LaTeX and Friends
Commands and Environments

http://csweb.ucc.ie/~dongen/LAF/LAF.html

M. R. C. van Dongen

UCC

Advantages of Automation

SE Tedious tasks can be automated.

- **reusability** Define once, use many times.
- **simplicity** Easier to use. Avoids errors.
- **refinement** Allows stepwise refinement.
- **maintenance** Make local changes with global effect.
- **consistency** Guarantees consistent typesetting.

computing Tasks and results are controlled by document options.

- **style control** Different style for different options.
- **content control** Commands may give different output.
- **typeset results** Do arithmetic and branching, and
  - typeset *results* of computations.

Disadvantages of LaTeX Commands

namespace No local identifiers.

parameters Two problems related to the parameters of the commands.

- No more than 9 parameters.
- Formal parameters are numbers.

Defining Commands

`\newcommand⟨cmd⟩{⟨subst⟩}`

- Defines ⟨cmd⟩.
- Using ⟨cmd⟩ results in ⟨subst⟩.

`\renewcommand⟨cmd⟩{⟨subst⟩}`

- Redefines command.

Example

LaTeX Usage

```
\documentclass{article}
\newcommand\CTAN{Comprehensive \TeX{} Archive Network}
\begin{document}
  I always download my packages from the \CTAN.
  The \CTAN{} is the place to be.
\end{document}
```

LaTeX Output

I always download my packages from the Comprehensive TeX Archive Network. The Comprehensive TeX Archive Network is the place to be.

How did that Work in Math?

$$f : \mathbb{N} \to \mathbb{N}$$
$$x \mapsto x^2 + 2x\,.$$

- The x is the *formal* parameter of f.
- The expression $x^2 + 2x$ defines the computation.
  - In computer science terminology, this is called the *substitution text.*
- You write $f(\langle \texttt{expr} \rangle)$ to apply f to the *actual parameter* $\langle \texttt{expr} \rangle$.
- To evaluate $f(\langle \texttt{expr} \rangle)$ you substitute the actual parameter $\langle \texttt{expr} \rangle$ for each occurrence of the formal parameter x in the substitution text of f.

Commands with Parameters (No Options)

\newcommand⟨cmd⟩[⟨digit⟩]{⟨subst⟩}

- Defines ⟨cmd⟩.
  - Command takes ⟨digit⟩ parameters (1–9).
  - The *i*th formal parameter is denoted #*i* in ⟨subst⟩.
  - Actual parameter #*i* is substituted for #*i* in ⟨subst⟩.

\renewcommand⟨cmd⟩[⟨digit⟩]{⟨subst⟩}

- Redefines ⟨cmd⟩.

Commands with Parameters (Example)

LaTeX Usage

```
\newcommand\opening[1]{%
    Dear #1,%
}

\begin{document}
  \opening{Mum},\\[2\baselineskip]
  {\LaTeX} is going great in~2013.
  We're studying user-defined macros now.
\end{document}
```

LaTeX Output

Dear Mum,

LaTeX is going great in 2013. We're studying user-defined macros now.

Parameters and Options

\newcommand⟨cmd⟩[⟨digit⟩][⟨default⟩]{⟨subst⟩}

- Defines ⟨cmd⟩.
  - Command takes ⟨digit⟩ parameters.
  - One parameter is optional.
  - Optional parameter is enclosed in square brackets.
  - Without optional parameter #1 is assigned ⟨default⟩.

\renewcommand⟨cmd⟩[⟨digit⟩][⟨default⟩]{⟨subst⟩}

- Redefines existing command.

Parameters and Options (Example)

LaTeX Usage

```
\newcommand\congratulations[2][a teddy bear]{%
    Congratulations #2. You've won #1.
}

\begin{document}
  \congratulations{John}

  \congratulations[a train set]{Luke}
\end{document}
```

LaTeX Output

Congratulations John. You've won a teddy bear.
Congratulations Luke. You've won a train set.

Fragile and Robust Commands

- *Moving parameters* are saved to be reread later on.
- Examples: parameters that are written to auxiliary files.
- Moving parameters are expanded before they are saved.
- Sometimes expansion leads to invalid TEX.
- Command is *robust* if it expands to valid TEX.
- Otherwise it's called *fragile*.
- The command `\protect` protects commands against expansion.
- Saving `\protect\cmd` saves `\cmd` without expanding.
- Protects fragile commands in moving arguments.

Defining Robust Commands

```
\DeclareRobustCommand⟨cmd⟩{⟨subst⟩}
\DeclareRobustCommand⟨cmd⟩[⟨digit⟩]{⟨subst⟩}
\DeclareRobustCommand⟨cmd⟩[⟨digit⟩][⟨default⟩]{⟨subst⟩}
\MakeRobustCommand⟨cmd⟩
```

Kinds of Tokens

character token	Corresponds to a single character (not \).
control sequence token	Correspond to a command.

Kinds of Parameters

primitive parameter Single character or control sequence token.
- No opening and closing brace token.

compound parameter Brace-delimited group.

Evaluating *n* Parameter Macro Call

- Remove macro and *n* actual parameters from the token stream.
- Carry out parameter substitution for *i* from 1 to *n*.
  - Substitution is done in substitution text:
    - Substitute *i*th actual parameter for *i*th formal parameter.
    - For compound parameter, remove the outermost brace pair.
- Puts the resulting expression in front of the token stream.

Example

LaTeX Usage

```
\newcommand\swop[2]{#2#1}
\newcommand\SWOP[2]{#2#1}
```

- \swop2\SWOP31

Example

LaTeX Usage

```
\newcommand\swop[2]{#2#1}
\newcommand\SWOP[2]{#2#1}
```

- `\swop2\SWOP31` $\mapsto$ `\SWOP231`

Example

LaTeX Usage

```
\newcommand\swop[2]{#2#1}
\newcommand\SWOP[2]{#2#1}
```

- `\swop2\SWOP31` ↦ `\SWOP231` ↦ 321

Combinatory Logic

LaTeX Input

```
\documentclass{article}

\newcommand\K[2]{#1}
\newcommand\S[3]{#1#3{#2#3}}
\newcommand\I{\S\K\K}
\newcommand\X{\S{\K{\S\I}}{\S{\K\K}\I}}

\begin{document}
   \X abc
\end{document}
```

Combinatory Logic (Continued)

```
 1          \X_1a_1b_1c_1
 2          \S_1 {_1\K_2{_2\S_3\I_3}_2}_1 {_1\S_2{_2\K_3\K_3}_2\I_2}_1 a_1b_1c_1
 3          \K_1{_1\S_2\I_2}_1a_1{_1\S_2{_2\K_3\K_3}_2\I_2a_2}_1b_1c_1
 4          \S_1\I_1{_1\S_2{_2\K_3\K_3}_2\I_2a_2}_1b_1c_1
 5          \I_1b_1{_1\S_2{_2\K_3\K_3}_2\I_2a_2b_2}_1c_1
 6          \S_1\K_1\K_1b_1{_1\S_2{_2\K_3\K_3}_2\I_2a_2b_2}_1c_1
 7          \K_1b_1{_1\K_2b_2}_1{_1\S_2{_2\K_3\K_3}_2\I_2a_2b_2}_1c_1
 8          b_1{_1\S_2{_2\K_3\K_3}_2\I_2a_2b_2}_1c_1
 9      b   {_1\S_2{_2\K_3\K_3}_2\I_2a_2b_2}_1c_1
10      b   \S_2{_2\K_3\K_3}_2\I_2a_2b_2}_1c_1
11      b   \K_2\K_2a_2{_2\I_3a_3}_2b_2}_1c_1
12      b   \K_2{_2\I_3a_3}_2b_2}_1c_1
13      b   \I_2a_2}_1c_1
14      b   \S_2\K_2\K_2a_2}_1c_1
15      b   \K_2a_2{_2\K_3a_3}_2}_1c_1
16      b   a_2}_1c_1
17     ba   }_1c_1
18     ba   c_1
      bac
```

TEX Commands Without Delimiters

\def⟨cmd⟩#1#2...#*n*{⟨subst⟩}

- Defines command ⟨cmd⟩ with *n* parameters.
- The substituion text of ⟨cmd⟩ is ⟨subst⟩.

\edef⟨cmd⟩#1#2...#*n*{⟨subst⟩}

- Defines command ⟨cmd⟩ with *n* parameters.
- The substitution text of ⟨cmd⟩ is the *expansion* of ⟨subst⟩.

TeX Commands Without Delimiters (Example)

LaTeX Input

```
\def\hi{hi}

\def\hello{\hi}
\edef\ehello{\hi}

\def\hi{HI}

\ehello. \hello.
```

LaTeX Output

hi. HI.

Local Macro Definitions

- For each level of local TEX macro definitions you double the number of #s.

LATEX Usage

```
\def\topLevelCommand#1{%
    \def\lowLevelCommand##1{%
        Top level: #1. Low level: ##1.%
    }
    \lowLevelCommand{LOW}%
}

\topLevelCommand{HIGH}
```

LATEX Output

Top level: HIGH. Low level: LOW.

Low-level TEX Commands

`\csname␣⟨tokens⟩\endcsname`
: Expands ⟨tokens⟩ and turns it into a control sequence.

`\noexpand⟨token⟩`
: Returns ⟨token⟩ without expanding it.

`\expandafter⟨token⟩⟨tokens⟩`
: Expands first token in ⟨tokens⟩ (once).

Low-level TeX Command (Example)

LaTeX Usage

```
\def\property#1{%
    % ``\def\#1##1{##1 is #1}''
    \expandafter\def\csname#1\endcsname##1{%
        ##1\ is #1%
    }%
}
\property{brilliant}
\property{excellent}
...
\excellent{\TeX} and
\brilliant{\LaTeX}.
```

LaTeX Output

TeX is excellent and LaTeX is brilliant.

Definitions with Delimiters

LaTeX Usage

```
\def\command|#1|#2|{...}
```

Example

LaTeX Usage

```
% allow @ in macro names
\makeatletter%
\def\cmd#1{%
  \@ifnextchar[%
    % use the given option
    {\cmd@relay{#1}}%
    % use the default option
    {\cmd@relay{#1}[dflt]}%
}
\def\cmd@relay#1[#2]{...}
% disallow @ in macro names
\makeatother
```

LaTeX Usage

```
\makeatletter
\def\cmd#1{%
  \def\cmd@relay##1[##2]{...}
  \@ifnextchar[%
    {\cmd@relay{#1}}%
    {\cmd@relay{#1}[dflt]}%
}
\makeatother
```

The \let Command

LaTeX Usage

```
\makeatletter
% Save meaning of old \section command.
\let\old@section=\section
\def\section#1#2{%
  % Define section using old \section command.
  \old@section{#2}
  % Define label for the section.
  \label{#1}
}
\makeatother
```

More than Nine Parameters

LaTeX Usage

```
\makeatletter
\def\cmd#1#2#3#4#5#6#7#8#9{%
  \def\cmd@arg@A{#1}%
  \def\cmd@arg@B{#2}%
  .
  .
  .
  \def\cmd@arg@I{#9}%
  \relay%
}
\def\relay#1{%
  Parameters: \cmd@arg@A, \cmd@arg@B, ..., and #1.%
}
\makeatother
```

More than Nine Parameters (Continued)

LaTeX Usage

```
\def\cmd#1#2#3#4#5#6#7#8#9{%
  \def\relay##1{Parameters: #1, #2, ..., and ##1.}%
  \relay%
}
```

Advantages of Environments

less ambiguity	If nested, makes it easier to read.
higher efficiency	Reduces need for extra stack space.

User-defined Environments

LaTeX Usage

```
\newenvironment{SectionalUnit}[2][section]
                {\csname#1\endcsname{#2}%
                 \begin{refsection}}
                {\printbibliography%
                 \end{refsection}}

\begin{document}
  \begin{SectionalUnit}[chapter]{Introduction}
    \begin{SectionalUnit}{Conventions}
        ...
    \end{SectionalUnit}
    \begin{SectionalUnit}{Notation}
        ...
    \end{SectionalUnit}
  \end{SectionalUnit}
  ⋮
\end{document}
```

Environment Definitions

```
\newenvironment{⟨name⟩}{⟨begin subst⟩}{⟨end subst⟩}
\newenvironment{⟨name⟩}[⟨digit⟩]{⟨begin subst⟩}{⟨end subst⟩}
\newenvironment{⟨name⟩}[⟨digit⟩][⟨default⟩]{⟨begin subst⟩}{⟨end subst⟩}
```

Bibliography

Acronyms and Abbreviations

AMS	American Mathematical Society
API	Application Programming Interface
APL	A Programming Language
CTAN	Comprehensive TEX Archive Network
CD	Compact Disk
FAQ	Frequently Asked Question
GUI	Graphical User Interface
IDE	Integrated Development Environment
ISBN	International Standard Book Number
OS	Operating System
SI	Système International d'Unités/International System of Units
TUG	TEX Users Group
URL	Uniform Resource Locator
WYSIWYG	What You See Is What You Get

About this Document

- This document was created with `pdflatex`.
- The LaTeX document class is `beamer`.