The "keywords.sty" style file*

Miguel Alabau
LaBRI, Université Bordeaux I (France)
e-mail: Miguel.Alabau@labri.u-bordeaux.fr

June 13, 2005

Abstract

This style file contains a set of definitions that allow keywords to be defined and printed in several convenient ways. For each keyword two definitions may be provided. This is to allow the use of keywords in two contexts (for instance English/French translation of the keywords). At the same time, keyword can be printed in roman, teletype, boldfaced or underlined. The user can define other styles for printing keywords. A set of keywords is provided by default. Of course a user can define new keywords or redefine existing ones without modifying the style file.

Contents

1 Introduction

When typing programs or algorithms, it often happens for a same text to appear as an algoritm (pseudo-code) or as a program. In such cases different fonts are used. For instance, algorithms may appear with keywords italicised or underlined, while the same keywords appear boldfaced in the program text. Moreover algorithms are often typed with keywords belonging to the native language of the writer while program keywords appear usually to be english ones.

The file *keywords.sty* provides simple solutions to both of these problems by giving the user capability to define each keyword in two simultaneaous different manners and by providing different printing styles.

Keywords are commands to be issued in text. Hence this style file is origimally intended to be used in conjunction with the features provided by the *programs.sty* style file [4]. However, it can be used independently.

For instance, let be the following piece of program:

```
\BEGIN
   statements
\END
```

*This file has version number v1.0 dated 95/04/01. The documentation was last revised on 96/01/31.

where \BEGIN and \END are two predefined commands in this style file. If we issue the command \ProgKeywords, then we get:

begin
    statements
end

If we prefer to use the second language, we can issue the \FProgKeywords:

début
    statements
fin

We can also use other typesetting rules, like underlined emphasized fonts (UAlgoKeywords):

begin
    statements
end

It is also possible to define new keywords or to redefine existing ones, by issuing a command like:

```
\NewKeyword{\BEGIN}{this is the beginning}
\NewKeyword{\END}{this is the end}
```

which leads to the following piece of code (default typesetting used is the last one specified, i.e. UAlgoKeywords):

this is the beginning
    statements
this is the end

2 User's Manual

A set of default keywords is provided in this file (see section 3.4). For each keyword, two definitions are provided: the first one in english, and the second one in french (because I am French). The switch between the two languages is performed by global commands wich serve also to define the font used for printing.

2.1 Selecting a printing style

\ProgKeywords
\FProgKeywords
\ttKeywords
\FttKeywords
\AlgoKeywords
\FAlgoKeywords
\NormalKeywords
\FNormalKeywords

The following printing styles are provided:

style	definition 1	definition 2
\bf	\ProgKeywords	\FProgKeywords
\tt	\ttKeywords	\FttKeywords
\em	\AlgoKeywords	\FAlgoKeywords
\rm	\NormalKeywords	\FNormalKeywords

\UAlgoKeywords
\UFAlgoKeywords
\FUAlgoKeywords

There are also two underlined styles:

style	definition 1	definition 2
underlined \em	\UAlgoKeywords	\FUAlgoKeywords

For compatability with previous versions of keywords.sty, the command \UFAlgoKeywords has been defined as a synonym for \FUAlgoKeywords.

2.2 Defining and Re-defining keywords

\NewKeyword

The command

```
\NewKeyword{\WORD}{SENTENCE1}[SENTENCE2]
```

defines the command \WORD to issue the text in SENTENCE1 or in SENTENCE2 when it is typed, according to the style currently in use.

For instance, one could use SENTENCE1 for defining english keywords and SENTENCE2 for defining their french translation.

If "[SENTENCE2]" is omitted, all happens as if the following command had been issued:

```
\NewKeyword{\WORD}{SENTENCE1}[SENTENCE1]
```

The command \NewKeyword serves also to redefine existing keywords.

2.3 Defining a printing style

\DefineKeywordsStyles The command

```
\DefineKeywordsStyles{MODE}{\STYLE}
```

allows for the user to define particular printing modes. Its main effect is to define the commands \MODEs and \FMODEs that will lead to print respectively the SENTENCE1 part or the SENTENCE2 part of the \NewKeyword definitions. For instance, the commands \ProgKeywords and \FProgKeywords have been automatically defined from a \DefineKeywordsStyles{ProgKeyword}{\bf} command.

\DefineUnderlinedKeywordsStyles

For SENTENCE$_i$ to be underlined, it is necessary to issue a

```
\DefineUnderlinedKeywordsStyles{MODE}{\STYLE}
```

command, instead of the "\DefineKeywordsStyles" described above. This is how the \UAlgoKeywords and \FUAlgoKeywords commands have been defined, by issuing a \DefineUnderlinedKeywordsStyles{UAlgoKeyword}{\em} command.

2.4 The Index File

In order for the processing of this file to be complete, an index format file is required. Let us assume that it is named keywords.ist, then the following command must be run and then another compilation of the current file:

```
⟨index⟩
⟨index⟩%% ------------------------------------------------------------
⟨index⟩%% Assuming this file is named "keywords.ist" (after being
⟨index⟩%% generated from "keywords.dtx" by running "latex docstrip"),
⟨index⟩%% the following command will produce a well formated index:
⟨index⟩%%
⟨index⟩%%                   makeindex -s keywords.ist keywords.idx
⟨index⟩%% ------------------------------------------------------------
⟨index⟩
```

Another possibility is to set the environment variable INDEXSTYLE to a directory name where the ".ist" files (index format files) may be found.

A possible index file is given below[1]:

```
⟨index⟩actual '='
⟨index⟩quote '!'
⟨index⟩level '>'
⟨index⟩preamble
⟨index⟩"\n \\begin{theindex} \n \\makeatletter\\scan@allowedfalse\n"
```

[1] It can be generated by invoquing the compilation of "docstrip" with the "index" option.

```
⟨index⟩postamble
⟨index⟩"\n\n \\end{theindex}\n"
⟨index⟩item_x1   "\\efill \n \\subitem "
⟨index⟩item_x2   "\\efill \n \\subsubitem "
⟨index⟩delim_0   "\\pfill "
⟨index⟩delim_1   "\\pfill "
⟨index⟩delim_2   "\\pfill "
⟨index⟩% The next lines will produce some warnings when
⟨index⟩% running Makeindex as they try to cover two different
⟨index⟩% versions of the program:
⟨index⟩lethead_prefix   "{\\bf\\hfil "
⟨index⟩lethead_suffix   "\\hfil}\\nopagebreak\n"
⟨index⟩lethead_flag       1
⟨index⟩heading_prefix   "{\\bf\\hfil "
⟨index⟩heading_suffix   "\\hfil}\\nopagebreak\n"
⟨index⟩headings_flag       1
```

2.5 The Driver File

There is also a driver file, called *programs.drv*, that is included in the distribution. It is devoted to control the latex compilation of the documentation. Its code is given below.

```
⟨*driver⟩
\newif\ifnoprogsfile
\openin1 programs.sty
\ifeof1 \noprogsfiletrue\else\noprogsfilefalse\fi\closein1
\ifnoprogsfile \relax\else
\openin1 keywords.sty
\ifeof1 \noprogsfiletrue\else\noprogsfilefalse\fi\closein1
\fi
\ifnoprogsfile
    \typeout{*****************************************************************}
    \typeout{To get a more complete documentation, you should:}
    \typeout{(1) generate the file 'programs.sty'(see 'programs.dtx'), and}
    \typeout{(2) copy the current file into 'keywords.sty'}
    \typeout{*****************************************************************}
\fi
\ifnoprogsfile
    \documentclass{ltxdoc}
\else
    \documentclass{ltxdoc}
    \usepackage{programs}
    \usepackage{keywords}
\fi
\MakePercentIgnore%
%
\setlength{\textwidth}{31pc}%
\setlength{\textheight}{54pc}%
\setlength{\parindent}{0pt}%
\setlength{\parskip}{2pt plus 1pt minus 1pt}%
\setlength{\oddsidemargin}{8pc}%
\setlength{\marginparwidth}{8pc}%
\setlength{\topmargin}{-2.5pc}%
\setlength{\headsep}{20pt}%
\setlength{\columnsep}{1.5pc}%
\setlength{\columnwidth}{18.75pc}%
%%
```

```
\setcounter{IndexColumns}{2}%
\EnableCrossrefs%
\RecordChanges
\CodelineIndex
%\OldMakeindex     % use if your MakeIndex is pre-v2.9%
\begin{document}%
    \DocInput{keywords.dtx}
\end{document}
⟨/driver⟩
```

2.6 Extracting the documents included in the file keywords.dtx

There are three documents included in the *keywords.dtx* file: the style file (*keywords.sty*), the index style file for printing a cross-referenced document (*keywords.ist*), and the driver file for printing the document: *keywords.drv*.

For file extraction it is necessary to use the `docstrip` utility, which is part of the `doc` distribution [3]. Normally, a file `docstrip.tex` should exist on the LaTeX style files directory. Extraction is performed by typing:

```
latex docstrip
```

This is an interactive program, and the dialogue for generating the style file should be:

```
**********************************************************
* This program converts documented macro-files into fast *
* loadable files by stripping off (nearly) all comments! *
**********************************************************

**************************************************
* First type the extension of your input file(s):  *
\infileext=doc
**************************************************

**************************************************
* Now type the extension of your output file(s)  : *
\outfileext=sty
**************************************************

**************************************************
* Now type the name(s) of option(s) to include   : *
\Options=style
**************************************************

**************************************************
* Finally give the list of input file(s) without   *
* extension seperated by commas if necessary     : *
\filelist=Programs
**************************************************
```

For generating the index file it suffices to rerun the `docstrip` utility and to answer "ist/index" instead of "sty/style" int the above steps 2 and 3.

The three files may be produced in a single pass, by simply latexing the file *keywords.ins* which goes along with the file *keywords.dtx*.

Generation of the documentation is then simply performed as follows (the *keywords.dtx* file includes its own driver):

```
latex keywords.dtx
latex keywords.dtx
latex keywords.dtx
makeindex  -s keywords.ist keywords.idx
latex keywords.dtx
```

```
75 ⟨*style⟩
```

3 Description of Macros

\AlreadyDefined@@Keywords

This macro can be tested by any style file to know if the file "keywords.sty" has been input. But it allows a modular programming style similar to the one used with the C header files. Hence, the first time the "keywords.sty" style file is included all of its body will be included; the second time, the body will not be included.

```
76 \expandafter\ifx\csname AlreadyDefined@@Keywords\endcsname\relax%
77 \expandafter\def\csname AlreadyDefined@@Keywords\endcsname{}%
78 \else\endinput\fi
```

A test for the existence of this macro is performed for compatability with ancient versions of LaTeX.

```
79 \@ifundefined{reset@font}{\global\let\reset@font\relax}{}
```

3.1 Defining and Re-defining keywords

\NewKeyword
\@@newkwrd

The `\NewKeyword` command has three parameters, but the third one is optional. By default it is assumed to be equal to the second one:

```
80 \def\NewKeyword#1#2{\@ifnextchar[{\@@newkwrd{#1}{#2}}{\@@newkwrd{#1}{#2}[#2]}}
```

The `\@@newkwrd` performs the real work. It calls the command `\@@KeywordsCurrentStyle` whose effect is to define the command `\@@kwrd` and then invokes this last command.

```
81 \def\@@newkwrd#1#2[#3]{\def#1{\@@KeywordsCurrentStyle{\@@kwrd}{#2}{#3}\@@kwrd}}
```

3.2 Defining printing styles

\@@TypeStyle

This command is expected to be called with a command name as first parameter. Its effect is to define `#1` as the command that print `#3` with style `#2` :

```
82 \def\@@TypeStyle#1#2#3{\def#1{\mbox{\reset@font#2{}#3\/}}}
```

\DefineKeywordsStyles

The macro `\DefineKeywordsStyles` has two parameters, let them be `toto` and `bf`. Its effect is to define two commands called `\@@toto` and `\F@@toto`. Each of these two new commands has three parameters: the first one must be the name of a command (the keyword to be defined) and the other two must be two texts associated to the keyword. The `\@@toto` command will select the first text while the `\F@@toto` command will select the second text.

```
83 \newif\if@@underline \@@underlinefalse
84 \def\DefineKeywordsStyles#1#2{
85     \if@@underline
86         \@namedef{@@#1}##1##2##3{\@@TypeStyle{##1}{#2}{\underline{##2}}}
87         \@namedef{F@@#1}##1##2##3{\@@TypeStyle{##1}{#2}{\underline{##3}}}
88     \else
89         \@namedef{@@#1}##1##2##3{\@@TypeStyle{##1}{#2}{##2}}
90         \@namedef{F@@#1}##1##2##3{\@@TypeStyle{##1}{#2}{##3}}
91     \fi
```

A boolean switch is used to select underlined fonts. By default non underlined fonts are used, and a reset to non boolean fonts is performed after every definition of keyword:

```
    \@@underlinefalse
```

At last two commands are provided to the user: `\totos` and `\Ftotos` whose effect is to set the command `\@@KeywordsCurrentStyle` respectively to `\@@toto` or `\F@@toto` :

```
    \@namedef{#1s}{\def\@@KeywordsCurrentStyle{\@nameuse{@@#1}}}
    \@namedef{F#1s}{\def\@@KeywordsCurrentStyle{\@nameuse{F@@#1}}}
}
```

By this way when a command

```
\NewKeyword{\WORD}{SENTENCE1}[SENTENCE2]
```

is issued, then `\WORD` is defined to

```
\@@KeywordsCurrentStyle{\@@kwrd}{SENTENCE1}{SENTENCE2}\@@kwrd}
```

Hence, every time the command `\WORD` is issued by the user in the text of its programs, `\@@kwrd` is redefined and invoked under the running definition of `\@@KeywordsCurrentStyle`. This complicated trick ensures that every keyword, even if it is not defined in the style file (e.g. if it is defined in the text typed by the user) will be typed with the correct font selection.

`\DefineUnderlinedKeywordsStyles` This macro serves to switch to underlined fonts:

```
\def\DefineUnderlinedKeywordsStyles#1#2{
    \@@underlinetrue
    \DefineKeywordsStyles{#1}{#2}
}
```

3.3 Predefined printing styles

The commands in the margin are automatically generated (see above and section 2.1) by issuing the following commands:

`\ProgKeywords` `\FProgKeywords`

```
\DefineKeywordsStyles{ProgKeyword}{\bf}
```

`\ttKeywords` `\FttKeywords`

```
\DefineKeywordsStyles{ttKeyword}{\tt}
```

`\AlgoKeywords` `\FAlgoKeywords`

```
\DefineKeywordsStyles{AlgoKeyword}{\em}
```

`\UAlgoKeywords` `\FUAlgoKeywords`

```
\DefineUnderlinedKeywordsStyles{UAlgoKeyword}{\em}
```

`\NormalKeywords` `\FNormalKeywords`

```
\DefineKeywordsStyles{NormalKeyword}{\relax}
```

`\UFAlgoKeywords` This macro is defined for compatability with previous versions of the style:

```
\let\UFAlgoKeywords\FUAlgoKeywords
```

3.4 Predefined keywords

The macros below are sorted alphabetically:

```
⟨style⟩%% DEFAULT KEYWORDS
\NewKeyword{\ABORT}{abort}[avorter]
\NewKeyword{\ABS}{abs}
\NewKeyword{\ABSTRACT}{abstract}[abstrait]
\NewKeyword{\ACCEPT}{accept}[accepter]
\NewKeyword{\ACCESS}{access}[acc\‘es]
\NewKeyword{\ALIASED}{aliased}[alias\’e]
\NewKeyword{\ALL}{all}[tout]
\NewKeyword{\ALT}{alt}
\NewKeyword{\AND}{and}[et]
\NewKeyword{\APPEND}{append}[ajouter\_\-en\_\-fin]
\NewKeyword{\ARRAY}{array}[tableau]
\NewKeyword{\ASSERT}{assert}[assertion]
\NewKeyword{\ASSIGN}{:=}
\NewKeyword{\AT}{at}
\NewKeyword{\BEGIN}{begin}[d\’ebut]
\NewKeyword{\BLOCK}{block}[bloc]
\NewKeyword{\BOOLEAN}{boolean}[bool\’een]
\NewKeyword{\BODY}{body}
\NewKeyword{\BOT}{$\bot$}
\NewKeyword{\BOX}{$<>$}
\NewKeyword{\BY}{by}[pas]
\NewKeyword{\CASE}{case}[choix]
\NewKeyword{\CATINDEX}{catindex}
\NewKeyword{\CHAN}{chan}[canal]
\NewKeyword{\CHANNEL}{channel}[canal]
\NewKeyword{\CHAR}{char}[car]
\NewKeyword{\CHARACTER}{character}[caract\‘ere]
\NewKeyword{\CLOSE}{close}[fermer]
\NewKeyword{\CO}{co}
\NewKeyword{\COBEGIN}{cobegin}
\NewKeyword{\COEND}{coend}
\NewKeyword{\COMMUTATIVE}{commutative}[commutatif]
\NewKeyword{\COMPLEX}{complex}[complexe]
\NewKeyword{\COMPUTE}{compute}[calculer]
\NewKeyword{\CONNECT}{$\longrightarrow$}
\NewKeyword{\CONNECTB}{$\Longrightarrow$}
\NewKeyword{\CONST}{const}
\NewKeyword{\CONSTANT}{constant}[constante]
\NewKeyword{\CONSTRAINTS}{constraints}[contraintes]
\NewKeyword{\CONTINUE}{continue}
\NewKeyword{\DATA}{data}[donn\’ee]
\NewKeyword{\DECLARE}{declare}
\NewKeyword{\DECOMPOSE}{decompose}
\NewKeyword{\DELAY}{delay}[d\’elai]
\NewKeyword{\DELTA}{delta}
\NewKeyword{\DEPTH}{depth}[profondeur]
\NewKeyword{\DIGITS}{digits}[chiffres]
\NewKeyword{\DIMENSION}{dimension}
\NewKeyword{\DIST}{dist}
\NewKeyword{\DISTRIBUTE}{distribute}[r\’epartir]
\NewKeyword{\DIV}{div}
\NewKeyword{\DO}{do}[faire]
\NewKeyword{\DOALL}{doall}[faire en parall\‘ele]
\NewKeyword{\DOM}{dom}
```

```
\NewKeyword{\DOMAIN}{domain}[domaine]
\NewKeyword{\DOMAINS}{domains}[domaines]
\NewKeyword{\DONE}{done}[fait]
\NewKeyword{\DOPAR}{dopar}[faire en parall\‘ele]
\NewKeyword{\DOWNTO}{downto}[jusqu’\‘a]
\NewKeyword{\DYNAMIC}{dynamic}[dynamique]
\NewKeyword{\EACH}{each}[chaque]
\NewKeyword{\EGO}{MyId}[EGO]
\NewKeyword{\ELSE}{else}[sinon]
\NewKeyword{\ELSIF}{elsif}[sinon si]
\NewKeyword{\END}{end}[fin]
\NewKeyword{\ENDCASE}{end case}[fin choix]
\NewKeyword{\ENDIF}{end if}[finsi]
\NewKeyword{\ENDDO}{end do}[fait]
\NewKeyword{\ENDLOOP}{end loop}[fait]
\NewKeyword{\ENTRY}{entry}[entr\’ee]
\NewKeyword{\EOT}{eot}
\NewKeyword{\EQ}{$=$}
\NewKeyword{\EXCEPTION}{exception}
\NewKeyword{\EXIT}{exit}[sortir]
\NewKeyword{\EXTERNAL}{external}[externe]
\NewKeyword{\FI}{fi}[finsi]
\NewKeyword{\FILE}{file}[fichier]
\NewKeyword{\FIRST}{first}[premier]
\NewKeyword{\FOR}{for}[pour]
\NewKeyword{\FORALL}{forall}[pour tout]
\NewKeyword{\FOREACH}{foreach}[pour chaque]
\NewKeyword{\FORWARD}{forward}
\NewKeyword{\FUNCTION}{function}[fonction]
\NewKeyword{\GE}{$\geq$}
\NewKeyword{\GENERIC}{generic}[g\’en\’erique]
\NewKeyword{\GETNODE}{getnode}[prendre\_\-noeud]
\NewKeyword{\GOTO}{goto}[aller\_\-\‘a]
\NewKeyword{\GRAPH}{graph}[graphe]
\NewKeyword{\GT}{$>$}
\NewKeyword{\IF}{if}[si]
\NewKeyword{\IMPLICATION}{$\Rightarrow$}
\NewKeyword{\IMPLY}{$\Rightarrow$}
\NewKeyword{\IMPORT}{import}[importer]
\NewKeyword{\IN}{in}[dans]
\NewKeyword{\IND}{ind}
\NewKeyword{\INDEX}{index}
\NewKeyword{\INIT}{init}
\NewKeyword{\INOUT}{inout}
\NewKeyword{\INPORT}{inport}
\NewKeyword{\INPUT}{input}
\NewKeyword{\INTEGER}{integer}[entier]
\NewKeyword{\INTO}{into}
\NewKeyword{\IS}{is}[est]
\NewKeyword{\LABEL}{label}[\’etiquette]
\NewKeyword{\LAST}{last}[dernier]
\NewKeyword{\LE}{$\leq$}
\NewKeyword{\LENGTH}{length}[longueur]
\NewKeyword{\LIMITED}{limited}[limit\’e]
\NewKeyword{\LOOP}{loop}[faire]
\NewKeyword{\LT}{$<$}
\NewKeyword{\MAP}{map}[placer]
\NewKeyword{\MOD}{mod}
\NewKeyword{\MODULE}{module}
```

```
\NewKeyword{\MODULO}{modulo}
\NewKeyword{\MULTIPLE}{multiple}
\NewKeyword{\MYID}{MyId}[EGO]
\NewKeyword{\NE}{$\neq$}
\NewKeyword{\NEIGHBOUR}{neighbour}[voisin]
\NewKeyword{\NEIGHBOURS}{neighbours}[voisins]
\NewKeyword{\NEW}{new}[nouveau]
\NewKeyword{\NEWBLOCK}{newblock}
\NewKeyword{\NIL}{nil}
\NewKeyword{\NODE}{node}[noeud]
\NewKeyword{\NOT}{not}[non]
\NewKeyword{\NUL}{nul}
\NewKeyword{\NULL}{null}[nul]
\NewKeyword{\OD}{od}[fait]
\NewKeyword{\ODPAR}{odpar}[fait]
\NewKeyword{\OF}{of}
\NewKeyword{\ON}{on}
\NewKeyword{\OPEN}{open}[ouvrir]
\NewKeyword{\OR}{or}[ou]
\NewKeyword{\OTHERS}{others}
\NewKeyword{\OUT}{out}
\NewKeyword{\OUTPORT}{outport}
\NewKeyword{\OUTPOUT}{outpout}
\NewKeyword{\PACKAGE}{package}[paquetage]
\NewKeyword{\PARALLEL}{parallel}
\NewKeyword{\PARFOR}{parfor}[en parall\‘ele: pour]
\NewKeyword{\PAR}{par}[en parall\‘ele]
\NewKeyword{\PERCENT}{\%}
\NewKeyword{\PLACE}{place}[placer]
\NewKeyword{\PORT}{port}
\NewKeyword{\PRAGMA}{pragma}
\NewKeyword{\PRI}{pri}
\NewKeyword{\PRIVATE}{private}[priv\’e]
\NewKeyword{\PROCEDURE}{procedure}[proc\’edure]
\NewKeyword{\PROCESS}{process}[processus]
\NewKeyword{\PROGRAM}{program}[programme]
\NewKeyword{\PROTECTED}{protected}[prot\’eg\’e]
\NewKeyword{\RAISE}{raise}[lever]
\NewKeyword{\RANGE}{range}[intervalle]
\NewKeyword{\READ}{read}[lire]
\NewKeyword{\READY}{ready}[pr\^et]
\NewKeyword{\REAL}{real}[r\’eel]
\NewKeyword{\RECORD}{record}[enregistrement]
\NewKeyword{\RECV}{recv}[recevoir]
\NewKeyword{\RECEIVE}{receive}[recevoir]
\NewKeyword{\REM}{rem}
\NewKeyword{\RENAMES}{renames}[renomme]
\NewKeyword{\REPEAT}{repeat}[r\’ep\’eter]
\NewKeyword{\REQUEUE}{requeue}
\NewKeyword{\RESET}{reset}
\NewKeyword{\RETURN}{return}[retour]
\NewKeyword{\REVERSE}{reverse}
\NewKeyword{\REWIND}{rewind}
\NewKeyword{\REWRITE}{rewrite}
\NewKeyword{\ROOT}{root}[racine]
\NewKeyword{\SELECT}{select}
\NewKeyword{\SEND}{send}[\’emettre]
\NewKeyword{\SENDEOT}{sendeot}[\’emettre eot]
\NewKeyword{\SEPARATE}{separate}[s\’epar\’ement]
```

```
\NewKeyword{\SEQ}{seq}
\NewKeyword{\SET}{set}
\NewKeyword{\SIZE}{size}[taille]
\NewKeyword{\SKIP}{skip}[sauter]
\NewKeyword{\STRING}{string}[cha\^{\i}ne de caract\‘eres]
\NewKeyword{\SUBTYPE}{subtype}[sous\_\-type]
\NewKeyword{\SWITCH}{switch}
\NewKeyword{\TAGGED}{tagged}[\’etiquett\’e]
\NewKeyword{\TASK}{task}[t\^ache]
\NewKeyword{\TERMINATE}{terminate}[terminer]
\NewKeyword{\THEN}{then}[alors]
\NewKeyword{\TO}{to}[jusqu’\‘a]
\NewKeyword{\TOWARDS}{towards}[vers]
\NewKeyword{\TRANSMIT}{transmit}[\’emettre]
\NewKeyword{\TUPLE}{tuple}[n\_\-uplet]
\NewKeyword{\TYPE}{type}
\NewKeyword{\UNDEF}{undef}[ind\’efini]
\NewKeyword{\UNTIL}{until}[jusqu’\‘a]
\NewKeyword{\USE}{use}
\NewKeyword{\VAR}{var}
\NewKeyword{\VARIABLE}{variable}
\NewKeyword{\WHEN}{when}[si]
\NewKeyword{\WHERE}{where}[si]
\NewKeyword{\WHILE}{while}[tant que]
\NewKeyword{\WITH}{with}[avec]
\NewKeyword{\WRITE}{write}[\’ecrire]
\NewKeyword{\XOR}{xor}
⟨style⟩%%
⟨style⟩%% French syntax
⟨style⟩%%
\NewKeyword{\EMETTRE}{send}[\’emettre]
\NewKeyword{\RECEVOIR}{receive}[recevoir]
\NewKeyword{\POUR}{for}[pour]
\NewKeyword{\FAIRE}{do}[faire]
\NewKeyword{\FAIT}{end do}[fait]
\NewKeyword{\SI}{if}[si]
\NewKeyword{\ALORS}{then}[alors]
\NewKeyword{\SINON}{else}[sinon]
\NewKeyword{\FINSI}{end if}[fin si]
\NewKeyword{\DEBUT}{begin}[d\’ebut]
\NewKeyword{\FIN}{end}[fin]
```

Then we terminate by instructing LaTeX to switch to the default font for typing keywords (which, in the current implementation is underlined `\em`).

```
\FUAlgoKeywords
```

⟨/style⟩

References

[1] D.E. Knuth. Computers & Typesetting (The TeXbook). Addison-Wesley, Vol. A, 1986.

[2] L. Lamport. *LaTeX: a Document Preparation System.* Addison-Wesley Publishing Company, 1986.

[3] F. Mittelbach. The doc-option. *TUGboat*, Vol. 10(2), *pp.* 245–273, July 1989.

[4] M. Alabau. The "programs.sty" style file. March 1995. *e-mail:* Miguel.Alabau@labri.u-bordeaux.fr