

LaTeX document class for J3 work

Van Snyder
August 11, 2003

Contents

1	Introduction	1
2	Large-scale document structure	1
3	Cross-reference labels	2
4	Font specifiers	2
5	Support for BNF	3
6	Constraints	6
7	Commands for indexing	7
8	Environment for notes	7
9	Support for the intrinsic procedures sections	8
10	Miscellaneous list environments	9
11	Miscellaneous commands	9
12	Generating the standard	10
13	Commands useful in generating meeting papers	11

1 Introduction

2 This paper describes a LaTeX document class designed to be used for constructing J3 documents.
 3 It is intended to be used both for setting the standard, and for writing meeting papers.

2 Large-scale document structure

5 LaTeX documents begin with a `\documentclass` command. The J3 document class is derived
 6 from the `book` document class. All of the options of that document class continue to work. An
 7 additional option `memo` has been added that makes `\section` the top level structure. If `memo`
 8 does not appear, `\chapter` is the top level structure. The `\documentclass` command at the
 9 beginning of this document is:

```
10 \documentclass[twoside,11pt,memo]{j3}
```

11 In addition to the `memo` option and options of the `book` class, one can put the following options
 12 in the `\documentclass` command:

13 `color` turns on background color for notes. This is the default but it's easier to change the
 14 default than to add new `\documentclass` options.

15 `nocolor` turns off background color for notes. The reason for this is that most versions of
 16 the X-windows previewer, `xdvi`, are not able to cope with the commands that generate
 17 background colors. Also, the "device independent" (dvi) file processor for output to Laser
 18 Jet printers, `dvilj`, also doesn't understand those commands.

19 The primary differences between the `book` document class and the `j3` document class are:

- 20 (1) The default page style is to have headers and footers on every page. The headers
 21 and footers have a flush-left part, a flush-right part and a centered part.
 22 If `memo` does not appear, the page heading is for the draft standard. If `memo` appears,
 23 the default is to put nothing into the center of the headers and footers, and the page
 24 numbering becomes "Page <this-page> of <last-page>." One is expected to put

1 `\label{lastpage}` immediately before the `\end{document}` command. The memo
 2 option is intended for producing meeting papers.
 3 Two commands, that you are expected to renew, are invoked during production of
 4 the page headers and footers. The first is `\hdate`, and the second is `\vers`. Neither
 5 one has an argument. Here are examples of the commands to renew them. You can
 6 put them immediately after the `\documentclass` command.
 7 `\renewcommand{\hdate}{\today\ \printtime}` % Date for headers and footers
 8 `\renewcommand{\vers}{<paper number>}` % Version for headers
 9 The `\hftitle` command is used to fill the center part of the header and footer. Its
 10 default if memo is absent is WORKING DRAFT. If memo appears, its default is empty. In
 11 this document, it's
 12 `\renewcommand{\hftitle}{\LaTeX\ document class for J3}`
 13 The `\hff` command, default `\sffamily\bfseries\large`, is used to set the header
 14 and footer font.
 15
 16 (2) There is a new sectioning command `\annex`. It generates the correct form of page
 17 heading for annexes of 007. It is a synonym for `\appendix`.
 18
 19 (3) The sectioning commands invoke a command `\secfont` to set the font for sections.
 20 The default is `\sffamily`. You can, of course, renew this command. The `\chapter`
 21 command puts "Section" before the chapter number, and a colon after. Unlike in
 22 the book class, our `\chapter` command puts the title all on one line.
 23
 24 (4) The page layout is adjusted to be the same as the draft standard.
 25
 26 (5) Numerous environments and commands have been added. These are described be-
 27 low.

26 3 Cross-reference labels

27 The document class provides a command `\divn` that takes two arguments. The first is expected
 28 to be a sectioning command, and the second is the title of the section. It invokes its first
 29 argument and gives it its second argument. Then it creates a label consisting of "D" followed
 30 by the chapter number in arabic numerals and a colon, and then the second argument. Blanks
 31 and everything else are significant in labels, and the case of letters is significant. T_EX special
 32 characters, e.g. `\` and `}`, are not allowed. The chapter number is inserted in an attempt to
 33 make labels unique. If memo is specified, the chapter number is zero. Remember that in L^AT_EX
 34 one can refer to the text of an entity's number with the `\ref` command, and to the text of its
 35 page number with the `\pageref` command. This section was begun with

```
36 \divn\section{Cross-reference labels}
```

37 This reference, i.e. (3), was produced with `\ref{D0:Cross-reference labels}`.

38 You can't use `\divn` if the section title has a command in it (because of the `\`).

39 In any case, you can create your own labels, on section commands or elsewhere, with the L^AT_EX
 40 `\label` command. If a label is in a table, an equation, a figure, an item in a list, the left-hand
 41 side of a BNF term, a constraint (6), a note (8), and perhaps a few other places, a `\ref` to that
 42 label will produce the object's number, not the section number.

43 The `\nref` command is provided to make references to note numbers consistent: It puts "Note"
 44 before the note number.

45 4 Font specifiers

46 There are several font specifiers:

1 `st` The `\st` command sets its argument in “syntax term” type face. The default definition is
 2 `\emph`, which in turn defaults to italic.

3 `obs` The `\obs` command sets its argument in “obsolete font”. The command
 4 `\obs{obsolete}` produces obsolete.

5 `cf` The `\cf` command sets its argument in “code font” font. The command
 6 `\cf{code font}` produces code font.

7 `obscf` is a combination of `\obs` and `\cf`

8 **5 Support for BNF**

9 Numerous commands are provided to support BNF.

10 **5.1 Commands to create BNF**

11 **5.1.1 The `bnf` command**

12 The `\bnf` command is the basic command to set BNF rules. It takes three arguments. The
 13 first is the syntax number and syntax term. The second is either **is** or **or** (of course, you can
 14 stick anything you want in there). The third is the right-hand side of the BNF rule. The first
 15 argument is set in a box 2.25 inches wide. The second is set in `\bf` font in a box equal to the
 16 width of **or** plus 1em. The third one is set in a L^AT_EX `\mbox`, so if it is long, it will extend
 17 into the margin instead of being folded. It isn’t folded automatically, because we want the
 18 continuation mark (see 5.1.7).

19 If an internal flag `@bnfindex` is **true** it puts the entire syntax rule in the index of syntax rules.
 20 This flag is set by `\bnfi` (5.1.4) and cleared by `\bnfn` (5.1.9) and `\bnfx` (5.1.8) commands.
 21 There is a `\bmf` command that doesn’t put things in the index.

22 For example, the command `\bnf{\st{abc}}{is}{DEF \st{ghi} JK}` produces
 23 *abc* **is** DEF *ghi* JK

24 The `\bnf` command doesn’t automatically start or finish a paragraph, so if you don’t put blank
 25 lines or `\\` around it, you will find a BNF rule in the middle of a line.

26 Other commands described below are usually easier to use, so you probably won’t use either
 27 `\bnf` or `\bmf` directly.

28 **5.1.2 The `xsn` command**

29 The `\xsn` (“explicit syntax number”) command takes two arguments. The first is an op-
 30 tional syntax rule number (optional arguments are enclosed in square brackets). The sec-
 31 ond argument is a syntax term. It puts “R” in front of the first argument and sets it in a
 32 box 0.5in wide, and then sets the second in the `\st` type face. This is one of the ways to
 33 create the first argument for the `\bnf` command. Using `\xsn` in the previous example, e.g.
 34 `\bnf{\xsn[604]{abc}}{is}{DEF \st{ghi} JK}` produces

35 R604 *abc* **is** DEF *ghi* JK

36 You probably won’t use `\xsn` directly.

37 **5.1.3 The `sn` command**

38 The `\sn` (“syntax name”) command takes one argument, a syntax term. It sets its argument
 39 in `\st` type face. Then it creates a new syntax number by incrementing the `sr` (“syntax rule”)
 40 counter, and concatenating it (with at least two digits) onto the chapter or section number

1 (the latter if `memo` is specified). Finally, it creates a label consisting of `sr:` (for “syntax rule”)
 2 followed by the argument.

3 Using `\sn` in the previous example, e.g. `\bnf{\sn{abc}}{is}{DEF \st{ghi} JK}` produces
 4 R501 *abc* is DEF *ghi* JK

5 Notice that we’re in section 5, and that is the leading digit of the syntax rule number. Also
 6 notice that “R” has been put ahead of the syntax number. This is because `\sn` uses `\xsn`
 7 (5.1.2) to combine the generated syntax number and the syntax term. You probably won’t use
 8 `\sn` directly.

9 5.1.4 The `bnfi` command

10 The `\bnfi` (“BNF is”) command takes two arguments. The first is the syntax rule name, and
 11 the second is the (first line of) its right-hand side. It generates a syntax rule number and
 12 sets the name, using the `\sn` command. Then it puts this as the first argument of the `\bnf`
 13 command, puts `is` as the second argument, and puts its second argument as the third argument
 14 of `\bnf`. If `memo` is not present in the `\documentclass` command, it enters the syntax term into
 15 the index of syntax terms, to be displayed with the syntax rule number and a bold face page
 16 number, enters the entire syntax rule in the index of syntax rules, and sets a switch that causes
 17 subsequent syntax rules also to be entered in that index. Our above example could have been
 18 written `\bnfi{abc}{DEF \st{ghi} JK}`, producing

19 R502 *abc* is DEF *ghi* JK

20 This would put “*abc* (R502), 4” into the index of syntax terms. By the way, the “index term”
 21 is *abc* alone, so if you put a reference to *abc* in the syntax term index (using the `\tindex`
 22 command – see section 7), it will come at the same place in the index.

23 Notice that the example syntax rule above is not exactly the same as in section 5.1.3, because
 24 a new syntax rule number has been invented. In this document, it also generates a duplicate
 25 label `sr:abc`, because the term *abc* was also defined in section 5.1.3. (If you have duplicate
 26 labels, a `\ref` command refers to the last one of them, so references to `sr:abc` will be to R502.)

27 5.1.5 The `bnfo` command

28 The `\bnfo` (“BNF or”) command takes one argument – the (first line of the) right-hand side
 29 of the `or` part of a syntax rule. It’s the same as `\bnf{}{or}{<right-hand-side>}`. We might
 30 continue our above example with `\bnfo{PQR \st{xyz}}`, which produces

31 or PQR *xyz*

32 5.1.6 The `bnfr` command

33 The `\bnfr` (“BNF right-hand-side”) command takes one argument – (one line of) the right-
 34 hand side of a syntax rule. It puts the `\bnfc` syntax rule continuation symbol (see 5.1.7) before
 35 its argument, and then uses the result as the third argument for `\bnf`, i.e. it’s the same as
 36 `\bnf{}{}{\bnfc <right-hand side>}`.

37 5.1.7 The `bnfc` command

38 The `\bnfc` (“BNF continuation”) command has no arguments. It produces the BNF continua-
 39 tion symbol, *viz.* ■. You need to put this at the end of the right-hand side of continued BNF
 40 rules, but `\bnfr` (see 5.1.6) will put it at the beginning of continuing lines for you.

41 5.1.8 The `bnfx` command

42 The `\bnfx` (“BNF is with eXplicit rule number”) command takes three arguments. The first
 43 is an explicitly specified rule number. The second is the syntax term. The third is the (first

1 line of the) right-hand side of the rule. The first two arguments are put together by `\xsn`.
2 This result is then used as the first argument of `\bnf`, with `is` for the second argument, and
3 the third argument of `\bnfx` is used as the third argument for `\bnf`. This one is intended to
4 be useful for producing meeting papers, wherein you want to refer to a syntax rule number in
5 the standard, not have L^AT_EX invent one for you. It does not enter its name into the index of
6 syntax terms, or the rule into the index of syntax rules, and it turns off the switch that causes
7 subsequent BNF-generation commands to put their rules into the index of syntax rules.

8 **5.1.9 The `bnfn` command**

9 The `\bnfn` (“BNF `is` with rule number gotten by reference to a Name”) takes two arguments:
10 the syntax term for the left-hand side, and the (first line of the) right-hand side. The syntax
11 number is gotten by reference to the name (the first argument). This command is used when
12 quoting a syntax rule in a place other than its home. The syntax rules that are defined in
13 section 7 of the standard but referenced in section 3 are set using the `\bnfn` command. It does
14 not enter its name into the index of syntax terms, or the rule into the index of syntax rules,
15 and it turns off the switch that causes subsequent BNF-generation commands to put their rules
16 into the index of syntax rules.

17 **5.1.10 The `bnfz` command**

18 The `\bnfz` (“BNF *zilch* – BNF `is` with no rule number”) command takes two arguments: the
19 syntax term for the left-hand side, and the (first line of the) right-hand side. No syntax rule
20 number is produced, but the left-hand side is indented the same amount it would be if a syntax
21 number were provided.

22 **5.1.11 The `bnfb` command**

23 The `\bnfb` (“BNF block”) command takes one argument: a part of the right-hand side of a
24 syntax rule. It doesn’t put `\bnfc` before the right-hand side. It is intended to be used for
25 constructs.

26 **5.2 Commands to reference syntax terms**

27 There are several commands to display and reference syntax terms and rule numbers. The ones
28 that claim to enter syntax terms into the index of syntax terms only do so if the `memo` option
29 is not present in the `documentclass` command.

30 **5.2.1 The `si` command**

31 The `\si` (“syntax index”) command takes one argument – a syntax term. It sets it in `\st` type
32 face, and enters the reference into the index of syntax terms.

33 **5.2.2 The `stdef` command**

34 The `\stdef` (“syntax term definition”) command takes one argument – a syntax term. It sets
35 it in `\st` type face, and enters the reference into the index of syntax terms with a bold-face
36 page number.

37 **5.2.3 The `sir` command**

38 The `\sir` (“syntax index with reference number”) command takes one argument – a syntax
39 term. It sets it in `\st` type face, then sets its rule number between parentheses, and finally
40 enters the reference into the index of syntax terms. For example, `\sir{abc}` produces *abc*
41 (R502) (*abc* was defined in section 5.1.4).

1 5.2.4 The `\sid` command

2 The `\sid` (“syntax index with definition page number”) command takes one argument – a
3 syntax term. It sets it in `\st` type face, then enters the reference into the index of syntax
4 terms, with its rule number, as a definition – i.e., with a bold-face page number.

5 5.2.5 The `\sidn` command

6 The `\sidn` (“syntax index with no syntax number but a definition page number”) command
7 takes one argument – a syntax term. It sets its argument in `\st` type face, and enters it in the
8 index with a bold face page number. This is intended to be used for the definition of terms
9 that are defined by explanation rather than BNF rules – e.g. *letter*.

10 5.2.6 The `\sinr` command

11 The `\sinr` (“syntax index with no syntax number”) command takes one argument – a syntax
12 term. It sets its argument in `\st` type face, and enters it in the index. This is intended to be
13 used for terms that are defined by explanation rather than BNF rules – e.g. *letter*.

14 5.2.7 The `\snref` command

15 The `\snref` (“syntax number reference”) command takes one argument – a syntax term. It
16 sets its syntax rule number (not between parentheses). For example `\snref{abc}` produces
17 R502 (*abc* was defined in section 5.1.4).

18 5.2.8 The `\sref` command

19 The `\sref` (“syntax reference”) command takes one argument – a syntax term. It does every-
20 thing that the `\sir` command does, except for putting a reference in the index.

21 6 Constraints

22 There is a list environment for setting several consecutive constraints, and a command for
23 setting one constraint. They both invent new constraint numbers in the same way that syntax
24 rule numbers are invented (but with a “C” instead of an “R”). See section 5.1.3.

25 6.1 The `\cons` environment

26 The `\cons` environment is a list environment for setting several consecutive constraints. Each
27 constraint is introduced by an `\item` command. For example,

```
28 \begin{cons}
29 \item First constraint.
30 \item Second constraint.
31 \end{cons}
```

32 produces

```
33 C601 First constraint.
34 C602 Second constraint.
```

35 As with any list environment, you can put your own labels in optional arguments of the `\item`
36 command.

37 The width of the label is the same as the space allowed for the syntax rule number in a BNF
38 definition (actually 0.5in + 1em).

1 6.2 The `\dcons` command

2 The `\dcons` command produces one constraint. It takes two arguments. The first one is optional
 3 (remember that optional arguments are enclosed in square brackets). It is an explicit constraint
 4 number (with “C” if you want it) to override the generated one. This is intended for meeting
 5 papers. The generated constraint number includes the section number. The constraint counter
 6 is incremented even if an explicit one is provided.

7 The second argument is the text of the constraint. Here is an example of a constraint on R502,
 8 produced by `\dcons{(\snref{abc}) The \st{ghi} shall be a ghi.}`:

9 C603 (R502) The *ghi* shall be a ghi.

10 This command does *not* comprise a paragraph. Moreover, its body is set using “hanging
 11 indentation,” which has a scope of the entire paragraph in which it appears. This
 12 paragraph is an example of the surprise you’ll get if you try to separate `\dcons` from
 13 adjacent text with `\\`.

14 7 Commands for indexing

15 There are three low-level commands to generate index terms. The reason for three is to have
 16 separate indices for general terms, syntax terms, and the syntax rules themselves.

17 The commands are `\mindex` to enter a term in the “main” index, `\rindex` to enter a complete
 18 syntax rule in the “syntax rule” index, and `\tindex` to enter a syntax term in the “syntax term”
 19 index. There is also a `\mindex*` command that sets its text *and* puts it in the index. There are
 20 also `\mindexd` and `\mindexd*` commands that are for definitions – they put a bold-face page
 21 number in the index.

22 The BNF commands use `\tindex` and `\rindex`. You will probably not use `\tindex` directly –
 23 it is preferable to use it by way of `\si` (5.2.1) or `\sir` (5.2.3). The `\tindex` command is not
 24 effective if `memo` appears in the `\documentclass` command. The `\rindex` command doesn’t do
 25 anything if the class-internal flag `@bnfindx` is false, so there’s no reason to try to use `\rindex`
 26 directly. It is also not effective if `memo` appears in the `\documentclass` command.

27 The `\kw` command puts a keyword into the index. The `\kw*` command puts the keyword into
 28 the text and the index. There are also `\kwd` and `\kwd*` commands that are for definitions –
 29 they put a bold-face page number in the index.

30 8 Environment for notes

31 The `note` environment increments a note counter, sets **NOTE** followed by the section and note
 32 numbers separated by a period, and then puts the body of the note in a box. If a note is split,
 33 the note heading is duplicated on the continuing page with “(cont.)”.

34 Note backgrounds can be colored. The color can be specified by defining `noteback`, e.g.,
 35 `\definecolor{noteback}{gray}{0.95}`. Note coloring can be turned off by putting the
 36 `nocolor` option in the `\documentclass` command, or by specifying the `\nocolor` command. It
 37 can be turned back on with the `\docolor` command. One reason to turn off note background
 38 coloring is that it is done by PostScript specials. Neither `xdvi` nor `dvilj` know what to do with
 39 these; they just throw up their hands in despair “Oh Dear! PostScript color specials! I better
 40 just do black (no matter what the color)!” So you get a black background with black text on
 41 it.

42 Here’s a note created by

```

1 \begin{note}
2   This is a note.  Its background color is noteback and its
3   foreground color is notefore.
4 \end{note}

```

NOTE 8.1

This is a note. Its background color is noteback and its foreground color is notefore.

5 **9 Support for the intrinsic procedures sections**

6 **9.1 Subsections in the intrinsic procedures sections**

7 The `\insubsection` (“intrinsic subsection”) command sets its argument with the same spacing,
 8 size and font as a `\subsection` command, but it doesn’t create a table-of-contents entry.

9 **9.2 Environment for the table of specific and generic names**

10 The `threecol` environment is used to set the three-column table of specific and generic names
 11 in section 13.6. Actually, there are four columns – one for the bullet that indicates the specific
 12 name is not allowed to be an actual argument, but the equivalent tag in Frame was named
 13 `threecol`.

14 **9.3 Environment to display intrinsic procedure summaries**

15 The `\insum` environment is a list environment intended for the intrinsic procedure summaries
 16 in section 13.

17 **9.4 Environment for arguments for intrinsic procedures**

18 The `args` environment is a list environment. Each item sets its optional argument (the one in
 19 square brackets) in bold face type in a 1.5in box. Also see 9.5.

20 **9.5 A command to display intrinsic procedure arguments**

21 The `\intrinarg` command takes two arguments. The first is an intrinsic procedure argument
 22 name, and the second is its description. It does the same thing as the `args` environment (9.4),
 23 but only for one argument.

24 **9.6 An enumeration environment for intrinsic function argument cases**

25 The `incase` environment is a list environment. The label of each item is set in `\emph` type
 26 face. It consists of the word “Case” followed by the optional argument of the `\item` command
 27 in parentheses, followed by a colon. If no item label argument is given, one is generated in
 28 lower-case roman numerals. The item label is set in a box 0.8125 inches wide.

29 **9.7 Paragraphs in intrinsic procedure descriptions**

30 In the intrinsic procedures sections, paragraphs are introduced by a word in bold-face type –
 31 or not – but in either case, the paragraph is indented using the `\inp` (11.2) command (11.2)
 32 and the length `\II` (“intrinsic indent”), which has a value of 0.5in.

33 The paragraphs that are introduced by a word in bold-face type are generated by the following
 34 commands. The commands that end in B are intended to be “bigger” – then have more line
 35 spacing.

command	introductory word		command	introductory word
---------	-------------------	--	---------	-------------------

argument	Argument	arguments	Arguments
class	Class	desc	Description
example	Example	exampleB	Example
examples	Examples	examplesB	Examples
reschar	Result Characteristics	restriction	Restriction
result	Result	resvalue	Result Value
resvalueB	Result Value		

1 These commands generate an `inpara` (“intrinsic paragraph”) command, which takes two ar-
 2 guments – the bold-faced word, and the rest of the paragraph. The `inpara` command puts a
 3 period after the bold-faced word, and sets the whole thing as an “indented paragraph” using
 4 the `\inp` command.

5 **10 Miscellaneous list environments**

6 There are several list environments, with their label widths and styles chosen to match the draft
 7 standard.

8 **10.1 A general enumeration environment**

9 The `enum` environment is similar to the L^AT_EX `enumerate` environment. The differences are

- 10 (1) the outermost label width is 3/4 inch
- 11 (2) the remaining label widths are 3/8 inch
- 12 (3) the numbering for the outermost level is arabic in parentheses
- 13 (4) the numbering for the second level is lower-case alphabetic in parentheses
- 14 (a) that is, like this one

15 The remainder of its behavior is the same as for the L^AT_EX `enumerate` environment. I looked
 16 superficially for three-level lists in the draft standard, but didn’t find any. If there are any, it
 17 will be easy to change `enum` to have the desired style.

18 **10.2 A “non-bold label” description environment**

19 The `nbdesc` environment is a list environment that works like the L^AT_EX `description` environ-
 20 ment, except that it doesn’t set the labels in bold face type.

21 **11 Miscellaneous commands**

22 **11.1 Commands to define a term**

23 The `\tdef` command sets its argument in bold face type, and creates an index entry for it that
 24 will have a bold face page number.

25 The `\tdeff` command just sets its argument in bold face type, without creating an index entry.

26 **11.2 A command to generate an indented paragraph**

27 The `\inp` command generates an “indented paragraph.” It takes one argument: The amount to
 28 indent the paragraph. The entire paragraph is indented this amount. It doesn’t matter where
 29 it appears in the paragraph.

1 11.3 A command to generate a hanging indented paragraph

2 The `\hin` command generates a “hanging indented paragraph.” It takes one argument: The
3 amount to indent the paragraph. The first line of the paragraph is not indented, but the rest
4 of the paragraph is indented the amount given by the first argument. It doesn’t matter where
5 it appears in the paragraph.

6 11.4 Captions in tables

7 The `\jcaption` (“J3 caption”) command generates a caption for a table that consists of the
8 word “Table” followed by the table number and a colon, and then its argument in bold-face
9 type. It also makes a label for the table that consists of “T” followed by a colon, followed by
10 the text of the caption.

11 The `\ccaption` (“continued caption”) command generates a caption for the part of a table
12 continued onto a subsequent page as for `\jcaption` but followed by “(cont.)”. It doesn’t
13 generate a label.

14 11.5 Double underline

15 Some of the table headings are formatted with a double underline. This is generated with the
16 `\dul` command.

17 12 Generating the standard

18 The standard is organized as a top-level document that includes low-level documents. L^AT_EX
19 provides an `\includeonly` command that allows to process only a part of the document, without
20 clobbering the cross references and indexes for the rest of it. If you want to generate just one
21 part, uncomment the `\includeonly` near the top of the main document and put a file name
22 in it (without `.tex`). Unfortunately, you frequently get an extra page or two at the beginning
23 and/or end of the section.

24 There is a `Makefile` to make the standard.

25 The command `make 007.dvi` runs `latex` twice on `007.tex`. It is necessary to run it twice to
26 get page numbers put into the index correctly. Then it runs `makeindex` using the J3 index style
27 file `j3.ist`. Then it postprocesses the index using the `hyperindex` program because hyperlinks
28 from the index to the n’t^h page of the document, not the page numbered n (these are different
29 because the page numbering is restarted after the front matter. Then it converts the index of
30 syntax rules to something that can be put back into the document. This consists of using `sed`
31 to remove `hyperpage` from the end of each line. Using `makeindex` wouldn’t be appropriate,
32 because that would sort the syntax rules incorrectly (e.g. all of the `or` rules would come at the
33 end) – but it would have replaced the hyperpage references. Finally, it runs `latex` twice more,
34 to make sure that all of the cross references and line numbers are correctly resolved. The result
35 is the T_EX “device independent” file `007.dvi`.

36 Having `007.dvi`, one can convert it for output on different printers. One can use `make 007.ps`
37 to make a PostScript file `007.ps`. One could also view it using `xdvi` on Unix systems or an
38 equivalent program on other systems, or convert it for different printers. There aren’t any
39 `Makefile` sections for other conversions.

40 PDF is generated by `make 007.pdf`. This section in the `Makefile` very much like the `007.dvi`
41 section, but it uses `pdflatex` instead of `latex`.

42 Text is made from PDF by `make 007.txt`.

43 The command `make all` makes `dvi`, PostScript, PDF and text.

- 1 The command `make clean` deletes all of L^AT_EX's output and intermediate files.
 2 The command `make ui-index` makes the “index of unresolved issues” paper. It uses the file
 3 `ui-index.tex`, into which you will need to insert the `\hdate` and `\vers` commands (2).

4 **13 Commands useful in generating meeting papers**

5 **13.1 The `edits` command**

6 The `\edits` command generates a description of the typographical conventions. It takes two
 7 arguments. The first one is optional (remember that optional arguments appear in square
 8 brackets). The section title is “Edits” followed by the optional first argument. The second one
 9 is the version of the draft standard to which the edits apply, e.g. `01-007r2`.

10 **13.2 The `sep` command**

11 The `\sep` command creates a vertical space of 5pt, and then generates a line that goes all the
 12 way across the page. It has no arguments. Here's what it does:

13 **13.3 The `mmpar` command**

14 The `\mmpar` command creates a marginal paragraph. Its primary use is to put page and line `\mmpar`
 15 numbers in the margin. There's a marginal paragraph adjacent to the first line of this paragraph.
 16 The `\mmpar*` command doesn't begin with an empty `mbox`. This changes vertical spacing, which
 17 usually makes it wrong, but improves things for marginal paragraphs adjacent to notes.

18 **13.4 The `mmpare` command**

19 The `\mmpare` command creates a marginal paragraph in `\emph` font (hence the “e” in the name). `\mmpare`
 20 There's also `\mmpare*` that works as for `\mmpar*`.

21 **13.5 Put boxes around stuff**

22 The `boxit` environment puts a box around its content. The `lbox` environment also puts a box
 23 around its content, but it does it by using the `longtable` environment, so it can be split at
 24 “new item” (`\`) signals at page boundaries.

25 **13.6 Note with explicit note number**

26 The `xnote` environment creates a note – in the same way that `note` (8) does, but instead of
 27 inventing a note number, you specify it. The command `\begin{xnote}{XYZ}` introduces an
 28 environment that creates a note box with **NOTE XYZ** above it.

29 **13.7 J3 internal note**

30 The `jnote` environment creates a note – in the same way that `note` does, but instead of inventing
 31 a note number, it puts **J3 internal note** above the box.

32 **13.8 References to the standard**

33 The `xr` package can be used to make “external references,” by putting the following in a paper:

```
34 \usepackage{xr}
35 \externaldocument{007}
```

36 This requires that the `007.aux` file be accessible. Using the `tetex` T_EX distribution on Linux
 37 or Unix, this can be done by naming the appropriate directory in the `TEXMFLOCAL` environment
 38 variable. The directory named in `TEXMFLOCAL` needs to have a `tex` subdirectory, and that sub-
 39 directory needs to have a `latex` subdirectory. All of the subdirectories there are automatically

1 searched. I put a “soft link” from there to `$HOME/f2000/007`, which is in turn a “soft link”
2 to the directory having the `.tex` files for the current standard. There are probably analogous
3 ways to set things up for Windows-based distributions of T_EX.
4 I make the standard, using the `Makefile` the editor prepared, to generate the `.aux` files. After
5 making the current draft of the standard, I run `texhash`.
6 Once things are set up, and the `.aux` files are generated, cross references can be put into meeting
7 papers using identical macros as for the same cross references in the standard, thereby saving
8 some work for the editor.