Eq – A Lout Package for Typesetting Mathematics

Jeffrey H. Kingston

Basser Department of Computer Science
University of Sydney 2006
Australia

ABSTRACT

This report describes the use of Eq, a package of definitions for typesetting mathematics with the Lout document formatter. For example,

```
@Eq { big int supp 1 on 0 ‘ dx over sqrt {1 - x sup 2} = pi over 2 }
```

produces the output

$$\int_0^1 \frac{dx}{\sqrt{1 - x^2}} = \frac{\pi}{2}$$

The advantages of Eq include careful attention to details of spacing, a repertoire of several hundred mathematical symbols, a simple syntax extensible by the user, and complete integration with the rest of Lout.

In addition, this report contains an appendix describing the use of Pas, a package of definitions for printing Pascal programs.

22 December, 1992

Eq – A Lout Package for Typesetting Mathematics

Jeffrey H. Kingston

Basser Department of Computer Science
University of Sydney 2006
Australia

1. Introduction

Eq is a package of definitions for typesetting mathematics with the Lout document formatter [4]. It includes several hundred mathematical and other special characters (including the entire PostScript[1] Symbol font [1]), and a smaller number of symbols for joining objects together in mathematical ways. Eq is based on the eqn language of Kernighan and Cherry [3], with spacing rules similar to Knuth's TEX formatter [6].

To use Eq in a Lout document, first ensure that its definition is included, either by having @SysInclude { eq } in the setup file, or -ieq on the command line. Then, at any point in the document, write @Eq { ... } and the symbols of Eq will be available between the braces. Any symbols available outside continue to be available inside, which means that Eq can be freely mixed with standard Lout and with symbols from other packages, without restriction.

In this report we show the Lout input at the left, and its result at the right:

@Eq { {x sup 2 + y sup 2} over 2 }	$\frac{x^2 + y^2}{2}$

Subsequent examples will omit the enclosing @Eq { ... }.

2. Symbols

Eq prints characters in the fonts appropriate for mathematics:

x - 2	$x - 2$

Here x is in Italic, 2 is in Roman, and – is from the Symbol font. The character - is a *symbol* which stands for –, and 2 is also a symbol, standing for 2. Eq includes a vast number of symbols:

Omega delta int partial club	$\Omega\delta\int\partial\clubsuit$

The summary at the end of this report has the complete list.

Symbols whose names are made from letters should be separated from each other by at least one space or end of line, as was done above, or else Eq will become confused:

Omegadelta	*Omegadelta*

[1]PostScript is a trademark of Adobe Systems, Incorporated.

Symbols whose names are made from digits and punctuation characters can, however, be run together with each other and with symbols made from letters:

```
Omega-delta<=2
```

$$\Omega - \delta \leq 2$$

This rule applies throughout the Lout world.

Some symbols join objects together in mathematical ways:

```
x sub 2
```

$$x_2$$

Here the sub symbol has taken the object just to its left, and the object just to its right, and joined them into one object in the form of a subscript. The two objects are called the left and right parameters of sub, and they may be arbitrary Lout objects.

Other symbols of a similar kind include sup for superscripting, over for built-up fractions, and from and to for the lower and upper limits of sums, products, etc. These symbols may be used together to produce complicated equations with astonishing ease:

```
big sum from i=0 to n r sup i
= {r sup n+1 - 1} over r-1
```

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}$$

Here sum is just the $\sum$ symbol; from and to do all the work of placing the limits. They are quite independent, so either or both may be omitted. To get a superscript directly over a subscript, use the supp and on symbols:

```
A supp 2 on 1
```

$$A_1^2$$

These two symbols should always be used together as shown.

As usual in Lout, braces are used to group something into an indivisible object. Leaving them out creates ambiguities:

```
a  sup  b  over  c
```

There are two possible interpretations for this:

```
{a  sup  b}  over  c
```

$$\frac{a^b}{c}$$

```
a  sup  {b  over  c}
```

$$a^{\frac{b}{c}}$$

Eq chooses between them in the following way. Every symbol that takes a parameter also has a *precedence*, which is a number. For example, sup has precedence 60 and over has precedence 54. The symbol with the highest precedence wins the object lying between them, so in the above case the first interpretation is chosen. If two symbols of equal precedence compete for an object, the association is towards the left:

```
a sup b sub 2
```

$${a^b}_2$$

In this case it is more probable that the following right association was actually wanted:

```
a sup { b sub 2 }
```

$$a^{b_2}$$

White space between two objects is considered to be a symbol with precedence 7, which is lower than the precedence of any Eq symbol; but if the two objects are immediately adjacent the precedence is 102, which is higher than the precedence of any Eq symbol. Compare these three examples:

```
big sum from i=0 to n
```

$$\sum_{i=0}^{n}$$

```
big sum from {i = 0} to n
```

$$\sum_{i=0}^{n}$$

```
big sum from i = 0 to n
```

$$\sum_{i} = \overset{n}{0}$$

and you will see that some care is needed on this point. Braces can always be used to override precedence and associativity, and when in doubt the easiest course is to insert them. Although Lout allows symbols to associate towards the left or right, Eq chooses to have only left associative symbols. The summary at the end of this report gives the precedence of every symbol.

The *matrix* symbol builds an array of objects:

```
matrix
  atleft { blpar }
  atright { brpar }
{ x sup 2 above x above 1
  nextcol
  y sup 2 above y above 1
  nextcol
  z sup 2 above z above 1
}
```

$$\begin{pmatrix} x^2 & y^2 & z^2 \\ x & y & z \\ 1 & 1 & 1 \end{pmatrix}$$

The atleft and atright options place vertically scaled versions of their values at each side; if either is omitted the value is taken to be an empty object of zero width by default. The right parameter of matrix is the array itself. It is a sequence of columns separated by nextcol symbols; each column is a sequence of objects separated by above symbols.

The nextcol and above symbols have low precedence, but not as low as white space between two objects. Therefore, unless the entries in the array are very simple, it is safest to enclose each of them in braces.

Columns built with the above symbol have their objects centred in the column. Also available are labove for left-justified columns, cabove meaning the same as above, rabove for right-justified columns, and mabove for alignment along column marks. Each column should contain only one kind of above symbol (although adventurous users might be able to get some mixtures to work), but different columns may differ. For example,

```
@R "Chain rule:"  labove  @R "Product rule:"
nextcol
{df over dx ^= df over dy cdot dy over dx}
mabove
{dfg over dy ^= f ‘ dg over dx + g df over dx}
```

has result

$$\begin{array}{ll} \text{Chain rule:} & \dfrac{df}{dx} = \dfrac{df}{dy} \cdot \dfrac{dy}{dx} \\ \text{Product rule:} & \dfrac{dfg}{dy} = f\dfrac{dg}{dx} + g\dfrac{df}{dx} \end{array}$$

As this last example shows, it is nextcol and the various above symbols that lay out the array; matrix attaches the atleft and atright options and makes sure the result appears in the correct vertical position relative to the rest of the equation. So the right parameter of matrix may be any object.

Each of the Eq symbols that takes parameters also has a gap option, which controls the amount of space inserted by the symbol:

x over y	$\frac{x}{y}$
x over gap { 3p } y	$\frac{x}{y}$

Eq usually gets the spacing right without help.

3. Spacing

There is a basic rule governing the use of white space characters (space, tab, and newline) in the input to Lout: white space between two objects affects the result; white space between a symbol and its parameter does not.

Although this rule is just right most of the time, it is not adequate for equation formatting. Getting the horizontal spacing right in equations is a very fiddly business, involving four different sizes of space (zero, thin, medium, and thick), and different rules for spacing within superscripts and subscripts to those applying outside, according to a leading authority [6]. Eq therefore takes the spacing decisions upon itself, and consequently chooses to ignore all white space in its input, even between two objects.[1]

Every symbol provided by Eq has a *full name*, which denotes the symbol without any space attached. Many symbols also have a *short name*, which denotes the same symbol with what Eq considers to be an appropriate amount of space for that symbol attached to it. For example, ≤

---

[1]This effect is produced by enclosing the entire equation in 0c @Space. The simplest way to restore the effect of white space to part of an equation is to enclose that part in a @Font symbol. Eq also changes the value of the v unit, so if a paragraph of filled text is desired within an equation, it may be necessary to enclose it in a @Break symbol.

has full name lessequal and short name <=:

a lessequal b — $a{\leq}b$

a <= b — $a \leq b$

Eq puts a thick space around relation symbols like <=, a medium space around binary operator symbols like +, and a thin space after punctuation symbols (; and ,); except that in places where the symbols appear in a smaller size (superscripts, subscripts, etc.), these spaces are omitted. No other horizontal space is ever inserted.

The short names have been carefully designed to produce good-looking mathematics most of the time. It is best to rely on them in the first instance and only think about spacing when the result is not pleasing. In that case, Eq's space can be removed by using the full names, and thin, medium and thick space can be added using the following symbols:

`	0.18f	(0.018f in subscripts, etc.)
``	0.24f	(0.024f in subscripts, etc.)
``	0.30f	(0.030f in subscripts, etc.)

where 1f is the current font size. These symbols have low precedence. The & symbol from standard Lout is also available; the s unit has value 0 and so is not very useful, but one can write &2m for example for a two em space. The full names are tedious to remember, so Eq provides a non symbol which removes spaces from its right parameter; thus non <= is equivalent to lessequal. There are also rel, bin, and punct symbols for telling Eq to add space to the following symbol as though it was a relation symbol, binary operator, or punctuation symbol.

4. Features from Standard Lout

In this section we summarize those features of standard Lout of most relevance to equation formatting. All are freely available within equations. Full details may be found in the Lout reference manual [4].

Standard Lout uses the symbols #, {, }, &, |, and / for special purposes (the braces are used for grouping, for example). To get these characters into equations without using their full names, enclose them in double quotes: "{", "}", etc. Any sequence of characters including spaces but not newlines may be so enclosed, and the effect is to turn off any special meaning that the symbols within it might have.

Eq sets letters in Slope (Lout's name for Italic), digits in Base (i.e. Roman), and other symbols in various fonts, mostly the Symbol font. To change fonts, use the @Font symbol:

Slope @Font "2" — *2*

In Eq it will often be necessary to enclose the right parameter in double quotes, because the symbol 2 includes a built-in change back to Base font. Changing fonts makes white space between objects in the right parameter appear in the result. The @Font operator also does size changes:

sum — Σ

```
"+2p" @Font sum
```
$\textstyle\sum$

```
2.0f @Font sum
```
$\displaystyle\sum$

Here "+2p" @Font sets its right parameter in a font two points larger than it would otherwise have been; 2.0f @Font sets its right parameter in a font twice the original size. Sizes should always be specified relative to the enclosing size as we have done here, since then they don't need to be changed if a decision is made to set the entire document in a different size. It is necessary to enclose +2p in double quotes within Eq, because otherwise the + will be taken as the Eq symbol for $+$. The right parameter may be any object.

Whenever similar equations or parts of equations are being typed repeatedly, *definitions* should be used to save time. Suppose for example that $p_i \log_2 p_i$ occurs frequently. Then

```
def epi { p sub i ‘ log sub 2 ‘ p sub i }
```

makes the symbol epi stand for the object between the braces:

```
big sum from i=1 to n ‘ epi
```
$$\sum_{i=1}^{n} p_i \log_2 p_i$$

Symbols may be given parameters:

```
def ep
  right x
{ p sub x ‘ log sub 2 ‘ p sub x
}
```

The parameter x will be replaced by the object just to the right of ep:

```
big sum from j=1 to k ‘ ep i +
big sum from j=k+1 to n ep j
```
$$\sum_{j=1}^{k} p_i \log_2 p_i + \sum_{j=k+1}^{n} p_j \log_2 p_j$$

The precedence of such symbols will be 100 by default.

To make the symbols of Eq available within such definitions, each must be preceded by import @Eq. The best place to keep them is in the setup file, which might then look like this:

```
@SysInclude { ft }
@SysInclude { dl }
@SysInclude { eq }

import @Eq
def epi { p sub i ‘ log sub 2 ‘ p sub i }

import @Eq
def ep right x { p sub x ‘ log sub 2 ‘ p sub x }

@Use { @DocumentLayout }
```

Use of epi and ep outside equations will cause an error.

Equations can appear within a paragraph of text, or they can be displayed. Eq's job is to produce a Lout object containing the equation; it neither knows nor cares where this equation goes. To get an equation within a paragraph, simply place @Eq { ... } at the desired point. To prevent it spreading over two lines, use @OneCol @Eq { ... }. To display an equation, use a display symbol from some other Lout package. For example, the DocumentLayout package [5] has @IndentedDisplay or @ID for an indented display, and @CentredDisplay or @CD for a centred display, so

```
@CD @Eq { int supp pi on 0 sin ‘ x = 0 }
```

produces

$$\int_0^{\pi} \sin x = 0$$

DocumentLayout also provides display symbols that make it easy to produce aligned and numbered equations.

5. Summary

This section is a complete list of the symbols provided by Eq. We divide them into auxiliary, parameterized, short names (further divided into relations, binary operators, and punctuation), and full names. The auxiliary symbols are:

‘	Thin space
‘‘	Medium space
‘‘‘	Thick space
bin x	Treat x as a binary operator
rel x	Treat x as a relation
punct x	Treat x as a punctuation symbol
non x	Remove spaces normally put into x
vctr x	Centre x vertically
big x	Make x larger

Here are all the parameterized symbols, shown in groups of equal precedence, with the precedence itself at right:

matrix not (100)
dot dotdot hat tilde vec dyad overbar underbar (62)
sup sub supp (60) on (61)
from to widefrom wideto (58)
sqrt root (56)
over frac (54)
above labove cabove rabove mabove (52)
nextcol (50)

See page 3 for examples of matrices. Here are some examples of the other symbols:

x dot	$\dot{x}$
x dotdot	$\ddot{x}$
x hat	$\hat{x}$
x tilde	$\tilde{x}$
x vec	$\vec{x}$
x dyad	$\overleftrightarrow{x}$
x+y overbar	$\overline{x + y}$
x+y underbar	$\underline{x + y}$

These marks are centred over the left parameter, except the last two which are extended to the width of the object.

a sup b	a^b
a sub b	a_b
a supp b on c	a_c^b

Note that supp and on must be used together as shown.

big sum from i	$\sum\limits_i$
big prod to j	$\prod\limits^j$
{a, ... , z} widefrom {90d @Rotate blbrace}	$\underbrace{a, \ldots, z}$
{a, ... , z} wideto minus	$\overline{a, \ldots, z}$

widefrom and wideto are like from and to except that they horizontally scale the right parameter to the width of the left.

sqrt {x over y}	$\sqrt{\frac{x}{y}}$
3 root {x over y}	$\sqrt[3]{\frac{x}{y}}$

The left parameter of root may be any object.

2 over 3	$\frac{2}{3}$

2 frac 3 $^{2}\!/_{3}$

The following short names define relations (that is, they have a thick space on each side):

<	<	>	>	=	=
<=	≤	prec	≺	preceq	⪯
<<	<<	subset	⊂	subseteq	⊆
sqsubseteq	⊑	in	∈	vdash	⊢
smile	⌣	frown	⌢	>=	≥
succ	≻	succeq	⪰	>>	>>
supset	⊃	supseteq	⊇	sqsupseteq	⊒
ni	∋	dashv	⊣	mid	∣
parallel	∥	==	≡	~	∼
-~	≃	asymp	≍	~~	≈
=~	≅	bowtie	⋈	propto	∝
models	⊨	doteq	≐	perp	⊥
notsub	⊄	notin	∉	!=	≠
<->	↔	<--	←	-->	→
up	↑	down	↓	<=>	⇔
<==	⇐	==>	⇒	dblup	⇑
dbldown	⇓	:	:	::	: :
:=	:=				

These can be negated by preceding them with not, as in not ==, for example, which yields ≢ .
The following short names define binary operators (medium space on each side):

+	+	-	−	+-	±
-+	∓	setminus	∖	cdot	·
times	×	*	∗	circ	○
div	÷	cap	∩	cup	∪
uplus	⊎	sqcap	⊓	sqcup	⊔
triangleleft	◁	triangleright	▷	wr	≀
bigcirc	◯	bigtriangleup	△	bigtriangledown	▽
vee	∨	wedge	∧	oplus	⊕
ominus	⊖	otimes	⊗	oslash	⊘
odot	⊙	dagger	†	daggerdbl	‡
amalg	∐				

The following names define punctuation symbols (thin space on the right-hand side):

;	;	,	,	col	:

The following symbols are used in ways typified by the large sum and product symbols. In display equations they should be preceded by the big symbol:

sum	Σ	prod	Π	coprod	∐
int	∫	oint	∮	bcap	⋂
bcup	⋃	bvee	⋁	bwedge	⋀

bodot	$\odot$	botimes	$\otimes$	boplus	$\oplus$
buplus	$\uplus$				

The following symbols are defined so that they will appear in Roman, as is conventional for them in equations:

arccos	arccos	arcsin	$\arcsin$	arctan	$\arctan$
arg	$\arg$	cos	$\cos$	cosh	$\cosh$
cot	$\cot$	coth	$\coth$	csc	$\csc$
deg	$\deg$	det	$\det$	dim	$\dim$
exp	$\exp$	gcd	$\gcd$	hom	$\hom$
inf	$\inf$	ker	$\ker$	lg	$\lg$
lim	$\lim$	liminf	$\liminf$	limsup	$\limsup$
ln	$\ln$	log	$\log$	max	$\max$
min	$\min$	Pr	$\Pr$	sec	$\sec$
sin	$\sin$	sinh	$\sinh$	supr	$\sup$
tan	$\tan$	tanh	$\tanh$	mod	mod

The following symbols are also defined to ensure that they will appear in Roman:

0	0	1	1	2	2
3	3	4	4	5	5
6	6	7	7	8	8
9	9	!	!	?	?
%	%	(	(	)	)
[	[	]	]		

The following symbols make good atleft and atright parameters of the matrix symbol:

lpar	$($	blpar	$\Bigg($	rpar	$)$
brpar	$\Bigg)$	lbrack	$[$	blbrack	$\Bigg[$
rbrack	$]$	brbrack	$\Bigg]$	lbrace	$\{$
blbrace	$\Bigg\{$	rbrace	$\}$	brbrace	$\Bigg\}$
lfloor	$\lfloor$	blfloor	$\Bigg\lfloor$	rfloor	$\rfloor$
brfloor	$\Bigg\rfloor$	lceil	$\lceil$	blceil	$\Bigg\lceil$
rceil	$\rceil$	brceil	$\Bigg\rceil$	langle	$\langle$

blangle	⟨	rangle	⟩	brangle	⟩

Here are some miscellaneous symbols:

hbar	ℏ	Re	ℜ	Im	ℑ
partial	∂	infty	∞	prime	′
nabla	∇	surd	√	top	⊤
bot	⊥	dbar	‖	triangle	△
backslash	\	forall	∀	exists	∃
neg	¬	circle	○	square	□
ldots	. . .	cdots	· · ·	vdots	⋮
ddots	⋱	del	∇	grad	∇
...	…	,...,	, …,	half	½
third	⅓	'	′	empty	∅

Finally, here is the long list of full names from the Adobe Symbol font:

space		exclam	!	universal	∀
numbersign	#	existential	∃	percent	%
ampersand	&	suchthat	∍	parenleft	(
parenright	)	asteriskmath	∗	plus	+
comma	,	minus	−	period	.
slash	/	zero	0	one	1
two	2	three	3	four	4
five	5	six	6	seven	7
eight	8	nine	9	colon	:
semicolon	;	less	<	equal	=
greater	>	question	?	congruent	≅
Alpha	Α	Beta	Β	Chi	Χ
Delta	Δ	Epsilon	Ε	Phi	Φ
Gamma	Γ	Eta	Η	Iota	Ι
thetaone	ϑ	Kappa	Κ	Lambda	Λ
Mu	Μ	Nu	Ν	Omicron	Ο
Pi	Π	Theta	Θ	Rho	Ρ
Sigma	Σ	Tau	Τ	Upsilon	ϒ
sigmaone	ς	Omega	Ω	Xi	Ξ
Psi	Ψ	Zeta	Ζ	bracketleft	[
therefore	∴	bracketright	]_	perpendicular	⊥
underscore	_	radicalex		alpha	α
beta	β	chi	χ	delta	δ
epsilon	ε	phi	ϕ	gamma	γ
eta	η	iota	ι	phione	φ
kappa	κ	lambda	λ	mu	μ
nu	ν	omicron	ο	pi	π
theta	θ	rho	ρ	sigma	σ

tau	τ	upsilon	υ	omegaone	ϖ
omega	ω	xi	ξ	psi	ψ
zeta	ζ	braceleft	{	bar	|
braceright	}	similar	∼	Upsilonone	ϒ
minute	′	lessequal	≤	fraction	⁄
infinity	∞	florin	ƒ	club	♣
diamond	♦	heart	♥	spade	♠
arrowboth	↔	arrowleft	←	arrowup	↑
arrowright	→	arrowdown	↓	degree	°
plusminus	±	second	″	greaterequal	≥
multiply	×	proportional	∝	partialdiff	∂
bullet	•	divide	÷	notequal	≠
equivalence	≡	approxequal	≈	ellipsis	…
arrowvertex	⏐	arrowhorizex	⎯	carriagereturn	↵
aleph	ℵ	Ifraktur	ℑ	Rfraktur	ℜ
weierstrass	℘	circlemultiply	⊗	circleplus	⊕
emptyset	∅	intersection	∩	union	∪
propersuperset	⊃	reflexsuperset	⊇	notsubset	⊄
propersubset	⊂	reflexsubset	⊆	element	∈
notelement	∉	angle	∠	gradient	∇
registerserif	®	copyrightserif	©	trademarkserif	™
product	∏	radical	√	dotmath	⋅
logicalnot	¬	logicaland	∧	logicalor	∨
arrowdblboth	⇔	arrowdblleft	⇐	arrowdblup	⇑
arrowdblright	⇒	arrowdbldown	⇓	lozenge	◊
angleleft	〈	registersans	®	copyrightsans	©
trademarksans	™	summation	∑	parenlefttp	⎛
parenleftex	⎜	parenleftbt	⎝	bracketlefttp	⎡
bracketleftex	⎢	bracketleftbt	⎣	bracelefttp	⎧
braceleftmid	⎨	braceleftbt	⎩	braceex	⎪
angleright	〉	integral	∫	integraltp	⌠
integralex	⎮	integralbt	⌡	parenrighttp	⎞
parenrightex	⎟	parenrightbt	⎠	bracketrighttp	⎤
bracketrightex	⎥	bracketrightbt	⎦	bracerighttp	⎫
bracerightmid	⎬	bracerightbt	⎭		

The names given are the same as Adobe's except in a few places where the Adobe name contains a digit, which is not possible in Lout.

Appendix A. Pas – a Lout Package for Printing Pascal Programs

Pas is a package of definitions for printing Pascal programs [2] neatly with the Lout document formatter [4]. No attempt is made to follow any particular printing standard; the design simply reflects the author's taste.

The package is so simple that there is very little to say about it. To use Pas, place @SysInclude { pas } in the setup file, or type -ipas in the command line. A Pascal program is entered like this, where the @ID symbol from the DocumentLayout package [5] has been used to obtain an indented display:

```
@ID @Pas {
procedure PriDelete(x: PriEntry; var Q: PriorityQueue);
   var i: integer;
begin
   with Q^ do begin
      size := size - 1;
      if x^.back <= size then
      begin
         i := x^.back;
         A[i] := A[size + 1];
         A[i]^.back := i;
         PriAddRoot(i, Q);
         PriAddLeaf(i, Q)
      end
   end
end;
}
```

The result will come out like this:

procedure *PriDelete*(*x*: *PriEntry*; **var** *Q*: *PriorityQueue*);
   var *i*: *integer*;
begin
   with *Q*↑ **do begin**
      size := *size* – 1;
      if *x*↑.*back* ≤ *size* **then**
      begin
         i := *x*↑.*back*;
         A[*i*] := *A*[*size* + 1];
         A[*i*]↑.*back* := *i*;
         PriAddRoot(*i*, *Q*);
         PriAddLeaf(*i*, *Q*)
      end
   end
end;

Blank lines, line breaks, indents and spaces in the input are respected, with a tab being considered equal to eight spaces. @Pas can also be used within a paragraph to produce Pascal fragments like $A[i..j]$. Use @OneCol @Pas { ... } to prevent the result from breaking over two lines.

@Pas does not attempt to rearrange the program in any way. Each item is simply printed according to the following plan:

and	**and**	0	0
array	**array**	1	1
begin	**begin**	2	2
case	**case**	3	3
const	**const**	4	4
div	**div**	5	5
do	**do**	6	6
downto	**downto**	7	7
else	**else**	8	8
end	**end**	9	9
file	**file**	.	.
for	**for**	,	,
forward	**forward**	:	:
function	**function**	;	;
goto	**goto**	’	’
if	**if**	‘	‘
in	**in**	+	+
label	**label**	-	−
mod	**mod**	*	∗
nil	**nil**	/	∕
not	**not**	(	(
of	**of**	)	)
or	**or**	[	[
otherwise	**otherwise**	]	]
packed	**packed**	^	↑
procedure	**procedure**	..	..
program	**program**	=	=
record	**record**	<	<
repeat	**repeat**	>	>
set	**set**	<>	≠
then	**then**	<=	≤
to	**to**	>=	≥
type	**type**	:=	:=
until	**until**		
var	**var**		
while	**while**		
with	**with**		

Anything not mentioned here will appear in italic font.

Pascal character strings need a little attention before formatting by Pas. Their interiors are best enclosed in double quotes to prevent the above transformations from occurring inside them. Any \ or " characters inside strings will need to be replaced by \\ and \" respectively, and the opening quote should be replaced by ‘.

Similar remarks apply to Pascal comments; don’t forget that { and } must be enclosed in double quotes. Alternatively, a @Com symbol can be placed in front of a comment enclosed in braces. It will add literal braces:

```
@Com { A Pascal comment }
```

has result

{ *A Pascal comment* }

It may still be necessary to enclose the interior in double quotes.

References

1. Adobe Systems, Inc., *PostScript Language Reference Manual, Second Edition*. Addison-Wesley, 1990.

2. Jensen, K. and Wirth, N., *Pascal User Manual and Report*. Springer-Verlag, 1975.

3. Kernighan, Brian W. and Cherry, Lorinda L., A system for typesetting mathematics. *Communications of the ACM* **18**, 182–193 (1975).

4. Kingston, Jeffrey H., *Document Formatting with Lout (Second Edition)*. Tech. Rep. 449 (1992), Basser Department of Computer Science F09, University of Sydney 2006, Australia.

5. Kingston, Jeffrey H., *A beginners’ guide to Lout*. Tech. Rep. 450 (1992), Basser Department of Computer Science F09, University of Sydney 2006, Australia.

6. Knuth, Donald E., *The TEXBook*. Addison-Wesley, 1984.