Whitesmiths, Ltd.
How to Use Ctext: An Introduction for Typists and Writers

How to Use ctext:

An Introduction for Typists and Writers

Date: August 1984

The C language was developed at Bell Laboratories by Dennis Ritchie; Whitesmiths, Ltd. has endeavored to remain as faithful as possible to his language specification. The external specifications of the Idris operating system, and of most of its utilities, are based heavily on those of UNIX, which was also developed at Bell Laboratories by Dennis Ritchie and Ken Thompson. Whitesmiths, Ltd. gratefully acknowledges the parentage of many of the concepts we have commercialized, and we thank Western Electric Co. for waiving patent licensing fees for use of the UNIX protection mechanism.

The successful implementation of Whitesmiths' compilers, operating systems, and utilities, however, is entirely the work of our programming staff and allied consultants.

This manual was written by TechWriting Affiliates, Inc. of Newton, Massachusetts.

CONTENTS

CHAPTER 1

Introduction

This manual will introduce you to ctext: a powerful program that formats text files (documents) for output.

ctext works with any text editor that you may be using on your terminal or desktop computer. ctext takes text files that you type with the editor and formats them for output on a printer or your screen.

Most ctext commands consist of a period and a two-letter code, such as

```
.SH
```

for Section Heading and

```
.LL
```

for Line Length. Always type ctext commands on a separate line in all caps.

This manual is set up as a series of before and after examples. In each example, the left-hand page shows how you would type the ctext commands in your input file. The right-hand page shows how the same file will look when ctext has output it on the printer.

By following these examples, you can easily create your own documents. Simply type the file the way it's shown on the left (substituting your own text), and it will come out looking the way it's shown on the right.

This manual assumes that you are typing at a terminal or microcomputer which is running Idris, and that you can direct files to a printer.

Summary of Chapters

The examples in Chapters 2-6 will show you how to prepare input files to produce:

- a letter or memo
- a chapter of a manual or report
- a simple table or chart

In each case, you'll see how you can dramatically change the appearance of the printed document simply by replacing a few ctext commands in the input file.

Chapter 7 contains a number of helpful hints for "fine-tuning" the appearance of your output files. The Appendix lists all of the ctext commands used in this manual, with a brief explanation of each.

For More Information about ctext

For most applications, this manual will tell you everything you need to know about ctext.

If you need to do more than is shown here, please read Whitesmiths' _ctext Users' Manual_, which explains many additional powerful features of ctext. You will find cross-references to the _Users' Manual_ throughout this manual.

CHAPTER 2

How to Type In a Letter with Block Paragraphs and Justified Lines

Preview

This chapter shows how to create a business letter using ctext.

This first letter will use block (unindented) paragraphs, and the body text will be justified (space will be added between words to create an even right margin).

Begin by carefully examining the example on the next two pages.

- The left-hand page shows the input file you will type in.

- The right-hand page shows the formatted letter that will appear on your printer.

- At the bottom of both pages, each command used in the letter is summarized.

Then look at the "Notes on the Example" section, which explains the commands in more detail.

Finally, there are instructions at the end of the chapter so you can type the example yourself and output it on your printer (or screen).

Letter - Block Paragraphs

YOUR TYPED INPUT

```
.LH
.LL 65
                                                  May 10, 1984
.SP 2
Mr. Jonathan Cobol
Customer Support
Whitesmiths, Ltd.
97 Lowell Road
Concord, Mass. 021730
.SP 2
Dear Mr. Cobol:
.SP
.FI
.LP
Our company currently uses your C compiler
package on the VAX.
We are considering providing each programmer with a desktop
micro, either a PC or a PRO,
possibly for offsite (at-home) use.
.LP
If this plan goes through, will we be able to
compile and debug C
programs on the desktop machines and then send them to the VAX
over regular telephone lines?  Will the resulting programs run
on the VAX without any further modification?
.LP
Also, could you please send me a catalog describing your complete
line of C compilers, cross-compilers, and assemblers?
.SP 2
.NF
Yours truly,
.SP 4
Pat Pascal
Software Project Leader
```

Command	Meaning
.LH	Advance the page past the printed Letterhead
.LL 65	Change the Line Length to 65 characters
.SP 2	Skip two lines
.SP	Skip one line
.FI	Fill each line with as many words as will fit (Fill mode)

<u>ctext's PRINTED OUTPUT</u>

May 10, 1984

Mr. Jonathan Cobol
Customer Support
Whitesmiths, Ltd.
97 Lowell Road
Concord, Mass. 021730

Dear Mr. Cobol:

Our company currently uses your C compiler package on the VAX. We are considering providing each programmer with a desktop micro, either a PC or a PRO, possibly for offsite (at-home) use.

If this plan goes through, will we be able to compile and debug C programs on the desktop machines and then send them to the VAX over regular telephone lines? Will the resulting programs run on the VAX without any further modification?

Also, could you please send me a catalog describing your complete line of C compilers, cross-compilers, and assemblers?

Yours truly,

Pat Pascal
Software Project Leader

Command	**Meaning**
.LP	Begin an unindented (Left-justified) Paragraph
.NF	Print each line exactly as typed (No-Fill mode)

Notes on the Example

Letterhead (.LH)

Always begin your letters with a .LH command. This advances the page eight lines to position your letter below a printed letterhead. It also suppresses the page number at the bottom of the page.

The .LH command also turns off Fill mode (see below).

Line Length (.LL n)

The .LL command sets the right margin. If you don't include this command, ctext will use the default value: 80 characters.

Add Space (.SP n)

Use the .SP command to add vertical whitespace. This command will add any number of blank lines you specify. If you type .SP without a number, ctext will add one line.

You can also specify half-line increments -- by typing .SP .5 or .SP 2.5, for example. If your printer can't advance in half-lines, ctext will add a full line instead.

Fill mode (.FI) and No-Fill mode (.NF)

By default, ctext is in Fill mode when you begin a document. In this mode, ctext will fill each output line with as many words as possible, up to the specified Line Length. The line endings you type in your input file are ignored.

When you want lines to appear in your output document exactly as you typed them in your input file, you must turn Fill mode off. In the example, a .NF command turns Fill mode off after the last paragraph. This is necessary to prevent ctext from running the complimentary closing and the signature together.

The .LH command turns Fill mode off at the beginning of the letter so the date and inside address will appear exactly as you type them. Before typing the first paragraph, you must turn Fill mode back on with a .FI command.

When typing your input file, be careful not to begin a line with a space or a tab character, since this will cancel Fill mode for that one line, even if the line is in the middle of a paragraph.

Block Paragraph (.LP)

A .LP command signals the beginning of a left-justified or "block" paragraph: a paragraph with no first-line indent.

ctext will add a half-line space above the paragraph. If the printer cannot advance in half-lines, ctext will add a full line.

How to Use the Example for Practice

Now you are going to produce the example yourself, by typing the input file with ctext commands and outputting the results on your printer.

1. Using your text editor, open a file with the filename "letter".

2. Type the example <u>exactly as it appears on the left-hand page</u>.

3. Use the ctext shell script **ctdbl** to format your input file. Store the results in a new file called "letter.doc". Type:

   ```
   % ctdbl letter>letter.doc
   ```

4. Observe the shell prompt, indicating that ctext has finished formatting the document:

   ```
   %
   ```

5. Now print the formatted letter by typing the following command:

   ```
   % lpr<letter.doc
   ```

   (If you do not have a printer but you do have a hardcopy terminal, type

   ```
   % cat letter.doc
   ```

   at the hardcopy terminal.)

6. Compare the printed letter to the right-hand page of the example. If your letter does not look exactly like the example, check your input file for typing errors. Correct any mistakes, then repeat steps 3 through 6.

Review

This chapter showed you how to type an input file with ctext commands, and how to format the file with the **ctdbl** shell script. It introduced all of the basic ctext commands you need to create a business letter.

For more information about the commands used in this chapter, see the following sections in the essay entitled "Intro" at the beginning of the _ctext Users' Manual_:

Section	Command
2.1	.FI (Fill mode)
2.2	.NF (No-Fill mode)
4.2	.LP (Block paragraph)
7.1	.FI and .NF (Fill and No-Fill modes)
7.3	.LL (Line length)
8.1	.SP (Add space)
8.2	.LH (Letterhead)

CHAPTER 3

How to Type In a Letter with Indented Paragraphs and Ragged Lines

Preview

One advantage of using ctext is flexibility: you can change the appearance of a document without retyping it, simply by changing a few commands in your input file.

This chapter shows how to produce the same letter you typed in Chapter 2 in a different format. This time, you're going to create a letter with indented paragraphs, a ragged right margin, and underlining for emphasis.

In the command summary under the example, the new commands are indicated by an arrow.

Letter - Indented Paragraphs

YOUR TYPED INPUT

```
.LH
.LL 65
                                                 May 10, 1984
.SP 2
Mr. Jonathan Cobol
Customer Support
Whitesmiths, Ltd.
97 Lowell Road
Concord, Mass. 021730
.SP 2
Dear Mr. Cobol:
.SP
.FI
.NA
.PP
Our company currently uses your C compiler
package on the VAX.
We are considering providing each programmer with a desktop
micro, either a PC or a PRO,
possibly for offsite (at-home) use.
.PP
If this plan goes through, will we be able to
compile and debug C
programs on the desktop machines and then send them to the VAX
over regular telephone lines?  Will the resulting programs run
on the VAX without
.IT
any
further modification?
.PP
Also, could you please send me a catalog describing your complete
line of C compilers, cross-compilers, and assemblers?
.SP 2
Yours truly,
.SP 4
.NF
Pat Pascal
Software Project Leader
```

Command	Meaning
.LH	Advance the page past the printed Letterhead
.LL 65	Change the Line Length to 65 characters
.SP 2	Skip two lines
.SP	Skip one line
.FI	Fill each line with as many words as will fit (Fill mode)

<u>ctext's</u> <u>PRINTED</u> <u>OUTPUT</u>

```
                                                May 10, 1984


Mr. Jonathan Cobol
Customer Support
Whitesmiths, Ltd.
97 Lowell Road
Concord, Mass. 021730


Dear Mr. Cobol:
```

     Our company currently uses your C compiler package on the VAX. We are considering providing each programmer with a desktop micro, either a PC or a PRO, possibly for offsite (at-home) use.

     If this plan goes through, will we be able to compile and debug C programs on the desktop machines and then send them to the VAX over regular telephone lines? Will the resulting programs run on the VAX without <u>any</u> further modification?

     Also, could you please send me a catalog describing your complete line of C compilers, cross-compilers, and assemblers?

Yours truly,

Pat Pascal
Software Project Leader

	Command	**Meaning**
-->	.NA	Print text with a ragged right margin
-->	.PP	Begin a Plain (indented) Paragraph
-->	.IT	Italicize (underline) the next input line
	.NF	Print each line exactly as typed (No-Fill mode)

Notes on the Example

Ragged Right (.NA)

The .NA (Not-Adjusted) command instructs ctext to set the text with a ragged right margin. ctext will still fill each line with as many words as possible, but it will not add extra spaces between words to justify each line.

By default, ctext justifies each line in a paragraph, adding spaces between words to create an even right margin. The letter in Chapter 2 had a justified right margin.

Plain (Indented) Paragraph (.PP)

A .PP command signals the beginning of a "Plain" Paragraph: a paragraph with an indented first line.

ctext will add a half-line space above the paragraph. If the printer cannot advance in half-lines, ctext will add a full line.

Underline (Italicize) (.IT *n*)

The .IT command will underline a specified number of lines from your input file. For example,

```
.IT 2
```

will underline the next two lines that you type. If you don't specify a number, ctext will underline one input line.

How to Use the Example for Practice

Now you'll edit the input file you created in Chapter 2, and "preview" the formatted document on your screen before printing it.

1. Using your text editor, edit your input file "letter" so it matches the left-hand page of the example.

2. Use the ctext shell script **ctdbl** to format this revised input file, storing the results under the filename "letter2.doc". Type:

```
% ctdbl letter>letter2.doc
```

3. Observe the shell prompt, indicating that ctext has finished formatting the document:

   ```
   %
   ```

4. This time, before doing a "hard" proof (on your printer), you're going to do a "soft" proof, by displaying the document on your screen. A soft proof lets you see the results and edit any mistakes before sending the document to the printer. Type:

   ```
   % cat letter2.doc
   ```

5. Observe the formatted letter on your screen. If the document is scrolling off the screen too quickly for you to read, press ctl-S (to stop scrolling) and ctl-Q (to resume).

6. If the formatted document does not look exactly like the right-hand page of the example, check your input file for typing errors. Correct any mistakes and return to step 2.

7. Now print the formatted letter by typing the following command:

   ```
   % lpr<letter2.doc
   ```

   (If you do not have a printer but you do have a hardcopy terminal, type

   ```
   % cat letter2.doc
   ```

   at the hardcopy terminal.)

Review

This chapter showed you how to change the appearance of a document by replacing some ctext commands in your input file. It also showed how you can preview the document on your screen before sending it to the printer.

For more information about the commands introduced in this chapter, see the following sections in the essay entitled "Intro" at the beginning of the _ctext Users' Manual_:

Section	Command
4.1	.PP (Plain paragraph)
6.2	.IT (Underline)
7.2	.NA (Ragged right margin)

CHAPTER 4

How to Produce a Draft Manual with Unnumbered Section Headings

Preview

This chapter shows how to use ctext to format a manual or report. The example introduces the commands that produce page headers, boldface text, centered lines, and two levels of boldface headings. It also demonstrates one way to create a simple table.

New commands are flagged with arrows.

YOUR TYPED INPUT

```
.LL 75
.IN 5
.PL 45
.EH "Idris Capabilities" "- <:<:pval:> -" "Draft - April 20, 1984"
.OH "Draft - April 20, 1984" "- <:<:pval:> -" "Idris Capabilities"
.CE 2
.BD 2
SECTION 2
Idris Capabilities and Organization
.SH "INTRODUCTION"
.LP
Section 1 gave you hands-on experience with Idris.  Now in this
section we'll discuss the system's general capabilities and
organization.  First we'll look at features that make
Idris different from other operating systems.  Then
we'll continue with a summary of the 75 or so Idris utilities.
They will be grouped into functional categories corresponding
roughly to the remaining sections of this manual:
.DS
             -   Filesystem            Sections 3 and 4
             -   Text processing       Sections 5 and 6
             -   Program development   Section 7
.DE
.LP
Finally, we'll introduce the Idris documentation library--the manuals
to look at when you need more information than you find here.
.SH "DISTINCTIVE FEATURES OF IDRIS"
.BH "Idris Is Compatible with UNIX"
.LP
The technical design of Idris is derived from that of UNIX. Programs
developed under UNIX can easily be run under Idris, and vice versa.
.LP
Idris improves on UNIX in some ways, of which the most
important is programming tools.  Whitesmiths, Ltd. has built
a family of proprietary C and Pascal compilers, cross-compilers
assemblers, and runtime libraries for use with Idris.
```

[continued on page 4 - 4]

	Command	Meaning
	.LL 75	Change the Line Length to 75 characters
-->	.IN 5	Indent the left margin five characters
-->	.PL 45	Change the Page Length to 45 lines
-->	.EH	Specify a Header for Even-numbered pages
-->	.OH	Specify a Header for Odd-numbered pages
-->	.CE 2	Center the next two lines

ctext's PRINTED OUTPUT

SECTION 2
Idris Capabilities and Organization

INTRODUCTION

Section 1 gave you hands-on experience with Idris. Now in this section we'll discuss the system's general capabilities and organization. First we'll look at features that make Idris different from other operating systems. Then we'll continue with a summary of the 75 or so Idris utilities. They will be grouped into functional categories corresponding roughly to the remaining sections of this manual:

-	Filesystem	Sections 3 and 4
-	Text processing	Sections 5 and 6
-	Program development	Section 7

Finally, we'll introduce the Idris documentation library--the manuals to look at when you need more information than you find here.

DISTINCTIVE FEATURES OF IDRIS

Idris Is Compatible with UNIX

The technical design of Idris is derived from that of UNIX. Programs developed under UNIX can easily be run under Idris, and vice versa.

Idris improves on UNIX in some ways, of which the most important is programming tools. Whitesmiths, Ltd. has built a family of proprietary C and Pascal compilers, cross-compilers assemblers, and runtime libraries for use with Idris.

- 1 -

[continued on page 4 - 5]

	Command	Meaning
-->	.BD 2	Print the next two lines in Boldface
-->	.SH	Create a boldface Section Heading
	.LP	Begin a Left-justified (block) Paragraph
-->	.DS	Begin a Display (turn off Fill mode)
-->	.DE	End a Display (turn Fill mode back on)
-->	.BH	Create a boldface Block Heading

YOUR TYPED INPUT (continued)

```
.BH "Idris Is Very Compact"
.LP
Idris is the most compact UNIX-style system, but its small
size is achieved with no sacrifice of standard UNIX functionality.
The kernel of Idris (the part that must always be resident
in program memory) takes as little as 40K bytes.  As a
result, Idris can run on microcomputers with only 128K of
memory.
.BH "Idris Shell Scripts Facilitate High-level Programming"
.LP
It is easy in Idris to tie together separately compiled
programs into larger application packages.  The shell provides
a programming language for building command files, called
"shell scripts."
Section 7 shows how to write shell scripts.  For now we
want to emphasize that you have three levels of
tools for application programming in Idris:
.IP "Compilers - "
for both C and Pascal, with their respective runtime libraries
.IP "UNIX-style utilities - "
for file management, text processing, program development,
telecommunications, etc.
.IP "The shell language itself - "
for tying together your C and Pascal programs, and Idris
utilities, into shell scripts (command files).
.BH "Idris Has a Full Set of UNIX-style Utilities"
.LP
Idris has about 75 UNIX-style utilities, occupying
about 500K bytes of disk storage.  The utilities perform
a wide range of operations for file maintenance, text
processing, program development, telecommunications,
and multiuser support.  See the summary of utilities
later in this section.
```

	Command	Meaning
-->	.IP	Begin an Itemized Paragraph

ctext's PRINTED OUTPUT (continued)

Idris Capabilities - 2 - Draft - April 20, 1984

Idris Is Very Compact

Idris is the most compact UNIX-style system, but its small size is achieved with no sacrifice of standard UNIX functionality. The kernel of Idris (the part that must always be resident in program memory) takes as little as 40K bytes. As a result, Idris can run on microcomputers with only 128K of memory.

Idris Shell Scripts Facilitate High-level Programming

It is easy in Idris to tie together separately compiled programs into larger application packages. The shell provides a programming language for building command files, called "shell scripts." Section 7 shows how to write shell scripts. For now we want to emphasize that you have three levels of tools for application programming in Idris:

Compilers - for both C and Pascal, with their respective runtime libraries

UNIX-style utilities - for file management, text processing, program development, telecommunications, etc.

The shell language itself - for tying together your C and Pascal programs, and Idris utilities, into shell scripts (command files).

Idris Has a Full Set of UNIX-style Utilities

Idris has about 75 UNIX-style utilities, occupying about 500K bytes of disk storage. The utilities perform a wide range of operations for file maintenance, text processing, program development, telecommunications, and multiuser support. See the summary of utilities later in this section.

Notes on the Example

Indent Left Margin (.IN n)

The .IN command indents the left margin a specified number of characters. You can see the effect of this command on the second page of the example: the text is indented five characters, while the page header remains unindented.

Page Length (.PL n)

The .PL command specifies the depth of the printed page, in lines. Ordinarily you will not need to use this command, since the default value is 66 lines (11 inches). It was used in the example to leave room at the bottom of the page for the command summary.

Even-numbered Page Header (.EH "left" "center" "right")

The .EH command specifies a header which will appear at the top of every even-numbered page. You can specify separate text strings for the left corner, the center of the page, and the right corner. Each string must be enclosed in quotation marks.

To include the current page number in a header, insert the sequence

```
<:<:pval:>
```

(for "page number value") wherever you wish the number to appear. For example, the command

```
.EH " "  " "  "Page 3 - <:<:pval:>"
```

will produce the heading

```
                                              Page 3 - 2
```

on the second page,

```
                                              Page 3 - 4
```

on the fourth page, and so on.

Odd-numbered Page Header (.OH "left" "center" "right")

The .OH command specifies a header which will appear at the top of every odd-numbered page, except for the first page. (You can specify a header for the first page with the .FH command.) In all other respects, the .OH and .FH commands are identical to the .EH command explained above.

Notice the page number at the bottom of the first page of the example. ctext automatically prints a centered page number, bracketed by hyphens, at the foot of the first page of a document. If you don't specify headers with the .OH and .EH commands, ctext will also center a page number at the top of all odd- and even-numbered pages.

Center Lines (.CE n)

The .CE command centers a specified number of input lines. For example,

```
.CE 2
```

will center the next two lines that you type. If you don't specify a number, ctext will center one line.

Boldface (Doublestrike) (.BD n)

The .BD command will set a specified number of input lines in boldface. For example,

```
.BD 2
```

will set the next two lines that you type in bold. If you don't specify a number, ctext will set one line in bold.

Section Heading (.SH "heading")

The .SH command creates a boldface Section Heading. ctext will print the specified text string in boldface, with two blank lines above it and one below. If the heading consists of more than one word, it must be enclosed in quotation marks.

Block Heading (.BH "heading")

The .BH command creates a boldface Block Heading, with one blank line above it and a half-line below. (If your printer cannot advance in half-lines, ctext will output a full line.)

Begin a Display (.DS)

Use the .DS command when you want to set a Display: a block of text that will appear in the output document line-for-line, exactly as you typed it.

.DS adds a half-line space (or a full line, depending on your printer) and turns off Fill mode.

In the example, the .DS command is used to create a simple three-line table. When typing this table, use spaces (or spaces and tabs) to position the text in columns. (No-Fill mode will pass extra blank spaces though to the formatted document. Fill mode will strip out any extra spaces that you type between words.)

By default, ctext sets a tabstop every four columns, up to column 36. When you type a tab character, ctext will advance the text to the next tabstop. Tables typed using default tabstops can be very difficult to edit, however. If you are typing a table with more than two or three lines or columns, read the recommended method in Chapter 6.

End a Display (.DE)

The .DE command functions exactly like .NF: it simply turns Fill mode back on. Use .DE to return to Fill mode when you finish typing a Display.

Itemized Paragraph (.IP "tag")

The .IP command signals the beginning of an Itemized Paragraph. In an Itemized Paragraph, a short piece of text (or "tag") is set in bold at the beginning of the first line. The rest of the first line and all subsequent lines in the paragraph will be indented. If the tag consists of more than one word, it must be enclosed in quotation marks.

ctext will add a half-line space above the itemized paragraph. If the printer cannot advance in half-lines, ctext will add a full line.

How to Use the Example for Practice

Once again, practice using the new commands by typing the example and formatting it with ctext.

1. Using your text editor, open a file with the filename "manual".

2. Type the example exactly as shown on the two left-hand pages, with one exception: omit the Page Length command, .PL 45.

3. Use the ctext shell script **ctdbl** to format your input file, storing the results in a file called "manual.doc". Type:

   ```
   % ctdbl manual>manual.doc
   ```

4. Observe the shell prompt, indicating that ctext has finished formatting the document:

   ```
   %
   ```

5. Print the example by typing the following command:

   ```
   % lpr<manual.doc
   ```

   (If you do not have a printer but you do have a hardcopy terminal, type

   ```
   % cat manual.doc
   ```

   at the hardcopy terminal.)

6. Compare your printed output to the right-hand pages of the example. You will discover that ctext has put more text on the first page, since it is now formatting the document for a full 11-inch page. In all other respects, though, your output should look exactly like the example.

   If there are any other differences, check your input file for typing errors. Correct your mistakes and repeat steps 3 through 6.

Review

In this chapter, you have seen how to use all the basic commands you will need to create a manual or report with ctext.

When you need to format a manual of your own, you can use the example as a guide. Simply copy the commands from the example for the ctext features you need to use.

For more information about the commands introduced in this chapter, see the following sections in the essay entitled "Intro" at the beginning of the _ctext_ _Users'_ _Manual_:

Section	Command
4.4	.IP (Itemized paragraph)
5.1	.SH (Section heading)
5.2	.BH (Block heading)
6.2	.BD (Boldface)
7.2	.CE (Center lines)
7.3	.IN (Indent left margin)
8.2	.DS and .DE (Display)
10.1	.PL (Page length)
10.2	.EH and .OH (Headers)

As you read these sections, you will also find descriptions of the following related features:

- **Outlines** - Using itemized paragraphs, you can easily prepare documents in outline format, with several levels of hanging indents.

- **Page Footers** - You can add footers to each page with three commands that function exactly like the header commands:

  .FF (first page footer)
  .EF (even-number page footer)
  .OF (odd-numbered page footer)

CHAPTER 5

How to Number the Section Headings in the Draft

Preview

This chapter shows how to use another ctext feature: numbered headings. ctext will automatically insert sequential section numbers for several different levels of headings.

In the example, you will make two changes that turn your draft manual into a final version: revised page headers, and numbered section headings. Once again, just by changing a few commands you will alter the appearance of the entire document.

As you look at the example, notice that the Section Heading (.SH) and Block Heading (.BH) commands have been changed to

```
          .NH "heading" 2
```

and

```
          .NH "heading" 3
```

commands, which specify second- and third-level numbered headings. (The section title, "SECTION 2", is assumed to be a first-level heading.)

YOUR TYPED INPUT

```
.LL 75
.IN 5
.PL 45
.NS 2
.EH "Idris Capabilities"  "- <:<:pval:> -"  "Rev 1.1 - May 1984"
.OH "Rev 1.1 - May 1984"  "- <:<:pval:> -"  "Idris Capabilities"
.CE 2
.BD 2
SECTION 2
Idris Capabilities and Organization
.NH "INTRODUCTION" 2
.LP
Section 1 gave you hands-on experience with Idris. Now in this
section we'll discuss the system's general capabilities and
organization.  First we'll look at features that make
Idris different from other operating systems.  Then
we'll continue with a summary of the 75 or so Idris utilities.
They will be grouped into functional categories corresponding
roughly to the remaining sections of this manual:
.DS
              -   Filesystem             Sections 3 and 4
              -   Text processing        Sections 5 and 6
              -   Program development    Section 7
.DE
.LP
Finally, we'll introduce the Idris documentation library--the manuals
to look at when you need more information than you find here.
.NH "DISTINCTIVE FEATURES OF IDRIS" 2
.NH "Idris Is Compatible with UNIX" 3
.LP
The technical design of Idris is derived from that of UNIX. Programs
developed under UNIX can easily be run under Idris, and vice versa.
.LP
Idris improves on UNIX in some ways, of which the most
important is programming tools.  Whitesmiths, Ltd. has built
a family of proprietary C and Pascal compilers, cross-compilers,
assemblers, and runtime libraries for use with Idris.
```

[continued on page 5 - 4]

	Command	Meaning
	.LL 75	Change the Line Length to 75 characters
	.IN 5	Indent the left margin five characters
	.PL 45	Change the Page Length to 45 lines
-->	.NS 2	Begin all numbered headings with the number "2"
	.EH	Specify a Header for Even-numbered pages
	.OH	Specify a Header for Odd-numbered pages

ctext's PRINTED OUTPUT

```
                          SECTION 2
            Idris Capabilities and Organization

  2.1.  INTRODUCTION

  Section 1 gave you hands-on experience with Idris.  Now in  this  sec-
  tion we'll discuss the system's general capabilities and organization.
  First  we'll  look  at  features  that make Idris different from other
  operating systems.  Then we'll continue with a summary of the 75 or so
  Idris utilities. They will be grouped into functional categories  cor-
  responding roughly to the remaining sections of this manual:

              -   Filesystem            Sections 3 and 4
              -   Text processing       Sections 5 and 6
              -   Program development   Section 7

  Finally,  we'll introduce the Idris documentation library--the manuals
  to look at when you need more information than you find here.

  2.2.  DISTINCTIVE FEATURES OF IDRIS

  2.2.1.  Idris Is Compatible with UNIX

  The technical design of Idris is  derived  from  that  of  UNIX.   You
  benefit  because programs developed under UNIX can easily be run under
  Idris, and vice versa.

  Idris improves on UNIX in some ways, of which the  most  important  is
  programming   tools.   Whitesmiths,   Ltd.  has  built  a  family  of
  proprietary C and Pascal compilers, cross-compilers,  assemblers,  and
  runtime libraries for use with Idris.

                                 - 1 -
```

[continued on page 5 - 5]

	Command	Meaning
	.CE 2	Center the next two lines
	.BD 2	Print the next two lines in Boldface
-->	.NH "heading" 2	Create a second-level Numbered Heading
	.LP	Begin a Left-justified (block) Paragraph
	.DS	Begin a Display (turn off Fill mode)
	.DE	End a Display (turn Fill mode back on)
-->	.NH "heading" 3	Create a third-level Numbered Heading

YOUR TYPED INPUT (continued)

```
.NH "Idris Is Very Compact" 3
.LP
Idris is the most compact UNIX-style system, but its small
size is achieved with no sacrifice of standard UNIX functionality.
The kernel of Idris (the part that must always be resident
in program memory) takes as little as 40K bytes.  As a
result, Idris can run on microcomputers with only 128K of
memory.
.NH "Idris Shell Scripts Facilitate High-level Programming" 3
.LP
It is easy in Idris to tie together separately compiled
programs into larger application packages.  The shell provides
a programming language for building command files, called
"shell scripts."
Section 7 shows how to write shell scripts.  For now we
want to emphasize that you have three levels of
tools for application programming in Idris:
.IP "Compilers - "
for both C and Pascal, with their respective runtime libraries
.IP "UNIX-style utilities - "
for file management, text processing, program development,
telecommunications, etc.
.IP "The shell language itself - "
for tying together your C and Pascal programs, and Idris
utilities, into shell scripts (command files).
.NH "Idris Has a Full Set of UNIX-style Utilities" 3
.LP
Idris has about 75 UNIX-style utilities, occupying
about 500K bytes of disk storage.  The utilities perform
a wide range of operations for file maintenance, text
processing, program development, telecommunications,
and multiuser support.  See the summary of utilities
later in this section.
```

Command	Meaning
.IP	Begin an Itemized Paragraph

Idris Capabilities - 2 - Rev 1.1 - May 1984

2.2.2. Idris Is Very Compact

Idris is the most compact UNIX-style system, but its small size is achieved with no sacrifice of standard UNIX functionality. The kernel of Idris (the part that must always be resident in program memory) takes as little as 40K bytes. As a result, Idris can run on microcomputers with only 128K of memory.

2.2.3. Idris Shell Scripts Facilitate High-level Programming

It is easy in Idris to tie together separately compiled programs into larger application packages. The shell provides a programming language for building command files, called "shell scripts." Section 7 shows how to write shell scripts. For now we want to emphasize that you have three levels of tools for application programming in Idris:

Compilers - for both C and Pascal, with their respective runtime libraries

UNIX-style utilities - for file management, text processing, program development, telecommunications, etc.

The shell language itself - for tying together your C and Pascal programs, and Idris utilities, into shell scripts (command files).

2.2.4. Idris Has a Full Set of UNIX-style Utilities

Idris has about 75 UNIX-style utilities, occupying about 500K bytes of disk storage. The utilities perform a wide range of operations for file maintenance, text processing, program development, telecommunications, and multiuser support. See the summary of utilities later in this section.

Notes on the Example

Initial Heading Number (.NS _n_)

The .NS command specifies the starting number for all of the numbered headings in a document. You must use this command at the beginning of any file in which you intend to use numbered headings. The default value is 1.

In the example, the starting number is set to 2, since the document is Section 2. Second-level headings will begin at 2.1, third level headings at 2.1.1, and so on.

Numbered Heading (.NH "_heading_" _n_)

The .NH command creates a boldface Numbered Heading. The command must specify the heading text (in quotation marks), followed by a number indicating the level of the heading. ctext will print the current heading number and the heading in boldface, with a blank line above and below.

How to Use the Example for Practice

Now you are going to edit the input file you created in Chapter 4, to produce the example yourself.

1. Edit your input file "manual" so it matches the left-hand pages of the example. Change the

   ```
   .SH "heading"
   ```

   commands to

   ```
   .NH "heading" 2
   ```

   commands (second-level headings), and change all .BH commands to .NH 3 commands (third-level headings).

   Also, edit the text in the header commands, .OH and .EH.

2. Use the **ctdbl** shell script to format your input file, and store the results in "manual2.doc". To activate automatic heading numbering, you must use the **-h** flag when you issue the **ctdbl** command. Type:

   ```
   % ctdbl -h manual>manual2.doc
   ```

3. Observe the shell prompt, indicating that ctext has finished formatting the document:

```
%
```

4. Display the document on your screen by typing:

```
% cat manual2.doc
```

5. Observe the formatted example on your screen. Compare the header on the second page and the numbered section headings to the example. If they are not identical, correct any typing errors in your input file and return to step 2.

6. Print the example by typing the following command:

```
% lpr<manual2.doc
```

   (If you do not have a printer but you do have a hardcopy terminal, type

```
% cat manual2.doc
```

   at the hardcopy terminal.)

Review

This chapter showed how to change the page headers in a document, and how to use the ctext feature that generates sequentially numbered section headings.

For more information about numbered headings, see Section 2 ("The Enhanced Heading Macro Package") in the essay entitled "Macros" in the *ctext Users' Manual*.

CHAPTER 6

How to Set Up Tables

Preview

Chapter 4 showed one way to produce a simple table: type a .DS command to turn Fill mode off, then type the table exactly as you want it to appear, using spaces and the default tabstops to position the columns.

Tables created this way, however, are very hard to edit. If a table is more than three lines long or has more than two columns, it's a good idea to set your own tabstops with the .TA command introduced in this chapter.

In the examples that follow, you'll see how to create and edit a table of contents for the manual you created in Chapter 5.

The first example shows how to type a table, using the .TA command to set tabstops. The second example shows how easy it is to edit a table created this way.

Since the tab character isn't visible on your screen or in printed output, we have used a backslash (\) in the examples to show you where to type a tab character in your input file.

YOUR TYPED INPUT

```
.DS
.TA 5 13 71
\\\Page
.SP
\2.1\PREVIEW\2-1
\2.2\DISTINCTIVE FEATURES OF IDRIS\2-1
\2.2.1\Idris is Compatible with UNIX\2-1
\2.2.2\Idris is Very Compact\2-2
\2.2.3\Idris Shell Scripts Facilitate High-level Programming\2-2
\2.2.4\Idris Has a Full Set of UNIX-style Utilities\2-2
.DE
```

NOTE: Type a tab character wherever you see a backslash \ in the example.

	Command	Meaning
	.DS	Begin a Display
	.DE	End a Display
-->	.TA 5 13 71	Set Tabstops at columns 5, 13, and 71

Setting tabstops with .TA command greatly simplifies the task of editing a table. For example, after printing out the example above you might decide to move the page numbers closer to the text. If you had used spaces or the default tabstops to position the columns, you would have to go back and remove the correct number of spaces or tab characters from each line--a tedious operation.

Since you have set your own tabstops, however, you can move the entire third column by simply changing one value in the .TA command, as shown in the example below.

YOUR TYPED INPUT

```
.DS
.TA 5 13 51
\\\Page
.SP
\2.1\PREVIEW\2-1
\2.2\DISTINCTIVE FEATURES OF IDRIS\2-1
\2.2.1\Idris is Compatible with UNIX\2-1
\2.2.2\Idris is Very Compact\2-2
\2.2.3\Idris Shell Scripts Facilitate
\\High-level programming\2-2
\2.2.4\Idris Has a Full Set of UNIX-style
\\Utilities\2-2
.DE
```

```
ctext's PRINTED OUTPUT

                                                                 Page

    2.1      PREVIEW                                              2-1
    2.2      DISTINCTIVE FEATURES OF IDRIS                        2-1
    2.2.1    Idris is Compatible with UNIX                        2-1
    2.2.2    Idris is Very Compact                                2-2
    2.2.3    Idris Shell Scripts Facilitate High-level Programming 2-2
    2.2.4    Idris Has a Full Set of UNIX-style Utilities          2-2
```

```
ctext's PRINTED OUTPUT

                                                 Page

    2.1      PREVIEW                              2-1
    2.2      DISTINCTIVE FEATURES OF IDRIS        2-1
    2.2.1    Idris is Compatible with UNIX        2-1
    2.2.2    Idris is Very Compact                2-2
    2.2.3    Idris Shell Scripts Facilitate
             High-level programming               2-2
    2.2.4    Idris Has a Full Set of UNIX-style
             Utilities                            2-2
```

Notes on the Examples

Set Tabstops (.TA n n2 n3...)

The .TA command erases all existing tabstops and sets new stops at the columns you specify.

How to Use Underlining and Boldface in Tables

Since the .IT (underline) and .BD (boldface) commands operate on entire lines of input, you can't use them to highlight part of a line in a table (or any filled text, for that matter).

To highlight an entry in a table, you will use a second set of style commands which can be inserted in the middle of an input line. To begin underlining within a line, insert the command

```
<:i:>
```

(for "italic"). To stop underlining, insert the command

```
<:r:>
```

(for "roman," or normal print). Both of these codes can be embedded anywhere in text, and will take effect immediately.

To begin setting boldface text in mid-line, insert a <:b:> command; insert a <:r:> to return to normal print.

YOUR TYPED INPUT

```
.DS
.TA 5 13 51
\\\<:i:>Page<:r:>
.SP
\<:b:>2.1\Preview<:r:>\2-1
\<:b:>2.2\Distinctive Features of Idris<:r:>\2-1
\2.2.1\Idris is Compatible with UNIX\2-1
\2.2.2\Idris is Very Compact\2-2
\2.2.3\Idris Shell Scripts Facilitate
\\High-level programming\2-2
\2.2.4\Idris Has a Full Set of UNIX-style
\\Utilities\2-2
.DE
```

Review

This chapter showed how to create and edit a simple table, using the .TA command to set tabstops. It also showed how to highlight text in tab columns.

Additional ctext commands not covered in this manual allow you to to set text centered or flush right in tab columns, and to fill the whitespace in a column with leaders. For a description of these advanced features, see Section 9 ("Commands for Formatting Tables") in the essay entitled "Intro" at the beginning of the <u>ctext Users' Manual</u>.

<u>ctext's PRINTED OUTPUT</u>

		<u>Page</u>
2.1	**Preview**	2-1
2.2	**Distinctive Features of Idris**	2-1
2.2.1	Idris is Compatible with UNIX	2-1
2.2.2	Idris is very Compact	2-2
2.2.3	Idris Shell Scripts Facilitate High-level programming	2-2
2.2.4	Idris Has a Full Set of UNIX-style Utilities	2-2

CHAPTER 7

Helpful Hints

This chapter shows how to correct two problems: unwanted hyphens and page breaks in the wrong places. You may notice these problems when you check the "soft" proof of your ctext-formatted document on your screen.

How to Prevent Unwanted Hyphens

When Fill mode is on, ctext attempts to hyphenate the last word in each output line, in order to fill as much of the line length as possible. Occasionally, particularly in proper names, ctext will break a word at a point that leaves something to be desired.

For example, if the last paragraph in the letter in Chapter 2 had read

```
.LP
Also, could you please send me your catalog describing
Whitesmiths' complete line of C and Pascal
compilers, cross-compilers, and assemblers?
```

ctext would have output

```
Also,  could  you please send me your catalog describing  Whites-
miths' complete line of C and Pascal compilers,  cross-compilers,
and assemblers?
```

Clearly, this is not the ideal point at which to hyphenate "Whitesmiths'". You can designate a preferable hyphenation point by inserting a "soft" hyphen

```
<:-
```

in the word when you type it. Then, if ctext has to hyphenate the word, it will only hyphenate it at that point.

For example, if you inserted a soft hyphen in your input file, like this

```
.LP
Also, could you please send me your catalog describing
White<:-smiths' complete line of C and Pascal
compilers, cross-compilers, and assemblers?
```

ctext would output

```
Also,  could  you  please send me your catalog describing  White-
smiths' complete line of C and Pascal compilers, cross-compilers,
and assemblers?
```

If you don't want ctext to hyphenate the word at all, place the soft hyphen at the <u>beginning</u> of the word. For example, if you typed

```
.LP
Also, could you please send me your catalog describing
<:-Whitesmiths' complete line of C and Pascal
compilers, cross-compilers, and assemblers?
```

ctext would output

```
Also,  could  you  please  send  me  your  catalog   describing
Whitesmiths' complete line of C and Pascal compilers,  cross-com-
pilers, and assemblers?
```

Finally, by inserting the command

```
.HY 0
```

you can cancel <u>all</u> hyphenation for the remainder of the document.

How to Force a Page Break

ctext breaks each document into pages automatically, starting a new page when the current page is full. In some cases you may want to override the breakpoint ctext has chosen.

For instance, if the letter in Chapter 2 had contained just enough additional text to cause it to run over to a second page, the first page might have ended with

```
Also, could you please send me a catalog describing your complete
line of C compilers, cross-compilers, and assemblers?


Yours truly,
```

and the entire second page might have looked like this:

```
Pat Pascal
Software Project Leader
```

In this case, you would want to force the entire complimentary closing (and probably the final paragraph as well) onto the second page, leaving the first page a little short. To force a page break, simply insert a Break Page command

```
.BP
```

wherever you want the new page to begin. ctext will start a new page as soon as it encounters this command, pushing all text that follows the .BP to the top of the following page.

For example, if you inserted a Break Page command before the last paragraph of the letter, like this:

```
programs on the desktop machines and then send them to the VAX
over regular telephone lines?  Will the resulting programs run
on the VAX without any further modification?
.BP
.LP
Also, could you please send me a catalog describing your complete
line of C compilers, cross-compilers, and assemblers?
.SP 2
.NF
Yours truly,
.SP 4
Pat Pascal
Software Project Leader
```

the second page of the letter would look like this:

```
Also, could you please send me a catalog describing your complete
line of C compilers, cross-compilers, and assemblers?


Yours truly,




Pat Pascal
Software Project Leader
```

APPENDIX

Summary of ctext Commands Used in this Manual

Command	Meaning
.BP	Break Page: begin a new page.
.BD n	Print the next n input lines in Boldface (doublestrike). If no number is specified, print one line in boldface.
.BH "heading"	Print a boldface Block Heading, preceded by a one-line space and followed by a half-line.*
.CE n	Center the next n input lines. If no number is specified, center one line.
.DE	End a Display: turn Fill mode on.
.DS	Begin a Display: add a half-line space* and turn Fill mode off.
.EH "left" "center" "right"	Specify a Header for Even-numbered pages.
.FH "left" "center" "right"	Specify a Header for the First page of a document.
.FI	Fill each line with as many words as will fit (Fill mode).
.IN n	Indent the left margin n characters.

* If the printer cannot advance in half-line increments, ctext will use a full line.

.IP "tag"	Begin an Itemized Paragraph, preceded by a half-line space.* Print the tag in boldface at the beginning of the first line, then indent the rest of the line and all subsequent lines in the paragraph.
.IT n	Italicize (underline) the next n input lines. If no number is specified, underline one line.
.LH	Format the document for printing on a Letterhead. Advance the page eight lines, turn on Fill mode, and supress the page number at the bottom of the first page.
.LL n	Change the Line Length to n characters.
.LP	Begin an unindented (Left-justified) Paragraph, preceded by a half-line space.*
.NA	Print text with a ragged (Not-Adjusted) right margin.
.NF	Print each line exactly as typed (No-Fill mode).
.NH "heading" n	Create a boldface Numbered Heading, preceded by a line of space and followed by a half-line.* n specifies the level of the heading.
.NS n	Use n as the starting number for numbered headings.
.OH "left" "center" "right"	Specify a Header for Odd-numbered pages.
.PP	Begin a Plain (indented) Paragraph, preceded by a half-line space.*
.SH "heading"	Create a boldface Section Heading, preceded by two lines of space and followed by one line.
.SP n	Skip n lines. If no number is specified, skip one line.
.TA n n2 n3...	Clear all existing tabstops and set new stops at the specified columns.

Whitesmiths, Ltd.
97 Lowell Road, Concord, Massachusetts 01742