Document Preparation

- text formatting (*pdflatex* and *bibtex*)
- picture editor (*xfig*)
- capturing, displaying, and manipulating images (*import* and *display*)
- spell checkers (*spell* and *ispell*)
- printing (*lpr*, *lpq*, *lprm*, *pr*, and *a2ps*)

Word Processors

- Word processors, like *Word*, use the model of WYSIWYG (What-You-See-Is-What-You-Get).
  - interactive
  - easier for novices to learn
  - good for creating short documents

Text Formatters

- Text formatters, like *latex*, use the model of creating a file with markup commands, processing the file, and viewing the output.
  - batch
  - more control over the output
  - output tends to look nicer
  - better for creating long documents
  - having the source in ASCII text allows it to be processed by other tools

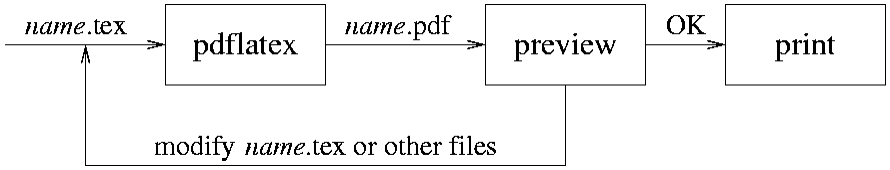
Tex and Latex

- *Tex* was invented by Donald Knuth in 1984. The main emphasis was to nicely print mathematical equations. *Tex* also had support for a variety of fonts.
- *LaTex* was invented by Leslie Lamport in 1985. It contains higher level support for tables of contents, citations, floating tables and figures, etc.

Generating a Latex Document

- The figure shows the usual sequence of commands for generating a *latex* document.

```
pdflatex name.tex       # produce a pdf file
okular name.pdf         # view a pdf file
```



Latex Commands

- A *latex* file must contain not only text, but also *latex* commands. Commands consist of special single characters or words that are preceded by the special character, the backslash \.
- Other special characters include:

%	*starts comments*	~	*represents a space*
&	*used in tables*	$	*delimits math expressions*
{	*start of argument list*	}	*end of argument list*
_	*precedes a subscript*	^	*precedes a superscript*
#	*for defining commands*		

- These special characters can be printed by preceding them with a \.

Latex Comments

- A comment in *latex* begins with a % and continues to the end of the line.
- *Latex* comments starting with % are similar to C++ comments starting with the // token.

Document Structure with Article Class

```
\documentclass[11pt]{article}    % specify class
\usepackage{graphics}            % preamble
\usepackage{graphicx}            %
\begin{document}                 % start document
\title{...}                      % article heading
\author{...}                     %
\date{...}                       %
\maketitle                       %
\tableofcontents                 % generate ToC
\begin{abstract}                 % abstract
   ...                           %
\end{abstract}                   %
\section{name1} \label{...}      % main body
   ...                           %
\section{name2} \label{...}      %
\bibliography{...}               % generate bibliography
\end{document}                   % end document
```

Latex Document Class

- The *latex* document class indicates the predefined way that the document will be formatted. Document classes include:

```
article    % relatively short documents like journal papers
report     % longer works that are broken into chapters
book       % has chapters, treats even and odd pages differently
slides     % used for generating slides
letter     % used for writing letters
```

- The following example specifies an article class that uses 11 point font.

```
\documentclass[11pt]{article}
```

Latex Use Package

- The *latex* \usepackage command indicates to latex to load a package, which contains *latex* information.
- This command allows styles for specific features or documents to be generated and applied by many users.
- The *graphics* and *graphicx* packages allow external graphics files to be imported and manipulated. Various formats are supported, including eps and pdf.
- Example:

```
\usepackage{graphics}
\usepackage{graphicx}
```

\begin{*environment*}...\end{*environment*}

- An environment has the effect that the text within it is treated differently according to the environment parameters. An environment is initiated with a \begin{*environment*} and is terminated by a \end{*environment*}.
- There are many predefined environments and most command names can also be used as environment names.

\begin{document}...\end{document}

- The \begin{document} ends the preamble. Everything between the \begin{document} and the \end{document} is referred to as the body. In contrast to the preamble, the commands in this portion only have a local effect within the section of the document in which they appear. The \ end{document} is typically placed at the end of the file.

Latex Article Heading

- The *latex* article heading consists of the title, author, and date.
- The \title{*title text*} is used to store the title of the document.
- The \author{*author information*} contains the author names, affiliations, and addresses. Multiple authors can be given by separating the information with a \ and.
- The \date{*date text*} contains the date of the article. If no date is given, then the current date is used.

Latex Article Heading (cont.)

- The \maketitle comand causes the title, author information, and date to be created.
- Depending on the style used, this information may appear on a separate page or just at the top of the first page.
- Example:

```
\title{Introduction to the GNU Debugger}
\author{John E. Student\\
        Florida State University}
\date{November 2012}
\maketitle
```

Latex Document Spacing

- *Latex* processes the input by creating output lines of equal width, formatting the text into paragraphs, and breaking the document into pages.
- Blanks and newlines indicate the end of a word. Extra blanks between words are ignored.
- A new paragraph is indicated by an empty line with multiple empty lines having the same effect as a single line.

Latex Abstracts

- An abstract in a *latex* document is specified within the \begin{abstract} and \end{abstract} commands. In the article class, the abstract comes after the article heading (title, authors, date). Other *latex* document classes can have the abstract appearing on a separate page.
- Example:

```
\begin{abstract}
This paper goes over the basics of the GNU
debugger, also known as \textit{gdb}.
\end{abstract}
```

Latex Sections

- A *latex* article is broken up into a hierarchy of sections. The following commands are used.

\section{*title*} % generates section number #

\subsection{*title*} % generates section number #.#

\subsubsection{*title*} % generates section number #.#.#

- Section numbers and titles are also saved to generate a table of contents if requested.
- Examples:

```
\section{Introduction}
\subsection{Invoking GDB}
\section{Main Activities}
```

Latex Labels and References

- Sections, figures, and tables are often referenced by number within the text of a document. However, the writer can decide to reorder these items. *Latex* allows these items to have labels and to reference these labels within the document to avoid renumbering these references.
- General form.

\section{*title*}
\label{*labelname*}
... Section~\ref{labelname} ...

Selecting Font Style in Latex

- One can select the font shape, series, and family.
  - The shape indicates the form of the font.

    \textup{*upright text*}, \textit{*italics text*}, \textsl{*slanted text*}, \textsc{*small caps text*}
  - The series indicates the width of the font.

    \textmd{*medium text*}, \textbf{*boldface text*}
  - The family indicates the overall style of the font.

    \textrm{*roman text*}, \textsf{*sans serif text*}, \texttt{*typewriter text*}

Selecting Font Size in Latex

- The following commands can be used to select the fontsize for the duration of the current scope or until another fontsize command is given.

\tiny (5pt)	\normalsize (10pt)	\LARGE (17.28pt)
\scriptsize (7pt)	\large (12pt)	\huge (20.74pt)
\footnotesize (8pt)	\Large(14.4pt)	\Huge (24.88pt)
\small (9pt)		

- One can also set an arbitrary fontsize. However, not all font sizes may be supported on a particular system. The predefined sizes are usually supported.

```
\fontsize{10}{12}   % sets size to 10pt and
                    % interline spacing to 12pt
```

Latex Tables

- *Latex* has two environments that can be used for producing tables. The *table* environment is used to place the location of the table and provide a caption. The *tabular* environment is used to format the actual table. The following general commands are typically used.

\begin{table}[*placement*]
\begin{tabular}{*format*}
...
\end{tabular}
\caption{*caption text for the table*}
\label{*labelname*}
\end{table}

Placement of Latex Tables or Figures

- The placement specifies the allowed locations for a table or a figure. Multiple placement options can be given to indicate a preference order and the placement selected is the one that obeys that style parameters and that can be placed earliest. If no placement is specified, then [tbp] is assumed.

h	*here*: place at that point in the text
t	*top*: place at the top of the page if room for it and the previous text, if not then place at top of next page
b	*bottom*: place at bottom of page if room, if not then place at bottom of next page
p	*page*: place in a special page reserved for only tables and figures

Formatting Columns in Latex Tables

- The \begin{tabular} *format* parameter indicates how the table columns are to be formatted. There should be a symbol for each column.

l	the column contents are left justified
r	the column contents are right justified
c	the column contents are centered
|	draws a vertical line
||	draws two vertical lines next to each other

- Example:

```
\begin{table}[htb]
\begin{tabular}{|c|c|l|}
...
```

Specifying the Data in Latex Tables

- Each horizontal row in a table is terminated with a \\. The column entries are separated by a & symbol. Horizontal lines can be drawn using the \hline command.
- Example:

```
Command & Arguments & Explanation\\
\hline\hline
break & [file:]function & Set a breakpoint at function.\\
\hline
...
```

Latex Figures

- A *latex* figure can be included into a document by the following commands. Options include specifying the width, height, angle, etc.

\begin{figure}[*placement*] % start figure environment
\includegraphics[*options*]{*filename*}
% specify options and filename
% containing the figure
\caption{*caption text for the figure*}
\end{figure} % end figure environment

Example Inclusion of a Latex Figure

- In the following example, the *dddpic.pdf* file is included into the document with a width that is 0.8 of the text width along with a specified caption and label.

```
\begin{figure}
\includegraphics[width=1.0\textwidth{dddpic}
\caption{A Screenshot of DDD}
\label{fig:ddd}
\end{figure}
```

Lists in Latex

- There are several types of list environments in *latex*. These include *itemize* (bulleted list), *enumerate* (numbered list), and *description* (customized list).
- General form.

\begin{*listtype*}
\item *text*
\item *text*
...
\item *text*
\end{*listtype*}

Example Latex List

```
\begin{enumerate}
\item   Start your program, specifying anything
        that might affect its behavior.
\item   Make your program stop on specified
        conditions.
\item   Examine what has happened when your program
        has stopped.
\item   Change the values of variables in your
        program so you can experiment with the
        effects of a bug.
\end{enumerate}
```

The Verbatim Environment

- The *verbatim* environment is used to print out text exactly as it was input. This is very useful for printing out code. Note the special characters lose their latex significance within the environment.

```
\begin{typett}
\begin{verbatim}
main()
{
   printf(“Hello World!\n”);
}
\end{verbatim}
\end{typett}
```

Generating a Bibliography in Latex

- The database for publications can be kept in a separate file that is processed by a tool called *bibtex*. The name of the database file has the form:

  databasename.bib

- The text references to publications that will appear in the bibliography are made in the following form:

  \cite{*key*}

- The generation of the bibliography, which is typically performed at the end of the document, is accomplished with the following commands:

  \bibliographystyle{*style*}
  \bibliography{*databasename*}

Creating a Bibliographic Database

- Each entry in the database contains information for one or more predefined fields. These fields include the author, title, journal, volume, number, pages, date, institution, publisher, url.
- The general form of each of the entries in a *.bib file is:

@entry_type{*key*,
field_name = “*field_text*”,
field_name = “*field_text*”,
...
field_name = “*field_text*”
}

Example Bibliographic Entries

```
@book{gdb,
   author = “R. Stallman and R. Pesch and S. Shebs”,
   title = “Debugging with GDB”,
   publisher = “Free Software Foundation”,
   month = “January”,
   year = “2002”
}

@manual{ddd,
   author = “A. Zeller”,
   title = “Debugging with DDD”,
   publisher = “Free Software Foundation”,
   month = “January”,
   year = “2004”,
   url = “http://www.gnu.org/manual/ddd/pdf/ddd.pdf”
}
```

Bibliography Style

- There are four style arguments to the following \ bibliography{*style*} command:
  - *plain*: entries ordered alphabetically and markers are a number inside square brackets
  - *unsrt*: entries ordered by appearance of citation in the paper
  - *alpha*: same as plain but markers are an abbreviation of the author's name and year
  - *abbrv*: same as plain, but bibliographic listing abbreviates first names, months, and journal names

Using Bibtex

- The general command for processing a *bibtex* file is to run *bibtex* on the *.tex file. Note that the \ bibliography latex command specifies the basename of the *.bib file. However, *latex* does prepare output that is used by *bibtex*. So a user has to run *pdflatex* first, *bibtex* next, and finally *pdflatex* again.

  pdflatex *basename*
  bibtex *basename*
  pdflatex *basename*

Viewing Latex Output

- *pdflatex* processes a *basename*.tex file as input and produces a *basename*.pdf as output. A user can view the pdf file using the following command:

  okular *basename*.pdf

Xfig: A Picture Editor

- *Xfig* is a menu-driven tool that allows a user to interactively create and manipulate figures. Features include:
  - Drawing lines, circles, ellipses, splines, polygons, rectangles, arcs, and arrows.
  - Entering text and inserting images.
  - Other operations on objects include scaling, moving, copying, deleting, flipping, and rotating.
  - Objects with lines can be colored and the thickness and style of the lines can also be varied.
  - The area inside of objects can also be filled with color and/or patterns.

Other Xfig Features

- Other features that *xfig* supports include:
  - Can export the figure in different formats.
  - Can use a grid to better control the placement of objects.
  - Can change the characteristics of an object by editing or updating them.
  - Can perform group operations on an object.

Capturing Screen Images

- One can use the *import* command to capture screen images. The command must be initiated from an x-term, not from a ssh console.
- General form of the command is:

  import *filename*
- After typing the command either click on a window or click and drag to define the screen area to be captured.
- You may find it easiest to *import* an image to an *eps* file and then using *epstopdf* to convert it to a *pdf* file before referencing it in a *latex* document.

Display: An Interactive Image Editor

- *Display* (ImageMagick) has many capabilities for processing or manipulating images. These include:
  - loading an image from a file
  - converting an image to a different format
  - scaling and rotating an image
  - cropping an image
  - color map editing
  - applying a variety of filters on a image
  - printing and saving an image

Spell and Ispell

- *Spell* is a batch Unix utility that can be used to report spelling errors. It is designed to work with *troff* source files, but can also be applied on other text, such as *latex* (*.tex) files. *Spell* takes a list of files as input and prints words that it cannot find in its dictionary to standard output.

  spell *filenames*
- *Ispell* is an interactive spell checker. *Ispell* will display each word that is not found in its dictionary and allow a user to change it. If there are other words that differ slightly, then they are displayed and the user can select one of them to replace it.

  ispell *filename*

Printing (45)

- lpr
- lpq
- lprm
- pr
- a2ps

The Lpr Command (45.2)

- The *lpr* command queues a print job for a given printer.
- The type of file is read on first line of the file.
- Commonly used *lpr* options include:

```
-#num      # number of copies to be printed
-Pprinter  # name of the printer
-Zoption   # option to be passed to the print spooler
           # (e.g. duplex to print on both sides)
```

Lpq and Lprm

- The *lpq* command is used to display the list of print jobs in the queue for a printer. You can pass "-P*printer*" as an argument to indicate the printer queue to be displayed. Among other information, it lists the job id, file to be printed, and the owner.

  lpq [-P*printer*]

- The *lprm* command is used to delete a job from a print queue. One uses the job id shown in the lpq command.

  lprm [-P*printer*] *jobid*

The Pr Command (45.6)

- The *pr* command is used to break up a text file into pages for printing. Page headings are also generated.

  pr [*options*] [*files*]

- Commonly used options include:

```
-ln        # set page length to n lines (default is 66)
-wm        # set page width to m columns (default is 72)
+num       # begin printing a page num (default is 1)
-m         # merge files printing them side by side, text is
           # chopped to fit in the page width
-t         # omit the page headings and trailing blank lines
```

A2ps: Ascii Formatter for Printing

- The *a2ps* Unix utility formats text files for printing on a postscript printer. In other words, it converts text to postscript.
- General form. It reads one or more files and writes to standard output.

  a2ps *text_files*
- Some of its features include:
  - putting multiple pages of input per output page
  - pretty printing (vary fonts for different lexical elements)
  - setting font size and selecting portrait or landscape mode
  - controlling what appears on the heading of each page

Some Commonly Used A2ps Options

-r	// print in landscape mode
-f #	// set font size to specified #
-o *filename*	// write to *filename* instead of stdout
--columns=#	// number of columns per page
--center-title=[*text*]	// put *text* as a centered title
--pages=*range*	// only print the *range* of pages