Appendix A

CTEXT AND THE FILESYSTEM

Much of the information ctext needs about individual output devices and the fonts associated with them resides in table files in the host filesystem. A table file is an ordinary ASCII textfile, whose first line is a special ctext command called <:mark:>, which identifies what the file defines. Following the command are lines of information, generally of fixed format, which may be interspersed with further <:mark:> commands, due to the file naming restrictions explained below. In any case, the contents of a table file are strongly line-oriented, and each new information unit or command must begin on a new line.

All ctext table files live in one of the directories on a user-specified "search path". By default, the Idris/UNIX path "|/usr/ctext" is used, which looks for files first in the current directory, then in the directory /usr/ctext. For maximum portability, this directory contains no further directory structure, but only ordinary files, with names of the format "xxxxxx.xx" (six-character name, two-character suffix). The first part of each filename is simply the first six characters of the full name of whatever the file defines. The first letter of the suffix identifies the class of device to which the definition applies, while the second letter gives the type of definition. An exception is a devicespec file, the extension of which is always ".ds". Device class letters are user-selectable (because defined in <:devicespec:> commands); default device letters are:

LETTER	DEVICE CLASS
d	Diablo-compatible printer
t	"tty" -- either plain printer or VDT
v	Varityper Comp/Edit photocomposer (5810/5900/6400)

File types are:

LETTER	FILE TYPE
c	charset definition
d	devicefont definition
f	fontset definition
ds	device specification file

Thus the file db640.ds might define the specifications of a Diablo model 640 printer, while goudyo.vd might define the devicefont Goudy Oldstyle for the Varityper photocomposer.

Because of the severe restraint on name length, a given table file may have to define more than one charset, devicefont, or fontset (if you want access to two or more whose names share the same first letters). So long as each <:mark:> command in the file begins on a new line, the corresponding definition will be correctly processed.

Appendix B

CTEXT INTERMEDIATE OUTPUT

The output from the formatter itself is not suitable for direct reproduction on any device. Instead, it is intended for submission to a driver for the output device desired. This intermediate output contains formatted text, interspersed with command bytes. Because the formatted text is not necessarily ASCII characters, the interpretation of command bytes is context-dependent.

Specifically, the components of an intermediate output file will always be the following:

1) a "device name follows" command, containing the name from the <:device:> command read at the start of the run.

2) a series of "font initialization follows" commands, one for each devicestyle eventually used by the <:fontset:> command read at the start of the run.

3) the formatted text, consisting of horizontal and vertical motion commands interspersed with escaped buffers of characters. That is, bytes to be sent to the output device will always be included as part of the buffer named by an "0240" command, so that they cannot be misinterpreted as command information.

Further notes about intermediate output:

1) the strings specified as "init" or "final" parameters for a devicestyle will be sent as buffers separate from any other output.

2) font and size change commands will appear between buffers of formatted text.

3) a "box level 1 ends" command follows the intermediate output for each line; a "box level 2 ends" command follows the intermediate output for each heading/footing, and the running text for each page; a "box level 3 ends" command follows the output for the entirety of each page.

Broadly speaking, command information serves three functions: it specifies horizontal or vertical motion, alters some driver state, or passes user information through to the driver for its own consumption. These three classes of command bytes are detailed below.

B.1. Cursor Motion

030x specify horizontal motion. The low five bits of the command byte are added to the left of the next byte in the output stream, to produce a two's-complement integer specifying a distance in device horizontal rasters.

034x specify vertical motion. The low five bits of the command byte are added to the left of the next byte in the output stream, to produce a two's-complement integer specifying a distance in device vertical rasters.

B.2. State Manipulation

0200 change font. The next byte in the output stream specifies the internal index of the new font (as established by font initialization).

0201 change size. The next two bytes in the output stream specify the new size in which the current font is to be set, in device vertical rasters. The size is written as unsigned, less significant byte first.

B.3. Information Pass-Through

0240 escape a buffer of characters. Use next byte in the output stream as a count of the number of characters immediately following that are to be passed to the output device with no interpretation.

0241 escape a string of characters. Use next byte in the output stream as a delimiter, then pass bytes to the output device, with no further interpretation, until the delimiter is seen. The delimiter itself is discarded. Two consecutive delimiters cause one delimiter to be passed to the device, and escaping to continue.

0244 device name follows. The next bytes in the output stream, up to a terminating ASCII NUL, contain the name of the output device this stream is intended for, as given on the command line to ctext.

0245 font initialization follows. The next byte in the output stream is an internal font index; following it is the full name of the "font" (i.e., devicestyle) to be associated with the index, which is taken to continue up to a terminating ASCII NUL. The two low-order bits of the index indicate the bold/italic characteristics of the font: if the 1-weighted bit is on, the font is bold; if the 2-weighted bit is on, the font is italic.

0251 box level 1 ends. Currently, this byte signals the end of the intermediate output for each output line.

0252 box level 2 ends. Currently, this byte signals the end of the intermediate output for the interior ("running text") of each page, and for each heading and footing. If heading and footing output is disabled (by setting <:ispace:> and <:ospace:> to zero), the corresponding control bytes are not sent.

0253 box level 3 ends. Currently, this byte signals the end of the intermediate output for the entirety of each page. It is always sent at that point.

Appendix C

ASCII CHARACTER MNEMONICS

Within string constants, ctext accepts special escape sequences to provide symbolic equivalents for arbitrary eight-bit bytes that are not printable ASCII characters. An escape sequence consists of a backslash '\' followed by one of: 1) one of the letters "btnvfr" (in lower case), to represent ASCII BS, HT, LF, VT, FF, CR; 2) one to three digits representing the value of the byte (these are taken as octal if preceded by a '0', and as decimal otherwise); or 3) one of the ASCII mnemonics listed below for a "control character" (these must be given in upper case).

MNEMONIC	VALUE	MNEMONIC	VALUE
NUL	000	DLE	020
SOH	001	DC1	021
STX	002	DC2	022
ETX	003	DC3	023
EOT	004	DC4	024
ENQ	005	NAK	025
ACK	006	SYN	026
BEL	007	ETB	027
BS	010	CAN	030
HT	011	EM	031
LF	012	SUB	032
VT	013	ESC	033
FF	014	FS	034
CR	015	GS	035
SO	016	RS	036
SI	017	US	037

Appendix D

INTERNAL LIMITS AND STANDARDS

The representation of data within ctext has two significant areas of impact on users: first, the processing of user-defined names in the symbol table and elsewhere, and second, the handling of numeric values in expressions and elsewhere.

D.1. User-defined Names

Names connected with fonts (i.e., fontset, font, stylecode, charset, or devicestyle names), are significant to at least 32 characters. All other names are significant to 14 characters. Longer names may be given, in either context, but characters beyond these maxima will be ignored.

D.2. Numbers

Numeric input is initially treated as floating-point (i.e., as a C-language "double", roughly, a double-precision number). If a number is immediately followed by one of the recognized "unit specifiers", it is taken to mean a distance, and is converted to device rasters with the appropriate multiplier. The use of either horizontal or vertical rasters is unambiguously implied by all commands that expect a measurement as an argument. Within a <:defctr:>, or anywhere else where a measurement cannot be anticipated, a measurement is taken to mean a vertical distance.

Expressions are permitted wherever a numeric constant may occur; numbers are retained as floating-point until any surrounding expression has been evaluated. Then, they are converted to signed short integers by rounding away from zero. If the integer part of the result is not representable in 15 bits plus sign, a warning message is output, but processing continues. (To be precise, results in the range [-32767, 32767] are permitted, because the value -32768 is pre-empted by the processor for its own use.)

Four values associated with output line control are maintained differently. The baseline and spacing characteristics of a devicestyle (given in the <:mark:> command for the style), and the <:spacing:> and <:spread:> characteristics of a galley, are maintained as signed short integers with an implied fractional part of three decimal digits. Thus numbers given for these values are significant to only three fractional digits, and numbers outside the range [-32, 32] may not be stored correctly.

Appendix E

HYPHENATION

ctext attempts to hyphenate words when, in creating a line of filled text, the next word to be output will not fit entirely within the current galley width. ctext performs hyphenation as two separate operations. First, potential hyphenation points in the word are marked; then, the latest possible point is selected for actual hyphenation.

Hyphenation points come from three sources. First, of course, are the "soft" hyphens indicated by the user in his input text. If a word contains soft hyphens, then no further hyphenation points are sought. Otherwise, any pre-existent hyphen in the word is considered a hyphenation point, so long as at least one vowel appears both before and after the hyphen. Thus, "pre-existent" contains such a hyphenation point; "1-weighted" or "<cs-name>" does not. Finally, if the word contained no soft hyphens, ctext tries to generate more potential hyphenation points from each non-hyphenated portion of the input word. It does so by conditional suffix stripping, followed by the marking of likely hyphenation points between adjacent pairs of letters in the resulting word prefix. The suffix stripping is often conditioned on the letter immediately preceding the suffix, to try to reduce invalid suffix removal.

This algorithm is applied to any non-white string appearing at the end of a line of filled text; the only special treatment added is that any of a set of "punctuation" characters appearing at the end of a word are removed before suffix stripping. The punctuation characters are ".,;':)!?]" and double-quote.

Appendix F

ERROR MESSAGES

Three kinds of diagnostic messages may be output during a ctext run. First, ctext outputs a few messages as warnings; these indicate that an unexpected but not fatal condition has arisen. After such a message, processing continues, though the resulting output may no longer be correct.

A larger class of error messages are output only when an error has occurred that makes further formatting impossible. Most of these concern the font mechanism; driven as it is from many different table files, the possibilities for erroneous setup are numerous.

Finally, aside from the messages documented here, ctext may (but should not) output abort messages indicating that some inconsistency in its internal state has been caused by nominally correct input. These messages will always contain an '!', and should be immediately reported to the maintainers.

F.1. Warning Messages

can't include <fname> - the file <fname>, given as the first argument to an <:include:> command, cannot be read. Note that ctext tries to open a file with exactly the name given; the table file prefixes (specified with "-t#" on the command line) are not used by the <:include:> command.

integer conversion overflow - the result of an arithmetic expression cannot be represented as a signed short integer. It has been stored in truncated form, but obviously not in its full magnitude, and may cause future arithmetic to go awry.

undefined name: <user-name> - the command name <user-name> was invoked, but had no definition at the time of invocation. This message can be output only if "-u" is given on the command line to ctext.

F.2. Fatal Error Messages

<:deffont:> must precede user text - no <:deffont:> command was encountered before the first input that creates formatted text. Generally, <:deffont:> commands are included in the ".xf" files named by a <:fontset:> command.

<:device:> must precede <:deffont:> - no <:device:> command was encountered before the first <:deffont:> command read. The <:device:> command provides the name of a ".ds" file containing a devicespec <:mark:> command. Since this <:mark:> command gives the "device letter" that helps to determine the names of all other table files referring to that device, no font processing can be done until the ".ds" file has been read.

can't invoke font <ft-name> - the name <ft-name>, which should have occurred as the first argument to a <:deffont:>, is not currently defined as a font name. Hence the <:font:> command that tried to invoke <ft-name> can't be executed.

can't open font table file: <fname> - the ".xf", ".xc" or ".xd" file <fname>, one of the table files used by the font currently being defined, could

not be found in any of the directories given by the command line flag "-t*".

can't open initialization file: <fname> - the ".ci" file <fname>, used to specify an initial set of commands that ctext will automatically read at startup, could not be found in any of the directories given by the command line flag "-t*".

can't read charset: <cs-name> - the ".xc" file for the charset <cs-name> exists, but the <:mark:> command for <cs-name> was not found inside.

can't read devicefont: <df-name> - the ".xd" file for the devicefont <df-name> exists, but the <:mark:> command for <df-name> was not found inside.

can't read devicespec: <ds-name> - the ".ds" file for the devicespec <ds-name> exists, but the <:mark:> command for <ds-name> was not found inside.

can't read fontset: <fs-name> - the ".xf" file for the fontset <fs-name> exists, but the <:mark:> command for <fs-name> was not found inside.

ef contents not complete - the processor state at the conclusion of an ef (even-page footing) command did not match the processor state at the start of the command. Typically, this indicates that a macro invocation was left incomplete inside the footing. This message is output when the footing is actually used.

eh contents not complete - the processor state at the conclusion of an eh (even-page heading) command did not match the processor state at the start of the command. See "ef contents not complete".

empty font invoked: <ft-name> - the name <ft-name>, used in a <:font:> command, is currently defined as a font; however, no stylecodes are defined in the font, and ctext is unable to establish a new stylecode for further processing.

ff contents not complete - the processor state at the conclusion of an ff (first-page footing) command did not match the processor state at the start of the command. See "ef contents not complete".

fh contents not complete - the processor state at the conclusion of an fh (first-page heading) command did not match the processor state at the start of the command. See "ef contents not complete".

macro invocation too long: <macro-name> - an invocation of <macro-name> has been read that is longer than the maximum specified by "-ms#" on the command line. This message is output as a debugging aid; the length counting mechanism can be disabled by specifying "-ms0".

no devicestyle precedes user text - font initialization did not succeed in setting up even one devicestyle. Since none are defined, no output can occur. This message should be output only if neither a <:device:> command nor a <:fontset:> command precedes the first input that creates formatted text. (If any error occurs once ctext has started to define devicestyles, one of the preceding messages will result.)

of contents not complete - the processor state at the conclusion of an of (odd-page footing) command did not match the processor state at the start of the command. See "ef contents not complete".

oh contents not complete - the processor state at the conclusion of an oh (odd-page heading) command did not match the processor state at the start

of the command. See "ef contents not complete".

unrecognized <size-expr> in devicespec - a <size-expr> was given that ctext cannot deal with. Though arbitrary expressions may be specified in devicespec <:mark:> commands to relate relative character width to output character width, only the simple special cases <:sz:> and <:wd:>*<:sz:> can currently be interpreted.