Management Information System Task Force Note October 20, 1986

Author: A.Berglund

Subject: **Audit of Central Text Processing on IBM**

DEFINITIONS

word processing: creation of a relatively short, simple document. Normally done on a stand-alone system which can be seen as an automated typewriter.

text processing: creation of either a longer or more complex document. Features like typographic quality text (e.g. multiple sizes of characters, different presentations – bold, italic, bold italic characters) and/or features like table of contents and index. Frequently the author adds logical information about the structure of the document over and above that contained in the word processing area.

SPECIFIC MARKUP VERSUS GENERIC MARKUP

There are two approaches one can take in describing a document – specific markup and generic markup.

Specific Markup:

Specific markup consists of adding low level formatting "commands" to the text, such as carriage return, center the following test, go to the next page. It is thus text processing at the "assembly" level. As an example let us consider the following example marked up in low level formatting commands:

```
.pa ;.sp 2 ;.ce ;.bd
Title of chapter
.sp
```

Specific markup has a number of disadvantages, for example:

- tied to a particular text-formatter and can thus only be exported to installations using the same text-formatter
- tied to a particular style (for a different style all occurences have to be edited)
- to a great extent device-dependent (a change from bold to e.g. underscored implies editing of all occurrences, dependent on pagesize and fonts available)
- effort for writer to think about *specific layout*
- major revision is needed if new material is added:
  - chapter (re)numbering is not automatic
  - placement of e.g. figures needs manual revision
  - table of contents is not automatically revised

  It is thus not easy to merge parts of documents from different authors
- formatting commands often long and cryptic – nothing for the casual user

Generic Markup:

Generic markup means adding information to the text indicating the logical components of a document, such as paragraphs, headers, footnotes. The formatting associated with the component is decoupled from the text, which gives a number of advantages. A header could be marked up as:

```
<H1>Title of chapter
```

Advantages of generic markup are:

- can be formatter independent
- style tailored in *one* place and it is thus easy to produce
  - different style from the same input data
  - refine the style at a late stage
- device dependence in *one* place
- "commands" can be made mnemonic and powerful and represent logical entities of the document and is thus closer to the thoughts of the author/writer
- it is easy to merge contributions from several authors and the style is automatically uniform
- many boring tasks are made easy
  - automatic numbering (& renumbering) of chapters, sections, figures, footnotes, ...
  - automatic generation of table of contents, list of figures, ...
  - automatically correct references

Note however:

- markup tag names have to be agreed for communication of texts
- one does not need to "manually" type the tags – a "WYSIWYG"[1] system can be made which remembers the "tags" in its data structure (see e.g. R.Watt: GML at Waterloo, From Micros to Mainframes, SEAS SM '84, proceedings I pp. 174 – 182).

WYSIWYG

Currently most standalone word processing systems are using specific markup, the codes entered by function keys and normally invisible on the screen.

Today most generic markup systems are processing an input file, created with a normal "text" editor, generating the formatted output file.

As was mentioned above there is no inherent reason why a system using generic markup should not be of the WYSIWYG type. Indeed one is starting to see systems that combine the best of both worlds[2]

---

1 What You See Is What You Get – i.e. the formatted text is shown on the screen as it is typed in.

2 One such commercial system, scheduled for release first quarter 1987, was shown by Datalogics at Markup '86. It follows the ISO 8879 (SGML) standard.

DEVELOPMENTS SEEN OVER THE NEXT 2 – 3 YEARS

It is clear that in the next few years more and more text processing is going to be performed on PCs and workstations and that mainframe (& PC) inputfile – processing – outputfile systems will be less and less used. SCRIPT, xROFF, TEX and other systems will fade away in preference for powerful generic markup systems of a WYSIWYG type. One should note, however, that such a system ***must*** have at least the same facilities as our current system to be viable. Thus almost all the systems currently offered on workstations are excluded.

SGML (STANDARD GENERALIZED MARKUP LANGUAGE)

SGML, ISO standard 8879, specifies a language for document representation. The design aims of the language include:

- making the structure of a document completely separated from its style of presentation
- being independent of the text formatting system (i.e. SGML is a common "layer on top")
- multiple processing of the same source document – e.g. to
  - format the complete document
  - extract parts of the text and keywords creating on-line help files
  - extract parts of the text to create a database of titles, authors and abstracts

SGML has caused a lot of interest in both the publishing industry and the computer industry. Implementations are "rumoured" to be developed by many firms. There have been created two user groups to stimulate and share experience with SGML: The SGML User's Group and EPMarkup.

It is of great interest to see that North-Holland are actively studying SGML and it may become possible for CERN authors to submit papers for publication in SGML form.

Large organizations that have adopted SGML include the Office of Official Publications of the EEC and the US Department of Defense.

SGML Applications

It should be made clear that the SGML standard ***does not***

> identify or specify "standard" document types, document architectures, or text structures

The definition of a Document Type Definition (DTD) is an SGML application. Such DTDs define the generic identifiers and attributes of the tags (e.g < FIG ID = name > being the tag to identify figures and having an attribute ID used as a name so one can refer to a particular figure) and the overall structure of the document (e.g. that a manual is composed of a front matter, followed by a body, followed by a back matter, and that the font matter is composed of a titlepage, followed by an abstract followed by a preface, followed by a table of contents and so on) from the highest level components down to the character level.

It is intended that industries, user groups and other sets of users with common needs for document components and interchange will develop DTDs for common use; these can be registered with ISO and thus made public.

ISO Technical Report

Within ISO/TC97/SC18/WG8 an ISO Technical Report "Information Processing — SGML Support Facilities — Techniques for Using SGML" is being written. It will contain sets of "recommended" tags for

- basic document (e.g. report, manual, preprint)
- letter
- memo
- spreadsheets
- mathematics
- graphics (from Computer Graphics Metafile) mixed with text

it will also contain examples of SGML applied to

- linguistics
- text with Hebrew as a base language (written right to left) with English text mixed in
- text with Latin as a base language with Arabic text (written right to left) mixed in

OTHER (RELATED) STANDARDS AND WORK ITEMS

Two SGML related standards have already reached Draft International Standard status (thus likely to become standards in 6 months time):

- ISO DIS/9069: Information Processing — Text and Office Systems — SGML Document Interchange Format
- ISO DIS/9070: Information Processing — SGML Support Facilities — Registration Procedures for Public Text

In various stages of preparation are:

- FDA (Future Document Architecture). A part of this project is TCL (Text Composition Language) which is moving towards the specification of an attribute system that could be used with SGML to define a publishing or documentation architecture. In other words define the layout to be associated with a particular tag.

  Another part is TPM (Text Presentation Metafile) where the work has active participation from the developers of both Interpress and PostScript.

- Fonts

- Text-entry systems

TEXT INTERCHANGE

I believe that the logical structure is the key to all document management. This is closely linked to the awareness that document interchange among unlike systems, particularly between systems of widely different capabilities, is best done on the basis of logical structures. Thus I see the future text interchange being made with documents with SGML markup rather than in any other format – e.g. already formatted text.

SGML, however, is not inherently limited to the description of document components, but is a very convenient way of identifying pieces of structured information and SGML applications could be created for the **interchange** (note: interchange format between systems and not what the end users sees at the workstation) of arbitrary information. Consider, for example, the following spread-sheet:

```
<ROW><CELL ID=item1>124                          </ROW>
<ROW><CELL ID=item2>296                          </ROW>
<ROW><CELL ID=total CAL="item1+item2">           </ROW>
```

WATERLOO SCRIPT AND ITS USER INTERFACES

System Characteristics

1. Hardware Configuration

   - Runs as an application on the central IBM and Siemens mainframes.
   - Waterloo SCRIPT is written in IBM assembler and thus only runs on computers using the S/370 architecture.

2. Basic Software

   - Waterloo SCRIPT can run both in a VM/CMS and an MVS (and TSO) environment.

3. Connected Terminals/Workstations/Printers/other peripherals

   - Waterloo SCRIPT does not require any special terminal for its use and is thus accessed from all terminals connected to the INDEX system, currently some 2000 terminals.
   - Even though Waterloo SCRIPT can format text to be printed on a wide range of output devices the special requirements at CERN of an extended character set makes flexible (laser, LED) printers highly desirable. Currently the connected "document" printers connected are
     - 1 APA6670 (in building 513). Price 90 k FrS (for a 6670 plus an amount of the same order for the APA interface – paid by IBM)
     - IBM 3812 printers. Price 16 k FrS each. Being bought by individual groups.
     - 3 Xerox 2700 laser printers, in buildings 2, 6, and 892. Price 43 k FrS each. They are also used for computer listings.
     - 2 IBM 6670 laser printers in buildings 4, and 376. Price 90 k FrS.
     - 1 IBM 3800 laser printer. The main use of this printer is computer listings.

4. Financial Aspects

- The annual license for Waterloo SCRIPT amounts to US $ 1800 per year ($ 900 per computer).
- The computing resources consumed by Waterloo SCRIPT for a typical document of 50 pages is 3.1 centimes a page (based on the accounted job cost, covering all the DD expenses for the equipment and divisional overhead).

5. Personnel Involved

- Definition of the user community
  - Like all of the facilities offered by the central computer system Waterloo SCRIPT is available to anyone who wishes to use it. Statistics from July and August for VM/CMS and for one week in October on MVS show that its use is

system used	times used per week	times used per working day	# different users
VM/CMS	1839	345	341
MVS	2506	471	271

  The paper quantity printed each month is approximately 100000 pages, of which 60000 is on the APA6670.
- Support personnel
  - First level support is carried out by the "Programming Enquiry Office". It is, however, clear that this support, whilst appropriate for the traditional (ie physicist) users of the central computers, is not adequate for support of secretarial staff. Experience from other sites shows that this support should be done by some number of interested and capable secretaries.
- Local developments
  - some tailoring of the basic Waterloo SCRIPT program for local output devices and some special requirements. Time: 3 man-weeks a year.
  - tailoring of output devices. Such tailoring is mostly needed in the area of special character sets required for scientific text. For the more recent printers it has been possible to purchase most of the needed characters, but the increased flexibility in number of sizes available on the devices the work needed is still non-negligible. Time: 2 – 4 man-weeks a year.
  - tailoring and modifying layout styles to CERN taste, installing optional changes to styles. Time: 2 – 3 man-weeks a year.
  - modifications to use SGML reference concrete syntax and tailoring layout styles. Time: 4 – 5 man-weeks (one-time).

6. Functional Description

- Objectives of the system
  - Provide text formatting service to the users of the central computers with emphasis of large and/or complex documents. Thus the advantages of marked-up text are of prime importance. The system is more intended to be used by physicists than by secretaries; no courses in the use are given as these users are assumed to be able to absorb a written manual.
- Language version used (English, French, ...) of program
  - English
- Language version used (English, French, ...) for data
  - English, French, German, Swedish, Danish, Norwegian, Italian, Spanish, ...
- Current performance
  - Resources on the central computers more than adequate for the processing resources needed.
- Backup of the data

  - As any other file on the central computers – i.e. good.
- Backup of the system
  - As any other application on the central computers – i.e. good.
- Technical limitations
  - typing of mathematical formulae difficult.
- Planned improvements/modifications/configuration changes
  - A mathematics processor ("EQN" syntax) will come with the next release of Waterloo SCRIPT – expected quite soon.
  - Introduction of SGML as the primary "user interface".
  - Introduction of the mixing of text and graphics.

7. Communication, what is done now and what is desirable

- interchange of text with:
  - systems of same type
    - – Can be done via numerous networks
  - systems of different type (which?)
    - – With SGML as markup scheme interchange of texts should be possible to many other systems using SGML.
- for bulk storage
  - Not applicable
- for printing on non-local printers
  - Not applicable
- use as terminal to other system
  - Not applicable

8. Specific Text-processing applications

- reuse of texts or "throw away" texts: see General User Audit
- document size (min, max, average number of pages): see General User Audit
- requirement for
  - National use characters (e.g. accents for French): Yes
  - Special symbols for γρεεκ and mathematics: Yes
  - Complex mathematical formulae: Yes
  - "typewriter" or "typographic" quality text: Yes
  - (automatically generated) table of contents: Yes
  - index: Yes
  - footnotes: Yes
  - graphics merged with the text: Yes

9. User comments

- see General User Audit

10. Relations with manufacturer

- Very good (made especially easy by EARN/BITNET).

OTHER TEXTFORMATTERS

It is clear that the market has been continously followed to see if one should move to a different text processing system, or add a second supported system. In this context one should point out that CERN (Computer Science Library) has been a member of the TEX User's Group since 1980. PC packages, with generic markup, have also been looked at.

TEX

TEX is a text formatter, first developed in the late '70s by D.Knuth, aimed at typesetting mathematical text. It was first implemented in the SAIL language. A PASCAL version was written later (and was completed a few years later than scheduled). TEX is in use in e.g. SLAC and DESY (where DCF – IBM's version of Script – is also in use), but notably not in RAL (where Waterloo SCRIPT and DCF are in use).

Strong points of TEX are:

- good mathematics processor
- written in WEB, which translates into Pascal, making it run on a number of different computers

Negative points of TEX include:

- The TEX output is always destined for a typographic (All Points Addressable) quality output device. This means that many printers can not be used to produce the output. In Europe, until a year or two ago, almost the only output devices that could be used were Versatec plotters (very low quality result), impact dot-matrix printers (very low quality result) and phototypesetters (good quality results, but expensive). More recently the appearance of APA laser and LED printers has changed this situation, but in CERN and collaborating labs these are still in the minority.
- TEX uses its own coding scheme for fonts and a large number of fonts are shipped with TEX. They are, however, not tuned to any particular output device — in contrast to fonts provided by the printer manufacturer — and produce very variable quality results. The TEX fonts made "specially"(!?) for the APA6670 are especially bad.
- The basic text formatting capabilities of TEX (i.e. of the text as opposed to the mathematical formulae) are not as developed as in other systems (e.g. SCRIPT). Some examples are:
  - TEX is a single pass system, which leads to problems in cross-references and table of contents (in that these refer to the next previous run and not the latest). The "LaTeX[3] User's Guide and Reference Manual" writes:

    > LaTeX's cross-referencing information is therefore always "old", since it was gathered on a previous execution. This will be noticeable mainly when you are first writing the document – for example, a newly added section won't be listed in the table of contents. However, the last changes you make to your document will normally be minor ones that polish prose rather than add new sections or equations. The cross-referencing information is unlikely to change the last few times you run LaTeX on your file, so all the cross-references will almost always be correct in the final version.

---

3 LaTeX is a popular macro package used on top of TEX.

— TEX has no built-in index capability. For creation of an index you have to write out the entries onto a separate file, then sort the entries and combine the page numbers and finally add the entries as normal text. The "LaTeX User's Guide and Reference Manual" writes:

> Compiling an index or glossary is not easy, but LaTeX can help by writing the necessary information onto a special file. ... As you write your document, you should type an \index command for every page reference you want in the index. When the document is complete except for the index, add the \makeindex command and run LaTeX on the entire document to produce the idx file. You must then process the information in the idx file yourself to create a theindex environment that will generate the index; the **Local Guide** tells if there are any programs available on your computer to help.

— LaTeX appears to have no facility for revision codes.

xROFF

xROFF is a family of text-formatters running under UNIX (with x being N, T, or DI). A number of pre-processors are used for special constructs. One of these – EQN – is used for entering mathematics in a very user-friendly language.

On the CERN UNIX machine some of these formatters are installed, but their use is limited. They can not be used under other operating systems.

WAYS OF ENTERING AND DESCRIBING MATHEMATICS

"Pictorial" approach to generating formulae

"Typewriting" on a grid of half-lines:

This is the method used on most stand-alone wordprocessors and many "PC" packages.

- mono-spaced characters only
- characters of only one size
- often difficult to make modifications and corrections to a typed formula (since there is only some or no linking of the different lines in a formula)
- pattern recognition impossible (or too difficult) to automatically turn into a structural form with all the "invisible brackets" needed for typographic quality printing

"Pick, Scale, and Position":

This scheme has been suggested for some packages on bit-map workstations (e.g. PERQ, APOLLO) and consists of a "menu" of greek characters and special symbols. The user (typist) picks up a symbol and places it in the appropriate place, building up the formula character by character.

- very tedious
- requires a lot of skill to make aesthetic formulae
- almost impossible to make modifications and corrections without re-doing most of the work

- easy to learn for a person who does not know the meaning of the formula

"Languages" describing formulae

Over the last decade several schemes for a "language" to describe mathematical formulae have been proposed and implemented. The best known schemes include:

"EQN"

- "spoken English" rendering of the formulae
- easy to learn and remember by people familiar with the meaning of mathematical formulae
- syntax implemented in many systems – e.g. xROFF, Waterloo SCRIPT, YMFL, DCF, ICEF2.

TEX

- geometric approach to the description
- many people complain about the "obscurity" of the notation and that typists do not find it easy to use
- public domain software running on a multitude of hosts, i.e. attractive for academic environments
- requiring an All Points Addressable output device with good font flexibility

SGML application. Based on ISO/TC97/SC18/WG8/N336.

- almost "spoken English" rendering of the formulae
- easy to learn and remember by people familiar with the meaning of the formulae
- very logical and consistent

First example:

$$\sum_{i=1}^{\infty}$$

"EQN": `sum from i=1 to infinity`

TEX: `\sum\downarrow{i=1}\uparrow\infty`

"SGML" minimimized form: `<SUM><FROM>i=1<TO>&infin;</SUM>`

Comments on first example:

"EQN":

- formula as read out verbally
- more complex limits require "invisible" brackets to be inserted

TEX:

- clear for non-mathematicians due to its "geometric" approach

"SGML":

- formula almost as read out verbally
- clear "bracketing" of the terms

Second example:

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

```
"EQN": a over b+c

TEX: a \over b+c

"SGML" minimized form: <FRAC>a<OVER>b+c</FRAC>
```

Comments on second example:

"EQN":

- operation is in the middle of the expression

TEX:

- operation is in the middle of the expression

"SGML":

- operation at the start of the expression

Third example:

$$\sin x$$

```
"EQN": sin x

TEX: \sin x

"SGML": &sin; x
```

- in "EQN" a list of standard functions to be set in normal roman is included (plus, naturally, a mechanism to over-rule this)
- in the other schemes the function is specified by a special reference

Fourth example:

$$\begin{matrix} 12 & 0 \\ 144 & -1 \end{matrix}$$

```
"EQN": matrix < rcol < 12 above 144 >
     rcol < 0 above -1 > >

TEX: \rpile { 12 \cr 144 \cr } \quad
     \rpile { 0 \cr -1 \cr }
```

```
"SGML" minimized form: <MATRIX><RCOL>12<ABOVE>144</RCOL>
                                <RCOL>0<ABOVE>-1</RCOL></MATRIX>
```

Comments on fourth example:

"EQN":

- quite clear and intelligible

TEX:

- somewhat confusing

"SGML":

- quite clear and intelligible

The system I would like to see

Design goals:

- Immediate feedback: the formula should appear as it is typed in, with correct fonts, with correct size (sometimes rescaling "on the fly").
- Assisted typing by automatic cursor positioning: say that you want to enter a sum. After defining it to be a sum the cursor should be positioned at the lower limit, a "tab" would move it to the place for the upper limit, and a further "tab" move it to the expression for the sum.
- Structure of the formula kept so modifications can be made – some problems, however, with deletions.
- Typographic as well as "typewriter" quality output

Implementations:

- Xerox 6085 has an embryonic system along these lines
- EdiMath, sold by Italsoft in Paris, implements a very nice system along these lines on a Macintosh. It is, however, stand-alone rather than integrated into (somebody elses) text formatter.

PRINTER DATA STREAMS

By printer data stream is mean the format of the file describing the formatted page that is generated as output by a text formatter. This file, containing the text plus printer dependent control codes, is sent to the output device with printed pages being the result.

In the last several years a number of page description languages have emerged. These are of a very different nature and it is in general much more difficult to convert from on of these languages to another than implementing a driver in a text formatter to produce output in the desired page description language. As mentioned above, however, text interchange is much better made at the level of marked up text than in any page description language. One should leave these to just be the data format used for the communication between the formatter and the printer.

The more wide spread page description languages include:

- CPDS: created and used by IBM
- Interpress: created and used by Xerox
- Impress: created and used by Imagen
- PostScript: created by Adobe Systems and used in e.g. Apple Laserwriter

From a "computer science" point of view PostScript is clearly the most interesting and neat one. It is, however, in my view an overkill in that it is really a programming language and much of the work the text formatter has already done is redone in the printer and the interpretation speed is not high on most printers.

It is not at all clear if all these page description languages will survive in the market place. As mentioned above there is also an ISO Standard being developed in this area.