pipEdit

Version 0.8.2

Users Guide

The_Piper
the_piper@web.de

January 2020

Contents

Chapter 1

License

This program is under the Gnu Public License, GPL.

Read the file `COPYING` or here:

`https://www.gnu.org/licenses/gpl-3.0.en.html`

for its content.

Excerpt:

Chapter 2

Introduction

pipEdit is meant to be a lookalike editor of the ISPF editor IBM offers on their mainframes, it is not meant to be a 100% clone of it or of the whole ISPF environment.

The plan is to create (and have) an editor which has a similar feeling like the IBM one, but some things might be different.

For example, I havent found out right now how to make a difference between the ENTER key and the ENTER key on the number pad, so one can be the XMIT/SEND key and the other wont.

That, and some other stuff, are some minor differences.

It can be compiled with ncurses and, if not available, with my tiny ncurses replacement, pipcurses, which should work with all vt52, vt100, ansi, xterm terminals.

Chapter 3

Compiling pipEdit

pipEdit ist developed with gcc, the Gnu C compiler under Linux.

To compile it with ncurses type:

```
gcc pipedit.c -opipedit.bin -O -lcurses
```

Without ncurses, using my ncurses replacement, pipcurses, type this:

```
gcc pipedit.c -opipedit.bin -O -D__USE_PIPCURSES__=1
```

Or use the Makefile, edit it to your needs and type `make`

Of course you can try to compile it for Windows, Dev-Cpp, for example, is a free compiler which should be able to compile source code for gcc.

`https://www.bloodshed.net/devcpp.html`

Chapter 4

Configuring pipEdit

pipEdit uses a configuration file to store/read its configuration data, named `pipedit.cfg`.

4.1 Environment variable

To know where `pipedit.cfg` is located, the editor reads an environment variable named `$PIPEDITCFG` which holds the complete path to the config file.

Example:

```
PIPEDITCFG="/home/myname/pipEdit/pipedit.cfg" ; export PIPEDITCFG
```

Define this variable in `.bashrc`, `.kshrc` or whereever it suits you.

4.2 Configuration file

The configuration file itself looks like this:

```
######################################
# Global configuration for pipEdit #
######################################
# target is used in *.par files to have multiple configurations
# for multiple compilers
#
target=GNUCOBOL
###target=MICROFOCUS
#
# Where are the macros located the edtior can use?
#
macros=/home/myname/pipEdit/macros
```

```
#
# language overwrites the environment variable $LANG
# This can be used for language files for the editor, macros and
# so on.
# If left blank, $LANG is used, if $LANG isn't defined "en" is
# used.
# default means, no file is read, the default values of the editor
# are used.
#
###language=default
language=
#
# various translations of texts and messages
#
langfiles=/home/myname/pipEdit/langfiles
#
# Function keys
#
F1=HELP
F2=
F3=END
F4=
F5=RFIND
F6=
F7=UP
F8=DOWN
F9=
F10=LEFT
F11=RIGHT
F12=RETRIEVE
PGUP=UP
PGDOWN=DOWN
```

Chapter 5

Installing pipEdit

To install pipEdit on a Linux system, follow these 3 easy steps:

One
Copy the compiled program, `pipedit.bin`, to a directory which is specified in your `$PATH`, like `/usr/local/bin`.
You might rename it to a shorter name, or create a shell script to call pipedit.bin, like `pe` or such.

Two
Copy the config file `pipedit.cfg` to a location where you like it to be. For example, create a directorey named `.pipedit` in your `$HOME` directory and move `pipedit.cfg` into it.
Edit the config file to your needs, language, PF-keys and such.

Three
Edit the profile of your shell, `.kshrc`, `.bashrc`, or whatever, and define the environment variable `$PIPEDITCFG` and set it to the location of `pipedit.cfg` like:

```
PIPEDITCFG=¨/home/myname/.pipedit/pipedit.cfg¨; export PIPEDITCFG
```

And thats it. Log off and log on and try to start pipEdit.

Chapter 6

Backups

pipEdit creates backups of the file being edited. Just to be safe..

The backups are stored in the `/tmp` folder and are named, for example, we are editing the file `hello.cob`:

`hello.cob.grandfather`	the oldest backup
`hello.cob.father`	the 2nd oldest backup
`hello.cob.son`	the current backup

As you might guess, every editing session, the grandfather is overwritten with the father, the father with the son and the son with the current file before being edited.

Chapter 7

Keys / Function keys

7.1 Editor

F1	Toggle between short and long (error) message
F3	Quit and save
F5	Repeat find (RFIND)
F7 / PgUp	Scroll one page up
F8 / PgDown	Scroll one page down
F10	Scroll left
F11	Scroll right
F12	Retrieve

The function keys are defined in the config file. If you want or must use other function keys for those actions, edit the config file.

For example, the terminal emulation I use, uses F11 to toggle full screen mode. So scroll right on F11 doesn't work very well.

Arrow up	Cursor up
Arrow down	Cursor down
Arrow left	Cursor left
Arrow right	Cursor right
INS	Insert one blank at current position
DEL	Deletes one character at current position

7.2 Cancel window

F1	Yes, discard all changes and leave the editor
F12	No, do not discard, stay in editor

Chapter 8

Command line commands

The command line is the line near the top of the screen marked `Command ===>`, where you can, well..., type commands.
Hit `ENTER` key and the editor will process the typed command.

Save	Save the current file
save4macro	Save the current file in a format macros use
res / reset	Remove all message lines
can / cancel	Cancel editing, must be confirmed again
cols	Toggles the display of columns above code
l / loc	Locate a line number
f / find	Find string in text

Chapter 9

Line commands

Line commands are typed at the line number of a text line.

d	Delete line
i	Insert blank line
r	Repeat line
x	eXclude line

Every command takes a number as a parameter. So `d3` deletes 3 lines, `i5` inserts 5 blank lines, `r2` repeats the current line two times and `x7` excludes 7 lines.

Chapter 10

Block commands

Block commands affect a block of lines (surprise, surprise...), like a block of lines you want to delete, copy or repeat.

A block is marked with the given block commands, so type `dd` in the line number area to mark the beginning of a block you want to delete, and again `dd` to mark the end of this block.

Example:

```
000005 line5
dd'''' line6
000007 line7
000008 line8
dd'''' line9
000010 line10
```

Then hit the `ENTER` key and the lines will be deleted.

The block commands are:

RR	Repeat a block of lines, duplicate it right after the last line of the block.
DD	Delete the block of lines marked with `dd dd`.
CC	Copy the block after a line marked with `a` or before a line marked with `b`.
XX	eXclude the lines of the block.

Chapter 11

Macros

pipEdit supports macros.

A macro is just a program, script, whatever, which is called by pipEdit, getting a defined number of parameters and reads and modifies a text file.

11.1 Parameters for macros

11.1.1 File name

This is the name of the temporary file pipEdit writes before calling the macro and reads after the macro is done.

The macro changes this file, like the comp macro, which inserts message lines of the error messages of the compile into the source code.

The format of this file is this (and the result pipEdit reads, must be the same format):

The first 5 lines:

```
msg=Done
message=Hex dump done
cursor_x=-1
cursor_y=-1
modified=0
```

`msg`	the short message the macro can set
`message`	the long message (F1) the macro can set
`cursor_x`	the x position of the cursor the macro can set
`cursor_y`	the y position of the cursor the macro can set
`modified`	did the macro modify the source code, yes (1) or no (0). This is needed to tell the editor if the source code must be saved or not.

All lines after the first 5 ones:

Bytes	**Type**	**Content**
0-5	Line number	6 digits line number
6	Line type	I=inserted line
		M=message
		E=Error
		N=normal text line
		X=eXcluded line
7-nnn	Text	The content of the line

The line number itself is ignored when pipEdit reads the result of the macro again. The line number is used by the comp macro to find the right line in the source code where the error messages are displayed.

11.1.2 Parameter file

A parameter file, `filename.par`, looks like this (this is a file for the comp macro):

```
##################################################
### M i c r o f o c u s ##############################################
@target=MICROFOCUS
pre=#!/bin/bash
pre=. /adm/config/basis.prof
pre=export COBCPY=$HOME/Projects/cpy
###——————————————-
compiler=cob
options=-x -P
binary=hello.bin
listing=hello.lst
movebinto=/home/yourname/yourlocation/Local_bin/
movelstto=/home/yourname/yourlocation/Listings/
removetmp=.idy .int .cs9Filename
###post=rm *.idy *.int *.cs9
##################################################
### G n u C o b o l
##################################################
@target=GNUCOBOL
pre=#!/bin/bash
compiler=cobc
options=-x -Thello.lst
binary=hello.bin
listing=hello.lst
movebinto=/home/yourname/yourlocation/Binaries/
movelstto=/home/yourname/yourlocation/Listings/
###——————————————-
post=exit
post=# And thats it
```

Macros can read this file, in this case the comp macro.

This file specifies how to compile COBOL programs for Microfocus or GnuCobol.

Which compiler is used is specified with the @target= tag. @target=MICROFOCUS or @target=GNUCOBOL

All lines following the right @target= tag will be used by the comp macro to compile the current source code.

The @target= itself is specified in the pipedit.cfg file.

The pre= lines are written at the beginning of the compile shell script.

The post= lines are written at the end of that script.

Between those lines the comp macro generates code for the compile from the tags compiler= name of the COBOL compiler options= options for the compiler binary= name of the output, the binary listing= name of the listing file movebinto= Where the result of the compile, the binary, is moved to movelstto= where the listing file should be moved to

This is an example of a *.par file for the comp macro. When you write your own macros, you will create your own parameter files, fitting to the needs of the macros.

11.1.3 Filename

The original name of the file currently being edited.

11.1.4 Config file

This is pipedit.cfg, or however you name it in the environment variable, the macro can read that config file too and use its values.

Those parameters are given to the macro, the macro can use them to read configuration values from the *.par file or from the config file, process and modify the termporary file, and thats it right now.

That is how macros work with pipEdit.

Further plans: Return something like error messages displayed where pipEdit displays its own error messages, relocating the cursor and such.

Chapter 12

comp Macro

The comp macro is a macro which compiles the (COBOL) source code loaded into pipEdit and displays error messages as message lines right into the source code.

Chapter 13

hex Macro

The hex macro inserts message lines of the hex code of every line of the source code into the source code itself.

Type hex to show the hex codes, type res or reset to get rid of them.

Chapter 14

cs Macro

The cs macro (Copy to Scratch file) can copy a block of lines into a temporary file in the `/tmp` folder and, after leaving the editor and editing a different file, copy this temporary file back into the editors content.

Copy a block of lines to a temporary file
Mark a block of lines by typing `.a` and `.b` in the line numbers area of the editor.
Then type `cs` in the command line to copy the lines into the temporary file.

Example:

```
    Command ==> cs
000001 line1
000002 line2
000003 line3
000004 line4
000005 line5
.a'''' line6
000007 line7
000008 line8
.b'''' line9
000010 line10
```

Then hit the `ENTER` key and the lines 6, 7, 8 and 9 will be copied into a temporary file.

Copy a block of lines from the temporary file into a different source code

Specify the location where the block of lines should be copied to with `a` (after this line) or `b` (before this line) (and without a dot ”.”) in the line numbers area of the editor.

Then type `cs` in the command line to copy the lines from the temporary file into the current source code.

Example:

```
    Command ==> cs
000001 line1
000002 line2
000003 line3
000004 line4
000005 line5
a’’’’’ line6
000007 line7
000008 line8
000009 line9
000010 line10
```