TECHNION – ISRAEL INSTITUTE OF TECHNOLOGY
Haifa 32000, ISRAEL

Computer Science Department

Fishbach 479, Tel.: +972-4-294325
Telex: 46406 TECON IL, Fax: +972-4-221581
BITNET: DBERRY@TECHSEL
CSNET: dberry@sel.technion.AC.IL

Multi-Lingual Word-Processing Research at the Technion

by

Daniel M. Berry

דניאל ברי

ダニエル・ベリ

丹
尼
儿
北
利

Computer Science Department
Technion
Haifa 32000
Israel

Software Engineering Institute

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

Room 4212, Tel.: 412-268-7778
Fax: 412-268-5758
CSNET & INTERNET: dberry@sei.cmu.edu
BITNET: dberry%sei.cmu.edu@CMUCCVMA

מחקר
בעבוד תמללים
רב שפות
בטכניון

ד״ר דניאל ברי
מכון להנדסת תכנה
אניברסיטת קרנגי מלון
פיטסבורג, פנסילווניה
ארה״ב

APPENDIX I

STAR TREK
THE NEXT GENERATION®

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!

מסע בין כוכבים – הדור הבא

החלל – הגבול הסופי.
אלה מסעותיה של החללית „אנטרפרייז”,
במשימתה המתמשכת לחקר עולמות חדשים מוזרים, לאתר חיים חדשים ותרבויות חדשות, משימה נועזת אל עבר הבלתי נודע!

الرحلة بين الكواكب ـ الجيل القادم

الفضاء ـ الحد الأقصى.
هذه هي رحلات السفينة الفضائية «إنترپرايز»،
في مهمتها المستمرَّة بالبحث عن عوالم غريبة، واكتشاف حياة جديدة وحضارات جديدة، مهمة جريئة في الذهاب إلى حيث لم يذهب احد من قبل!

Technion – Israel Institute of Technology
Faculty of Computer Science, Technion City, Haifa 32000, Israel

丹尼儿北利

Даниэль М. Бэри
دانيال بيري
Δανιηλ Μ. Μπερι
דניאל ברי

Daniel M. Berry

Professor

ढाणीयळ बेरी
ダニエル・ベリ
ዳንኤል ቤሪ
다니엘 베리

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
ß æ Æ ø Ø ...

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 Languags
  - 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

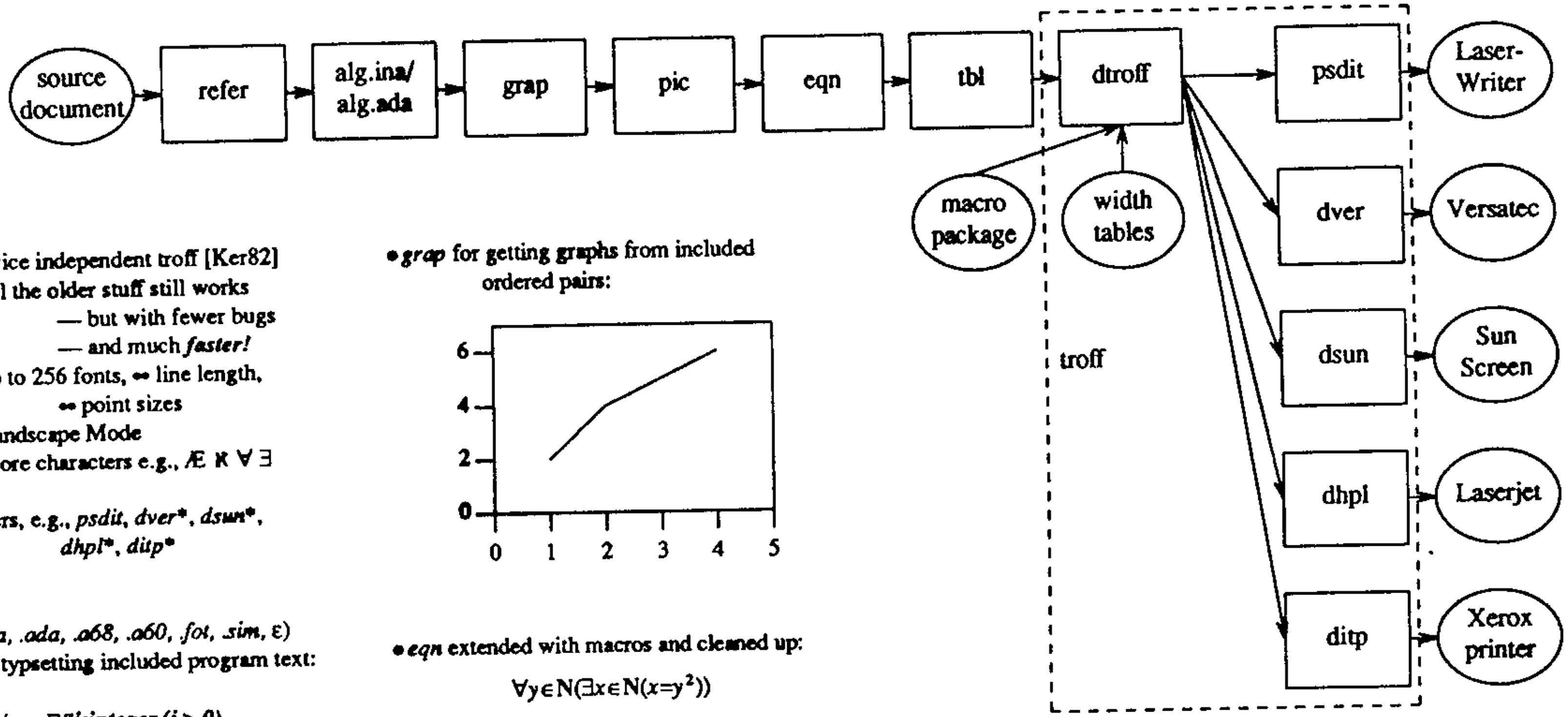
Daniel M. Berry

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., Æ ℵ ∀ ∃
- device drivers, e.g., *psdit*, *dver**, *dsun**, *dhpl**, *ditp**
- *refer* fixed
- *algχ* (χ=*.ina*, *.ada*, *.a68*, *.a60*, *.fot*, *.sim*, ε) for typsetting included program text:

  `axiom` ***E"i:integer (i > 0)***

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

- *grap* for getting graphs from included ordered pairs:

- *eqn* extended with macros and cleaned up:

$$\forall y \in N(\exists x \in N(x=y^2))$$

- tbl cleaned up slightly

x	y
1	2
2	4
4	6

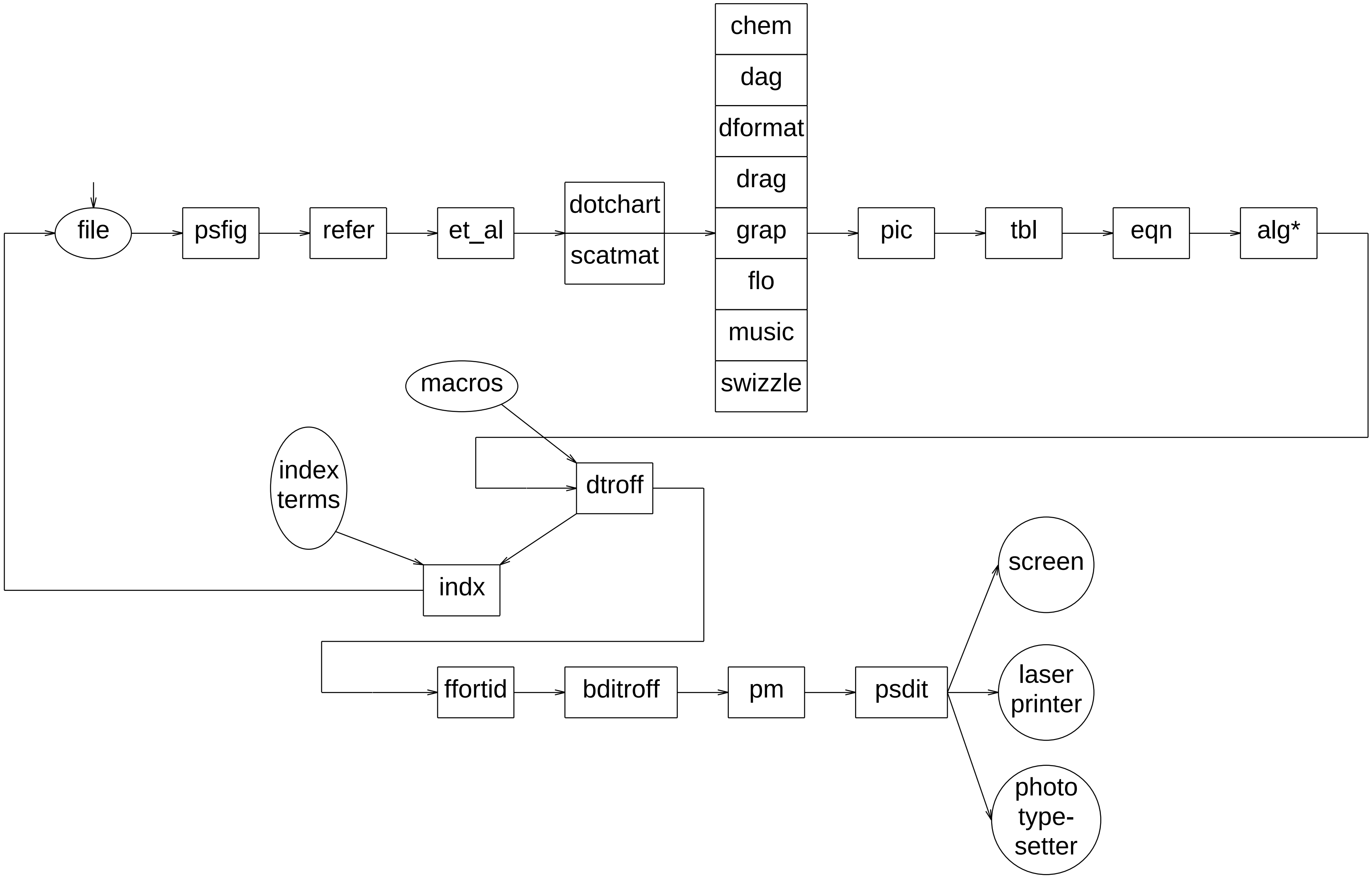
References

[Ker82] Kernighan, B.W., "A Typesetter-Independent TROFF," Computing Science Technical Report No. 97, Bell Laboratories, Murray Hill, NJ 07974 (March, 1982).

---

* Not yet up

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

Program	Pre/Post	of What?	Purpose
Batch			
eqn	pre	dtroff	Formulae
tbl	pre	dtroff	Tables
pic	pre	dtroff	Line Drawings
ideal	pre	dtroff	Line and Filled Drawings
psfig	pre	dtroff	Including Arbitrary POSTSCRIPT Files
refer	pre	dtroff	Bibliographical Citations
indx	post	dtroff	Indexing
ffortid	post	dtroff	Handling Right-to-Left Text
bditroff	post	dtroff	Handling Top-to-Bottom Text
pm	post	dtroff	Page Markup and Figure Placement
psdit	post	dtroff	Translating dtroff Output to POSTSCRIPT
dformat	pre	pic	Data Format
swizzle	pre	pic	Sorting Exchanges
chem	pre	pic	Chemical Diagrams
m2p	pre	pic	Musical Score (that drives speaker) to pic
dag	pre	pic	Directed Acyclic Graphs
drag	pre	pic	Drawing Graghs
flo	pre	pic	Flow Charts
dotchart	pre	grap	Dot Charts
scatmat	pre	grap	Scatter Matrices
et_al	post	refer	Replace “Al, Et” by “*et al*”

Interactive

monk	pre	dtroff	Preparing dtroff Input from Window Interface
picasso	pre	pic	Preparing pic Input from Window Interface
fig	pre	pic	Preparing pic Input from Window Interface
suntroff	post	dtroff	Translating dtroff Output to Sunview
xtroff	post	dtroff	Translating dtroff Output to X window

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 $\TeX$?

MONOLITHIC

formulae
tables
limited line drawing

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., –ms, –me, –mm, –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:

Latin — Hebrew (ASCII/ESCII)

		x	x	x	
A	A	E	E	E	A

Latin — Japanese (ASCII/Shift JIS)

	x	x		x	x
A	J	J	A	J	J

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)

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

```
|40   @ |41   A |42   B |43   C |44   D |45   E |46   F |47   G |
|48   H |49   I |4A   J |4B   K |4C   L |4D   M |4E   N |4F   O |
|50   P |51   Q |52   R |53   S |54   T |55   U |56   V |57   W |
|58   X |59   Y |5A   Z |5B   [ |5C   \ |5D   ] |5E   ˆ |5F   _ |
|60   ‘ |61   a |62   b |63   c |64   d |65   e |66   f |67   g |
|68   h |69   i |6A   j |6B   k |6C   l |6D   m |6E   n |6F   o |
|70   p |71   q |72   r |73   s |74   t |75   u |76   v |77   w |
|78   x |79   y |7A   z |7B   { |7C   | |7D   } |7E   ˜ |7F  DEL|
```

Hebrew Half

```
|80  NUL|81  SOH|82  STX|83  ETX|84  EOT|85  ENQ|86  ACK|87  BEL|

|88  BS |89  HT |8A  NL |8B  VT |8C  NP |8D  CR |8E  SO |8F  SI |

|90  DLE|91  DC1|92  DC2|93  DC3|94  DC4|95  NAK|96  SYN|97  ETB|

|98  CAN|99  EM |9A  SUB|9B  ESC|9C  FS |9D  GS |9E  RS |9F  US |

|A0  SP |A1   ! |A2   " |A3   # |A4   $ |A5   % |A6   & |A7   ’ |

|A8   ( |A9   ) |AA   * |AB   + |AC   , |AD   - |AE   . |AF   / |

|B0   0 |B1   1 |B2   2 |B3   B |B4   4 |B5   5 |B6   6 |B7   7 |

|B8   8 |B9   9 |BA   : |BB   ; |BC   < |BD   = |BE   > |BF   ? |
```

```
|C0   @ |C1   A |C2   B |C3   C |C4   D |C5   E |C6   F |C7   G |
|C8   H |C9   I |CA   J |CB   K |CC   L |CD   M |CE   N |CF   O |
|D0   P |D1   Q |D2   R |D3   S |D4   T |D5   U |D6   V |D7   W |
|D8   X |D9   Y |DA   Z |DB   [ |DC   \ |DD   ] |DE   ^ |DF   _ |
|E0   א |E1   ב |E2   ג |E3   ד |E4   ה |E5   ו |E6   ז |E7   ח |
|E8   ט |E9   י |EA   ך |EB   כ |EC   ל |ED   ם |EE   מ |EF   ן |
|F0   נ |F1   ס |F2   ע |F3   ף |F4   פ |F5   ץ |F6   צ |F7   ק |
|F8   ר |F9   ש |FA   ת |FB   { |FC   | |FD   } |FE   ~ |FF DEL|
```

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

UNICODE -2

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 dtroff
and of input to device drivers.

```
This is a line.
```

is translated by dtroff 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 “םולש” in Hebrew.`

After formatting with dtroff (schematically)

He said to
Dan “םולש”
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:

הוא אמר
לדן „Hello”
באנגלית.

Time-ordered input:

הוא אמר לדן “Hello” באנגלית.

After formatting with dtroff (schematically)

אוה רמא
ןדל ”Hello„
תילגנאב.

Can get what we want line-by-line by first flipping whole line

הוא אמר
לדן „olleH”
באנגלית.

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)

```
for each line in the file do
    if the current document direction is L-R then
        reverse each contiguous string of RL
                characters in the line
    else (the current document direction is R-L)
        reverse the whole line;
        reverse each contiguous string of LR
                characters in the line
    fi
od
```

Input:

```
\fRThe next sentence contains one verb.
\fHהמשפט הזה כולל פועל אחד\fR.
The previous sentence contains one verb.
```

Output:

> The next sentence contains one verb. **המשפט הזה כולל פועל אחד**. The previous sentence contains one verb.

Input:

```
\fHהמשפט הבא כולל פועל אחד.
\fRThis sentence contains one verb\fH.
המשפט הקודם כולל פועל אחד.
```

Output:

המשפט הבא כולל פועל אחד. This sentence contains one verb. המשפט הקודם כולל פועל אחד.

שם

ffortid—

מצא והפוך את הטקסט הנכתב מימין לשמאל, ועמד שורות הכתובות בשפה הערבית ע״י מתיחת המילים. של troff,

תמצית

ffortid [**–r***font-position-list*] [**–w***paperwidth*] [**–a***arabic-position-list*] [**–s**[nfla]] [file] ...

תאור

התפקיד של *ffortid* , הוא לקרוא את הפלט של *ditroff*, המעובד משמאל לימין, לחפש טקסט הנכתב מימין לשמאל, להפוך אותו, ולסדר מחדש את השורות, כך שהטקסט בכ״א מהפונטים יראה בכוון הטבעי שלו. *ffortid* מקבל את הקלט שלו מ-*ditroff*, ולכן הוא לא יודע ,ולא צריך לדעת, על אף אחד מקדם המבדים שלו. בנוסף לכך, הפלט של *ffortid* זהה מבחינה סינטקטית לפלט של *ditroff*, ועובר דרך אותם נוהגי התקנים (device drivers).

ברירת המחדל, שכל הפונטים נכתבים משמאל לימין, והארגומנט **r**–, משמש לקביעת כוון הפוך (מימין לשמאל) של כ״א מהפונטים הניתנים ברשימת ה־font-position-list.

תוצאות העיבוד של *ffortid* תלויות ,בנוסף לכוון של כ״א מהפונטים בטקסט, בכוון העיבוד הנוכחי של הטקסט, שעלול להיות משמאל לימיו או מימין לשמאל. כוון העיבוד ההתחלתי הוא משמאל לימין, וניתן לשינוי ע״י שתי הפקודות (macros) **.PL** ו־ **.PR** הקובעות את כוון העיבוד משמאל לימין ומימין לשמאל, בהתאמה.

אם כוון העיבוד הנוכחי משמאל לימיו, עימוד השורות והאינדנטציה נראים משמאל לימין. בכל שורה, אוסף המילים הנכתבות מימין לשמאל, נהפכות סביב הציר שלהן.

חשוב לציין ש-*ffortid* משתמש ברוחב הדף (paper width), על מנת לקבוע את רוחב השוליים. אם למשל, הטקסט מעובד להדפסה על רוחב דף 8.5inch, ורוחב שוליים שמאליות (page offset) של 1.5inch, ורוחב השורה הוא 6inch, אזי רוחב השוליים הימניות יהיה 1inch (כאשר כוון ההדפסה משמשל לימין). אם לעומת זאת, כוון ההדפסה הוא מימין לשמאל, אזי רוחב השוליים הימניות יהיה 1.5inch , והשמשליות

7. יתרונות הגישה

המערכת שהורחבה בעבודה זו, פותרת את כל בעיות עיבוד השפה הערבית שהוזכרו, ועונה על הדרישות. התכונות העיקריות של המערכת, במשולב עם ditroff הקיים, הן:

1- סדר דפוס טקסט רב לשוני הנכתב בכוונים שונים (מימין לשמאל, משמאל לימין, ומלמעלה למטה).

2- יכולת הקישור בין אותיות המילים הנכתבות בשפה הערבית, ולקבוע את מיקומן במילה.

3- האפשרות לעימוד שורות טקסט ערבי ע״י קשידה.

4- זיהוי וטיפול בהרכבות וניקודים. הטיפול בניקודים, כולל גם עדכון דינמי של מיקומם האנכי ביחס לאותיות.

5- יכולת סדר דפוס טקסט מדעי המכיל ציורים, משוואות, גרפים, וכו׳.

6- מתן איכות הדפסה גבוהה. הפלט עובר דרך נוהג התקן, ומתורגם לשפת מדפסת laser בפורמט של POSTSCRIPT [AdbT85, AdbR85] .

חשוב לציין, שההרחבה נעשתה על ffortid, שהוא אחר-מעבד של ditroff, ללא שום שינוי בתוכנית ditroff עצמה. הדבר היה אפשרי בגלל שתי סיבות עיקריות:

1- המודולריות של ditroff (ראה פרק 5).

2- הפלט של ffortid מספק את כל האינפורמציה הדרושה לביצוע קשידה, ללא שינוי בתוכנית עצמה, שלא כמו בפלט של TEX למשל. המידע הדרוש להיפוך כיוון סדר הדפוס של טקסט, חישוב ערך המתיחה, וביצוע קשידה (כפי שהוסבר בפרק הקודם), הוא:

- המיקום האופקי של כ״א מהתווים בשורה.
- סוג הפונט של כ״א מהתווים בשורה.
- סמן לסוף מילה.
- סמן לסוף שורה.
- האפשרות לחישוב רוחב התו (character width). רוחב התו הוא ערך התזוזה האופקית שצריך לבצע אחרי כתיבתו.

כל הנתונים הנ״ל מסופקים בפלט של ditroff, לעומת הפלט של TEX, שאינו מכיל סמנים

Input:

```
.ll 4.5i
.EQ
delim $$
define circint %% "\s+8\f(sy\z\(is\s0\fP\(ci" %%
define thf %% ".\v'-.5m'.\v'.5m'." %%
.EN
.PR
\f(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
$circint bold B cdot d bold l = mu sub o epsilon sub o {d PHI sub {bold E}}
over {d t} + mu sub o i$
.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
\f(HF%Nvayehi "or
```

ויאמר אלוקים.

AND G-D SAID

$$\epsilon_o \oint \mathbf{E} \cdot d\mathbf{S} = q$$

$$\oint \mathbf{B} \cdot d\mathbf{S} = 0$$

$$\oint \mathbf{B} \cdot d\mathbf{l} = \mu_o \epsilon_o \frac{d\Phi_{\mathbf{E}}}{dt} + \mu_o i$$

$$\oint \mathbf{E} \cdot d\mathbf{l} = \frac{-d\Phi_{\mathbf{B}}}{dt}$$

$\therefore$

$$c = \frac{1}{\sqrt{\mu_o \epsilon_o}}$$

ויהי אור

AND THERE WAS LIGHT

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.

אין חניה: 8:00-17:00.

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 REALLY SAME SET
HANZI OF KOREAN

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

dtroff allows up to 255 fonts and 254 characters per font

So 94×94 matrix is made a collection of 94 fonts, each with 94 characters

Character ab,cd of matrix is called

\f(ab	\(cd
font	character
ab	cd

Just need filters from standard codings to this input form

d
i
t
ditroff/ffortid/r
o
f
f

An Adaptation of the UNIX ditroff for Formatting Tri-Directional Text

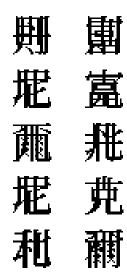
by

Zeev Becker
Daniel M. Berry

Computer Science Department
Technion
Haifa 32000
Israel

זאב בקר
דניאל ברי

ゼエブ・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 מערכת לעריכת מסמכים המכילים שפות הנכתבות בשלושה כוונים שונים, משמאל לימין, מימין לשמאל ומלמעלה למטה. לדוגמא, מאמר זה הנו מסמך כזה, מכיוון שהוא מכיל טכסט הכתוב באנגלית, עברית, יפנית וסינית. המערכת מניחה כי הקלט נתון בסדר בו הוא נקרא בקול רם, והיא מפיקה פלט שבו כל שפה מודפסת בכוון הנכון, אך אדם המודע למוסכמות הקריאה ייצר מחדש את סדר הקלט. המערכת מורכבת משלושה חלקים עיקרים: ditroff של Ossana ו־Kernighan לעיבוד טכסט הכתוב רק משמאל לימין או בכוון אחד, ffortid של Buchman ו־Berry לסידור הטכסט הכתוב מימין לשמאל הטמון בפלט של ה־ditroff כך שיודפס מימין לשמאל, ותכנית חדשה \b’ditroff’, לסידור הטכסט מלמעלה למטה הטמון בפלט של ה־ditroff כך שידפס מלמעלה למטה.

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、読み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。

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、読み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、Ｏｓｓａｎｎａ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。

擇要：

在 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， 伯 文，日文及中文，故此同一文件之中，便有三種編排格式，分別爲：由右至左、由左至右、及由上至下，此類系統的排列，蓋以誦讀時的排序爲原則，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，分別爲：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’（的排列爲上至下的正文）。

在 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， 伯 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，蓋以誦讀時的排序為原則，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)。

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
עברית
.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

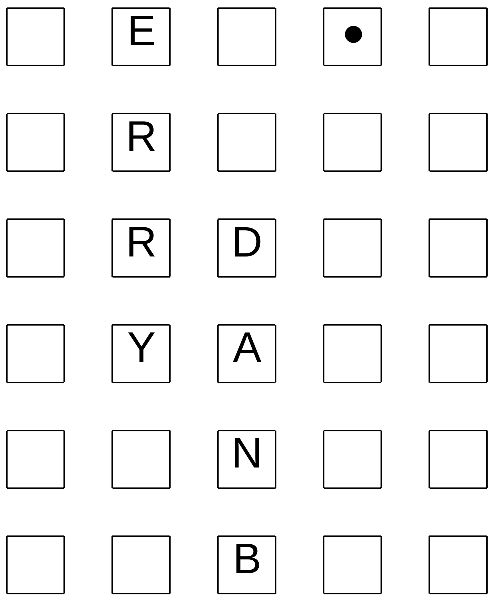
漢字

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!!!!

STILL OTHER PROPERTIES OF DOCUMENT CONTAINING L-R, R-L, AND T-B TEXT

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:

---

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 4×8 rectangle to get the following:

---

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
\s9ada is a trademark of the u.s. dept. of defense.
ms-dos is a trademark of microsoft, inc.\s(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
\s9ffortid is a trademark of berry computer scientists,
unix is a trademark of at&t bell laboratories.\s(10
```

OUTPUT AFTER ditroff ONLY

ada is a trade-
mark of the
u.s. dept. of
defense. ms-
dos is a trade-
mark of mi-
crosoft, 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 comput-
er scientists,
ltd. unix is a
trademark of
at&t bell la-
boratories.

OUTPUT AFTER ditroff/bditroff

Assuming all input fits on one output page:

ada is a trade-
mark of the
u.s. dept. of
defense. ms-
dos is a trade-
mark of mi-
crosoft, 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 comput-
er scientists,
ltd. unix is a
trademark of
at&t bell la-
boratories.

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 trade-
mark   of   the
u.s.  dept.  of
defense.    ms-
dos is a trade-
mark   of   mi-
crosoft, inc.
M J G D A
N K H E B
O L I F C
```

____

```
 Y V S P
 Z W T Q
   X U R
```

ffortid is a trademark of berry 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.iv, a Bi-Directional vi by Habusha

Same trick is played for vi.iv, 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

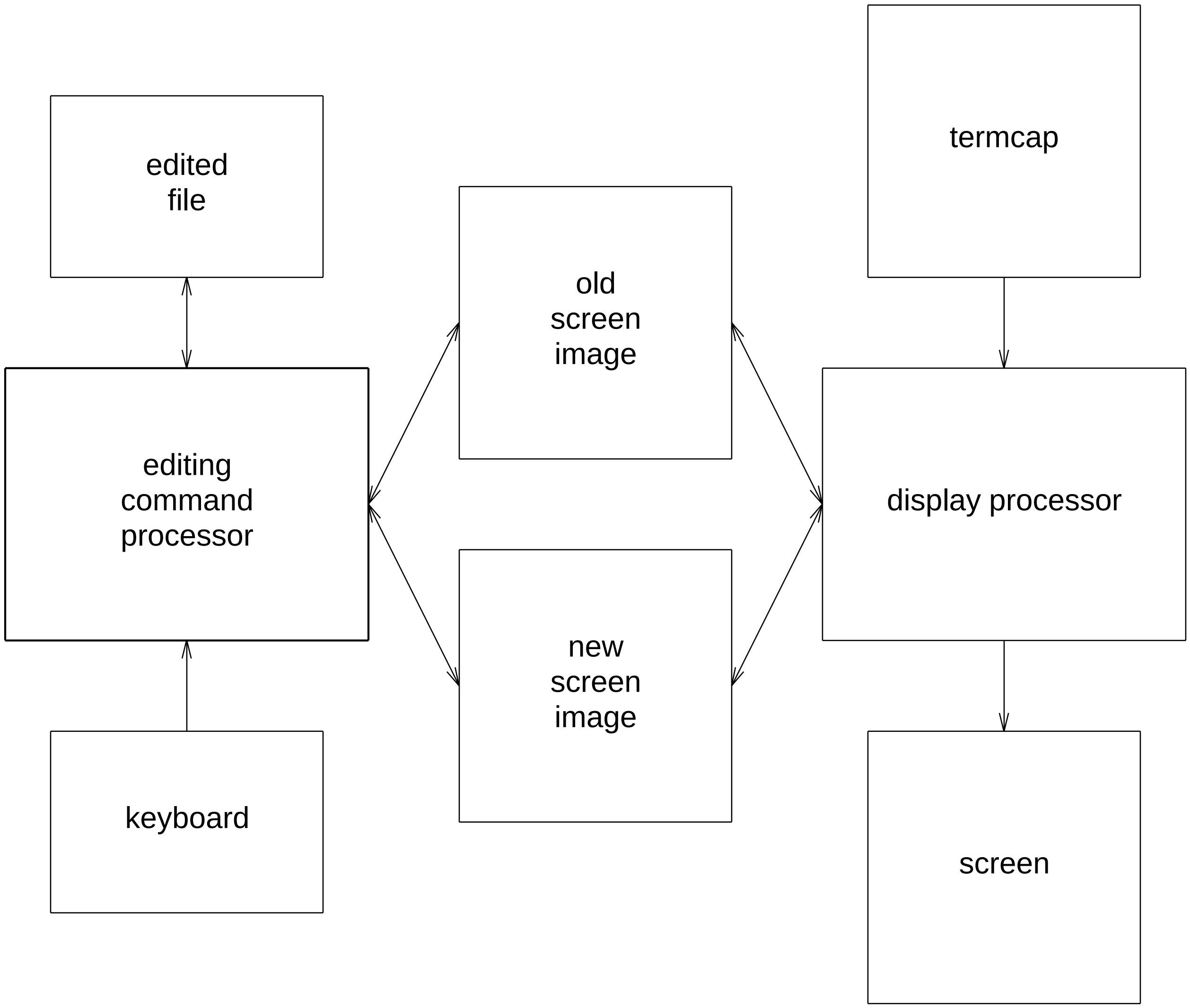
Terminal described by termcap

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

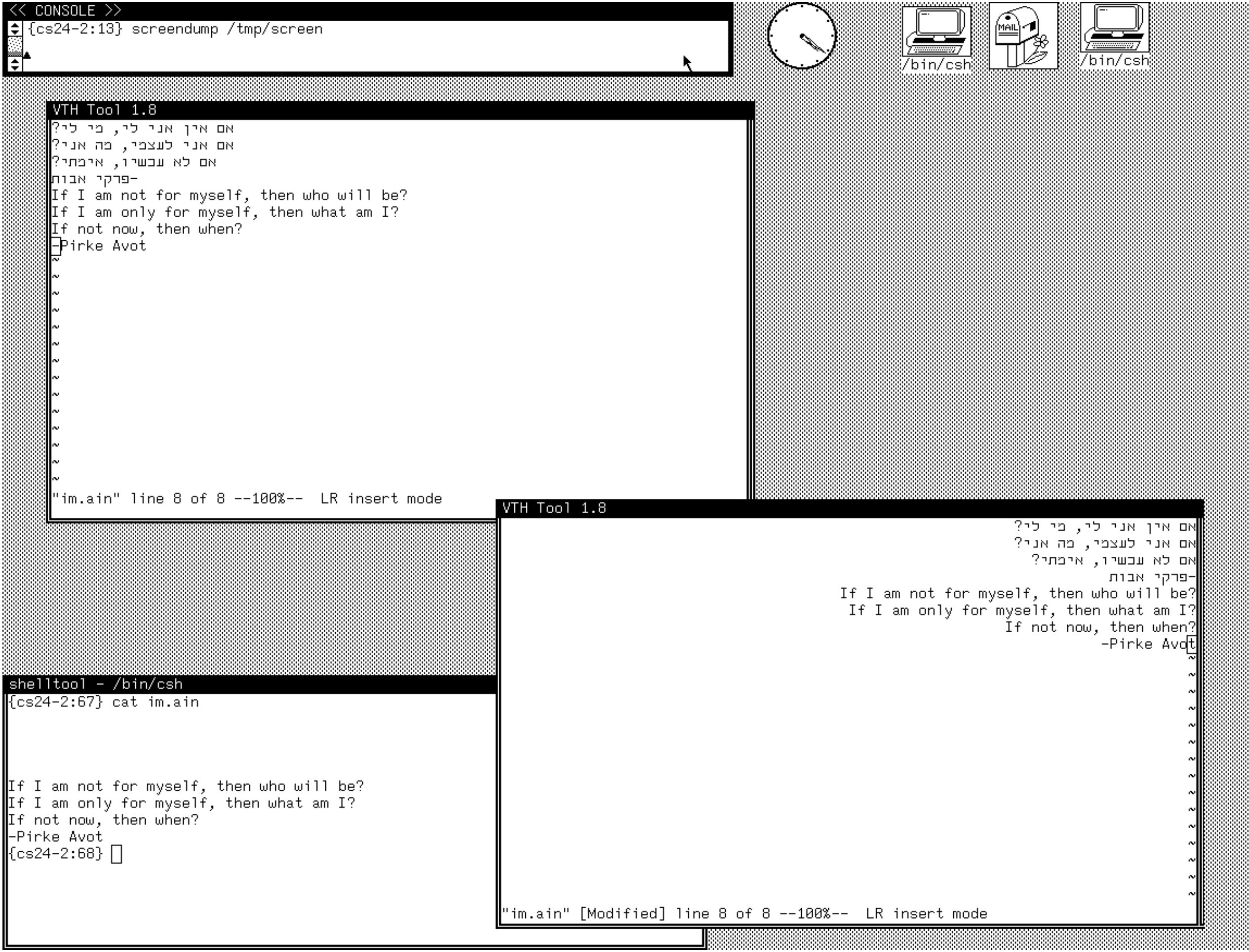
Get screen manager to apply layout algorithm for each line changed as a result of an editing command and to send ONLY these lines to screen

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!!



<< CONSOLE >>
{cs24-2:13} screendump /tmp/screen
/bin/csh
MAIL
/bin/csh
VTH Tool 1.8
אם אין אני לי, מי לי?
אם אני לעצמי, מה אני?
אם לא עכשיו, אימתי?
-פרקי אבות
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 Avot
"im.ain" line 8 of 8 --100%-- LR insert mode
VTH Tool 1.8
אם אין אני לי, מי לי?
אם אני לעצמי, מה אני?
אם לא עכשיו, אימתי?
-פרקי אבות
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 Avot
"im.ain" [Modified] line 8 of 8 --100%-- LR insert mode
shelltool - /bin/csh
{cs24-2:67} cat im.ain
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 Avot
{cs24-2:68}

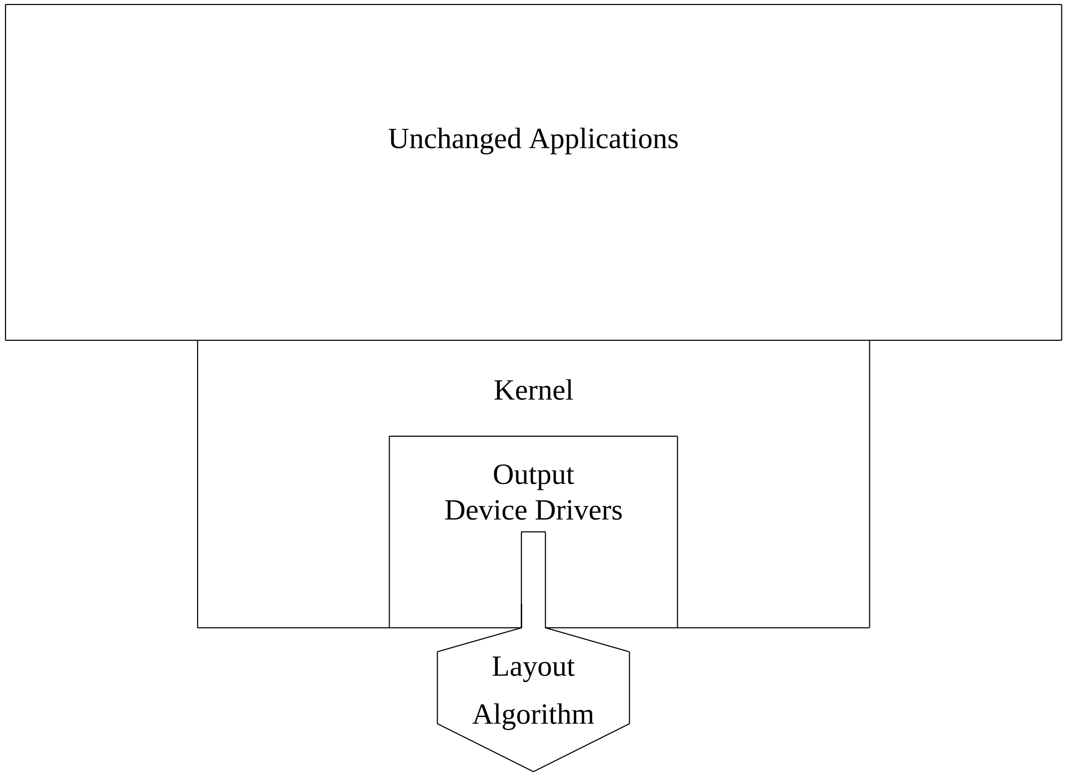
MINIX.XINIM, a Di-Directional MINIX, a mini-UNIX
by Allon

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 throough 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.


Unchanged Applications
Kernel
Output
Device Drivers
Layout
Algorithm

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 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 δ from end of text to end of line

Divide δ 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 δ units before
> last connecting-before letter in the line
>
> divides δ 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

indx finds page numbers
(and even line numbers, if so requested)
of all occurences 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, dtroff 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 dtroff 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 dtroff
still, its line-breaking and justification
leaves much aesthetics to be desired

Since 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/XeT

But it is clearly harder to make TeX/XeT
as one monolithic program
(without damaging pure TeX part)
than it is to make a XeT
(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 Compuer 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 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