pattern space
F	print the filename(current input filename) with trailing newline character
g	copy hold space to pattern space
G	append hold space to pattern space
h	copy pattern space to hold space
H	append pattern space to hold space
i	insert text before current line
l	print pattern space in escaped form
n	get next line into pattern space
N	append next line to pattern space
p	print pattern space to output
P	print first line of pattern space
q	exit the stream editor
s	regular-expression substitute
t	branch on last substitute successful
T	branch on last substitute failed
w	write pattern space to file
W	write first line of pattern space
x	exhange pattern and hold spaces
y	transliterate text
z	clear pattern space

Conversion to C and Structure of C

Sed scans each line of inputfile in buffer(patternspace) and maintains a secondary buffer(holdspace, can be used for several operations). Then commands are executed one by one on patternspace, after completion next line is loaded onto patternspace.
produced C file is simiar to how sed works, conatins a while loop to simulate sed-cycles.
Several commands often correspond to large C-code with minor varitaions in C-code with Sed-parameters. Therefore all C-code strings that are long are included in "strs.h"(constant strings) in unescaped form.
[Selectors] are mostly conditions on line numbers or regular expressions.
[commands] are commands which mostly have same C-code.
[NEG] is "!" which negates selector [flags] are for additional control over command. s-command(substitution) supports flags.

flag	Function

flag	Function
g	apply replacement to all matches
p	if substitution is made print the pattern space
i	case-insensitive
I	case-insensitive
number	replace only _number_th match

basic indentation is made for most commands in produced C file. Error checking is minimal and user supplied sed file is assumed to be correct.