Multi-Lingual Word-Processing Research at the Technion

by

Daniel M. Berry

多言語文書処理研究

ダニエル・ベリ

Dan

Nie

儿

北

利

Computer Science Department
Technion
Haifa 32000
Israel
מתכון
בעבור טמילים
רב שפורט
bstcniw

דייר דניאל בר
مقון להנדסת תכנית
אוניברסיטת קורנגי מלון
פיטסבורג, פנסילבניה
ארהיב
APPENDIX I

Space — the final frontier.
These are the voyages of the starship Enterprise, its continuing mission, to explore strange new worlds, to seek out new life and new civilizations, to boldly go where no one has gone before!

issent bin dokhbar - haddar ba'da

החלל — הגבול הסופי.
אלה מסעותיה של תחנת "אנטרפרייז", המשמשת את המשימה לבקר בעולמות חדשים, למצוא חיים חדשים ותרבותים חדשים ל defStyleAttr ולערער את הבדלי נושאים.

ה𬬭בינה בין הקנוקב — הجيل القادم

הisspace — תחילת האהדה.
هذه هي רحلות הספינה הפלאטִית "אנטרפרייז", בממשיתה המשימה בה ביקרת אל תחומי חדשוניות, מצוירת חיים חדשים ותרבויות חדשות.

STAR TREK and STAR TREK: THE NEXT GENERATION are registered trademarks of Paramount Pictures Corporation.
Technion – Israel Institute of Technology
Faculty of Computer Science, Technion City, Haifa 32000, Israel

Daniel M. Berry
Professor

E-mail: dberry@cs.technion.ac.il
HTTP://www.cs.technion.ac.il/~dberry/

Secy: +972-4-829-4313
Fax: +972-4-822-1128
Outline of Talk
Need
Goal
Software Engineering Concerns
  Modular Formatting System
  Existing Formatting Tools
Why ditroff and
  not WYSIWYG or \TeX?

Solved Problems

Requirements of Multilingual Systems
ditroff Intermediate Form
Basic Trick
Hebrew R-L Formatting
Chinese and Japanese Alphabet
Top-Bottom Formatting
Bi-Directional vi
Bi-Directional MINIX
Arabic and Farsi Formatting
Indexing
Page Mark Up

Why \TeX{} Cannot Do Trick

Typesetting for Journals
Open Problems
Need for Multi-Lingual Word-Processing

First computers developed in English-speaking countries

First mass-marketing of computers in English-speaking countries

Spread next to countries whose languages are written with the Latin alphabet; some minor fudging needed for accents and unusual letters

\[\beta \varepsilon \& \phi \emptyset \ldots\]
Finally have spread to countries with totally different alphabets:
- Arabic/Farsi family
- Chinese family
- Cyrillic family
- Greek
- Hebrew family
- Hindi

Very large alphabets for which one byte is not enough to encode all the characters:
- Chinese family

Written in other directions
- R-L:
  - Hebrew family
  - Arabic/Farsi family
- T-B:
  - Chinese family
Need

Formatters

Editors

Applications

Operating Systems
Goal

Complete environment for preparation, proofing, and printing of technical and non-technical multi-lingual documents

Need to be able to edit, preview, and typeset documents with
- bibliography and citations
- formulae
- tables
- pictures
  - line
  - filled
  - half-tone
- plots
- flow diagrams
- flow charts
- graphs
- trees
- data structure
- program code
in

LR Languages
Latin
Greek
Cyrillic

RL Languages
Arabic
Farsi
Hebrew

TB Languages
Chinese
Japanese
Korean
A good software engineer is a lazy one!

Use existing system as much as possible

   good if user level compatible

   better if existing code is modified

   best if existing code is externally extended

Choose UNIX environment
   for ease in development
   for portability of applications
   because of its wide availability

Choose vi for editing
   because of uniformity world over

Choose ditroff collection for formatting
   because of its modularity
OVERVIEW OF DEVICE-INDEPENDENT TROFF FACILITIES AT SDC

Daniel M. Berry
9 June 1986
10:00am to 12:00pm
Fish Bowl

Abstract (A picture is worth a thousand words):

- **dtroff** — device independent troff [Ker82]
  - All the older stuff still works
    - but with fewer bugs
    - and much faster!
  - up to 256 fonts, == line length,
  - point sizes
  - Landscape Mode
  - More characters e.g., $\mathbb{R}$ \( \forall \exists \)
  - device drivers, e.g., **psdit**, **dver***, **dsun***, **dhpl***, **dip***

- **refer** fixed
- **algx** \( y = \text{ina}, \text{ada}, \text{a68}, \text{a69}, \text{for}, \text{sim}, \varepsilon \)
  for typesetting included program text:
  \[
  \text{axiom } E^\mathbb{Z}: \text{integer}(i \geq 0)
  \]

- **eqn** extended with macros and cleaned up:
  \[
  \forall y \in \mathbb{N}(2x \in \mathbb{N} (x = y^2))
  \]
- **tbl** cleaned up slightly

- **pic** for drawing pictures like the above
  flow diagram
  - Alternatively, **ideal** for drawing pictures — very C-like

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

*Not yet up

References

DITROFF FLOW:
Offers hope of implementing new functionality simply by inserting new pre- and post-processors.

UNIX philosophy:
Separate language processors, each understanding part of the job, and leaving all the rest to the others

Each is easily modified independently of the others

All existing pre- and postprocessors and macro packages continue to work as each new processor or macro package added!

No license needed to source code of ditroff to write a pre- or post-processor!
Examples of added functionality

<table>
<thead>
<tr>
<th>Program</th>
<th>Pre/Post</th>
<th>What?Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eqn</td>
<td>pre</td>
<td>dtroff</td>
</tr>
<tr>
<td>tbl</td>
<td>pre</td>
<td>dtroff</td>
</tr>
<tr>
<td>pic</td>
<td>pre</td>
<td>dtroff</td>
</tr>
<tr>
<td>ideal</td>
<td>pre</td>
<td>dtroff</td>
</tr>
<tr>
<td>psfig</td>
<td>pre</td>
<td>dtroff</td>
</tr>
<tr>
<td>refer</td>
<td>pre</td>
<td>dtroff</td>
</tr>
<tr>
<td>indx</td>
<td>post</td>
<td>dtroff</td>
</tr>
<tr>
<td>ffortid</td>
<td>post</td>
<td>dtroff</td>
</tr>
<tr>
<td>bditrof</td>
<td>post</td>
<td>dtroff</td>
</tr>
<tr>
<td>pm</td>
<td>post</td>
<td>dtroff</td>
</tr>
<tr>
<td>psdit</td>
<td>post</td>
<td>dtroff</td>
</tr>
<tr>
<td>dformat</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>swizzle</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>chem</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>m2p</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>dag</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>drag</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>flo</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>dotchart</td>
<td>pre</td>
<td>grap</td>
</tr>
<tr>
<td>scatmat</td>
<td>pre</td>
<td>grap</td>
</tr>
<tr>
<td>et_al</td>
<td>post</td>
<td>refer</td>
</tr>
<tr>
<td>Program</td>
<td>Pre/Post</td>
<td>Tool</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>monk</td>
<td>pre</td>
<td>dtroff</td>
</tr>
<tr>
<td>picasso</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>fig</td>
<td>pre</td>
<td>pic</td>
</tr>
<tr>
<td>suntroff</td>
<td>post</td>
<td>dtroff</td>
</tr>
<tr>
<td>xtroff</td>
<td>post</td>
<td>dtroff</td>
</tr>
</tbody>
</table>
Why not WYSIWYG?

MONOLITHIC

everything it does

Also WYSIOAWYG, WYSIAYG, and WYCGINE!

Better to have WYSAAFSIWYG if WYS is exact and WYG is everything you need

With today’s Hardware,
Multi-window environment
  with batch previewer in one
  and editor in another
on dedicated workstation, almost as fast as WYSIWYG
Why not \TeX?

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
\textbf{MONOLITHIC} & formulae & tables \\
\hline
 & limited line drawing & \\
\hline
\end{tabular}
\end{center}

Not really pipeable

Also other problems

More on this later!
Note that from the user’s point of view, it really matters not which of ditroff or \TeX{} is used.

Both are assembly languages of comparable functionality, and both have higher level interfaces, in the forms of macro or style packages (e.g., \texttt{-ms}, \texttt{-me}, \texttt{-mm}, \texttt{-mX}, and \LaTeX{}).

Interactive front ends

Someone who uses plain ditroff or \TeX{} is asking for it!

Reality: You like what you are used to and hate what you are not!
Problems We Have Solved:

Right-to-Left formatting and Hebrew

Chinese and Japanese Characters

Top-to-Bottom formatting

vi.iv

MINIX.XINIM

Arabic, connecting, and stretching

Indexing without flooding input with indexing commands

Problems Others Have Solved:

Page Mark Up, page balancing and figure and footnote placement
Requirements for multilingual formatters, editors, systems, and applications

Input

Time Order (Logical Order)
as if ALL languages were written left-to-right
Each language (including computer languages) in its own standard encoding

Output

Visual Order
Fonts with glyphs selected by standard encoding for each language

File storage

Time order (Logical Order)
Each language in its own standard encoding (SAME AS INPUT!)
Thus conversion
  from time order to visual order
done at output time AND at EACH output time
NOT at input time

More general
  In files, most significant character of each line
  is in same place, so, e.g., sorting
  applications work with no change

If line length changed after input,
  much easier to get into new visual order
  from original time order than
  from another visual order
Mixing languages
Generally mixing latin and local language

Latin for computers, scientific math & technical
Local for people!

two languages only

So use eighth bit as Latin/Local flag
0 → Latin
1 → Local, e.g.,
   ESCII — Hebrew
   Shift JIS — Japanese

In two-language mixed text, easy to distinguish which byte is in which language:
If have more than two languages
use in-line escape sequences to
distinguish
Character Coding Issues -4

The other coding issue is the order of the codes for the letters.

The ESCII code is a Hebrew extension of the ASCII code. Characters in the range of 0 to 127 are considered Latin and follow the ASCII coding. Characters in the range of 128 to 255 are considered Hebrew. The Hebrew letters appear in alphabetical order, with final letters immediately preceding their non-final counterpart. A character that is in both Hebrew and Latin appears twice in the table, their codes separated by 127.
Latin Half (Hexadecimal → Character)

<table>
<thead>
<tr>
<th>00</th>
<th>NUL</th>
<th>01</th>
<th>SOH</th>
<th>02</th>
<th>STX</th>
<th>03</th>
<th>ETX</th>
<th>04</th>
<th>EOT</th>
<th>05</th>
<th>ENQ</th>
<th>06</th>
<th>ACK</th>
<th>07</th>
<th>BEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>BS</td>
<td>09</td>
<td>HT</td>
<td>0A</td>
<td>NL</td>
<td>0B</td>
<td>VT</td>
<td>0C</td>
<td>NP</td>
<td>0D</td>
<td>CR</td>
<td>0E</td>
<td>SO</td>
<td>0F</td>
<td>SI</td>
</tr>
<tr>
<td>10</td>
<td>DLE</td>
<td>11</td>
<td>DC1</td>
<td>12</td>
<td>DC2</td>
<td>13</td>
<td>DC3</td>
<td>14</td>
<td>DC4</td>
<td>15</td>
<td>NAK</td>
<td>16</td>
<td>SYN</td>
<td>17</td>
<td>ETB</td>
</tr>
<tr>
<td>18</td>
<td>CAN</td>
<td>19</td>
<td>EM</td>
<td>1A</td>
<td>SUB</td>
<td>1B</td>
<td>ESC</td>
<td>1C</td>
<td>FS</td>
<td>1D</td>
<td>GS</td>
<td>1E</td>
<td>RS</td>
<td>1F</td>
<td>US</td>
</tr>
<tr>
<td>20</td>
<td>SP</td>
<td>21</td>
<td>!</td>
<td>22</td>
<td>&quot;</td>
<td>23</td>
<td>#</td>
<td>24</td>
<td>$</td>
<td>25</td>
<td>%</td>
<td>26</td>
<td>&amp;</td>
<td>27</td>
<td>'</td>
</tr>
<tr>
<td>28</td>
<td>(</td>
<td>29</td>
<td>)</td>
<td>2A</td>
<td>*</td>
<td>2B</td>
<td>+</td>
<td>2C</td>
<td>,</td>
<td>2D</td>
<td>-</td>
<td>2E</td>
<td>.</td>
<td>2F</td>
<td>/</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>31</td>
<td>1</td>
<td>32</td>
<td>2</td>
<td>33</td>
<td>3</td>
<td>34</td>
<td>4</td>
<td>35</td>
<td>5</td>
<td>36</td>
<td>6</td>
<td>37</td>
<td>7</td>
</tr>
<tr>
<td>38</td>
<td>8</td>
<td>39</td>
<td>9</td>
<td>3A</td>
<td>:</td>
<td>3B</td>
<td>;</td>
<td>3C</td>
<td>&lt;</td>
<td>3D</td>
<td>=</td>
<td>3E</td>
<td>&gt;</td>
<td>3F</td>
<td>?</td>
</tr>
</tbody>
</table>
UNICODE -1

UNICODE is a two-byte code for the whole world, containing *one* occurrence of each letter that appears in any of most of the alphabets in the world.

One code for period to be shared by all languages; same for other punctuation and digits

One code for each Hebrew letter, to be used by Hebrew and Yiddish
One code for each Chinese letter to be used by Chinese, Japanese, and Korean

In text using this code, must use an escape to indicate language change, because language is not inherent in the codes for the characters.
**ditroff** intermediate format, 
i.e., format of output of **ditroff**
and of input to device drivers.

This is a line.

is translated by **ditroff** to

```
H576
V96
cT
-
49h40i22sw51i22sw51aw56l22i22n40e36.n96 0
--- =-- =- =---- -_____
```

V#  = absolute vertical position
H#  = absolute horizontal position
C   = Character
M   = Movement
w   = end-of-word marker
L   = end-of-line marker
    (also end-of-word unless last character is hyphen)

\[
\{H# V# ((CM)^+ w)^+ (CM)^+ l}\}^+
\]

Pure ASCII
\TeX’s intermediate form, DVI, does not have end-of-word and end-of-line markers, and is NOT ASCII

\[(H\# V\# (CM)^+)^+\]

Note that ends-of-word and ends-of-line markers are NOT needed by device drivers

These were frosting put in by Kernighan to allow an editor, that was never built, to work directly with ditroff intermediate form.
Basic Trick for Multidirectional Formatting

Let ditroff format time-ordered text as if all languages were written from left to right.

Then for each non-LR direction $D$,
- have a postprocessor for $D$
- reorganize the ditroff intermediate form output from dtroff
so that
  - all text to be written in direction $D$
  - is in position to be printed in direction $D$

The output of the postprocessor is again in ditroff intermediate form.

This is possible because of end-of-line markers in ditroff intermediate form.

Without these markers,
  - the reorganization is impossible

ffortid for $D =$ right-to-left
bditroff for $D =$ top-to-bottom
ffortid by Buchman

First,
How to Read LR-RL Bi-Directional Document:

Defs: uni-directional chunk
maximal length string of text within
one line, all of whose characters are
in langauges of same direction

In the line,

He said “שלום לך” to Uri.

3 uni-directional chunks are

He said “
שלום לך
” to Uri.
Invariant:
Cannot move on to the next line until all the text on a given line has been read

Within line, one bounces around within a line to read the uni-directional chunks in order = current document direction:
Each chunk read in its own direction

In example above, current document direction is LR, so read

He said “שלום לך” to Uri.
What we want:

He said to
Dan “שלום”
in Hebrew.

Time-ordered input:
He said to Dan “שלום” in Hebrew.

After formatting with dtroff (schematically)

He said to
Dan “שלום”
in Hebrew.

Can get what we want line-by-line by
flipping Hebrew phrases in place

Works because

Not move to next line (in reading)
until whole line has been read

Permuting characters of line
NOT change total length of line!
What we want:

Hello

Time-ordered input:
Hello

After formatting with dtroff (schematically)

Can get what we want line-by-line by first flipping whole line
and then flipping Latin phrases in place

Works for same reasons
Conversion Algorithm

Assume that file is stored in time order

Language of a character determined by its font and ffortid knows which fonts are R-L

Current direction of document set globally by command (actually macro)

\[
\textbf{for} \text{ each line in the file } \textbf{do} \\
\quad \textbf{if} \text{ the current document direction is L-R } \textbf{then} \\
\qquad \text{reverse each contiguous string of RL characters in the line} \\
\quad \textbf{else} (\text{the current document direction is R-L}) \\
\qquad \text{reverse the whole line;} \\
\qquad \text{reverse each contiguous string of LR characters in the line} \\
\quad \textbf{fi} \\
\textbf{od}
\]
Input:

The next sentence contains one verb.

The previous sentence contains one verb.

Output:

The next sentence contains one verb. The previous sentence contains one verb.
This sentence contains one verb.

This sentence.

This sentence contains one verb.

This sentence.

This sentence.
-ftoffrid

었א הזף קר שהתקסע הנכתב מימין לשמאל, י퓨 שרוחות הכותבות בשפה העברית.
עי מתייחסים שול.

-1

ןמאית

-ftoffrid [-rfont-position-list] [-wpaperwidth] [-aarabic-position-list] [-s[nfl]] [ file ] ...

תעזור

התcksåד לש קלחאוא ות פלט של ftoffrid , המועבד משמע Reid, להפוך סקטס הנכתב מימין לשמאל, להפוך אותות, לוסר מזויג דיאג נקודת, כז שקטס
בכ"א מתפתיות או הא בוקו העבורי של. ftoffrid ממקבל את הפקס של מ- difoff-° לכל
הוא לא פייע, זלא צרי פלט, לע רא איון פמקד הימני של. difoff לכל, הפקס של
device) difoff (הוא מבהלה סקטסית לפצל של ftoffrid זוגר ד"ר אזהז נוגאי חטקיים
, difoff (הוא מבהלה סקטסית לפצל של ftoffrid זוגר ד"ר אזהז נוגאי חטקיים
, difoff )

drivers

ברירת המחדל, של חפסית נכתבים משמע Reid, והאなるוגיט 3- משמשים קבקעה.

.font-position-list (מעים לדמאל) של כ"א מתפתיות הינוורנגי בՊים Reid ה-
 consc לופן של ffortid difoff-°, עון פצק נכתבים משמע Reid, י퓨 שרוחות
difference) difoff (הוא מבהלה לيمنי, יゞכז ליזני ע"י שרי הפקדות

וקובע את הכ"א העבורי משמע Reid לيمنי, יפו שרוחות התאגיד銷售 יציא משמע Reid

יא כ"א עבורי הנכתב משמע Reid, יפו שרוחות התאגידсал יציא משמע Reid

לםי. בכלל, אוסף חספיס נכתבין מימין לשמאל, הנכובות סביר חצר.

אש ש還是 לש ffortid-"משמע Reid" מובן חזרב, (paper width) לע מנט הלעב את רוחב
chers. 8.5inch (page offset) מוסר ffortid-"משמע Reid" לע רוחב ד"ף, 8.5inch (page offset) מוסר ffortid-"משמע Reid" לע רוחב ד"ף, 8.5inch (page offset) מוסר ffortid-"משמע Reid" לע רוחב ד"ף, 8.5inch (page offset) מוסר ffortid-"משמע Reid" לע רוחב ד"ף, 8.5inch (page offset) מוסר ffortid-"משמע Reid" לע רוחב ד"ף, 8.5inch (page offset) מוסר ffortid-"משמע Reid" לע רוחב ד"ף, 8.5inch (page offset) מוסר ffortid-"משמע Reid" לע רוחב ד"ף.
7. יגורות הגישה

המשרתו של החידוש בברודר, או פתרת את כל בעיות לעיבוד השפה העורבית של והקימיקים, זה
דרישותנות. התכונות העיקריות של המשרתו, בṀמשלב על הקימיקים, זה:

1. סדר דפים טקסט בין לשוני הכתבים בשפה העורבית, לבין כתיבתי

2. הימלע situéית לשוני.

3. כלכל התכונה בין אתיות המילים הכתביות בשפה העורבית, הקיצוב והניקום
במקומם.

4. האפ疝ואר החדש של יינו טקסט ערב בועה קשיה.

5. יונח ו읏וייל בחרכבות ובניקודים. הספואל בניקודים, כלכל על בעויי דינמי של מיקומם
האנימי בין לאותיות.

6. כלכל סדר דפים טקסט מדעיה המכלי קנייה, והञואות, לarParams, וחבר
laser בｫפרסר של POSTSCRIPT

בשגרת לציון, שחזרתי הושארת על ffidort, שניה אוח-מעבד על שלא שיניב
בתוכנות עזוב.フリー, הזבר את א PARTICULAR, שלית סיבוב עידוניות:

1. המודליזציה של diyff (ראה פרק 5).

2. התפוש של diyff מספק את כל האניופוסים הדורות לביצוע קשיה, ליא שוגי
בתוכנות עזוב, של החופשת על diyff של T_EX\\tex. המודע הדורות להינפנף כון סדר
דפוס של diyff, שעושי עדח התוכנית, ויצא קשיה (כפי שהוסבר בפרק היקוד),

תואם:

- המיקום האופקי של יינו מתחוונות בשורה.
- סוף חופן של יינו מתחוונות בשורה.
- סמן לסף מילה.
- סמן לסף מילה.
- סמן לסף מילה.

ה апр訴ות לחתות בוחה חתנות
- האפópezות שעריך לברק אוחרי כיתוב.
ככל מתחוונות נייל מוספים מแปลก של diyff, לועמק הפולט של T_EX\\tex, שיאני เมיל ספוגים
$ll 4.5i$
.EQ
delim $$
define circint \% \% \"s+8f(sy\z\{ls\s0fP\(cl\) \%$
define thf \% \% \".\v^-5m'.\v'.5m'\." \%$
.EN
.PR
VF(HF%Nvay"mer "eloqim:%F\fR
.sp
.ce 100
$\epsilon sub o circint bold E cdot d bold S = q$
.sp
$circint bold B cdot d bold S = 0$
.sp
$circint bold B cdot d bold l = \mu sub o \epsilon sub o \{d PHI sub \{bold E\}\}
 over \{d t\} + \mu sub o l$
.sp
$circint bold E cdot d bold l = \{- d PHI sub \{bold B\}\} over \{d t\}$
.ce 0
.sp
$thf$
.sp
.ce 100
$c = 1 over \{sqrt \{\mu sub o \epsilon sub o \}\}$$
.ce 0
.sp
VF(HF%Nvayehi "or
\[ \varepsilon_0 \oint E \cdot dS = q \]

\[ \oint B \cdot dS = 0 \]

\[ \oint B \cdot dA = \mu_0 \varepsilon_0 \frac{d\Phi_E}{dt} + \mu_0 i \]

\[ \oint E \cdot dA = \frac{-d\Phi_B}{dt} \]

\[ c = \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \]
Basic Multi-Lingual Nature of Modern Hebrew

In modern Hebrew, even if you think you do not have to worry about mixed-language and mixed-direction text, you do!

Modern Hebrew uses the same numerals as do Latin languages, the so-called Arabic numerals, which are written from left to right (i.e., most significant digit to the left), with the same glyphs as in Latin languages.

When is it legal to park? Nu?
Typesetting Chinese and Japanese characters
by Ip and Chow

Their character sets

> 256 Characters
   CHINESE — HANZI
   KANJI OF JAPANESE
   HANZI OF KOREAN
   \{REALLY SAME SET\}

All are ideographic
> 10K and for all practical purposes
< 64K characters

2 bytes needed

character set organized as matrix:
   1 byte = row index
   1 byte = column index

JIS: 94×94 (JAPANESE)
GB2312: 94×94 (PRC CHINESE)
KOR: 94×94 (KOREAN)
???: 80×120 (ROC CHINESE)
Easy to add capability to print JIS, GB2312, etc character set to ditroff with NO change to dtroff itself.

**ditroff** allows up to 255 fonts and 254 characters per font.

So $94 \times 94$ matrix is made a collection of 94 fonts, each with 94 characters.

Character $ab,cd$ of matrix is called

\[ \text{\textbackslash f}(ab) \quad \text{\textbackslash c}(cd) \]
font character

\[ ab \quad cd \]

Just need filters from standard codings to this input form.
An Adaptation of the UNIX ditroff for Formatting Tri-Directional Text

by

Zeev Becker
Daniel M. Berry

Computer Science Department
Technion
Haifa 32000
Israel

泽夫·贝克尔
丹尼尔·M·贝利

ゼフ・ベケル
ダニエル・ベリ
Abstract

This paper describes a system for formatting documents consisting of text written in languages printed in three different directions, left-to-right, right-to-left, and top-to-bottom. For example, this paper is such a document because it contains text written in English, Hebrew, Japanese, and Chinese. The system assumes that the input is in the order in which the text is read aloud, and it produces output in which each language is printed in its own correct direction, but for which a human cognizant of the reading conventions will reproduce the input order. The system consists of three major pieces of software: Ossana and Kernighan’s ditroff, for formatting text consisting of only left-to-right or unidirectional text, Buchman and Berry’s ffortid™ for arranging that right-to-left text buried in ditroff output is printed from right to left, and a new program \b’ditroff’, for arranging that top-to-bottom text buried in ditroff output is printed from top to bottom.
תקציר

מאמר זה מתאר שערכת עלירעכת מסמכים המכלים שפות הנכתבות בשילושה בשפה שגרית, משלilarity לים, מימי למשאל וממלעצלת לפשה. לדורגה, מאמר זה הורג המסך בצורת ממומת של מימי סמליםとなっている באנגלית, עברית, יפנית וסריגה. הערכה מبذלה כיוון הקטן ביותר ובו זוה נקודת בול, זו היא מכילה של כשיהמודת בכותב הנכון, או אדום המודע לממדות המקרא של איז שפה סמלית shader Kernighan ר Ossana של ditroff אispens מחודש את סדר הקטן, הערכה מורכבת משילושה הלכית עיקריית:UNG נרitu בתסום הקטוב רכ משמיא ליימי נירitta מביתית הדגיטאלה בפלס השי ديuffed, נירitta מחודש התסום הממלעצלת לפשה התסום בפלס של השי, b'ditroff' נירitta ממלעצלת לפשה.
アブストラクト

左から右、右から左、上から下、の異なる3つの方向に印字される複数の言語によって記述されたドキュメントをフォーマットするシステムを紹介する。例えば本論文は、英語、ヘブライ語、日本語、中国語で記述されており、そのようなドキュメントの1つである。本システムは、テキストの読まれる順に入力が与えられたものとして、それぞれの言語が正しい方向に印字されるように出力を生成するものであるが、読む方を知っている人間なら、出力から入力順序を再構成することも可能である。本システムは、左から右または単一方向のテキストのみからなる文書をフォーマットするための、OssanaとKernighanのditroff、ditroffの出力に埋め込まれた右から左のテキストを正しく印字するための、BuchmanとBerryのffortid、そしてditroffの出力に埋め込まれた上から下のテキストを正しく印字するための、新しいプログラムb'ditroff、の3つの主要ソフトウェアから構成されている。
ドキュメント

下記のテキストを正しく入れ込むためには、右から左に書き込むことが必要です。テキストの方向を変えると、読む方向が逆になることがあります。英文の場合は、右から左に書きます。日本語の場合、左から右に書きます。中国語、ヒンディー語、アラビア語など、言語によって別の規則があります。
在一些情況下，一文件可能會用上幾種不同語文，而本章就是介紹如何替此等正文輸入正當的格式。不同語文有不同的編排規則，即如此文便可作一例，因文中同時套用了英文，德文，日文及中文，故此同一文件之中，便有三種編排格式，分別為：由右至左、由左至右、及由上至下。此類系統的排列，蓋以讀讀時的排序為原則，從而輸入合乎各語文的編排方向，但操作人員必須熟識各語文的傳統格式，此類編制格式系統，主要有三部軟件，分別為：Ossana and Kernighan’s ditroff（用作理由左至右或單一方向排列的正文），Buchman and Berry’s ffortid（用作理貯於），'b'ditroff'（的排列為上至下的正文）。
正 o  m 方 f g a 三，员各的此左种，英可不输本可
文 f 理 s a 向（h n 都此必语排类、篇故文作同入章能
）f 贰 n 排用 a a 類须文序系由排此，一的正就会在
。'於 f 列作 n 軟編熱的為統左格同例編當是用
（）fa 的' a 件制識編原的至式一伯，排的介上些
的，o n 正理 s n ，格各排則排右，文 因規格紹幾情
排 r d 文由 d 分式語方，列，分件文文則式如種況
列 b t ) 左 d 別系文向従，及別之，中，何不下
為 ' i B，至 i K 爲統的，而蓋由為中日同即不替同，
上 d d e B 右 t e ：，傳但輸以上：，文時如同此語一
至 i ( r u 或 r O 主統操入誦至由便及套此語等文
下 t 用 r c 署 o n s 要格作合讀下右有中用文文正，文
的 r 作 y h f i s 有式人乎時，至三文了便有文而件
NEED FOR TRI-DIRECTIONAL FORMATTING

In PRC, Xinjinang Uighur autonomous region, have documents in:
- Uighur, Kazak, Kirgiz, Mongol R-L
- English (Technical) L-R
- Chinese T-B

In SINGAPORE, all over:
- English, Malay, Tamil L-R
- Chinese T-B
- Arabic, Urdu R-L

In HONG KONG newspapers:
- Chinese Text T-B
- Chinese Headlines R-L
- English Advertisements L-R

etc.
Input (shown in stylized form),

.ft R "Roman

English
.ll 4i " line length 4 inches
.br
.PR " predominantly right-to-left
.ft HB "Hebrew

יריבי
.br
.PL " predominantly left-to-right
.ft KT "Katakana
カタカナ
.br
.BT " begin top-to-bottom
.ft HR "Hiragana
ひらがな
.sp
.ft CH "Chinese
漢字
.br
.ET " end top-to-bottom
Assuming English and Katakana printed from left to right
Hebrew printed from right to left
Hiragana and Chinese printed from
top to bottom with columns laid out from right to left,
Output is something like:

English

עברית

カタカナ

ひらがな

漢字
OBSERVED PROPERTIES OF CHINESE/JAPANESE TOP-TO-BOTTOM TEXT

Characters are printed in rectangular grid with NO extra space between "words" and NO attempt to avoid breaking lines in middle of "words" and just before punctuation —

EVEN when Latin text is embedded within.

Grid arrangement fits in nicely with constant sized square Hanzis:
OTHER PROPERTIES OF DOCUMENT CONTAINING L-R, R-L, AND T-B TEXT

1. Given horizontal line contains
   Either
   L-R and R-L (horizontal) text
   Or
   T-B (vertical) text
   NOT both

2. Within region of horizontal text, not move down to next line, until have read ALL text in one line, possibly bouncing within the line

   ─────────────────────────────────────────
   ─────────────────────────────────────────

   NEVER, EVER, EVER MOVE UPWARD!!!!
3. Within horizontal line, any permutation of the characters in the line yields a line of the same length (can fix kerning to preserve this property)

4. Within contiguous rectangle of Japanese/Chinese text, any permutation of the characters in the rectangle yields a rectangle of the same size (here, trailing blanks to fill out the rectangle are considered characters also).
INPUT — in Time Order:

blah blah

break-line

.BT

{ 

start-T-B-text-marker

1 2 3 4 5 6
7 8 9 10 11
12 13 14 15
16 17 18 19 20 21
22 23 24 25

end-T-B-text-marker

.ET

{ 

break-line

bleh bleh


OUTPUT:
ditroff formats this into:

```plaintext
blah blah
```

```
1  2  3  4  5  6  7
8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25  
```

bleh bleh

bditroff reorganizes the $4 \times 8$ rectangle to get the following:

```plaintext
blah blah
```

```
25 21 17 13 9 5 1
22 18 14 10 6 2
23 19 15 11 7 3
24 20 16 12 8 4
```

bleh bleh
EXAMPLE INPUT:

.ft R
\$9ada is a trademark of the u.s. dept. of defense.
ms-dos is a trademark of microsoft, inc.\$10
.br
%!x TS
.ft C
A B C D E F
G H I J K L
M N O P Q R
S T U V
W X Y Z
.br
%!x TE
.ft R
\$9ffortid is a trademark of berry computer scientists,
unix is a trademark of at&t bell laboratories.\$10
ada is a trademark of the u.s. dept. of defense. ms-dos is a trademark of microsoft, inc.

A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y
Z

ffortid is a trademark of berry computer scientists, ltd. unix is a trademark of at&t bell laboratories.
OUTPUT AFTER ditroff/bditroff
Assuming all input fits on one output page:

ada is a trademark of the u.s. dept. of defense. ms-dos is a trademark of microsoft, inc.

Y S M G A
Z T N H B
U O I C
V P J D
W Q K E
X R L F

ffortid is a trademark of berry computer scientists, ltd. unix is a trademark of at&t bell laboratories.
Page break ends last explicitly began region and begins another on the next page. This new region is ended by the first explicit region-end mark.
(which matches the begin-marker for the page-break-ended region)

OUTPUT AFTER ditroff/bditroff
Assuming page break between “O” and “P”:

```
ada is a trademark of the u.s. dept. of defense. ms-dos is a trademark of microsoft, inc.
M J G D A
N K H E B
O L I F C
```
fortid is a trademark of
berkeley computer scientists, ltd. unix is a trademark of at&t bell laboratories.
In all of the above,
    there were NO changes to dtroff
except to recompile it with larger table sizes
to allow up to 255 fonts
    as opposed to the default 10
and to fix bugs (sigh!)
    exposed by this size change
vi.îv, a Bi-Directional vi by Habusha

Same trick is played for vi.îv, except that because vi, as all editors must be, is a monolithic program, the change is made to the base program, vi

The line reorganization algorithm is inserted into the screen manager

Each time any character in any line is changed, the line is subjected to the algorithm and is redrawn on screen
vi.iv assumes

Files stored in time order

LR language text with eighth bit off
RL language text with eighth bit on

Terminal described by `termcap`

Displays all text in visual order as a function of
text itself
AND current view direction

which is `:set table`

NOTE: nothing particular to Hebrew here!
Structure of vi:

Structure used by emacs
used by anything based on curses
Ideal structure for vi.iv

Trying to build vi.iv with as few changes as possible to vi

Clear that have to change editing part for new commands, e.g.,
   change view direction
   change input language/direction

But try to do ALL other changes ONLY in

   Screen Manager

Get screen manager to apply layout algorithm for each line changed as a result of an editing command and to send ONLY these lines to screen
Why is this approach good?

It’s lazy!!!

Largely unchanged editing part insures downward compatibility

Note that because files and input are in time order, unchanged editing part, INCLUDING pattern matching, works!!
If I am not for myself, then who will be?
If I am only for myself, then what am I?
If not now, then when?
-Pirke Avoth
MINIX.XINIM, a Di-Directional MINIX, a mini-UNIX
by Allon

Same trick is played to build
MINIX.XINIM from MINIX.

The line reorganization algorithm
is put into the device drivers for
the screen, the line printer,
and any other device whose output
is human-read.

Each line that passes through is
subject to reorganization before
being thrown at the device

The result is that from this simple change,
the kernel and ALL line-oriented
application programs become bi-directional
with NO change to the rest of the kernel
or to any of the application programs.
Line-oriented applications include

    sh csh cat more grep gres sort uniq ed

and do not include programs that
    write to the full screen such as

    vi emacs and
    curses-based applications

However the latter can be made bi-directional
by inserting the line reorganization algorithm
into curses, as was done to vi
Arabic and Farsi formatting by Srouji

Arabic and Farsi are R-L like Hebrew

But ... other problems

Different forms of letters based on position in word

Have input preprocessor that identifies position and translates letter code into glyph code

So input is in pure spelling form the form assumed by standard codes ASMO and IISCI

People think pure spelling and it is suited for alphabetization
Connecting letters

just have base line segment of each letter be in

same vertical position,
same thickness, and
flush to bounding box on connecting side

keshide —
stretch either the last letter or
the connection to the last letter
in a line
rather than spreading words to achieve
left justification

How to achieve keshide without changing \texttt{dtroff}?
In any case, ffortid works by totally reformating line

Input in ditroff intermediate form

Extract the text of the line from input

Reorganize text of line so that
R-L-font text is printed R-L

Use dtroff’s line-filling algorithm with
dtroff’s width tables to build
unjustified line and to calculate
distance $\delta$ from end of text to end of line

Divide $\delta$ by number of interword gaps
(with a smidgen more after sentence punctuation)
and distribute that amount of white space
to each interword gap
For Arabic and Farsi, add an option that either

inserts a base-line filler of $\delta$ units before last connecting-before letter in the line

divides $\delta$ by number of words having a connecting-before letter and inserts base-line fillers of that length before last connecting-before letter of each word in the line

To stretch last letters themselves needs dynamic font; J. André has proposed and illustrated them

We get keshide WITHOUT changing `dtroff` itself!!
Indexing
(Not really multilingual, but making use of trick!)

Usual technique in ditroff:
   flood document file with commands
   .tm term \n%
   which send term and current page number
   to standard error

Similar device in \TeX
Abe wrote program `indx` with input:

one file with index terms

ditroff intermediate form output of document to be indexed

Document broken into lines AND pages whose numbers are known

Note that because of end-of-word markers, input words can be found
idx finds page numbers
   (and even line numbers, if so requested)
of all occurrences of each index term

and builds ditroff input file for index

Exact format of index is determined
by definitions of macros invoked in
this file

NO need to flood document file with
   indexing instructions
Document file is kept clean for editing,
   greping, spelling, proofing,
   dictioning, doubleing, etc.
Page Markup

Problems with all batch, one-pass formatters, such as ditroff

1. Widow and orphan lines

   Last line of paragraph at top of page
   first line of paragraph at bottom of page

   Second is avoidable by not beginning a paragraph UNLESS there is room for at least two lines (.ne command)

   But avoiding first requires look ahead
2. Figure placement

Presently handled by macro packages using diversions, but very UNSTABLE

3. Multiple column text
   figure placement
   starting single column after a bit of double-column output

4. Balancing pages
   all pages same length

The solutions to all of these really require two-pass algorithms
Kernighan’s solution: Program pm

First turn off pagination

In any case, \texttt{dttroff} knows NOTHING about page length

Page length implemented by macro packages using traps that can be set at particular distances from top of current page

So now \texttt{dttroff} thinks that whole document is ONE page!
Change macros for UNITs to output
BEGIN-UNIT and END-UNIT markers only
   in particular NO floating

So get text broken into lines on
   one LONG page with
beginnings and ends of
   paragraphs, figures, footnotes, etc.
marked and appearing in the order
that they were input

pm accepts ditroff intermediate format
and outputs in the same format

However, it uses the fact that it is a
second pass over the text to do
a good job of figure and footnote placement
in balanced, possibly multi-column pages,
with no widows and orphans

The algorithm for figure placement is greedy
So it is stable
Because this was done with NO change to \texttt{dtroff}
still, its line-breaking and justification leaves much aesthetics to be desired

Since \texttt{dtroff} does it line-by-line,
interword gaps of two different lines can be radically different (albeit all the same within each line)

\TeX{}’s algorithm does line balancing for whole paragraph BEFORE outputting any line

Results are visibly better
Note that NONE of the above tricks can be performed with \TeX!!!

\TeX DVI format does not have the necessary information
end-of-line marker
end-of-word marker

Thus to do the above tricks, it is necessary to either change DVI format OR do the trick inside a modified \TeX

Necessitates change to \TeX in either case.

Nightmare of one or more of

necessitating all DVI processors out there to be changed
distributing a new version of \TeX
maintaining several programs with mostly identical code

None is very appetizing!
To make bi-directional $\TeX$, Knuth and Mackay opted to change $\TeX$ itself to make $\TeX/\XeTeX$.

But it is clearly harder to make $\TeX/\XeTeX$ as one monolithic program (without damaging pure $\TeX$ part) than it is to make a $\XeTeX$ (in the model of $\ffortid$).
Modularity of ditroff system paid dividends in speed and security.

We were able to QUICKLY get the new functions in

WITHOUT having to diddle with any of AT&T’s widely-distributed ditroff programs

WITHOUT risk of changing ditroff functionality and of eliminating bugs that had become features!
time line
Journals I have typeset my own articles for (in ditroff, of course)

- Electronic Publishing
- Journal of Logic Programming
- Journal of Computer Languages
- Journal of Systems and Software
- Software—Practice and Experience
- ACM Transactions on Programming Languages and Systems
- TUGboat (!)

Journals that use ditroff technology

- Electronic Publishing (also accepts TE\TeX input)
- Interactive Learning International
- Journal of Software Maintenance
- AT&T Technical Journal
Open Research Problems

Extensible WYSIWYG formatter

Bi-directional UNIX

Tri-directional vi, MINIX, and UNIX
what does scrolling mean?

Better Paragraphing in dtroff

Multilingual X Windows

Dynamic fonts with stretchable letters