7 Environments

Most LaTeX commands can only be given between the `\begin{document}` and `\end{document}` commands. This process can be taken further. LaTeX provides functions called "environments" which are a pair of commands `\begin{`*thing*`}` and `\end{`*thing*`}` between which the formatting of the text is subject to different rules and extra commands are available. Many of LaTeX's facilities are provided via environments.

7.1 Quotations

Quotations are typically indented on either side to make them stand out from the main running text. This is achieved by setting up a "quotation environment". The quotation is placed between `\begin{quotation}` and `\end{quotation}`.

```
Over on the back the same hand had written
this further information:---

\begin{quotation}
``Tall tree, Spy-glass shoulder, bearing a
point to the N.  or N.N.E.

``Skeleton Island E.S.E. and by E.

``Ten feet.

``The bar silver is in the north cache; you
can find it by the trend of the east
hummock, ten fathoms south of the black crag
with the face on it.

``The arms are easy found, in the sand hill.
N. point of north inlet cape, bearing E. and
a quarter N.''

\end{quotation}
That was all; but brief as it was, and to
me, incomprehensible, it filled the squire
and Dr.~Livesey with delight.
```

Over on the back the same hand had written this further information:—

> "Tall tree, Spy-glass shoulder, bearing a point to the N. or N.N.E.
>
> "Skeleton Island E.S.E. and by E.
>
> "Ten feet.
>
> "The bar silver is in the north cache; you can find it by the trend of the east hummock, ten fathoms south of the black crag with the face on it.
>
> "The arms are easy found, in the sand hill. N. point of north inlet cape, bearing E. and a quarter N."

That was all; but brief as it was, and to me, incomprehensible, it filled the squire and Dr. Livesey with delight.

7.2 Poetry

Poetry provides another example of a simple environment:

```
\begin{verse}
While Dawn, Day's herald straddling
the whole sky,\\
Offers the drowsy world a toast 'To Wine',\\
The Sun spills early gold on city roofs---\\
Day's regal Host, replenishing his jug.

Then shouts ring out among us at the tavern:\\
'Rise too, you good-for-nothing tavern lad!\\
Refill our empty bowls with today's measure\\
Before the measure of our lives be filled!'
\end{verse}
```

This generates the output:

> While Dawn, Day's herald straddling the
>   whole sky,
> Offers the drowsy world a toast 'To Wine',
> The Sun spills early gold on city roofs—
> Day's regal Host, replenishing his jug.
>
> Then shouts ring out among us at the tavern:
> 'Rise too, you good-for-nothing tavern lad!
> Refill our empty bowls with today's measure
> Before the measure of our lives be filled!'

7.3 Lists

So far, the environments we have seen simply cause differences in formatting. Environments have the potential to create new commands that exist within them, but not outside. A simple example of this is given by the three environments for generating lists. Consider the following:

```
\begin{itemize}
\item The first topic is dull
\item The second topic is duller
\begin{itemize}
\item The first subtopic is silly
\item The second subtopic is stupid
\end{itemize}
\item The third topic is dullest
\end{itemize}
```

which generates the following output:

- The first topic is dull
- The second topic is duller
  - The first subtopic is silly
  - The second subtopic is stupid
- The third topic is dullest

The `\item` command does not exist except within the `itemize` environment. It takes an optional argument which is the "bullet" to use.

If we replace `itemize` with `enumerate` to give

```
\begin{enumerate}
\item The first topic is dull
\item The second topic is duller
\begin{enumerate}
\item The first subtopic is silly
\item The second subtopic is stupid
\end{enumerate}
\item The third topic is dullest
\end{enumerate}
```

we get

1. The first topic is dull
2. The second topic is duller
   (a) The first subtopic is silly
   (b) The second subtopic is stupid
3. The third topic is dullest

The final list environment of the three provided is `description`. In this environment the optional argument of `\item` is always used:

```
\begin{description}
\item[The first topic] is dull
\item[The second topic] is duller
\begin{description}
\item[The first subtopic] is silly
\item[The second subtopic] is stupid
\end{description}
\item[The third topic] is dullest
\end{description}
```

gives the modified output:

The first topic is dull

The second topic is duller

  The first subtopic is silly

  The second subtopic is stupid

The third topic is dullest

7.4 What you see is what you get!

It is sometimes necessary to reproduce exactly what is typed, with no interpretation being done by LaTeX. A good example of this is the writing of LaTeX documentation! LaTeX provides two mechanisms for doing this: a command for exactly reproducing a small amount of data in-line, and an environment for exactly reproducing, and displaying, a larger amount of data.

The commands `\verb` and `\verb*` do not take an argument passed in braces like most other LaTeX commands, but instead look at the next character passed. Everything between that character and the next occurrence of that character is duplicated exactly, without any interpretation by LaTeX. The `\verb*` command renders spaces as “␣” rather than as “ ”:

- `The \verb!\LaTeX! command gives the ``\LaTeX'' logo.`

  The `\LaTeX` command gives the “LaTeX” logo.

The `verbatim` environment will render whole chunks of text without LaTeX processing:

```
\begin{verbatim}
Line
breaks in the input
are respected and
look at all these weird characters:
!@#$%^&*()_+|~-=\`[]{};':"<>/?
\end{verbatim}
```

```
Line
breaks in the input
are respected and
look at all these weird characters:
!@#$%^&*()_+|~-=\`[]{};':"<>/?
```

Just as the `\verb*` command renders spaces differently from `\verb`, there is a `verbatim*` environment that renders spaces as “␣”.

7.5 Tabular data

A common demand is for the display of tabulated data and LaTeX provides the `tabular` environment for this purpose. For example the input

```
\begin{tabular}{|l|c|r|}
\hline
abcdef & abcdef & abcdef \\
abcde & abcde & abcde \\
abcd & abcd & abcd \\
\hline
abc & abc & abc \\
ab & ab & ab \\
a & a & a \\
\hline
\end{tabular}
```

generates the following output:

abcdef	abcdef	abcdef
abcde	abcde	abcde
abcd	abcd	abcd
abc	abc	abc
ab	ab	ab
a	a	a

- The `\begin{tabular}` command takes an argument. It is the first environment we have seen that does.
- The argument `{|l|c|r|}` specifies the alignment of the contents of the various columns:

  l: left aligned,

  c: centre aligned,

  r: right aligned.
- The vertical bars indicate where the vertical lines of the table are to be inserted. They are not compulsory; omitting them simply gives a table without its vertical lines.
- The `\\` command is used to indicate the end of the line.
- (The final line does *not* end with `\\`.)
- The `&` is used to delimit entries within a line.
- The `\hline` command gives a horizontal line across the table.

Now consider the table:

Userid	UID	Name
root	0	Superuser
rjd4	2049	Bob
mug99	2193	Software Testing
spqr1	No such user	
fjc55		

This was created with the input:

```
\begin{tabular}{|l|r|l|}
\hline
Userid & UID & Name \\
\hline
root & 0 & Superuser \\
\hline
rjd4 & 2049 & Bob \\
\hline
mug99 & 2193 & Software Testing \\
\hline
spqr1 & \multicolumn{2}{l|}{No such user}\\
\cline{1-1}
fjc55 & \multicolumn{2}{l|}{}\\
\hline
\end{tabular}
```

There are two new features in this example:

- `\multicolumn` takes an argument and spreads it over several columns. Its three arguments are, in order:
  1. `{2}` The number of columns to be spanned.
  2. `{l|}` The formatting to be used. The *trailing* vertical bar must be provided here, not the leading bar.
  3. `{No such user}` The text to be placed in the wide column.
- `\cline{1-1}` Places a horizontal line over just those columns in the range it takes as its argument. The argument *must* be a range; even if it ranges over only one value, as in this example.