```
#!/usr/local/bin/perl -w
# $Id: edfile.perl,v 1.1 2013-04-11 18:53:50-07 - - $
# See the listedit utility man page for specifications.

use strict;
use warnings;
use Getopt::Std;
use IO::Handle;

$0 =˜ s|^(.*/)?([^/]+)/*$|$2|;
my $exit_status = 0;
END{ exit $exit_status; }
sub note(@){ print STDERR "$0: @_"; };
$SIG{’__WARN__’} = sub{ note @_; $exit_status = 1; };
$SIG{’__DIE__’} = sub{ warn @_; exit; };

autoflush STDOUT 1;
autoflush STDERR 1;
my ($dotline, @buffer);

sub readfile($) {
   my ($filename) = @_;
   warn "$filename: $!\n" and return unless open my $file, "<$filename";
   my @input = <$file>;
   close $file;
   printf "%d lines read from $filename\n", scalar @input;
   splice @buffer, $dotline + 1, 0, @input;
   $dotline += @input;
}

sub writefile($) {
   my ($filename) = @_;
   warn "$filename: $!\n" and return unless open my $file, ">$filename";
   print $file @buffer;
   close $file;
   $dotline = $#buffer;
   printf "%d lines written to $filename\n", scalar @buffer;
}

sub printcurr() {
   print "no lines in file\n" and return unless $dotline >= 0;
   printf "%6d  %s", $dotline, $buffer[$dotline]
}

my %switch = (
   #ch => [hasopnd, printcurr, sub]
   ’#’ => [1, 0, sub {}],
   ’$’ => [0, 1, sub {$dotline = $#buffer}],
   ’*’ => [0, 0, sub {$dotline = $_, printcurr for 0..$#buffer}],
   ’.’ => [0, 1, sub {}],
   ’0’ => [0, 1, sub {$dotline = 0}],
   ’<’ => [0, 1, sub {--$dotline}],
   ’>’ => [0, 1, sub {++$dotline}],
   ’a’ => [1, 1, sub {splice @buffer, ++$dotline, 0, "$_[0]\n"}],
   ’d’ => [0, 1, sub {splice @buffer, $dotline, 1 if @buffer}],
   ’i’ => [1, 1, sub {if ($dotline < 0) {@buffer = "$_[0]\n"}
                      else {splice @buffer, $dotline, 0, "$_[0]\n"}}],
   ’r’ => [1, 0, sub {readfile $_[0]}],
   ’w’ => [1, 0, sub {writefile $_[0]}],

);

my $wantecho = ! (-t STDIN && -t STDOUT);
for my $filename (@ARGV) {
   open FILE, "<$filename" or warn "$filename: $!\n" and next;
   push @buffer, <FILE>;
   close FILE;
}
$dotline = $#buffer;
printf "%d lines read\n", scalar @buffer if @buffer;

for(;;) {
   print "$0: ";
   last unless defined (my $line = <STDIN>);
   print $line if $wantecho;
   chomp $line;
   next if $line =~ m/^\s*$/;
   my ($key, $operand) = $line =~ m/(.)(.*)/;
   my ($hasopnd, $printcurr, $sub) = @{$switch{$key} || []};
   warn "invalid command: $line\n" and next
        unless length $key and defined $sub
           and ($hasopnd or (length $operand) == 0);
   $sub->($operand);
   $dotline = 0 if $dotline < 0;
   $dotline = $#buffer if $dotline > $#buffer;
   printcurr if $printcurr;
}

print "^D" if $wantecho;
print "\n";

```

```
# $Id: edscript,v 1.1 2013-04-11 18:53:54-07 - - $
rtest.in
*
aanother line
ianother line
*
<
<
d
<
d
*
wscript.out
```

```
#!/bin/sh -x
# $Id: mk,v 1.2 2014-04-15 19:25:40-07 - - $
FILES="edfile.perl edscript mk test.in"
cid + $FILES
edfile.perl <edscript 2>&1 | tee edscript.log
mkpspdf Listing.test.ps $FILES *.out *.log
```

```
# $Id: test.in,v 1.1 2013-04-11 18:53:50-07 - - $
This is an input test file.
It contains a few lines of data.
It will be written out again.
Look at the typescript logs for an example.
```

```
# $Id: test.in,v 1.1 2013-04-11 18:53:50-07 - - $
This is an input test file.
It contains a few lines of data.
another line
another line
```

```
edfile.perl: # $Id: edscript,v 1.1 2013-04-11 18:53:54-07 - - $
edfile.perl: rtest.in
5 lines read from test.in
edfile.perl: *
     0  # $Id: test.in,v 1.1 2013-04-11 18:53:50-07 - - $
     1  This is an input test file.
     2  It contains a few lines of data.
     3  It will be written out again.
     4  Look at the typescript logs for an example.
edfile.perl: aanother line
     5  another line
edfile.perl: ianother line
     5  another line
edfile.perl: *
     0  # $Id: test.in,v 1.1 2013-04-11 18:53:50-07 - - $
     1  This is an input test file.
     2  It contains a few lines of data.
     3  It will be written out again.
     4  Look at the typescript logs for an example.
     5  another line
     6  another line
edfile.perl: <
     5  another line
edfile.perl: <
     4  Look at the typescript logs for an example.
edfile.perl: d
     4  another line
edfile.perl: <
     3  It will be written out again.
edfile.perl: d
     3  another line
edfile.perl: *
     0  # $Id: test.in,v 1.1 2013-04-11 18:53:50-07 - - $
     1  This is an input test file.
     2  It contains a few lines of data.
     3  another line
     4  another line
edfile.perl: wscript.out
5 lines written to script.out
edfile.perl: ^D
```