Linköping university
Department of Computer and Information Science
EMTEK unit

PM 1996:12
1996-10-27

Erik Sandewall

The Abbreviated Markup Language, AML

This memo is primarily intended as a reference for users of the ADL input notation. It will also be used as working materials by the participants in the AIMS project, and for the information of the members of IT-rådet.

Related information can be obtained via the following WWW pages:

Linköping university:	http://www.liu.se/
Department of CIS:	http://www.ida.liu.se/
EMTEK group memos:	http://www.ida.liu.se/~erisa/org/spai/
This article:	http://www.ida.liu.se/~erisa/org/spai/ntd96/12/
The author:	http://www.ida.liu.se/~erisa/

To be completed.

1 Background

1.1 Purpose

AML (Abbreviated Markup Language) is a simple coding scheme for so-called markup of texts which occur as the *main contents* of *stereotypical* documents. “Markup” means that it is used to indicate which parts of the text are to occur in italics or boldface, what are section and subsection headings, etc. Being a very simple “language” or (more properly speaking) coding scheme, it is not adequate for defining brochures or other materials where layout is important. It is intended for office memos, letters, and other similar documents which consist only of text. Moreover, it assumes that the general appearance of the finished document, the design of the front page or letterhead, etc. are defined separately as a “style”.

A text which has been prepared using AML must therefore be combined with (1) a “style” or appearance definition, and (2) a document descriptor indicating the information that is to go into front page and headings (such as the author, the title, and the serial number if any); this information together is fed into a *document preparation system* in order to produce the finished document. By using several alternative styles as well as several document preparation systems, one can obtain different variants of the document as desired.

AML has been defined within the NTD project (Network Transportable Documents) at Linköping university, and its first uses will be for a prototype project involving meeting-related documents as well as for submissions to the Linköping University Electronic Press. Two document preparation systems are provided at present: *Latex* which is used for preparing printer-quality appearance, and an HTML generator developed within the project. Connections to additional document preparation systems (DPS) are foreseen. However, the author of the original text does not have to worry about the choice of DPS unless special appearance effects are desired.

It is expected that additional uses of AML will be found within the broader AIMS initiative (Academic Information Management System). In that context, it is intended to complement AML with a more powerful notation tentatively called CML (Core Markup Language), which will provide a broader range of facilities. However, by its very nature, CML will be a computer-internal notation which will not be suitable as input notation, so we expect that AML will continue to be used for input. Automatic transformation from AML to CML will be provided, so documents prepared using AML will be usable in the broader framework as well.

An additional usage of AML will be in the WWDB (World-Wide Data Base) project which is another part of the AIMS initiative. In that case, a limited part of AML will be used for short pieces of text which are stored as properties of objects in the WWDB database.

1.2 Design considerations

The main design considerations for AML are that it should contain the most frequent structures used in common texts, it must have simple syntax and semantics, and it should be easy to read and write so that it is inviting to use it directly.

A file in AML format is an ASCII file containing ordinary text in free format (blanks and carriage returns do not count), but where special significance is given to AML commands. All AML commands begin with a < character; they end with a matching > character, a blank space (in some cases), or another special character such as a full stop (.) or a double quote (").

AML commands serve the following purposes:

Character codes are used for indicating characters with accents, umlauts, and other so-called national characters. For example, ä can be written as <a", ç can be written as <c, etc. Only two of these codes are obligatory, namely writing <lt> if a < character is required in the final document, and similarly <gt> for a > character. (This is inorder to avoid confusion with an AML command). All other AML character codes are optional, but they may be useful whenever a particular character is missing on the keyboard. The full list of character codes is specified below.

Style commands specify that a particular part of the text is to be produced in italics, boldface, etc. On purpose, only the most basic choices of such styles are available.

Header commands specify that a particular part of the text is to appear as a section heading on either of three levels: section, subsection, and subsubsection.

Item-list commands specify that a sequence of short parts of text are to be presented "itemized" or as an enumeration.

Inclusion and escape commands: Inclusion commands specify that the contents of another text file are to be included in the present document. Escape commands are used if some part of the intended document can not be properly expressed in AML, and allows the user to include a piece of text in the more extensive notation required by the document preparation systems, for example, text in Latex or in SGML format. Combined inclusion/escape commands allow the inclusion of a separate file entirely written in the foreign notation.

If the same AML text is to be used in more than one document preparation system, then its escape-coded parts have to be re-written for each of those systems. Plain documents without any special complications should not need to use escape commands.

We now proceed to define the commands in each of these groups.

2 Specification of AML commands

2.1 Character codes

The following character codes are defined. Recall that <lt> and <gt> are the only obligatory ones; for all other characters one can use the character itself if it is available on the keyboard.

<a" or <a: for ä
<a. for å
<o. for ȯ (for all letters except A)
<a´ for á
<a` for à
<a^ for â
<a~ for ã
<c, for ç
<o/ for ø (used in Danish and Norwegian)
<l~ for ł (used in Polish)
<a- for ā (for all letters except L)
<ae> and <oe> for æ and œ

All the codes mentioned above may also be used with capital letters, for example <AE> for Æ. Furthermore we have:

<s/ for German ß (sharp s)
<? and <! for ¿ and ¡ (used in Spanish)
<c/ for ©
<L- for £
<Y- for Y
<+| for †
<S| for $
<c| for c
</ for * (elevated)
<minus> for –
<slash> for /
<bslash> for \
<mid> for |
<amp> for &
<perc> for %
<pnd> for #
<lcurl> for {
<rcurl> for }
<usco> for _
<acute> for ´
<grave> for `
<circ> for ^
<tilde> for ~
<lt> is used when you really want a < sign in the result text
<gt> similarly for a > sign

2.2 Style commands

The following commands are available:

<e text> marks the text as italic (emphasized)

<b text> marks the text as boldface

<t text> marks the text as fixed-width ("teletype")

<sa text> marks the text as being in sans-serif font.

Sans-serif is represented as italic boldface in HTML pages.

2.3 Header commands

The following commands are available:

<s text> specifies the beginning of a section, with the indicated text as the section heading.

<ss text> specifies the beginning of a subsection, with the indicated text as the subsection heading.

<sss text> specifies the beginning of a subsubsection, with the indicated text as the subsubsection heading.

In a corresponding HTML page, these will normally correspond to levels H2, H3, and H4, respectively, whereas the H1 level will correspond to the title of the whole page and is specified in the document descriptor. In Latex, the correspond to the *section*, *subsection*, and *subsubsection* commands, respectively.

The following are sectioning commands although not properly speaking header commands:

<np

requires a new page to be taken. (In HTML it is recommended to represent this by a fat horizontal line containing a page number and a possibility to click in order to proceed to the next "screen page" within the present "HTML page".

<> specifies that a new line is to begin in the produced text.

2.4 Item-list commands

These are commands for defining enumerations of items, corresponding to ul, ol, etc in HTML and to blocks of itemize, enumerate, etc. in Latex:

<bi

<i ...

<i ...

...

<ei

This construct defines an itemization where each item is marked by a special character such as a "bullet". Each individual item consists of text which only ends at the occurrence of the next <i or of a <ei command. Note that no > is put after these commands.

Similarly, if it is desired that the items are enumerated from 1 and up, one uses be and ee instead of bi and ei. Finally, if the enumeration is to contain a sequence of named items, one writes

<bd
<di *text* > ...
<di *text* > ...
...
<ed
where the *text* contained inside the angle brackets is short and is used as the thing to be described.

2.5 Inclusion and escape commands

There are a number of commands for inclusion of information in separate files and/or in source codings.

<inc name> includes the CML coded file name.cml in the present document

<ins name> includes the source coded file name.xxx in the generated document, where xxx may be e.g. tex or frm. Thus, if one is going to generate several versions of the present transportable document in order to go with Latex, Frame, etc., one has to write several versions of the name file.

<tex text> includes *text* in the generated file if Latex code is being generated, and nothing otherwise.

<frm text> includes *text* in the generated file if Frame code is being generated, and nothing otherwise.

<html> *text* <end> includes *text* in the generated file if HTML code is being generated, and nothing otherwise.

Other analogous commands will of course be introduced for other formatting systems at the time that they will be supported.

<$ *expression* $ allows one to write a formula *expression* in the Latex formula language

<verb> Begin verbatim, that is, the following lines are copied literally into the result document (generating a verbatim environment in Latex and a pre segment in HTML) until the occurrence of a line containing only \end starting in the first column. No commands are allowed within the verbatim scope. (More precisely, if they occur there, then the effects may differ between target systems).

2.6 Facilitation of diagnosis

The programs operating on ADL files attempt to identify and explain any violations of the specified syntax. The following conventions are recommended in order to facilitate good diagnoses: All AML commands that were defined in sections 2.3 to 2.5 above should start on a new line, that is, they should have their initial < in the first column of a line. They may continue over several lines, but if they do, the first column of a succeeding line must not contain a < character.

If these rules are not followed, then a warning message will be given, but otherwise nothing will happen as long as the file is correct.

3 Implementation considerations

Programs operating on AML format assume those file to have the extension aml.

The simple structure of the AML notation means that a scanner for AML-coded text can be implemented by looking for any occurrence of $<$ or $>$; in their absence all characters can be copied directly from input to output. Of course, each time that a $<$ is encountered, the resulting action depends on the next following characters. Each time a $>$ is encountered, the resulting effect depends on the current context, that is, on what has been read before.

It is readily seen that the translation of this AML notation to and from subsets of Word printout, Latex, HTML, etc is very straightforward. For example, the CommonLisp program for translating AML to Latex is only four pages long. Note, however, that for more advanced usage we would prefer the translation to go via CML, which is eventually to be defined. CML should contain the same facilities that have been described here, but for a notation which is chosen in a more systematic way, for example as a subset of SGML.