

Published: 05/12/67

Identification

Index and Errata for BN.4.02
Robert R. Fenichel

Purpose

It is nearly impossible to find stuff in BN.4.02 without using this appendix.

Method

The code "(e)" denotes an example; "(d)" denotes a description.

Index

activating components	3(d),8,22
addop	16(d)
aexpr	26(d)
ALEX	21(d),36(e)
ALLOC	20(d)
anomalous component	5,16(d)
ARBNO	22(d)
argument	16
assignment	26(d)
assignment list	26(d)
BCD stream	3,19
bexpr	24(d)
blanks	16(d),20
BLANKS	18(d),28,30(e),32(e),36
bprimary	24(d)

break	15(d)
bterm	24(d)
CARDOF	18(d), 35(e)
CARDON	18(d)
CHAR	19(d)
character class component	3, 16(d), 30(e)
character class declaration	13(d), 28(e), 31(e), 34(e)
character class name	13(d)
character set	9(d)
CHKFLG	21(d), 35(e)
CLRFLG	21(d)
comment	11(d), 14, 29(e), 30(e), 31(e), 32(e), 33(e), 34(e), 35(e)
components	2ff
COMPUTE	5(e), 24(e), 25(d), 32(e), 33(e), 35(e), 36
console stream	3, 25
continuation	16(d), 22(e), 28, 35(e)
CONTXT	20, 27(d)
current symbol	5, 20, 21, 27(d)
CVTD	25(d)
CVTO	25(d)
dc	11(d)
decimal integer	13(d)
declaration	11(d)

declaration division	10,11(d)
defc	17(d)
definition	16(d),30(e),33(e)
definition component	6,8
definition division	10,17(d),30(e),35(e)
definition element	8(d),17(d)
definition line	18(d)
definition name	18(d),33(e)
definition text	17(d)
DELETE	19(d),20,32(e)
description list	26,34(e),35(e)
detour	16(d),28,29(e),33(e),36
DICT	21(d)
dirtyies	36
earg	24(d)
ee list	25(d)
element	15(d)
element list	15(d)
ENCODE	24(d)
end card	10,18(d),30(e)
ENTER	20(d)
EOLMRK	19(d)
e pseudo	24(d)
EQUADR	5,20,27(d),35(e)

es interior	24(d)
es name	24(d)
EXIT	6,22(d),24(e)
external name	5(d),20,21
fbreak	14(d)
fc with	16(d),36,37
fc without	16(d)
fd	12(d)
F definition	17(d),35(e)
FIND	20(d)
fixed string declaration	13(d),32(e)
flag declaration	14(d),34(e)
flaglist	14(d)
flagname	14(d)
Fn	8
free definition	17(d)
free variables in definitions	8,33(e)
function call component	4,16(d),30(e),35(e)
function name	12(d)
GETCRN	21(d)
GETNAM	5,20(d),35(e)
GETREE	23(d)
GETVAL	21(d),27

GLOT	19(d)
header division	10(d), 34(e)
id	11(d)
IF	24(d), 35(e), 36
INSTAL	5, 20(d), 30(e), 32(e), 35(e)
integer	13(d)
internal name	5, 20, 21
INTVAL	20, 21, 27(d)
J	5, 18, 27(d), 32(e)
larg	27(d)
lhs	26(d)
LISTS	19(d)
literal string component	3, 16(d), 32(e)
l pseudo	27(d)
ls interior	27(d)
ls name	27(d)
LOCAL	26(d)
MARK	19(d)
MARKS	4, 20(d), 30(e), 32(e)
matching components	2(d)
NAME	5, 20(d)
NAMEST	20(d)
NL definition	17(d), 30(e)
NOBLKS	18(d), 20, 28, 36

NOCOM	19(d)
NOFLGS	21(d), 34(e)
NOGO	22(d)
NOLSTS	19(d)
non-definition component	15(d)
NOT	22(d), 32(e)
NULL	6, 22(d)
object stream	3
octal integer	13(d)
OR component	4, 16(d), 32(e)
output	3
P1	6, 22(d), 36
P arglist	17(d)
PARSDO	22(d), 35(e), 36, 37
parsdo argument	22(d)
P def	17(d), 30(e)
P definition	9, 17(d), 30(e)
Pig Latin example	31
primary	26(d)
pseudo sentence	15, 23(d)
Q definition	9, 17(d), 33(e)
quoted character def	9, 17(d), 35(e)
REFOFF	19(d), 29(e)

REFON	19(d)
relation	24(d), 35(e)
relative detour	16(d), 28, 32(e), 36
relop	24(d)
RESET	19(d)
SAVTRE	23(d), 36
sc	15(d)
sentence	14, 15(d)
sentence beginning	15(d)
sentence division	10, 14(d)
sentence fragment	16(d), 28
sentence label	15(d)
sentence middle	15(d)
sentence name	6, 12(d)
sentence name component	6, 16(d), 27
SETFLG	21(d), 35(e)
signed integer	13(d)
starred character class component	4, 16(d), 30(e), 37
STRING	19(d)
string name	13(d)
subject	14, 15(d)
SUBSAV	21(d)
SUBST	21(d)

super sentence	7,22,29(e),32(e)
super sentence declaration	12(d),29(e)
super sentence name	7
symbol buffer	4,20
symbol table	5
system cell declaration	12(d),32(e)
system cells	12,25,26,27(d)
system function declaration	12(d)
tab declaration	14(d),28(e)
tab definition	17(d),35(e)
tab list	14(d)
tab stop	14(d)
term	26(d)
tmgl program	10(d),29(e),31(e),34(e)
tmgs	13(d)
TMG string	13(d)
trail	6,7,25,27(d)
TREE	23,27(d)
TYPE	25(d),30(e)
VALBRA	23(d)
Varg	23(d)
variable declaration	12(d),32(e)
variable name	12(d)

vdesc	14(d)
vector declaration	14(d)
vector list	14(d)
vector name	14(d),26
ve list	23(d)
vs interior	23(d)
vs name	23(d)
v pseudo	23(d)
word-count example	34
word-reversing example	7,29
YESCOM	18(d)
**	5,29(e),30(e),36(e)
//	6,30(e)

Errata

<u>Page</u>	<u>Line</u>	<u>For</u>	<u>Read</u>
9	6	in page 10	on page 8
	9	S	\$
12	5	of B	B
13	-8	fixed	string
15	7	<sentence label>	<sentence label>
	7.5	~	<sentence label><sentence middle>
16	-6	(<	<
17	-2	definitions	definition
	-3	definitions	definition
22	10	See subsection	_____
	11	J, item #3.	p1 Acts like "=\$(\$p1\$)", even to the point of treating its detour like a continuation. See subsection J, item #3.
22	15	thet	that
	-1	\$)	\$/x1
24	6.5	_____	<bexpr> ::= <bterm> <bterm> . OR. <bexpr>
	7	<bexpr> ::=	<bterm> ::=
25	-7]	.
27	4.5	_____	<ls name> ::= <id>
28	20	successful	successful, the <u>blanks/</u> <u>noblks</u> mode is restored and

The following is a replacement for page 33.

```

*      Here we go.  You should know that "porcify" is a definition which
*      takes a split word (say, "squid," split into "squ" and "id") and
*      produces Pig Latin.  So the point of "bigword" is mainly just to split
*      words into pieces for "porcify."
*      We start by noting the input scan pointer, J.  We will use J
*      to compute the quantity of output which we have produced; in the
*      case of words starting with consonants, for example, the output
*      produced is as long as the input [ (ending J)-(starting J) ]
*      plus two for the added "ay."
*      Now we use "marks" and we start collecting letters.
*      Suppose the first letter is a vowel.  In this case, the detour of
*      the "vowel" component is irrelevant, and we proceed to "big-word-4."
*      The "letter*" component collects the rest of the word and
*      the "compute" ups the column-count to show the word and the
*      coming "yay".  Now we go to "porcify" with an imaginary word,
*      split into a leading "y" and a trailing string which is the
*      word we really found.
*      Suppose the first letter of the word is not a vowel ("big-word-1"),
*      but rather it is "y".  Then we skip the detour of the "y" component, and
*      we continue with big-word-3.  If a consonant follows the "y", then we
*      drop immediately to the next line, which is exactly where we were a
*      minute ago when the word began with a vowel.  If a non-consonant follows
*      the "y," however, we slyly reinitialize the letter-collector with a new
*      "marks" component.  Only then do we drift down to the next line, where
*      the code which thinks it is converting "am" into "amyay" is actually
*      converting "yam."
*      Finally, suppose the word begins with some letter not one of
*      [a,e,i,o,u,y].  We collect its initial string, collect the remainder,
*      and porcify the word split into this pair.
*      The punctuation-handler is rather an anticlimax.
*
punctuation.. blanks compute(startword=j)
      marks puncts* install/(
          // paragraph/(=$( $)) compute(column-count=5)  =$(///$)
      compute(column-count=column-count+j -startword)=p1
*
*      Finally, the definitions
*
.definitions.
porcify=(2)$( $q2$qlay$)
p1=$( $p1$)
p2=$( $p2$)
p3=$( $p3$)
y=$(y$)
ay=$(ay$)
end

```