II. Ctext Component Programs

NAME

Programs - components of the ctext system

FUNCTION

This section describes the individual programs that make up the ctext system. The descriptions are in the form of "manual pages": summaries of how the programs work, and how they are run. In particular, the flags accepted by each program are documented here. These programs fall into four groups:

1) the command scripts that run ctext for a specific output device. These are ctdbl, to run ctext for plain or Diablo-compatible printers; cth19, to run ctext for VT-52 compatible VDT's; and ctvt, to run ctext for Varityper Comp/Edit devices.

2) the ctext formatter itself. Like the other programs, the formatter can be run separately; its intermediate output can then be saved in a permanent file for later submission to one of the device drivers.

3) device drivers. These programs translate the intermediate output from the formatter into the command stream expected by a particular class of output device. They are: vt52, for VT52-compatible VDT's; dbl, for plain or Diablo-compatible printers; and vtyper, for Varityper Comp/Edit devices driven through the STS interface.

4) utility programs. Currently, this is just the program toctext, which converts troff-style escape sequences into ctext commands.

Normally, ctext is run using the command scripts. If the scripts are used, then only the conversion program toctext, if wanted, needs to be run separately. The other programs are available for special-purpose use, however, and this section tells you everything you need to know in order to run them.

NAME

alarm - send alarm signal periodically

SYNOPSIS

alarm -[p# s#]

FUNCTION

alarm is used by vtyper, the device driver for Varityper Comp/Edit photocomposers, to provide timeout capability on reads by vtyper from the typesetter.

alarm simply sends a signal to the process whose id is given on its command line, after pausing for the number of seconds also given there. It is started by vtyper before each read from the typesetter. When the read is complete, the alarm process is killed. If, however, the read is not complete after the delay specified, alarm interrupts vtyper, indicating a timeout has occurred.

The flags are:

-p# send alarm signal to process #.

-s# sleep for # seconds between sending signals.

If the signal cannot be sent, alarm exits.

RETURNS

alarm loops until unable to send the signal, or until killed. In either case, it then returns failure.

NAME

ctdbl - drive ctext for Diablo or plain printers

SYNOPSIS

```
ctdbl -[c d* fs* f# h hi i* ms# o* p# t* u vi w +# #] <files>
```

FUNCTION

ctdbl is a shell script that runs first ctext, then the device driver dbl, to produce formatted text for Diablo-compatible printers, or for "vanilla" printers with no special capabilities. The entire command line given to ctdbl is passed directly to ctext, except for any occurrence of the four flags explained below.

Four flags are interpreted by ctdbl instead of being passed to ctext. These are:

-h include the enhanced heading macro package, ctmac.h, on the command line to ctext.

-hi enable the use of incremental horizontal spacing in the Diablo driver dbl.

-vi enable the use of incremental vertical spacing in the Diablo driver dbl.

-w include the Whitesmiths-style macro package, ctmac.w, on the command line to ctext.

By default, only the standard macro package, ctmac.s, is passed to ctext.

If either "-hi" or "-vi" is given, ctext will format with a Diablo in mind, using device "dbl". These flags are then passed through to the device driver dbl. As shipped, the devicespec for device "dbl" specifies both horizontal and vertical incremental spacing, so if one of these flags is given, both should be.

If neither "-hi" nor "-vi" is given, ctext will format for a "vanilla" printer, using device "tty", and the output of ctdbl is not Diablo-specific.

To lessen the demand ctdbl places on Idris, ctext and dbl are run sequentially, communicating via a temporary file. On systems where this is not an issue, the two programs may be run in a pipeline, permitting dbl to output text to the printer while ctext is still formatting.

RETURNS

ctdbl returns success if both ctext and dbl ran successfully.

BUGS

It would be nice if ctdbl recognized and passed to dbl the other flags that the driver accepts.

NAME

ctext - a processor for text

SYNOPSIS

```
ctext -[c d* fs* f# i* ms# o* p# t* u +# #] <files>
```

FUNCTION

ctext is a user-programmable utility for creating formatted text suitable for reproduction by a variety of output devices. This first pass of the ctext system is a formatter that outputs a device-independent stream of graphics and control information, suitable for interpretation by a driver program for a specific output device.

ctext takes input from each file named in <files>; if any file named does not exist, ctext does not complain but goes on to the next file in sequence. If no <files> are given, STDIN is read. A filename of "-" also causes STDIN to be read, at that point in the list of files.

The formatter accepts these flags:

-c continue formatting (without output) after page given with "-#".

-d* format text for device *. The string given is used as the argument to a <:device:> command. A device given with "-d" will override any specified elsewhere; <:device:> commands after the first are ignored.

-fs* use fontset *. The string given is used as the argument to a <:fontset:> command. A fontset given with "-fs" will override any specified elsewhere; <:fontset:> commands after the first are ignored.

-f# make # the page number that will activate the "first page" heading and footing commands. The default, of course, is 1.

-i* make * the name of the initialization file ctext will read at the start of the run. To the name given is appended the string ".ci"; the resulting filename is then searched for as a table file would be. The default name is "ctext", meaning that the file "ctext.ci" is looked for.

-ms# specify the maximum size of a legitimate macro invocation; # is given in bytes, and may be no larger than 8160. ctext will complain and abort if it encounters a longer definition than specified. By default, a maximum size of 1024 bytes is imposed. A value of 0 disables macro size checking.

-o* write intermediate output to file *, and error output to STDOUT. By default, intermediate output goes to STDOUT, and error messages to STDERR. This flag is mandatory on non-Idris/UNIX systems that distinguish between text and binary files, since STDOUT is treated there as a textfile.

-p# make # the number of the first page formatted.

-t# use # as a series of prefixes (presumably directory names) that ctext will prepend to table filenames in the order given. Prefixes are separated by a vertical bar '|'; the current directory is indicated by a null prefix. The default string is the Idris/UNIX "|/usr/ctext/" (i.e., current directory, then directory /usr/ctext).

-u warn of user-defined names that are undefined when used. By default, commands with names having no current definition simply disappear.

+# start output at each page numbered #. Earlier pages will be formatted, but will generate no output. By default, all pages are output.

-# stop output after each page numbered #. If "-c" is not given, ctext exits after the first such page is completed.

RETURNS

ctext returns success if all file and macro processing was performed with no errors, that is, with no diagnostic messages produced. If ctext finds an error, it will issue an appropriate message, and will return failure.

FILES

ctext searches the directories specified by "-t#" for the ".ci" initialization file it needs, and for the table files implied by the first <:device:> and <:fontset:> commands it encounters.

NAME

cth19 - drive ctext for H19 (VT52) terminals

SYNOPSIS

```
cth19 -[c d* fs* f# h i* ms# o* p# t* u w +# #] <files>
```

FUNCTION

cth19 is a shell script that runs first ctext, then the device driver vt52, to produce formatted text for terminals compatible with the DEC VT52. The entire command line given to cth19 is passed directly to ctext, except for any occurrence of the two flags explained below.

Two flags are interpreted by cth19 instead of being passed to ctext. These are:

-h include the enhanced heading macro package, ctmac.h, on the command line to ctext.

-w include the Whitesmiths-style macro package, ctmac.w, on the command line to ctext.

By default, only the standard macro package, ctmac.s, is passed to ctext.

To lessen the demand cth19 places on Idris, ctext and vt52 are run sequentially, communicating via a temporary file. On systems where this is not an issue, the two programs may be run in a pipeline, permitting vt52 to output text to the screen while ctext is still formatting.

RETURNS

cth19 returns success if both ctext and the device driver ran successfully.

NAME

ctvt - drive ctext for Varityper Comp/Edit devices

SYNOPSIS

```
ctvt -[c d* fs* f# h i* l* ms# o* p# t* u w +# #] <files>
```

FUNCTION

ctvt is a shell script that runs first ctext, then the device driver vtyper, to produce formatted text for Varityper Comp/Edit typesetters, using the slave typesetter (STS) protocol. The entire command line given to ctvt is passed directly to ctext, except for any occurrence of the three flags explained below.

Three flags are interpreted by ctvt instead of being passed to ctext. These are:

-h include the enhanced heading macro package, ctmac.h, on the command line to ctext.

-l* use * as the name of the communications link to the typesetter. By default, the name built into ctvt is used.

-w include the Whitesmiths-style macro package, ctmac.w, on the command line to ctext.

By default, only the standard macro package, ctmac.s, is passed to ctext.

To lessen the demand ctvt places on Idris, ctext and vtyper are run sequentially, communicating via a temporary file. On systems where this is not an issue, the two programs may be run in a pipeline, permitting vtyper to output text to the printer while ctext is still formatting.

RETURNS

ctvt returns success if both ctext and the device driver ran successfully.

BUGS

It would be nice if ctvt recognized and passed to vtyper the other flags that the driver accepts.

NAME

dbl - drive plain or Diablo-compatible printers for ctext

SYNOPSIS

dbl -[bi hi h# l# p vi v# +# #] <files>

FUNCTION

dbl converts the ctext intermediate language into a command stream suitable for "vanilla" printers or CRT's, or printers compatible with the Xerox Diablo. A "vanilla" output device is presumed to be capable of generating overstruck lines (by accepting successive text lines terminated only by a carriage return), but to have no other special capabilities. Specifically, dbl reads an intermediate command stream from each file in <files>, and a writes a converted command stream to STDOUT. If no <files> appear, STDIN is read; a filename of "-" also causes STDIN to be read, at that point in the list of files.

dbl accepts these flags:

-bi use incremental spacing only in producing boldfaced text, in the manner described below.

-hi use incremental spacing horizontally. Implies a horizontal raster of 1/120 of an inch.

-h# output # characters per inch horizontally. The "hmi" is initialized to correspond to this value. The normal value used is 12.

-l# make "pages" # lines in length; used in conjunction with the "+#" and "-#" flags described below. Default is 66 lines.

-p pause after each page of output. dbl will pause until a newline is read from STDOUT (presumably the terminal).

-vi use incremental spacing vertically. Implies a vertical raster of 1/48 of an inch.

-v# output # lines per inch vertically. The "vmi" is initialized to this value, and reset to it after each incremental spacing (if any). The normal value used is 6.

+# start output at page #, where pages are measured by "-l#" and numbered from one. Default is first page.

-# stop output after page #, where pages are measured by "-l#" and numbered from one. By default, all pages are output.

Note that the "page numbering" here is just a count of the pages processed, where page length is given by "-l". No numbering information is passed from ctext to the driver.

dbl will output text as overstruck whenever the current font has an internal index with the "bold" bit (1-weighted bit) set. Overstriking is accomplished by outputting the text three times in place. dbl will output text as underlined whenever the current font has an internal index with the "italic" bit (2-weighted bit) set. Currently, only alphanumeric characters are underlined.

If none of -[bi hi h# vi v#] is specified, dbl as far as possible avoids outputting Diablo-specific escape sequences, and is thus usable as a general-purpose driver. In this case, dbl outputs spaces (ASCII ' ') and backspaces (ASCII BS) for right or left cursor motion, line feeds (ASCII LF) and -- for want of anything better -- "reverse line feeds" ("\ESC\LF") for down or up cursor motion.

If -hi is given, dbl will use incremental spacing in outputting horizontal whitespace, and will offset the first overstrike by one increment in overstriking text (to enhance emboldening). If -bi is given, only this offseting will occur. If -vi is given, dbl will similarly use incremental spacing in outputting vertical whitespace. Naturally, "-hi" or "-vi" may be used only if the front end was run with a devicespec that allowed for them (by specifying the appropriate horizontal and vertical raster units).

If "-h#" or "-v#" is given, dbl will set the default horizontal or vertical spacing as specified; they are reset at the end of the run to the "normal" values given above.

No other escape sequences are ever output; further, if none of -[hi h# vi v#] is given, dbl is guaranteed never to output any Diablo-specific commands.

dbl expects to be installed with a filename that contains the name of the device used by ctext in preparing the input to the driver. In the standard package, this driver is installed under the names "dbl" and "tty".

RETURNS

dbl returns success if able to read all of the named input <files>. It will return failure (but output no disagnostic) if unable to do so.

NAME

toctext - convert nroff/troff escape sequences to ctext markup

SYNOPSIS

toctext -[+a* a* c2* cc* +c* c* ec* eo* hc* ho* +q* q* +t +u] <files>

FUNCTION

toctext is intended to ease the conversion of documents originally meant for formatting with nroff/troff into a format suitable for use with ctext. For the most part, this amounts to converting in-line escape sequences to a more regular format that ctext can interpret. toctext also recognizes the troff requests meant to cause some change in the format of troff markup, and provides other conversion facilities as options.

Note that, aside from a few rudimentary heuristics, toctext is purely a translator from one markup syntax to another. It does nothing to provide ctext-compatible definitions for any troff escape sequences.

Most of the flags accepted by toctext serve to change the components of the markup syntax it outputs. By default, toctext converts escape sequences to the default markup syntax described in the System Reference Manual, and passes through line-oriented requests unchanged. (The initialization file "ctext.ci" shipped with the package defines a markup format that allows ctext to recognize line requests.)

Flags are:

+a* make * the output argument open sequence.

-a* make * the output argument close sequence.

-c2* make the first character of * the initial troff secondary command character.

-cc* make the first character of * the initial troff command character.

+c* make * the output command open sequence.

-c* make * the output command close sequence.

-ec* make the first character of * the initial troff escape character.

-eo turn off troff escape character initially.

-hc* make the first character of * the initial troff hyphenation character.

-ho turn off troff hyphenation character initially.

+q* make each character of * a potential open quote character in the output markup.

-q* make each character of * a potential close quote character in the output markup.

+t convert .ta or .TA requests by prepending a "0" to the list of tabstops given, and appending the unit specifier "ch" to each tabstop value.

+u change lowercase alphabetic request names to uppercase.

toctext prepends two characters to the names of most of the escape sequences it encounters. This is done both to bring names into at least closer conformance with the rules for ctext identifiers, and to differentiate objects that in troff are in separate namespaces. Two escape sequences, "\|" and "\^", are transformed outright into "trs" and "ths" (for "narrow space" and "half-width narrow space"). The other troff escape sequences named by a single special character ("\&", "\ ", and so on) all have direct equivalents in ctext to which they are translated. The prefixes used for the remaining sequences follow:

PREFIX	**USED FOR**
tc	character names "\(xx"
ts	strings
tn	number registers
tf	font names or numbers
te	all other escape sequences

Thus the sequence "\n(AC" would become the command "<:tnAC:>", "\f1" would become "<:tf1:>", and so on.

toctext also translates line requests introduced by the secondary command character into ordinary requests preceded by the command ".nobrk". Finally, toctext does two kinds of argument processing. In line requests or escape sequences, numeric arguments of the form "+n" or "-n" are rewritten as the ctext-style "++n" or "--n". And in line requests, a closing quote is added to quoted arguments that lack one.

toctext will output warning messages, each identifying the current input file and line number, if confronted with a request or escape sequence it is unable to convert. Nominally, however, anything legal in troff will be converted to a corresponding ctext command.

RETURNS

toctext returns success if able to read its input <files>, and if able to convert every escape sequence it encounters.

NAME

vt52 - drive VT52-compatible terminals for ctext

SYNOPSIS

vt52 -[l# +# #] <files>

FUNCTION

vt52 converts the ctext intermediate language into a command stream suitable for any terminal compatible with the DEC VT52. Specifically, it reads an intermediate command stream from the files in <files>, and a writes a converted command stream to STDOUT. If no <files> appear, STDIN in read; a filename of "-" also causes STDIN to be read, at that point in the list of files.

vt52 accepts these flags:

-l# make "pages" # lines in length; used in conjunction with the "+#" and "-#" flags described below. Default is 66 lines.

+# start output at page #, where pages are measured by "-l#" and numbered from one. Default is first page.

-# stop output after page #, where pages are measured by "-l#" and numbered from one. By default, all pages are output.

Note that the "page numbering" here is just a count of the pages processed, where page length is given by "-l". No numbering information is passed from ctext to the driver.

vt52 will turn on reverse video when it encounters a font change command to any font with a non-zero internal index; vt52 turns reverse video off when the current font is changed back to the one with font index zero. Whitespace is also output with reverse video off; this is done purely for aesthetic reasons.

vt52 outputs spaces (ASCII ' ') and backspaces (ASCII BS) for right or left cursor motion, line feeds (ASCII LF) and "cursor up one line" commands ("\ESC[1A") for down or up cursor motion.

No other escape sequences are ever output.

vt52 expects to be installed with a filename that contains the name of the device used by ctext to prepare the input to the driver. In the standard package, this driver is installed under the name "vt52".

RETURNS

vt52 returns success if able to read all of its input <files>. It returns failure (but outputs no diagnostic) if unable to do so.

NAME

vtyper - drive Varityper Comp/Edit photocomposer

SYNOPSIS

```
vtyper <dev> -[cl## f* i* pl## t* +# #] <files>
```

FUNCTION

vtyper converts the ctext intermediate language into a command stream suitable for a Varityper Comp/Edit phototypesetter, driven through the STS (Slave Typesetter) interface. vtyper reads an intermediate command stream from each file in <files>, writing a converted command stream to <dev>, which vtyper treats as a filename that names the communications link to the typesetter (presumably a serial line). If no <files> appear, STDIN is read; a filename of "-" also causes STDIN to be read, at that point in the list of files.

vtyper reads a private table file to learn what typestyles are mounted at what turret positions in the typesetter. (ctext itself has no access to this information.) This table file consists of one or more lines, each containing two fields: first, the full name of the typestyle (as it appears in a ctext devicestyle table file); and second, the "style number" corresponding to the turret and row position where the typestyle appears. Style numbers count up from one (turret A, row 1), with adjacent rows on a type disc having consecutive numbers and adjacent type discs having consecutive ranges of numbers (e.g., style number 4 is turret A, row 4; style number 5 -- on a model 5410 -- is turret B, row 1). The typestyle name is scanned as a single string with no embedded whitespace; the style number is scanned as a decimal string, and is separated from the name by whitespace.

Flags are:

-cl## specify a "cassette length" of ## inches. When a vertical motion command is encountered that would cause vertical leading to exceed this amount, vtyper presumes the paper takeup cassette on the typesetter is full. vtyper leads up to the full cassette length, then displays a "cassette full" message to the typesetter console, issues a "suspend processing" command, and waits for a "resume processing" command to be typed at the console keyboard. Default is 110 inches (ten 11-inch pages).

-f* use * as the name of the typestyle location table file. By default, the name "vtyper.in" is used.

-i* write the string *, followed by a carriage return (ASCII CR), to <dev>, before entering the slave typesetter protocol.

-pl## specify a "page length" of ## inches; this length is used only in conjunction with the "+#" and "-#" flags described below. Default is 11 inches.

-t* treat * as a series of directory prefixes, with adjacent prefixes separated by a vertical bar '|'. Each prefix in turn will be prepended to the typestyle location table filename in searching for the file. By default, the single Idris/UNIX directory /usr/ctext is used.

+# start output at page #, where pages are measured by "-l#" and numbered from one. By default, output begins with the first page.

-# stop output after page #, where pages are measured by "-l#" and numbered from one. By default, all pages are output.

vtyper uses a relatively small set of the commands defined by the slave typesetter interface. Specifically: text, of course, is output with "escape then expose" triplets, a one-byte character position followed by a two-byte horizontal escapement, with the escapement output as a two's-complement integer, less-significant byte first. Changes in point size are output as "select point size" (PS, or 0x93) commands; changes in devicestyle are output as "select style" (FT, or 0x92) commands. Vertical motion is output as one or more "feed exposure media" (LD, or 0x94) commands; horizontal motion is output as one or more "escape then expose" triplets, with a (non-existent) character index of 0x72 given -- this causes an escapement only.

Several other commands are used to administer the line protocol to the typesetter. The "suspend processing" command (SP, or 0x98) is used to halt the typesetter so that the paper cassette can be changed, or when an error condition reported from the typesetter (like being out of paper) requires it. Any "suspend processing" command is always preceded by a "display message" command (DM, or 0x9b), to notify the operator of why the suspension is occurring. When the typesetter notifies vtyper of an error condition, other commands may be used in remedying the condition. These are: "request status" (RS, or 0x9c), "resume processing" (RP, or 0x99), and "initialize" (GO, or 0xa2). Finally, vtyper always terminates a run by sending an "end of job" command (ET, or 0x97).

vtyper expects to be installed with a filename that contains the name of the device used by ctext in preparing input to the driver. In the standard package, this driver is installed under the name "vtyper".

RETURNS

vtyper returns success if able to read all of its input <files>, and able to transmit them successfully to the typesetter. No diagnostic is issued if an input file does not exist; any problem in talking to the typesetter, however, will be indicated with an appropriate message.

BUGS

vtyper has been tested with versions of the slave typesetter interface up to revision seven (released in March, 1983). As of that revision, the DM command did not work properly. The use of that command by vtyper is believed correct; however, this remains unproven. vtyper was developed using a Model 5410 Comp/Edit system; hence, there was no opportunity to use the new commands for the Model 6400 digital typesetter.