Brian W. Kernighan

An Introduction

Daniel M. Berry

Why I am Introducing BWK

I wanted to introduce Brian because I am one of the few people on Earth that is still using his Device-Indpendent Typesetter Run-Off (*ditroff*) program.

I use *ditroff* for almost all of my typesetting.

Even these slides were done by *ditroff*; I *ditroff* to *PostScript* and then *distill* that *PostScript* to *pdf* so that I can use the slide-show features of *acroread* to display them now.

Why do I stick to this ancient program that hasn’t been changed since 1985?

Why do I stick to this ancient program that hasn't been changed since 1985?

Well, why *should* I switch?

Why do I stick to this ancient program that hasn't been changed since 1985?

Well, why *should* I switch?

Has the set of things one does to a printed document changed since 1985?

Why do I stick to this ancient program that hasn't been changed since 1985?

Well, why *should* I switch?

Has the set of things one does to a printed document changed since 1985?

Also, compare the first paragraph of Brian's abstract done with *ditroff*, Microsoft's *word*, and TEX.

ditroff Version

This talk is based on my experience teaching "Computers in Our World," a course for students in the humanities and social sciences. The course describes how computing works–hardware, software, networks, and systems built upon them–for a very non-technical audience. The intent, or perhaps just fond hope, is to help students understand specific technologies better, but also how to reason about how systems work and how to be intelligently skeptical about technology and technological claims.

word Version

This talk is based on my experience teaching "Computers in Our World," a course for students in the humanities and social sciences. The course describes how computing works–hardware, software, networks, and systems built upon them–for a very non-technical audience. The intent, or perhaps just fond hope, is to help students understand specific technologies better, but also how to reason about how systems work and how to be intelligently skeptical about technology and technological claims.

TEX Version

This talk is based on my experience teaching “Computers in Our World,” a course for students in the humanities and social sciences. This course describes how computing works–hardware, software, networks, and systems built upon them–for a very non-technical audience. The intent, or perhaps just fond hope, is to help students understand specific technologies better, but also how to reason about how systems work and how to be intelligently skeptical about technology and technological claims.

In the MS *word* version, notice the hockey player's mouth effect and the two beady eye balls staring at you from the word "specific".

In the MS *word* version, notice the hockey player's mouth effect and the two beady eye balls staring at you from the word "specific".

Notice how much nicer the *ditroff* and TEX versions are.

Admittedly *TEX*, with its multipass, optimizing placement algorithm, does a nicer job of spacing and line breaking than *ditroff*.

Admittedly *TEX*, with its multipass, optimizing placement algorithm, does a nicer job of spacing and line breaking than *ditroff*.

However, *ditroff*'s simple one-pass greedy placement algorithm makes it much easier to control the placement of footnotes and floating figures.

You are guaranteed that if there is enough room for a footnote on the current page, it will be placed there, …

You are guaranteed that if there is enough room for a footnote on the current page, it will be placed there, …

and if you change text that follows a figure, there is no chance that the figure will move up from where it was placed before you changed the text, …

You are guaranteed that if there is enough room for a footnote on the current page, it will be placed there, …

and if you change text that follows a figure, there is no chance that the figure will move up from where it was placed before you changed the text, …

unlike in *TEX*.

You are guaranteed that if there is enough room for a footnote on the current page, it will be placed there, …

and if you change text that follows a figure, there is no chance that the figure will move up from where it was placed before you changed the text, …

unlike in *TEX*.

The latter problem is known to happen also with MS *word*.

Moreover, since *ditroff* has not been modified since 1985, its speed doubles every 18 months, … ☺

Moreover, since *ditroff* has not been modified since 1985, its speed doubles every 18 months, … ☺

and its size is still only 272K! Yes, 272K and not 8.39M and growing!

Moreover, since *ditroff* has not been modified since 1985, its speed doubles every 18 months, … ☺

and its size is still only 272K! Yes, 272K and not 8.39M and growing!

ditroff now formats a 100-page document faster than MS *word* updates the page you are looking at!

More about BWK

I note that in our Distinguished Lecturer Series, we have had Alfred **A**ho, and today we have Brian **K**ernighan. You could say that we have had both the front end and the back end of ***awk***. ☺

Our visitor's login is *bwk*, just one letter up from *awk*! ☺

Local Roots

Brian is a local boy, born in Toronto, teenager in Milton, educated at UT, with relatives in Cambridge, Mississauga, and even at UW!

As Brian says, “waterloo is friends and family.”

A Promise

I promised not to do the standard boring introduction, reading from his biography.

You can read it yourself on the flyer announcing this lecture or at his website.

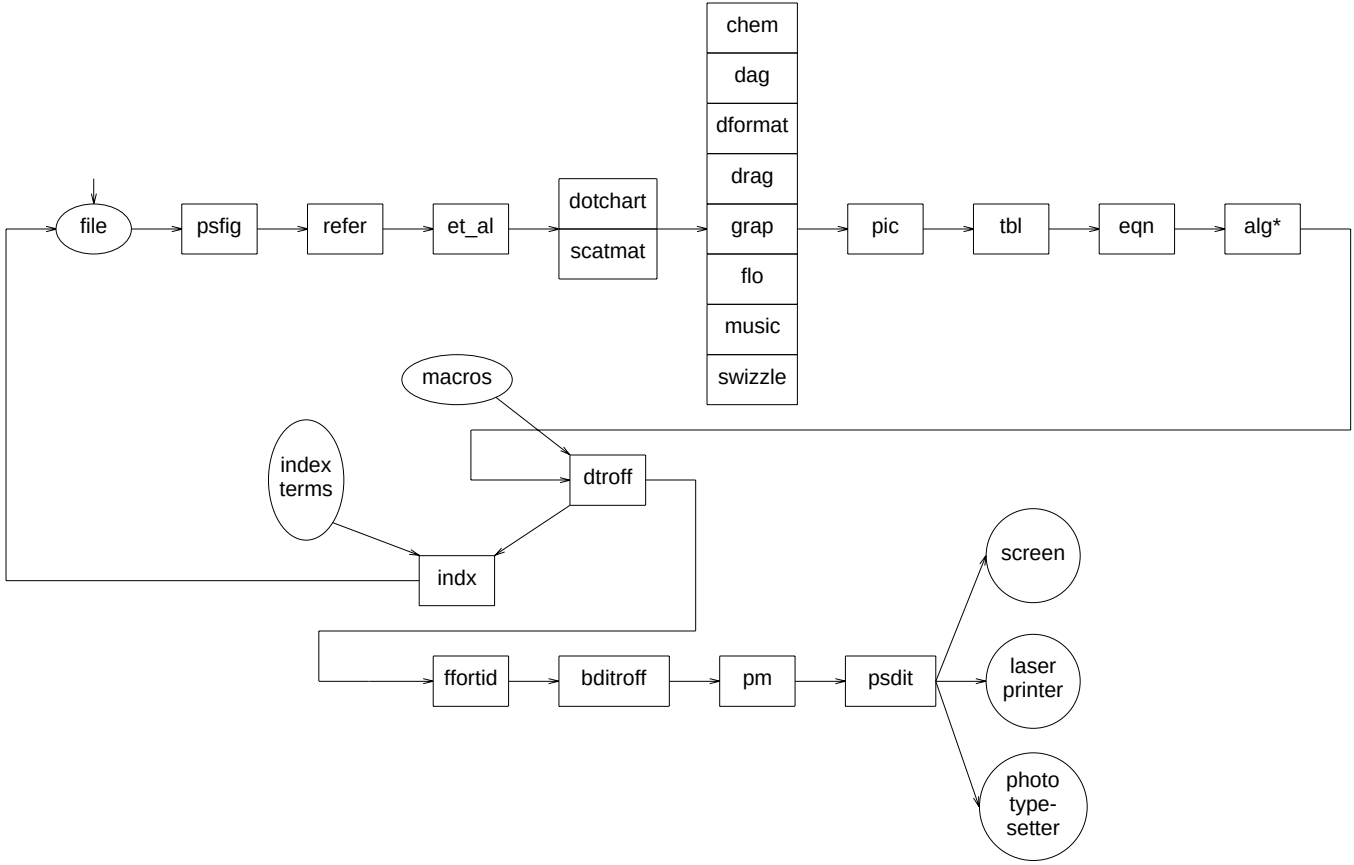
I will talk, as I always do when I am introducing someone, about the lessons I have learned from the speaker's work!

I sort of promised also to be short.. Oh well.. ☺

Lessons Learned from BWK

What did I learn from Brian?

To explain them, let's look at a *pic*-generated diagram of the architecture and dataflow of the entire *ditroff* system. (*pic* is another of Brian's programs that I still use!)


file
psfig
refer
et_al
dotchart
scatmat
chem
dag
dformat
drag
grap
flo
music
swizzle
pic
tbl
eqn
alg*
macros
index terms
dtroff
indx
ffortid
bditroff
pm
psdit
screen
laser printer
photo type-setter

I have learned from Brian the concepts of

I have learned from Brian the concepts of

- piped architectures,

I have learned from Brian the concepts of

- piped architectures,
- editable input and output, and

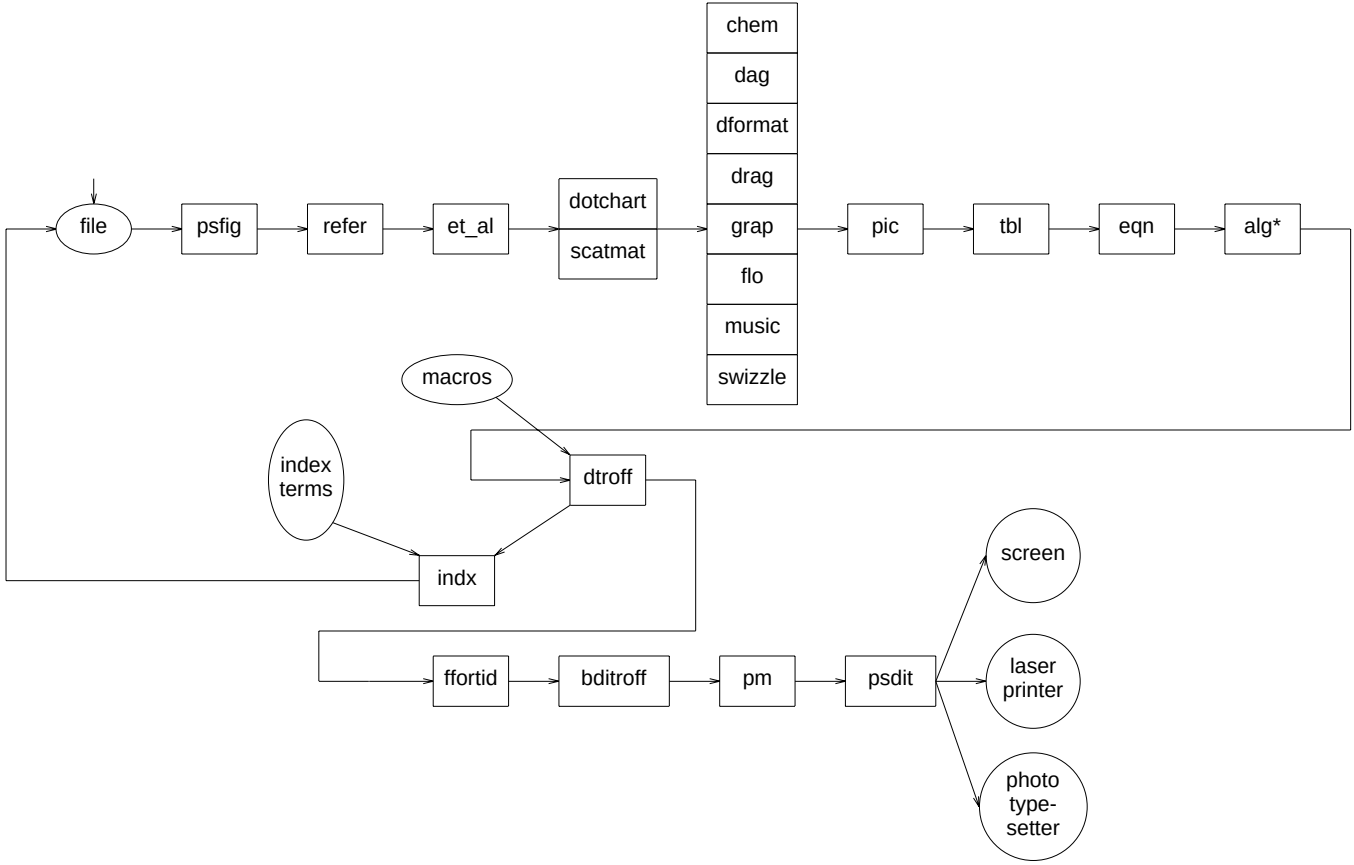
I have learned from Brian the concepts of

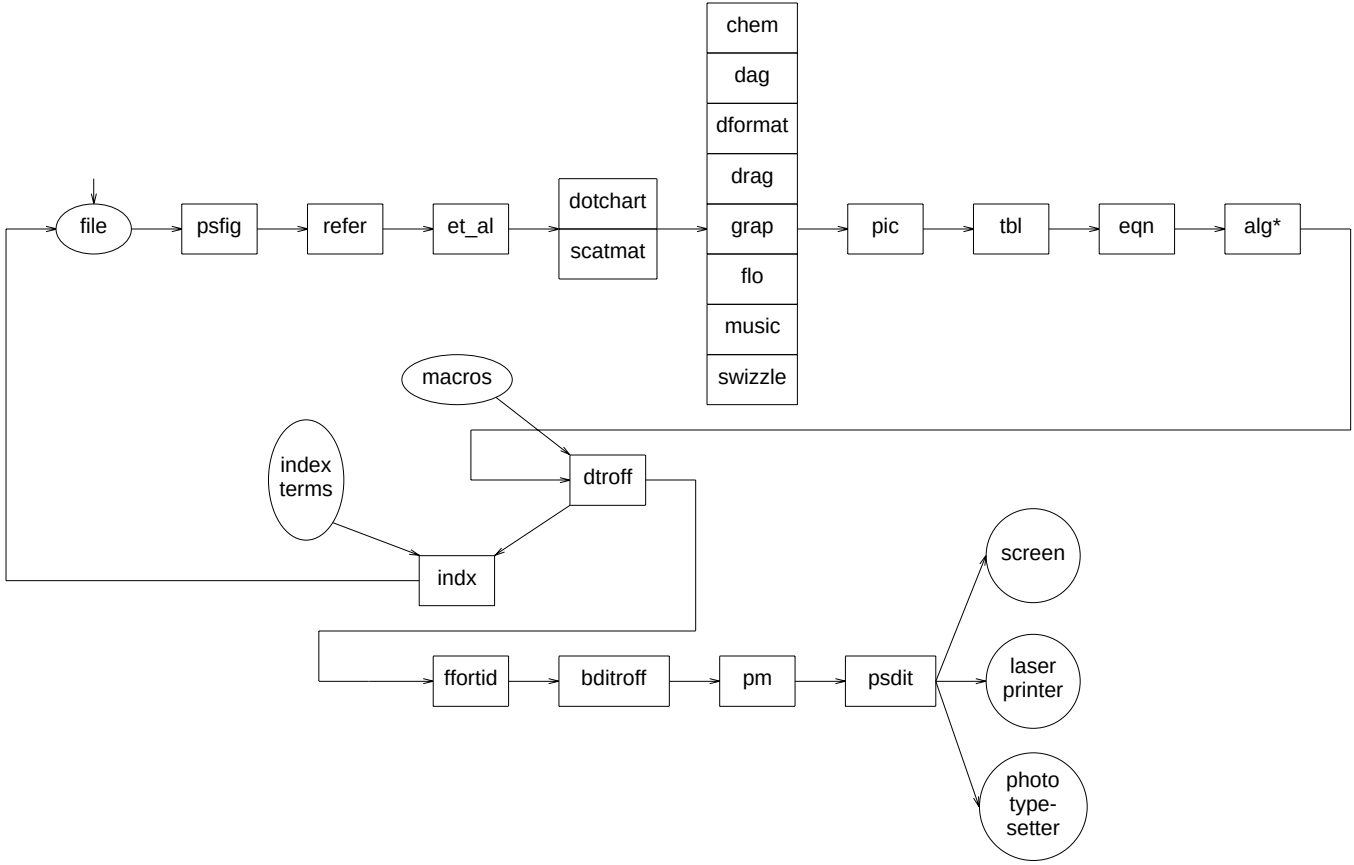
- piped architectures,
- editable input and output, and
- little languages.

The first two permit highly modular systems that can be extended easily by the third, written a variety of languages, including the language of a human-operated editor.

The first two permit highly modular systems that can be extended easily by the third, written a variety of languages, including the language of a human-operated editor.

The first and third really separate the concerns; each kind of document element, e.g., tables or formulae, has its own little language and processor, which can be modified independently of all others, including the main module *ditroff*.


file
psfig
refer
et_al
dotchart
scatmat
chem
dag
dformat
drag
grap
flo
music
swizzle
pic
tbl
eqn
alg*
macros
index terms
dtroff
indx
ffortid
bditroff
pm
psdit
screen
laser printer
photo type-setter


file
psfig
refer
et_al
dotchart
scatmat
chem
dag
dformat
drag
grap
flo
music
swizzle
pic
tbl
eqn
alg*
macros
index terms
dtroff
indx
ffortid
bditroff
pm
psdit
screen
laser printer
photo type-setter

By adding the boxes with red-colored outline, my students and I were able to add the following functions to the *ditroff* system:

By adding the boxes with red-colored outline, my students and I were able to add the following functions to the *ditroff* system:

- Right-to-left formatting for Arabic, Hebrew, Persian, and Urdu, with stretchable letters

By adding the boxes with red-colored outline, my students and I were able to add the following functions to the *ditroff* system:

- Right-to-left formatting for Arabic, Hebrew, Persian, and Urdu, with stretchable letters
- Top-to-bottom formatting for Chinese, Japanese, and Korean

By adding the boxes with red-colored outline, my students and I were able to add the following functions to the *ditroff* system:

- Right-to-left formatting for Arabic, Hebrew, Persian, and Urdu, with stretchable letters
- Top-to-bottom formatting for Chinese, Japanese, and Korean
- Back-of-the-book indexes without cluttering the source of the book with indexing commands

By adding the boxes with red-colored outline, my students and I were able to add the following functions to the *ditroff* system:

- Right-to-left formatting for Arabic, Hebrew, Persian, and Urdu, with stretchable letters
- Top-to-bottom formatting for Chinese, Japanese, and Korean
- Back-of-the-book indexes without cluttering the source of the book with indexing commands
- Flowcharting: Pascal $\rightarrow$ flowchart

By adding the boxes with red-colored outline, my students and I were able to add the following functions to the *ditroff* system:

- Right-to-left formatting for Arabic, Hebrew, Persian, and Urdu, with stretchable letters
- Top-to-bottom formatting for Chinese, Japanese, and Korean
- Back-of-the-book indexes without cluttering the source of the book with indexing commands
- Flowcharting: Pascal $\rightarrow$ flowchart
- Replacing all but first author in a bibliographical reference by "*et al*"

We were able to do all this *without* modifying the functionality of *any* program in the existing collection.

We did correct a few bugs that were exposed by our use of rarely used features; these corrections were passed on to Brian for distribution to *ditroff* licensees.

Each of the additions is effectively a little language processor that sits in the pipe with other programs.

So I can still do

- graphs
- line drawings
- tables
- formulae

In the midst of a tri-directional, multilingual document with flowcharts, and index, and reduced author lists in the bibliography!

The extreme modularity of the *ditroff* system allowed my group to finish building the bidirectional *ditroff* one year before the TEX group finished building the bidirectional TEX, even though we started one year after they did.

Because the input and output of *all* of these programs is simple, editable ASCII, it is very easy to prototype a new little language processor

Because the input and output of *all* of these programs is simple, editable ASCII, it is very easy to prototype a new little language processor

- manually, using a human-operated editor that is invoked by one command in the pipe inside a *makefile*,

Because the input and output of *all* of these programs is simple, editable ASCII, it is very easy to prototype a new little language processor

- manually, using a human-operated editor that is invoked by one command in the pipe inside a *makefile*,

- with a scripting language, such as *sed*, *awk*, or *perl*, or

Because the input and output of *all* of these programs is simple, editable ASCII, it is very easy to prototype a new little language processor

- manually, using a human-operated editor that is invoked by one command in the pipe inside a *makefile*,
- with a scripting language, such as *sed*, *awk*, or *perl*, or
- with *C* or *C++*.

Also, I have been known to cheat!

If a processor lacks a feature I need, I arrange for a placeholder to be output by the processor, and I edit the output manually to produce the output that would be there if the feature were available.

I have done this to the output of *refer*, *pic*, and *eqn*.

Is This All Useful?

Is This All Useful?

Well, I have a publication from each of the first four added functionalities, and I typeset the paper about each processor in the journal's own format, using the software we wrote and the *ditroff* collection!

Is This All Useful?

Well, I have a publication from each of the first four added functionalities, and I typeset the paper about each processor in the journal's own format, using the software we wrote and the *ditroff* collection!

I published, in a rabbinical journal, a commentary about the first sentence of *Genesis*; this paper quotes the original Hebrew.

Is This All Useful?

Well, I have a publication from each of the first four added functionalities, and I typeset the paper about each processor in the journal's own format, using the software we wrote and the *ditroff* collection!

I published, in a rabbinical journal, a commentary about the first sentence of *Genesis*; this paper quotes the original Hebrew.

Also I made my own visiting card using this software!

다니엘 베리
ダニエル・ベリ

Fax: +1-519-746-5422
E-mail: dberry@uwaterloo.ca
HTTP://se.uwaterloo.ca/~dberry/

丹尼儿北利

Daniel M. Berry, Ph.D.
Professor

Даниэль М. Бэри
Δανιήλ Μ. Μπέρι
ढाणीयळ बेरी
ዳንኤል ቤሪ

دانيال بيري
דניאל ברי

다니엘 베리
ダニエル・ベリ

Fax: +1-519-746-5422
E-mail: dberry@uwaterloo.ca
HTTP://se.uwaterloo.ca/~dberry/

丹尼儿北利

Daniel M. Berry, Ph.D.
Professor

Даниэль М. Бэри
Δανιήλ Μ. Μπέρι
ढाणीयळ बेरी
ዳንኤል ቤሪ

دانيال بيري
דניאל ברי

Brian's Code

In the process of doing all what I have described, I happened to read a lot of Brian's code.

His code is the most readable I have ever seen, and this is in spite of his use of short identifiers.

Real works of art!

Conclusion

This is what I have learned from Brian!

Thank you, Brian!

Now, we shall find out if this is what I should have learned from him!

ditroff Version

This talk is based on my experience teaching "Computers in Our World," a course for students in the humanities and social sciences. The course describes how computing works–hardware, software, networks, and systems built upon them–for a very non-technical audience. The intent, or perhaps just fond hope, is to help students understand specific technologies better, but also how to reason about how systems work and how to be intelligently skeptical about technology and technological claims.

word Version

This talk is based on my experience teaching "Computers in Our World," a course for students in the humanities and social sciences. The course describes how computing works–hardware, software, networks, and systems built upon them–for a very non-technical audience. The intent, or perhaps just fond hope, is to help students understand specific technologies better, but also how to reason about how systems work and how to be intelligently skeptical about technology and technological claims.

TEX Version

This talk is based on my experience teaching “Computers in Our World,” a course for students in the humanities and social sciences. This course describes how computing works–hardware, software, networks, and systems built upon them–for a very non-technical audience. The intent, or perhaps just fond hope, is to help students understand specific technologies better, but also how to reason about how systems work and how to be intelligently skeptical about technology and technological claims.