

```
1 ;Lundi 11 Mars 2002 7:1:48 file : prt-pg.vlisp ;
2
3 ;----- pretty-print, par [pg];
4 ; 18-Jan-82 : prettyfie les fonctions tracees.;
5 ;(setq --x-- careful);
6 ;(setq careful ());
7
8 (setq --x-- 'careful careful nil)
9
10 (df pretty (l) pretty)
11 (mapc l (lambda (f) (terpri) (pprint f)))
12
13 (de pprint (f ;; x) pprint)
14 (status print 3)
15 ; bit 0 : "" , bit 1 : pas d'espace.;
16 (setq lmargin 0)
17 (setq x (cdr (assoc (ftype f) '((7 . de) (8 . df) (9 . macro)))))
18 (and x (p-p [x f . (or (get f 'trace) (fval f))])
19 (terpri)
20 (status print 0)
21 f)
22
23 (de p-p (l ;; x xx) p-p)
24 (status print 3)
25 ; pour les appels exterieurs;
26 (cond
27 ((null l) (princh "()"))
28 ((atom l) (prin1 l))
29 ((and (eq (car l) quote) (null (cddr l))) (princh """) (p-p (cadr l)))
30 (t (if (and (listp (car l)) (eq (caar l) lambda))
31 (setq
32 l
33 ['let
34 (let ((v (cadar l)) (a (cdr l)))
35 (if (null v) nil [(nextl v) (nextl a)] . (self v a)))
36 . (cddar l))))
37 (setq xx (outpos))
38 (princh "(")
39 (p-p (car l))
40 ; attention a un non-atome en tete;
41 (selectq (and (litatom (setq x (nextl l))) (get x 'ptyp))
42 (1 ; format PROG ; (p-progn))
43 (2 ; format WHILE ; (p-p1) (p-progn t))
44 (3 ; format DEF ; (p-p1) (p-p1) (p-progn t))
45 (4 ; format COND ; (p-cond))
46 (5 ; format SELECTQ ; (p-p1) (p-cond))
47 (t ; format standard ; (t+3) (while (listp l) (p-p1)) (t-3)))
48 (and l (princh " . ") (princh l))
49 (princh ")"))))
50
51 (de p-p1 () p-p1)
52 (princh " ") (p-p (nextl l))
53
```

```

54 (de t+3 ()
55   (setq lmargin (+ lmargin 3)))
56
57 (de t-3 ()
58   (setq lmargin (- lmargin 3)))
59
60 (de p-progn (?)
61   (if (and (null (cdr l)) (null ?))
62       (p-p1)
63       ; un seul argument;
64       (t+3)
65       ; plusieurs;
66       (while (listp l) (if (> lmargin (outpos)) (outpos lmargin) (terpri)) (p-p (nextl l)))
67       (t-3)))
68
69 (de p-cond ()
70   (t+3)
71   (while (listp l)
72     (terpri)
73     (princh "(")
74     (let (l (nextl l)) (p-p (nextl l)) (if l (p-progn)))
75     (princh ")"))
76   (t-3))
77
78 (mapc '(progn ; type progn; prog1 and exit or) (lambda (x) (put x 'ptyp 1)))
79
80 (mapc '(lambda ; type lambda; escape if ifn let mapc mapcar while until)
81   (lambda (x) (put x 'ptyp 2)))
82
83 (mapc '(de ; type def; df dm dmc) (lambda (x) (put x 'ptyp 3)))
84
85 (mapc '(cond ; type cond;) (lambda (x) (put x 'ptyp 4)))
86
87 (mapc '(selectq ; type selectq;) (lambda (x) (put x 'ptyp 5)))
88
89 (mapc '(setq ; typesetq multiple;) (lambda (x) (put x 'ptyp 6)))
90
91 ;(de e () (sh "ed pretty.vlisp") (lib pretty));
92

```

Cross Reference

p-cond de 69
p-p de 23
p-p1 de 51
p-progn de 60
pprint de 13
pretty de 10
t+3 de 54
t-3 de 57

1	--x--	8
2	?	60 61
3	a	34 35 35
4	f	11 11 13 17 18 18 18 21
5	l	10 11 23 27 28 28 29 29 29 30 30 32 34 34 36 39 41 47 48 48 52 61 66 66 71 74 74 74 74
6	macro	17
7	p-cond	45 46 69
8	p-p	18 23 29 39 52 66 74
9	p-p1	43 44 44 46 47 51 62
10	p-progn	42 43 44 60 74
11	pprint	11 13
12	pretty	10
13	ptyp	41 78 81 83 85 87 89
14	t+3	47 54 64 70
15	t-3	47 57 67 76
16	trace	18
17	v	34 35 35 35
18	x	13 17 18 18 23 41 41 78 78 81 81 83 83 85 85 87 87 89 89

19

xx

23 37

;Lundi 11 Mars 2002 7:1:48 end of file : pvt-pg.vlisp ;