NAME

diffpp − pretty−print diff outputs with GNU enscript

SYNOPSIS

diffpp *currentfile* < *diff−file*

DESCRIPTION

Diffpp converts **diff(1)**−program's output files to a format suitable to be printed with GNU **enscript(1)**. Program annotates the changes with enscript's special escapes so enscript can highlight the modified portions of the file. All changed and added lines are printed with gray−background, deleted lines are marked with minus ('−') characters; **diffpp** prints one minus character for each deleted line.

EXAMPLES

The easiest way to use **diffpp** is to use it as an input filter for enscript. If an input filter is specified for enscript it is used to pre−process the incoming data−stream. Filtering does not alter any header strings or file−timestamps which might be printed on enscript headers; only the incoming data is modified.

enscript −G2re −−filter='rcsdiff %s | diffpp %s' *.c *.h
Print the changes between current source files and their latest RCS−versions.

enscript −G2re −−filter='diff %s~ %s | diffpp %s' *.c *.h
Print changes between source files and the corresponding backup−files.

SEE ALSO

diff(1), enscript(1)

AUTHOR

Markku Rossi <mtr@iki.fi> <http://www.iki.fi/~mtr/>

NAME

enscript – convert text files to PostScript, HTML, RTF, ANSI, and overstrikes

SYNOPSIS

enscript [**–123456789BcgGhjkKlmOqrRvVzZ**] [**–#** *copies*] [**–a** *pages*] [**–A** *align*] [**–b** *header*] [**–C**[*start_line*]] [**–d** *printer*] [**–D** *key*[**:***value*]] [**–e**[*char*]] [**–E**[*lang*]] [**–f** *font*] [**–F** *header_font*] [**–H**[*num*]] [**–i** *indent*] [**–I** *filter*] [**–J** *title*] [**–L** *lines_per_page*] [**–M** *media*] [**–n** *copies*] [**–N** *newline*] [**–o** *outputfile*] [**–o –**] [**–p** *outputfile*] [**–p –**] [**–P** *printer*] [**–s** *baselineskip*] [**–S** *key*[**:***value*]] [**–t** *title*] [**–T** *tabsize*] [**–u**[*text*]] [**–U** *num*] [**–w** *language*] [**–X** *encoding*] [*filename ...*]

DESCRIPTION

Enscript converts text files to PostScript or to other output languages. **Enscript** can spool the generated output directly to a specified printer or leave it to a file. If no input files are given, **enscript** processes the standard input **stdin**. **Enscript** can be extended to handle different output media and it has many options which can be used to customize the printouts.

OPTIONS

–# *num* Print *num* copies of each page.

–1, –2, –3, –4, –5, –6, –7, –8, –9, ––columns=*num*
Specify how many columns each page have. With the long option **––columns=***num* you can specify more than 9 columns per page.

–a *pages***, ––pages=***pages*
Specify which pages are printed. The page specification *pages* can be given in the following formats:

begin–end	print pages from *begin* to *end*
–end	print pages from 0 to *end*
begin–	print pages from *begin* to end
page	print page *page*
odd	print odd pages
even	print even pages

–A *align***, ––file–align=***align*
Align separate input files to even *align* page count. This option is useful in two-side and 2-up printings (––file–align=2).

–b *header***, ––header=***header*
Use the text *header* as a page header. The default page header is constructed from the name of the file and from its last modification time.

The header string *header* can contain the same formatting escapes which can be specified for the **%Format** directives in the user defined fancy headers. For example, the following option prints the file name, current data and page numbers:

enscript ––header='$n %W Page $% of $=' *.c

The header string can also contain left, center and right justified fields. The fields are separated by the '|' character:

enscript ––header='$n|%W|Page $% of $=' *.c

now the file name is printed left justified, the date is centered to the header and the page numbers are printed right justified.

−B, −−no−header
Do not print page headers.

−c, −−truncate−lines
Cut lines that are too long for the page. As a default, **enscript** wraps long lines to the next line so no information is lost.

You can also use the **−−slice** option which slices long lines to separate pages.

−C[*start_line*]**, −−line−numbers**[=*start_line*]
Precede each line with its line number. The optional argument *start_line* specifies the number of the first line in the input. The number of the first line defaults to 1.

−d *name*
Spool output to the printer *name*.

−D *key*[**:***value*]**, −−setpagedevice=***key*[**:***value*]
Pass a page device definition to the generated PostScript output. If no value is given, the key *key* is removed from the definitions.

For example, the command

enscript −DDuplex:true foo.txt

prints file foo.txt in duplex (two side) mode.

Page device operators are implementation dependant but they are standardized. See section **PAGE DEVICE OPTIONS** for the details.

−e[*char*]**, −−escapes**[=*char*]
Enable special escapes interpretation (see section **SPECIAL ESCAPES**). If the argument *char* is given, it changes the escape character to *char*. The default escape character is 0.

−E[*lang*]**, −−highlight**[=*lang*]
Highlight source code by creating a special input filter with the **states** program. The optional argument *lang* specifies the language to highlight. As a default the **states** makes an educated guess.

You can print a short description of the supported highlighting languages and file formats with the command:

enscript −−help−highlight

The highlighting rules are defined in the ‘c:/progra˜1/enscript/share/enscript/st/*.st’ files which can be edited to create highlighting definitions for new languages.

Note! You can not use your own input filters with this option.

−f *name***, −−font=***name*
Select a font that is used for the body text. The default body font is **Courier10**, unless multi-column landscape printing mode is selected, in which case the default font is **Courier7**.

The font specification *name* contains two parts: the name of the font and its size in PostScript points. For example, "**Times−Roman12**" selects the "Times−Roman" font with size 12pt.

The font specification *name* can also be given in format ‘*name@ptsize*’, where the name of the font and its point size are separated by a ‘@’ character. This allows **enscript** to use fonts which contain digit characters in their names.

The font point size can also be given in the format *width/height* where the *width* and the *height* specify the size of the font in x- and y-directions. For example, "**Times−Roman@10/12**" selects a 10 points wide and 12 points high "Times−Roman" font.

You can also give the font sizes as decimal numbers. For example, "**Times−Roman10.2**" selects a 10.2pt "Times−Roman" font.

−F *name*, **−−header−font=***name*
Select a font for the header texts.

−g, −−print−anyway
Print a file even if it contains binary data. The option is implemented only for compatibility purposes. **Enscript** prints binary files anyway regardless of the option.

−G, −−fancy−header[=*name*]
Print a fancy page header *name* to the top of each page. The option **−G** specifies the default fancy header. See section **CONFIGURATION FILES** to see how the default fancy header can be changed.

−h, −−no−job−header
Suppress printing of the job header page.

−H[*num*], **−−highlight−bars**[=*num*]
Specify how high the highlight bars are in lines. If the *num* is not given, the default value 2 is used. As a default, no highlight bars are printed.

−i *num*, **−−indent=***num*
Indent every line *num* characters. The indentation can also be specified in other units by appending an unit specifier after the number. The possible unit specifiers and the corresponding units are:

c	centimeters
i	inches
l	characters (default)
p	PostScript points

−I *filter*, **−−filter=***filter*
Read all input files through an input filter *filter*. The input filter can be a single command or a command pipeline. The filter can refer to the name of the input file with the escape '**%s**'. The name of the standard input can be changed with the option '**−−filter−stdin**'.

For example, the following command prints the file 'foo.c' by using only upper-case characters:

```
enscript −−filter="cat %s | tr 'a-z' 'A-Z'" foo.c
```

The following command highlights changes which are made to files since the last checkout:

```
enscript −−filter="rcsdiff %s | diffpp %s" −e *.c
```

To include the string "%s" to the filter command, you must write it as "%%s".

−j, −−borders
Print borders around columns.

−J *title* An alias for the option **−t**, **−−title**.

−k, −−page−prefeed
Enable page prefeed.

−K, −−no−page−prefeed
Disable page prefeed (default).

−l, −−lineprinter
Emulate lineprinter. This option is a shortcut for the options **−−lines−per−page=66**, and **−−no−header**.

−L *num*, **−−lines−per−page=***num*
Print only *num* lines for each page. As a default, the number of lines per page is computed from the height of the page and from the size of the font.

−m, −−mail
Send a mail notification to user after the print job has been completed.

−M *name*, **−−media=***name*
Select an output media *name*. **Enscript**'s default output media is **A4**.

−n *num*, **−−copies=***num*
Print *num* copies of each page.

−N *nl*, **−−newline=***nl*
Select the *newline* character. The possible values for *nl* are: **n** (unix newline, 0xa hex) and **r** (mac newline, 0xd hex).

−o *file* An alias for the option **−p**, **−−output**.

−O, −−missing−characters
Print a listing of character codes which couldn't be printed.

−p *file*, **−−output=***file*
Leave the output to file *file*. If the *file* is '−', enscript sends the output to the standard output **stdout**.

−P *name*, **−−printer=***name*
Spool the output to the printer *name*.

−q, −−quiet, −−silent
Make **enscript** really quiet. Only fatal error messages are printed to *stderr*.

−r, −−landscape
Print in the landscape mode; rotate page 90 degrees.

−R, −−portrait
Print in the portrait mode (default).

−s *num*, **−−baselineskip=***num*
Specify the baseline skip in PostScript points. The number *num* can be given as a decimal number. When **enscript** moves from line to line, the current point *y* coordinate is moved (*font point size* + *baselineskip*) points down. The default baseline skip is 1.

−S *key*[**:***value*], **−−statusdict=***key*[**:***value*]
Pass a statusdict definition to the generated PostScript output. If no value is given, the key *key* is removed from the definitions.

The statusdict operators are implementation dependant; see the printer's documentation for the details.

For example, the command

enscript −Ssetpapertray:1 foo.txt

prints the file *foo.txt* by using paper from the paper tray 1 (assuming that the printer supports paper tray selection).

−t *title*, **−−title=***title*
Set banner page's job title to *title*. The option sets also the name of the input file **stdin**.

−T *num*, **−−tabsize=***num*
Set the tabulator size to *num* characters. The default is 8.

−u[*text*], **−−underlay**[=*text*]
Print the string *text* under every page. The properties of the text can be changed with the options **−−ul−angle**, **−−ul−font**, **−−ul−gray**, **−−ul−position**, and **−−ul−style**.

If no *text* is given, the underlay is not printed. This can be used to remove an underlay text that was specified with the '**Underlay**' configuration file option.

−U *num*, **−−nup=***num*
Print *num* logical pages on each output page (N−up printing). The values *num* must be a power of 2.

−v, −−verbose[=*level*]
Tell what **enscript** is doing.

−V, −−version
Print **enscript** version information and exit.

−w [*lang*], **−−language**[=*lang*]
Generate output for the language *lang*. The possible values for *lang* are:

PostScript	generate PostScript (default)
html	generate HTML
overstrike	generate overstrikes (line printers, less)
rtf	generate RTF (Rich Text Format)
ansi	generate ANSI terminal control codes

−X *name*, **−−encoding=***name*
Use the input encoding *name*. Currently **enscript** supports the following encodings:

88591, latin1	ISO−8859−1 (ISO Latin1) (**enscript**'s default encoding).
88592, latin2	ISO−8859−2 (ISO Latin2)
88593, latin3	ISO−8859−3 (ISO Latin3)
88594, latin4	ISO−8859−4 (ISO Latin4)
88595, cyrillic	ISO−8859−5 (ISO Cyrillic)
88597, greek	ISO−8859−7 (ISO Greek)
88599, latin5	ISO−8859−9 (ISO Latin5)
885910, latin6	ISO−8859−10 (ISO Latin6)
ascii	7−bit ascii
asciifise, asciifi, asciise	7−bit ascii with some scandinavian (Finland, Sweden) extensions
asciidkno, asciidk, asciino	7−bit ascii with some scandinavian (Denmark, Norway) extensions
ibmpc, pc, dos	IBM PC charset
mac	Mac charset
vms	VMS multinational charset
hp8	HP Roman-8 charset
koi8	Adobe Standard Cyrillic Font KOI8 charset
ps, PS	PostScript font's default encoding

pslatin1, ISOLatin1Encoding
PostScript interpreter's 'ISOLatin1Encoding'

−z, −−no−formfeed
Turn off the form feed character interpretation. The form feed characters are interpreted as they were newline characters.

−Z, −−pass−through
Pass through all PostScript and PCL files without any modifications. This allows that **enscript** can be used as a lp filter.

The PostScript files are recognized by looking up the '%!' magic cookie from the beginning of the file. **Note! Enscript** recognized also the Windoze damaged '^D%!' cookie.

The PCL files are recognized by looking up the '^[E' or '^[%' magic cookies from the beginning of the file.

−−color[=*bool*]
Use colors in the highlighting outputs.

−−continuous−line−numbers
Don't reset the printed page number to one on every file. If you print many files, the page numbers will continue incrementing throughout all of the files.

−−download−font=*fontname*
Include the font description file (*.pfa* or *.pfb* file) of the font *fontname* to the generated output.

−−extended−return−values
Enable extended return values. As a default, **enscript** returns 1 on error and 0 otherwise. The extended return values give more details about the printing operation. See the section **RETURN VALUE** for the details.

−−filter−stdin=*name*
Specify how the **stdin** is shown to the input filter. The default value is an empty string ("") but some programs require that the **stdin** is called something else, usually "-".

−−footer=*footer*
Use the text *footer* as a page footer. Otherwise the option works like the **−−header** option

−−h−column−height=*height*
Set the horizontal column height to be *height* PostScript points. The option sets the formfeed type to **horizontal−columns**.

−−help Print a short help message and exit.

−−help−highlight
Describe all supported **−−highlight** languages and file formats.

−−highlight−bar−gray=*gray*
Specify the gray level which is used in printing the highlight bars.

−−list−media
List the names of all known output media and exit successfully.

−−margins=*left***:***right***:***top***:***bottom*
Adjust the page marginals to be exactly *left*, *right*, *top* and *bottom* PostScript points. Any of the arguments can be left empty in which case the default value is used.

−−mark−wrapped−lines[=*style*]
Mark wrapped lines in the output with the style *style*. The possible values for the *style* are:

none	do not mark them (default)
plus	print a plus (+) character to the end of each wrapped line
box	print a black box to the end of each wrapped line
arrow	print a small arrow to the end of each wrapped line

−−non−printable−format=*format*
Specify how the non-printable characters are printed. The possible values for the *format* are:

caret caret notation: ‘^@’, ‘^A’, ‘^B’, ...

octal octal notation: ‘\000’, ‘\001’, ‘\002’, ... (default)

questionmark
replace non-printable characters with a question mark ‘?’

space replace non-printable characters with a space ‘ ’

−−nup−columnwise
Change the layout of the sub-pages in the N−up printing from row-wise to columnwise.

−−nup−xpad=*num*
Set the page x-padding of the *n*-up printing to *num* PostScript points. The default is 10 points.

−−nup−ypad=*num*
Set the page y-padding of the *n*-up printing to *num* PostScript points. The default is 10 points.

−−page−label−format=*format*
Set the page label format to *format*. The page label format specifies how the labels for the ‘%%Page:’ PostScript comments are formatted. The possible values are:

short Print the current pagenumber: ‘%%Page: (1) 1’ (default)

long Print the current filename and pagenumber: ‘%%Page: (main.c: 1) 1’

−−ps−level=*level*
Set the PostScript language level that **enscript** uses for its output to *level*. The possible values are **1**, and **2**.

−−printer−options=*options*
Pass extra options to the printer command.

−−rotate−even−pages
Rotate each even−numbered page 180 degrees.

−−slice=*num*
Print the vertical slice *num*. The slices are vertical regions of input files. A new slice starts from the point where the line would otherwise be wrapped to the next line. The slice numbers start from 1.

−−style=*style*
Set the highlighting style to *style*. The possible values are: **a2ps**, **emacs**, **emacs-verbose**, **ifh**, and **msvc**.

−−swap−even−page−margins
Swap left and right page margins for even−numbered pages.

−−toc Print a table of contents to the end of the output.

−−word−wrap
Wrap long lines from word boundaries.

−−ul−angle=*angle*
Set the angle of the underlay text to *angle*. As a default, the angle is **atan(−d_page_h, d_page_w)**.

−−ul−font=*name*
Select a font for the underlay text. The default underlay font is **Times-Roman200**.

−−ul−gray=*num*
Print the underlay text with the gray value *num* (0 ... 1), the default gray value is .8.

−−ul−position=*position_spec*
Set the underlay text's starting position according to the *position_spec*. The position specification must be given in format: ‘*sign xpos sign ypos*’, where the *sign* must be ‘+’ or ‘-’. The positive dimensions are measured from the lower left corner and the negative dimensions

from the upper right corner. For example, the specification '+0-0' specifies the upper left corner and '-0+0' specifies the lower right corner.

−−ul−style=*style*
Set the underlay text's style to *style*. The possible values for *style* are:

outline	print outline underlay texts (default)
filled	print filled underlay texts

CONFIGURATION FILES

Enscript reads configuration information from the following sources (in this order): command line options, environment variable **ENSCRIPT**, user's personal configuration file (**$HOME/.enscriptrc**), site configuration file (**c:/progra˜1/enscript/etc/enscriptsite.cfg**) and system's global configuration file (**c:/progra˜1/enscript/etc/enscript.cfg**).

The configuration files have the following format:

Empty lines and lines starting with '#' are comments.

All other lines are option lines and have format:

option [*arguments* ...].

The following options can be specified:

AcceptCompositeCharacters: *bool*
Specify whether PostScript font's composite characters are accepted as printable or if they should be considered as non-existent. The default value is false (0).

AFMPath: *path*
Specifies the search path for the *AFM* files.

AppendCtrlD: *bool*
Specify if the Control-D (ˆD) character should be appended to the end of the output. The default value is false (0).

Clean7Bit: *bool*
Specify how characters greater than 127 are printed. The valuee true (1) generates 7-bit clean code by escaping all characters greater than 127 to the backslash-octal notation (default). The value false (0) generates 8-bit PostScript code leaving all characters untouched.

DefaultEncoding: *name*
Select the default input encoding. The encoding name *name* can be one of the values of the option **−X**, **−−encoding**.

DefaultFancyHeader: *name*
Select the default fancy header. The default header is used when the option **−G** is specified or the option **−−fancy−header** is given without an argument. The system−wide default is '**enscript**'.

DefaultMedia: *name*
Select the default output media.

DefaultOutputMethod: *method*
Select the default target to which the generated output is sent. The possible values for the *method* are:

printer	send output to printer (default)
stdout	send output to **stdout**

DownloadFont: *fontname*
Include the font description file of the font *fontname* to the generated output.

EscapeChar: *num*
Specify the escape character for the special escapes. The default value is 0.

FormFeedType: *type*
Specify what to do when a formfeed character is encountered from the input. The possible values for *type* are:

column	move to the beginning of the next column (default)
page	move to the beginning of the next page

GeneratePageSize: *bool*
Specify whether the **PageSize** page device setting is generated to the PostScript output. The default value is true (1).

HighlightBarGray: *gray*
Specify the gray level which is used to print the highlight bars.

HighlightBars: *num*
Specify how high the highlight bars are in lines. The default value is 0 which means that no highlight bars are printed.

LibraryPath: *path*
Specifies the **enscript**'s library path that is used to lookup various resources. The default path is: 'c:/progra~1/enscript/share/enscript:*home*/.enscript'. Where the *home* is the user's home directory.

MarkWrappedLines: *style*
Mark wraped lines in the output with the style *style*. The possible values for the *format* are the same which can be given for the **−−mark−wrapped−lines** option.

Media: *name width height llx lly urx ury*
Add a new output media with the name *name*. The physical dimensions of the media are *width* and *height*. The bounding box of the Media is specified by the points (*llx*, *lly*) and (*urx*, *ury*). **Enscript** prints all graphics inside the bounding box of the media.

User can select this media with option **−M** *name*.

NoJobHeaderSwitch: *switch*
Specify the spooler option to suppress the print job header page. This option is passed to the printer spooler when the **enscript**'s option **−h**, **−−no−job−header** is selected.

NonPrintableFormat: *format*
Specify how the non-printable characters are printed. The possible values for *format* are the same which can be given for the **−−non−printable−format** option.

OutputFirstLine: *line*
Set the PostScript output's first line to *line*. The default value is **PS-Adobe-3.0**. Since some printers do not like DSC levels greater than 2.0, this option can be used to change the output first line to something more suitable like **%!PS-Adobe-2.0** or **%!**.

PageLabelFormat: *format*
Set the page label format to *format*. The possible values for *format* are the same which can be given for the **−−page−label−format** option.

PagePrefeed: *bool*
Enable / disable page prefeed. The default value is false (0).

PostScriptLevel: *level*
Set the PostScript language level, that **enscript** uses for its output, to *level*. The possible values for *level* are the same which can be given for the **−−ps−level** option.

Printer: *name*
Names the printer to which the output is spooled.

QueueParam: *name*
The spooler command switch to select the printer queue, e.g. **−P** in **lpr −Pps**. This option can also be used to pass other flags to the spooler command. These options must be given before the queue switch.

SetPageDevice: *key*[**:***value*]
Pass a page device definition to the generated PostScript output.

Spooler: *name*
Names the printer spooler command. **Enscript** pipes generated PostScript to the command *name*.

StatesBinary: *path*
Define an absolute path to the **states** program.

StatesColor: *bool*
Should the **states** program generate color outputs.

StatesConfigFile: *file*
Read highlighting states configuration from the file *file*. The default config file is 'c:/progra~1/enscript/share/enscript/hl/enscript.st'.

StatesHighlightStyle: *style*
Set the highlight style to *style*.

StatesPath: *path*
Define the path for the **states** program. The **states** program will lookup its state definition files from this path. The default value is '$HOME/.enscript:c:/progra~1/enscript/share/enscript/hl'.

StatusDict: *key*[**:***value*]
Pass a statusdict definition to the generated PostScript output.

TOCFormat: *format*
Format table of contents entries with the format string *format*. The format string *format* can contain the same escapes which are used to format header strings with the '%Format' special comment.

Underlay: *text*
Print string *text* under every page.

UnderlayAngle: *num*
Set the angle of the underlay text to *num*.

UnderlayFont: *fontspec*
Select a font for the underlay text.

UnderlayGray: *num*
Print the underlay text with the gray value *num*.

UnderlayPosition: *position_spec*
Set the underlay text's starting position according to the *position_spec*.

UnderlayStyle: *style*
Set the underlay text's style to *style*.

FANCY HEADERS

Users can create their own fancy headers by creating a header description file and placing it in a directory which is in **enscript**'s library path. The name of the header file must be in format: '*headername*.hdr'. Header can be selected by giving option: **−−fancy−header=***headername*.

Header description file contains PostScript code that paints the header. Description file must provide procedure **do_header** which is called by **enscript** at the beginning of every page.

Header description file contains two parts: comments and code. Parts are separated by a line containing text:

```
% −− code follows this line −−
```

Enscript copies only the code part of description file to the generated PostScript output. The comments part can contain any data, it is not copied. If separator line is missing, no data is copied to output.

Enscript defines following constants which can be used in header description files:

d_page_w	page width
d_page_h	page height
d_header_x	header lower left *x* coordinate
d_header_y	header lower left *y* coordinate
d_header_w	header width
d_header_h	header height
d_footer_x	footer lower left *x* coordinate
d_footer_y	footer lower left *y* coordinate
d_footer_w	footer width
d_footer_h	footer height
d_output_w	width of the text output area
d_output_h	height of the text output area
user_header_p	predicate which tells if user has defined his/her own header string: **true/false**
user_header_left_str	if **user_header_p** is **true**, this is the left field of the user supplied header string.
user_header_center_str	if **user_header_p** is **true**, this is the center field of the user supplied header string
user_header_right_str	if **user_header_p** is **true**, this is the right field of the user supplied header string
user_footer_p	predicate which tells if user has defined his/her own footer string: **true/false**
user_footer_left_str	if **user_footer_p** is **true**, this is the left field of the user supplied footer string.
user_footer_center_str	if **user_footer_p** is **true**, this is the center field of the user supplied footer string
user_footer_right_str	if **user_footer_p** is **true**, this is the right field of the user supplied footer string
HF	standard header font (from **−F**, **−−header−font** option). This can be selected simply by invoking command: '**HF setfont**'.
pagenum	the number of the current page
fname	the full name of the printed file (/foo/bar.c)
fdir	the directory part of the file name (/foo)
ftail	file name without the directory part (bar.c)
gs_languagelevel	PostScript interpreter's language level (currently 1 or 2)

You can also use the following special comments to customize your headers and to specify some extra options. Special comments are like DSC comments but they start with a single '%' character; special comments start from the beginning of the line and they have the following syntax:

%commentname: *options*

Currently **enscript** support the following special comments:

%Format: *name format*

Define a new string constant *name* according to the format string *format*. Format string start from the first non-space character and it ends to the end of the line. Format string can contain general '%' escapes and input file related '$' escapes. Currently following escapes are supported:

%%	character '%'
$$	character '$'
$%	current page number
$=	number of pages in the current file
$p	number of pages processed so far
$(*VAR***)**	value of the environment variable *VAR*.
%c	trailing component of the current working directory
%C (**$C**)	current time (file modification time) in 'hh:mm:ss' format
%d	current working directory
%D (**$D**)	current date (file modification date) in 'yy-mm-dd' format
%D{*string***}** (**$D{***string***}**)	format string *string* with the strftime(3) function. '**%D{}**' refers to the current date and '**$D{}**' to the input file's last modification date.
%E (**$E**)	current date (file modification date) in 'yy/mm/dd' format
%F (**$F**)	current date (file modification date) in 'dd.mm.yyyy' format
%H	document title
$L	number of lines in the current input file. This is valid only for the toc entries, it can't be used in header strings.
%m	the hostname up to the first '.' character
%M	the full hostname
%n	the user login name
$n	input file name without the directory part
%N	the user's pw_gecos field up to the first ',' character
$N	the full input file name
%t (**$t**)	current time (file modification time) in 12-hour am/pm format
%T (**$T**)	current time (file modification time) in 24-hour format 'hh:mm'
%* (**$***)	current time (file modification time) in 24-hour format with seconds 'hh:mm:ss'
$v	the sequence number of the current input file
$V	the sequence number of the current input file in the 'Table of Contents' format: if the **−−toc** option is given, escape expands to '*num*−'; if the **−−toc** is not given, escape expands to an empty string.
%W (**$W**)	current date (file modification date) in 'mm/dd/yy' format

All format directives except '$=' can also be given in format

escape width directive

where *width* specifies the width of the column to which the escape is printed. For example, escape "$5%" will expand to something like " 12". If the width is negative, the value will be printed left-justified.

For example, the 'emacs.hdr' defines its date string with the following format comment:

%Format: eurdatestr %E

which expands to:

/eurdatestr (96/01/08) def

%HeaderHeight: *height*
Allocate *height* points space for the page header. The default header height is 36 points.

%FooterHeight: *height*
Allocate *height* points space for the page footer. The default footer height is 0 points.

According to Adobe's Document Structuring Conventions (DSC), all resources needed by a document must be listed in document's prolog. Since user's can create their own headers, **enscript** don't know what resources those headers use. That's why all headers must contain a standard DSC comment that lists all needed resources. For example, used fonts can be listed with following comment:

%%DocumentNeededResources: font *fontname1 fontname2*

Comment can be continued to the next line with the standard continuation comment:

%%+ font *fontname3*

SPECIAL ESCAPES

Enscript supports special escape sequences which can be used to add some page formatting commands to ASCII documents. As a default, special escapes interpretation is off, so all ASCII files print out as everyone expects. Special escapes interpretation is activated by giving option **−e**, **−−escapes** to **enscript**.

All special escapes start with the escape character. The default escape character is ^@ (octal 000); escape character can be changed with option **−e**, **−−escapes**. Escape character is followed by escape's name and optional options and arguments.

Currently **enscript** supports following escapes:

bgcolor change the text background color. The syntax of the escape is:

^@bgcolor{*red green blue*}

where the color components *red*, *green*, and blue are given as decimal numbers between values 0 and 1.

bggray change the text background color. The syntax of the escape is:

^@bggray{*gray*}

where *gray* is the new text background gray value. The default value is 1.0 (white).

color change the text color. The syntax of the escape is:

^@color{*red green blue*}

where color components *red*, *green* and *blue* are given as decimal numbers between values 0 and 1.

comment
comment the rest of the line including the newline character. Escape's syntax is:

^@comment *text newline_character*

escape change the escape character. The syntax of the escape is:

^@escape{*code*}

where *code* is the decimal code of the new escape character.

epsf inline EPS file to the document. The syntax of the escape is:

^@epsf[*options*]{*filename*}

where *options* is an optional sequence of option characters and values enclosed with brackets and *filename* is the name of the EPS file.

If *filename* ends to the '|' character, then *filename* is assumed to name a command that prints EPS data to its standard output. In this case, **enscript** opens a pipe to the specified command and reads EPS data from pipe.

Following options can be given for the **epsf** escape:

c	print image centered
r	print image right justified
n	do not update current point. Following output is printed to that position where the current point was just before the **epsf** escape
nx	do not update current point *x* coordinate
ny	do not update current point *y* coordinate
x*num*	move image's top left *x* coordinate *num* characters from current point *x* coordinate (relative position)
x*num***a**	set image's top left *x* coordinate to column *num* (absolute position)
y*num*	move image's top left *y* coordinate *num* lines from current line (relative position)
y*num***a**	set image's top left *y* coordinate to line *num* (absolute position)
h*num*	set image's height to *num* lines
s*num*	scale image with factor *num*
sx*num*	scale image in *x* direction with factor *num*
sy*num*	scale image in *y* direction with factor *num*

As a default, all dimensions are given in lines (vertical) and characters (horizontal). You can also specify other units by appending an unit specifier after number. Possible unit specifiers and the corresponding units are:

c	centimeters
i	inches
l	lines or characters (default)
p	PostScript points

For example to print an image one inch high, you can specify height by following options: **h1i** (1 inch), **h2.54c** (2.54 cm), **h72p** (72 points).

font select current font. The syntax of the escape is:

^@font{*fontname*[:*encoding*]}

where *fontname* is a standard font specification. Special font specification **default** can be used to select the default body font (**enscript**'s default or the one specified by the command line option **−f**, **−−font**).

The optional argument *encoding* specifies the encoding that should be used for the new font.

Currently the encoding can only be the **enscript**'s global input encoding or **ps**.

loadx load the current point X-coordinate from a register. The syntax of the escape is:

^@loadx{*register*}

ps include raw PostScript code to the output. The syntax of the escape is:

^@ps{*code*}

savex save the current point X-coordinate into a register. The position can later be restored with the **loadx** escape. The syntax of the escape is:

^@savex{*register*}

shade highlight regions of text by changing the text background color. Escape's syntax is:

^@shade{*gray*}

where *gray* is the new text background gray value. The default value is 1.0 (white) which disables highlighting.

PAGE DEVICE OPTIONS

Page device is a PostScript level 2 feature that offers an uniform interface to control printer's output device. **Enscript** protects all page device options inside an if block so they have no effect in level 1 interpreters. Although all level 2 interpreters support page device, they do not have to support all page device options. For example some printers can print in duplex mode and some can not. Refer to the documentation of your printer for supported options.

Here are some usable page device options which can be selected with the **–D**, **––setpagedevice** option. For a complete listing, see *PostScript Language Reference Manual*: section 4.11 Device Setup.

Collate *boolean*
how output is organized when printing multiple copies

Duplex *boolean*
duplex (two side) printing

ManualFeed *boolean*
manual feed paper tray

OutputFaceUp *boolean*
print output 'face up' or 'face down'

Tumble *boolean*
how opposite sides are positioned in duplex printing

PRINTING EXAMPLES

Following printing examples assume that **enscript** uses the default configuration. If default actions have been changed from the configuration files, some examples will behave differently.

enscript foo.txt
Print file **foo.txt** to the default printer.

enscript –Possu foo.txt
Print file **foo.txt** to printer **ossu**.

enscript –pfoo.ps foo.txt
Print file **foo.txt**, but leave PostScript output to file **foo.ps**.

enscript –2 foo.txt
Print file **foo.txt** to two columns.

enscript –2r foo.txt
Print file to two columns and rotate output 90 degrees (landscape).

enscript −DDuplex:true foo.txt
Print file in duplex (two side) mode (printer dependant).

enscript −G2rE −U2 foo.c
My default code printing command: gaudy header, two columns, landscape, code highlighting, 2-up printing.

enscript −E −−color −whtml −−toc -pfoo.html *.h *.c
A nice HTML report of your project's C source files.

ENVIRONMENT VARIABLES

The environment variable **ENSCRIPT** can be used to pass default options for **enscript**. For example, to select the default body font to be Times−Roman 7pt, set the following value to the **ENSCRIPT** environment variable:

−fTimes−Roman7

The value of the **ENSCRIPT** variable is processed before the command line options, so command line options can be used to overwrite these defaults.

Variable **ENSCRIPT_LIBRARY** specifies the **enscript**'s library directory. It can be used to overwrite the build-in default 'c:/progra˜1/enscript/share/enscript'.

RETURN VALUE

Enscript returns value 1 to the shell if any errors were encountered or 0 otherwise. If the option **−−extended−return−values** was specified, the return value is constructed from the following flags:

0	no errors or warnings
2	some lines were truncated or wrapped
4	some characters were missing from the used fonts
8	some characters were unprintable

FILES

c:/progra˜1/enscript/share/enscript/*.hdr	header files
c:/progra˜1/enscript/share/enscript/*.enc	input encoding vectors
c:/progra˜1/enscript/share/enscript/enscript.pro	PostScript prolog
c:/progra˜1/enscript/share/enscript/afm/*.afm	AFM files for PostScript fonts
c:/progra˜1/enscript/share/enscript/font.map	index for the AFM files
c:/progra˜1/enscript/share/enscript/hl/*.st	states definition files
c:/progra˜1/enscript/etc/enscript.cfg	system−wide configuration file
c:/progra˜1/enscript/etc/enscriptsite.cfg	site configuration file
˜/.enscriptrc	personal configuration file
˜/.enscript/	personal resource directory

SEE ALSO

diffpp(1), ghostview(1), gs(1), lpq(1), lpr(1), lprm(1), states(1)

AUTHOR

Markku Rossi <mtr@iki.fi> <http://www.iki.fi/˜mtr/>

GNU Enscript WWW home page: <http://www.iki.fi/˜mtr/genscript/>

NAME

sliceprint – slice documents with long lines.

SYNOPSIS

sliceprint [*enscript_options*] [*files*]

DESCRIPTION

XXX

SEE ALSO

enscript(1)

AUTHOR

Markku Rossi <mtr@iki.fi> <http://www.iki.fi/˜mtr/>

NAME

states – awk alike text processing tool

SYNOPSIS

states [**–hvV**] [**–D** *var***=***val*] [**–f** *file*] [**–o** *outputfile*] [**–p** *path*] [**–s** *startstate*] [**–W** *level*] [*filename* ...]

DESCRIPTION

States is an awk-alike text processing tool with some state machine extensions. It is designed for program source code highlighting and to similar tasks where state information helps input processing.

At a single point of time, **States** is in one state, each quite similar to awk's work environment, they have regular expressions which are matched from the input and actions which are executed when a match is found. From the action blocks, **states** can perform state transitions; it can move to another state from which the processing is continued. State transitions are recorded so **states** can return to the calling state once the current state has finished.

The biggest difference between **states** and awk, besides state machine extensions, is that **states** is not line-oriented. It matches regular expression tokens from the input and once a match is processed, it continues processing from the current position, not from the beginning of the next input line.

OPTIONS

–D *var***=***val***, ––define=***var***=***val*
: Define variable *var* to have string value *val*. Command line definitions overwrite variable definitions found from the config file.

–f *file***, ––file=***file*
: Read state definitions from file *file*. As a default, **states** tries to read state definitions from file **states.st** in the current working directory.

–h, ––help
: Print short help message and exit.

–o *file***, ––output=***file*
: Save output to file *file* instead of printing it to **stdout**.

–p *path***, ––path=***path*
: Set the load path to *path*. The load path defaults to the directory, from which the state definitions file is loaded.

–s *state***, ––state=***state*
: Start execution from state **state**. This definition overwrites start state resolved from the **start** block.

–v, ––verbose
: Increase the program verbosity.

–V, ––version
: Print **states** version and exit.

–W *level***, ––warning=***level*
: Set the warning level to *level*. Possible values for *level* are:

light	light warnings (default)
all	all warnings

STATES PROGRAM FILES

States program files can contain on *start* block, *startrules* and *namerules* blocks to specify the initial state, *state* definitions and *expressions*.

The *start* block is the main() of the **states** program, it is executed on script startup for each input file and it can perform any initialization the script needs. It normally also calls the **check_startrules()** and

check_namerules() primitives which resolve the initial state from the input file name or the data found from the begining of the input file. Here is a sample start block which initializes two variables and does the standard start state resolving:

```
start
{
  a = 1;
  msg = "Hello, world!";
  check_startrules ();
  check_namerules ();
}
```

Once the start block is processed, the input processing is continued from the initial state.

The initial state is resolved by the information found from the *startrules* and *namerules* blocks. Both blocks contain regular expression - symbol pairs, when the regular expression is matched from the name of from the beginning of the input file, the initial state is named by the corresponding symbol. For example, the following start and name rules can distinguish C and Fortran files:

```
namerules
{
  /.(c|h)$/    c;
  /.[fF]$/     fortran;
}

startrules
{
  /- [cC] -/      c;
  /- fortran -/   fortran;
}
```

If these rules are used with the previously shown start block, **states** first check the beginning of input file. If it has string **-*- c -*-**, the file is assumed to contain C code and the processing is started from state called **c**. If the beginning of the input file has string **-*- fortran -*-**, the initial state is **fortran**. If none of the start rules matched, the name of the input file is matched with the namerules. If the name ends to suffix **c** or **C**, we go to state **c**. If the suffix is **f** or **F**, the initial state is fortran.

If both start and name rules failed to resolve the start state, **states** just copies its input to output unmodified.

The start state can also be specified from the command line with option **–s**, **––state**.

State definitions have the following syntax:

state { *expr* {*statements*} ... }

where *expr* is: a regular expression, special expression or symbol and *statements* is a list of statements. When the expression *expr* is matched from the input, the statement block is executed. The statement block can call **states**' primitives, user-defined subroutines, call other states, etc. Once the block is executed, the input processing is continued from the current intput position (which might have been changed if the statement block called other states).

Special expressions **BEGIN** and **END** can be used in the place of *expr*. Expression **BEGIN** matches the beginning of the state, its block is called when the state is entered. Expression **END** matches the end of the state, its block is executed when **states** leaves the state.

If *expr* is a symbol, its value is looked up from the global environment and if it is a regular expression, it is matched to the input, otherwise that rule is ignored.

The **states** program file can also have top-level expressions, they are evaluated after the program file is parsed but before any input files are processed or the *start* block is evaluated.

PRIMITIVE FUNCTIONS

call (*symbol***)**
: Move to state *symbol* and continue input file processing from that state. Function returns whatever the **symbol** state's terminating **return** statement returned.

calln (*name***)**
: Like **call** but the argument *name* is evaluated and its value must be string. For example, this function can be used to call a state which name is stored to a variable.

check_namerules ()
: Try to resolve start state from **namerules** rules. Function returns **1** if start state was resolved or **0** otherwise.

check_startrules ()
: Try to resolve start state from **startrules** rules. Function returns **1** if start state was resolved or **0** otherwise.

concat (*str***, ...)**
: Concanate argument strings and return result as a new string.

float (*any***)**
: Convert argument to a floating point number.

getenv (*str***)**
: Get value of environment variable *str*. Returns an empty string if variable *var* is undefined.

int (*any***)**
: Convert argument to an integer number.

length (*item***, ...)**
: Count the length of argument strings or lists.

list (*any***, ...)**
: Create a new list which contains items *any*, ...

panic (*any***, ...)**
: Report a non-recoverable error and exit with status **1**. Function never returns.

print (*any***, ...)**
: Convert arguments to strings and print them to the output.

range (*source***,** *start***,** *end***)**
: Return a sub–range of *source* starting from position *start* (inclusively) to *end* (exclusively). Argument *source* can be string or list.

regexp (*string***)**
: Convert string *string* to a new regular expression.

regexp_syntax (*char***,** *syntax***)**
: Modify regular expression character syntaxes by assigning new syntax *syntax* for character *char*. Possible values for *syntax* are:

'w'	character is a word constituent
' '	character isn't a word constituent

regmatch (*string***,** *regexp***)**
: Check if string *string* matches regular expression *regexp*. Functions returns a boolean success status and sets sub-expression registers **$***n*.

regsub (*string*, *regexp***,** *subst***)**
: Search regular expression *regexp* from string *string* and replace the matching substring with string *subst*. Returns the resulting string. The substitution string *subst* can contain **$***n* references to the *n*:th parenthesized sup-expression.

regsuball (*string*, *regexp***,** *subst***)**
: Like **regsub** but replace all matches of regular expression *regexp* from string *string* with string *subst*.

require_state (*symbol*)
Check that the state *symbol* is defined. If the required state is undefined, the function tries to autoload it. If the loading fails, the program will terminate with an error message.

split (*regexp*, *string*)
Split string *string* to list considering matches of regular rexpression *regexp* as item separator.

sprintf (*fmt*, ...)
Format arguments according to *fmt* and return result as a string.

strcmp (*str1*, *str2*)
Perform a case−sensitive comparision for strings *str1* and *str2*. Function returns a value that is:

-1	string *str1* is less than *str2*
0	strings are equal
1	string *str1* is greater than *str2*

string (*any*)
Convert argument to string.

strncmp (*str1*, *str2*, *num*)
Perform a case−sensitive comparision for strings *str1* and *str2* comparing at maximum *num* **characters.**

substring (*str*, *start*, *end*)
Return a substring of string *str* starting from position *start* (inclusively) to *end* (exclusively).

BUILTIN VARIABLES

$.	current input line number
$*n*	the *n*:th parenthesized regular expression sub-expression from the latest state regular expression or from the **regmatch** primitive
$‘	everything before the matched regular rexpression. This is usable when used with the **regmatch** primitive; the contents of this variable is undefined when used in action blocks to refer the data before the block’s regular expression.
$B	an alias for **$‘**
argv	list of input file names
filename	name of the current input file
program	name of the program (usually **states**)
version	program version string

FILES

c:/progra˜1/enscript/share/enscript/hl/*.st enscript’s states definitions

SEE ALSO

awk(1), enscript(1)

AUTHOR

Markku Rossi <mtr@iki.fi> <http://www.iki.fi/˜mtr/>

GNU Enscript WWW home page: <http://www.iki.fi/˜mtr/genscript/>