Corso Intensivo di LaTeX

Terza Giornata

Autore e speaker: Luca Tesei

Corso offerto dal Camerino Linux User Group

http:\\www.camelug.org

Tabelle

- Abbiamo visto l'ambiente **`array`** in math mode
- Un analogo in paragraph mode è l'ambiente **`tabular`**
- Funziona esattamente come **`array`**
- La differenza è che il contenuto delle “caselle” viene interpretato in paragraph mode invece che in math mode

Altri comandi per le tabelle/array

- \hline traccia una linea orizzontale che taglia tutte le colonne
- \cline{i-j} traccia una linea orizzontale che taglia dalla colonna i alla colonna j
- \cline può essere inserito nello stesso punto in cui può essere inserito \hline
- Si possono mettere più \cline consecutivi per lasciare dei “vuoti” nella riga orizzontale

Altri comandi per le tabelle/array

- A volte una certa casella di una tabella deve occupare il posto di più colonne
- Per far questo c'è il comando

`\multicolumn{`*`n`*`}{`*`allineamento`*`}{`*`contenuto`*`}`

- La casella generata prenderà lo spazio di *n* colonne, sarà centrata come la lettera in *allineamento* (`c`, `l` o `r` con eventuali `|`) e conterrà *contenuto*
- Il comando annulla le linee verticali ai lati, che quindi vanno reinserite in *allineamento*

Altri comandi: esempio

```
\begin{tabular}{|l||r|r|}
\hline \hline
Nome e Cognome & \multicolumn{2}{c|}{Voti} \\
\hline
Paolino Paperino & 18 & 24\\
\cline{2-3}
Paperon De Paperoni & 28 & 30\\
\hline
Medie & 23 & 27\\
\hline \hline
\end{tabular}
```

tabbing

- L'ambiente **`tabbing`** è una variante delle tabelle
- Il suo comportamento ricorda le vecchie macchine da scrivere meccaniche
- Si possono fissare dei **Tab Stop** su certe “colonne”
- In ogni riga si può quindi far scorrere il carrello tante volte quante servono per allineare il testo alla “colonna” giusta

tabbing: comandi interni

- `\=` inserisce un Tab Stop nel punto della riga in cui compare
- `\>` posiziona il testo che lo segue allineandolo rispetto al successivo Tab Stop, anche se si trova indietro!
- `\\` va a capo. Le righe vuote non contano
- `\kill` registra tutti i Tab Stop della riga, ma non produce nessun output
- All'interno vale sempre il principio del blank (una serie di spazi è contato come uno)

tabbing: esempio

```
\begin{tabbing}
mmmm\=mmmm\=mmmm\=mmmm\= \kill \\
1 \> 2 \> 3 \> 4\\
\> \> tre % Non va a capo

\\ % In tabbing si può andare a capo solo con \\

\> \> \> quattro\\
\end{tabbing}
```

E la struttura logica?

- Abbiamo visto molti comandi che ci consentono di operare sulla formattazione del documento
- Ma la premessa era che una cosa di questo genere non ci doveva interessare!
- Chiariamo meglio

L'arcano

- Finché i comandi e gli ambienti standard di LaTeX ci bastano allora non dobbiamo preoccuparci di niente
- Ma i comandi e gli ambienti standard non possono coprire tutti i concetti logici relativi a tutti i possibili documenti
- Quando il nostro documento contiene un qualcosa che non è previsto negli standard allora possiamo definire noi un comando o un ambiente per rappresentarlo

Aumentiamo il LaTeX

- Possiamo quindi definire dei nuovi comandi e dei nuovi ambienti
- La definizione crea un nuovo concetto logico che potrà essere inserito nella struttura del documento
- Oltre alla definizione logica del concetto di solito vogliamo anche definire come questo verrà evidenziato nel documento
- È qui che ci vengono in aiuto tutti i comandi e gli ambienti che abbiamo visto!

\newcommand

- Permette di definire un comando nuovo
- Permette di definire un certo numero di argomenti per il comando, di cui uno opzionale

```
\newcommand{\nomecom}[narg][argdefault]{testo}
```

- È una dichiarazione e quindi soggetta alle regole che conosciamo per il campo d'azione
- Se inserito nel preambolo vale per tutto il documento (a differenza di altre dichiaraz.)

\newcommand

- Il nome del comando deve essere una sequenza di lettere
- *narg* è un numero intero compreso tra 1 e 9 e indica il numero di argomenti che il comando richiede
- Il numero di argomenti fornito può essere *narg -1* In questo caso viene preso *argdefault* come primo argomento
- Dentro **`testo`** possiamo scrivere tutto quello che vogliamo e inserire gli argomenti con #1, #2, #3, ... #9

\newcommand

- È molto utile per definire la notazione matematica con un nome e degli argomenti
- In questo modo se si vuole cambiare notazione si deve cambiare solo la definizione del comando!
- Analogamente per strutture e sequenze di testo ripetute che rappresentano un aggregato logico nel documento

\newcommand: esempio

```
\newcommand{\grazie}[2][Dott]{Si ringrazia
    il {\em #1.\/#2} per la collaborazione.}
```

- Il comando **\grazie** produce un ringraziamento personalizzato
- Mette in evidenza il nome e il titolo
- Il titolo di default è Dott
- L'argomento obbligatorio è il cognome della persona da ringraziare

\newcommand

- Quando LaTeX incontra il comando nel testo semplicemente:
  - Controlla che siano specificati tutti gli argomenti obbligatori
  - Rimpiazza il comando con il testo della definizione sostituendo a #1, #2, ... gli argomenti specificati
  - #1 rappresenta, se c'è, l'argomento opzionale altrimenti il primo argomento obbligatorio

\newcommand: uso

- `\grazie{Topolino}` produce: “Si ringrazia il *Dott. Topolino* per la collaborazione.”
- `\grazie[Sig]{Paperino}` produce: “Si ringrazia il *Sig. Paperino* per la collaborazione.”

Graffe e Blocchi

- Attenzione: le graffe che racchiudono il **`testo`** in **`\newcommand`** non creano un nuovo blocco!
- Una dichiarazione nel **`testo`** produce i suoi effetti come se fosse inserita nel documento senza un proprio blocco apposito!

\newcommand a più argomenti

- `\newcommand{\appl}[3]{#1 $(#2, #3)$}`
- `\appl{FIRST}{A}{x}` produce FIRST(A,x)
- `\appl{\em FIRST}{A}{x}` produce *FIRST(A,x)* ed estende l'effetto della dichiarazione `\em` fino alla chiusura del blocco successivo!
- Questo perché le graffe che racchiudono gli argomenti non creano nuovi blocchi, che vanno inseriti esplicitamente:
- `\newcommand{\appl}[3]{{#1} $({#2}, {#3})$}`

\newcommand e math mode

- Spesso si definisce un nuovo comando per rappresentare una notazione matematica
- `\newcommand{\fun}[2]{\Gamma(#1,#2)}`
- Se `\fun` viene usato in math mode allora non ci sono problemi
- Se però `\fun` viene usato in paragraph o LR mode allora si genera un errore perché `\Gamma` può essere usato solo in math mode

\newcommand e math mode

- Vorremmo poter definire la notazione matematica in modo che funzioni sia in math mode che in paragraph o LR mode
- Per questo si può usare il comando `\ensuremath{`***`formula`***`}`
- Se viene incontrato in math mode non fa niente e viene interpretato come ***`formula`***
- Altrimenti LaTeX viene forzato ad entrare in math mode, ad interpretare ***`formula`*** e a tornare poi al mode precedente

\ensuremath

- `\newcommand{\fun}[2]`

  `{\ensuremath{\Gamma(#1,#2)}}`
- In questo modo `\fun` funziona in qualsiasi contesto:
- `Sia \fun{a}{b} in $\sqrt[1/a]{\fun{a}{b}}$.`

\renewcommand

- LaTeX dà errore se si prova a definire un comando con un nome già esistente
- Per ridefinire un comando si può usare \renewcommand
- Si usa esattamente come \newcommand
- Da usare con molta cautela se si ridefiniscono comandi standard!

Nuovi ambienti

- Allo stesso modo dei comandi possiamo definire nuovi ambienti
- Il comando è

```
\newenvironment{nome}
                [narg][argdefault]
                  {testo inizio}
                    {testo fine}
```

\newenvironment

- Il nome **non** comincia con `\`
- **`narg`** e **`argdefault`** funzionano allo stesso modo di \newcommand
- Invece di un solo **`testo`** **`\newenvironment`** ha un testo di inizio e uno di fine
- Gli argomenti **`#1`**, **`#2`**, ... possono essere usati sia nel testo di inizio che in quello di fine

Come funziona?

- Nel documento .tex ogni occorrenza di **`\begin{nome}`** viene sostituita con il testo di inizio e di **`\end{nome}`** con il testo di fine
- Tutto ciò che sta in mezzo viene interpretato così com'è
- Le sostituzioni degli argomenti avvengono esattamente come per **`\newcommand`** (accorgimenti sui blocchi e il campo di azione delle dichiarazioni negli argomenti o nei testi)

Come usarlo

- In genere si usano altri ambienti predefiniti con dichiarazioni aggiuntive per rendere visivamente la struttura logica che il nuovo ambiente rappresenta
- `\newenvironment{emphit}`

```
{\begin{itemize} \em}
{\end{itemize}}
```

Come usarlo

```
... \begin{emphit}
      \item Ciao
      \item A dopo
\end{emphit} ...
```

- Produce:
  - *Ciao*
  - *A dopo*

Un esempio più complicato

- Un ambiente per la scrittura di algoritmi

```
\newenvironment{alg}
    [3][Pseudocodice]
        { \ \\ \vspace{4mm} \ \\
              \begin{sf}
               {\sc Linguaggio:} #1.\\
               {\sc Input:} #2.\\
               {\sc Output:}  #3.
              \begin{tabbing}
              mm\=mm\=mm\=mm\=mm\=mm\= \kill \\}
        {  \end{tabbing} \end{sf} }
```

Algoritmo

- L'argomento opzionale è il linguaggio usato per descrivere l'algoritmo
- Il primo argomento obbligatorio è un testo che descrive l'inuput
- Il secondo argomento obbligatorio è la descrizione dell'output
- All'interno si può indentare di due “m” con il meccanismo del tabbing
- Il font è senza grazie (sf) per indicare la diversità rispetto al testo normale

\renewenvironment

- Stesso discorso che per \renewcommand
- Da usare con cautela se si ridefiniscono ambienti standard

Strutture numerate

- Nei testi scientifici spesso si usano strutture logiche di testo che vengono numerate progressivamente:
  - Definizioni
  - Teoremi
  - Assiomi
  - ....
- Anche altri tipi di testi hanno strutture simili: ad esempio gli articoli di una legge, ...

Strutture numerate

- Chiaramente LaTeX non può prevedere tutte le possibili strutture di questo tipo
- Quindi mette a disposizione una dichiarazione che permette di definirne di personalizzate
- Solo per un fatto di origine matematico-scientifica del linguaggio, tale dichiarazione si chiama **`\newtheorem`**

\newtheorem

- Definisce una struttura logica che viene evidenziata nel testo da
  - Nome della struttura
  - Numerazione progressiva
  - Nelle classi standard, testo enfatizzato
- **`\newtheorem{`*`nome`*`}[nseguetheorem]`**
  `{`*`TestoVisualizzato`*`}`
  `[structnum]`

\newtheorem

- È una dichiarazione che può avere due argomenti opzionali
- Il campo di azione è definito come per una qualsiasi dichiarazione
- Se inserita nel preambolo è visibile in tutto il documento
- Il ***nome*** è la sequenza di lettere che verrà usata in `\begin{`***nome***`}` e `\end{`***nome***`}` per inserire una nuova struttura nel documento

\newtheorem

- ***TestoVisualizzato*** è il testo che verrà apposto, in evidenza, all'inizio della struttura Es:
  - “Teorema”, “Articolo”, “Comma”, ...
- **`structnum`** è il nome di parte del documento entro la quale le strutture saranno numerate (1.1, 1.2, 2.1, 2.2, 2.3,...)
  - **`chapter`**, **`section`**, **`subsection`**, ...
  - Se non indicato la numerazione è progressiva per tutto il documento (1,2,3,4,...)

\newtheorem

- **`nseguetheorem`** è il nome di una struttura numerata con la quale il nuovo “theorem” condivide la numerazione
- \newtheorem{theorem}{Teorema}[section]
- \newtheorem{proposition}[theorem]
  {Proposizione}
- La proposizione che segue, nella sezione 2, il teorema 2.2 sarà la 2.3 (e viceversa)

Esempio

- `\newtheorem{algorithm}{Algoritmo}[section]`
- `\newtheorem{procedure}[algorithm]{Procedura}`
- Gli algoritmi e le procedure seguono la stessa numerazione che si riazzera ad ogni sezione

Uso

- L'uso di una struttura numerata è molto semplice
- Basta iniziare con **`\begin{nome}`** e terminare con **`\end{nome}`**
- È possibile aggiungere un argomento opzionale a **`\begin{nome}[`*`Testo`*`]`**
- ***`Testo`*** di solito è una descrizione della struttura logica in questione
- Ad esempio “Teorema di Euclide”, “Algoritmo del banchiere”, ...

Esempi

- Nel file .tex di esempio allegato ci sono diversi esempi di applicazione combinata di
- Strutture logiche **`algorithm`** e **`procedure`**
- Ambiente **`alg`** per la scrittura di algoritmi