NAME

cps – construct C to POSTSCRIPT language interface

SYNOPSIS

cps [**–c** [%] | **–C** [%]] [**–o** *style*] [**–f** *filespec*] [**–D** *symbol*] [**–I** *filename*] [**-s** [*number*]] [**–i**] [**–l** [**on** | **off**]] [*file***.cps**]

DESCRIPTION

cps compiles a specification file containing C procedure names and POSTSCRIPT language code into a header file (*file***.h**) that can be included in C programs. The header file associates the C procedure names with macros that transmit a compressed form of the POSTSCRIPT language code to the X11/NeWS server stream. Only one input file can be specified. If the *file***.{c,h,cc}** file(s) already exist, a backup will be generated of the form *file***.{c,h,cc}.BAK** before the new file is generated.

The convention is for the input specification file to end with the suffix **.cps**.

OPTIONS

–c Compiles the file of POSTSCRIPT language code for faster loading by NeWS, but does not generate a specification file for C programs. For example, the command line

example% cps -c < input_file > output_file

converts the input file from the ASCII form of the POSTSCRIPT language to a compressed binary form. When read by NeWS, the output file will execute exactly the same as *input_file*, except that it will be parsed faster.

NOTE: The –**c** option will not work if the input file uses constructs such as **currentfile readstring**, which are often used with the **image** primitive.

–c% Identical with -c, but will copy without compression until the first percent (%) character is found. This is to prevent cps from modifying shell commands in a file which is intended to be directly executed.

–C Compiles the file of POSTSCRIPT language code in the same way that the **-c** option does: however, when **-C** is used, the file can contain usertoken specifications. The tokens are automatically set up at the start of the output file; they are used throughout the output file to compress the POSTSCRIPT language even further than occurs with the **-c** option.

–C% Identical with -C, but will copy without compression until the first percent (%) character is found. This is to prevent cps from modifying shell commands in a file which is intended to be directly executed.

–D *symbol* Defines symbols to be passed to the C language preprocessor (**cpp**(1)), which processes the input file.

–f *filespec* Indicates the connection file specification. In an environment where there may be multiple PSFILE or wire connections open it is convienient to be able to directly specify which file is to be used to send/recieve the function data. (The examples below assume the ansi output option is in effect.) Valid specifications are:

current Generates code which uses the global PostScript and PostScriptInput variables. "cdef mumble(int foo)" becomes "...

mumble (int foo)"

wire Generates code which expects a wire to be passed to the cdef'd functions/macros. "cdef mumble (int foo)" becomes "... mumble (wire_Wire The__Wire, int foo)"

psfile Generates code which expects the input side of a PSFILE pair to be passed to the cdef'd functions/macros. "cdef mumble (int foo)" becomes "... mumble (PSFILE *The__File, int foo)"

–I *filename* Specifies include files or include paths. Passed on to the C preprocessor.

–o *style* Specifies the output format.

Most of these options are useful for minimizing the size impact of CPS interfaces that contain procedures called from several places in the C code. The *file***.c** generated would only need to be compiled once. Each file that needs to use the interface could then include only the *file***.h** and use the macros in that file multiple times. Each repeated invocation of the macro would refer to the shared POSTSCRIPT language code in the *file***.c** rather than its own static copy of the POSTSCRIPT language code.

In all cases if the output file(s) exist, **.BAK** backup copies will be generated.

styles recognized are:

macros Generates a single specification file of the form file.h, which contains the string constants and macro definitions required for the C-POSTSCRIPT interface.

ansi Generates two specification files: the first file contains only ANSI-c function prototypes; the second file contains the function definitions required for the C-POSTSCRIPT interface. The first file is of the form *file***.c**; the second file is of the form *file***.h**.

c++ Generates code identical to the ansi option, but the second file is of the form file.cc.

conststr Generates two specification files: the first file contains only the compressed form of the POSTSCRIPT language code; the second file contains the macro definitions required for the C-POSTSCRIPT interface. For example, **ps_open_PostScript()** and **ps_close_PostScript()** would be defined in the second file. The second file references compressed POSTSCRIPT language code as **extem char** arrays. The first file is of the form *file***.c**; the second file is of the form *file***.h**.

functions Generates two specification files: the first file contains only traditional (K&R) c function prototypes; the second file contains the function definitions required for the C-POSTSCRIPT interface. The first file is of the form *file***.c**; the second file is of the form *file***.h**.

inline Generates a single specification file of the form file.h, which contains the function definitions required for the C-POSTSCRIPT interface expressed as ANSI static inline functions.

portable Generates two specification files: the first file contains only function prototypes declared using the extended portability macros defined in <portable/c_varieties.h>. The second file contains the traditional (K&R) function definitions required for the C-POSTSCRIPT interface. The first file is of the form *file*.**c**; the second file is of the form *file*.**h**.

–i Obsolete. Replaced by '-o conststr'.

-s [*number*]
Specifies the threshold at which compiled POSTSCRIPT language code will be output as decimal arrays instead of string constants. If *number* is missing, all POSTSCRIPT language code will be ouput as string constants in the resulting *file*.**h**. This may be useful for debugging purposes, even though the POSTSCRIPT language code is in compressed form. If *number* is 0, all POSTSCRIPT language code will be output as decimal arrays. The default threshold is 400 characters, which is less than the maximum limit of string constants for most compilers. Note that there must be no space before *number*, since it is optional.

–l[on | off]
Specify the handling of '(literal string) gettext' sequences in cdef functions. When turned on ('-l' or '-lon'), the c code generated will perform a gettext() on the literal string before sending it down the wire to the NeWS server; and the 'gettext' token won't be sent at all. The default is '-loff'.

SEE ALSO

cpp(1)
NeWS 3.1 Programmer's Guide, Appendix B - Byte Stream

TRADEMARK

POSTSCRIPT is a registered trademark of Adobe Systems, Inc.