Web Tools for Mathematics and Computer Science—A Primer

William Slough
Department of Mathematics and Computer Science
Eastern Illinois University

September 2002

Contents

1 Introduction

To create a "web presence" for a course, some decisions about file formats need to be made. Popular choices include html and pdf. Although some web designers use proprietary formats (such as Microsoft's doc), this is much less desirable than the others, since one cannot assume that such files can be read by a wide variety of computers. In addition, mathematics and computer science places extra demands upon the content: one often needs or benefits from a wide selection of fonts and special symbols. TEX and its related tools are particularly strong in this area.

This primer provides a guided tour of selected tools and packages which can be used to produce materials for a web site and course use. The intent of this primer is to provide a quick start for those wishing to produce classroom presentations and/or course web sites containing mathematical or scientific content. All of the examples shown were derived from the documentation and books which appear in the references.

This major focus of this primer is on two primary tools—pdfLATEX and TEX4ht—along with a number of subsidiary packages that can be used in conjunction with these. It is assumed that the reader has a basic grounding in the use of LATEX and has access to at least one source of documentation, e.g. [6], regarding its use.

The tools illustrated here are based upon TEXLive 7, the latest distribution of TEX provided by TUG, the TEX Users Group. Although these tools can be installed on a wide variety of computers and operating systems, this primer assumes Linux. For those lacking an office computer with Linux, EIU's math server provides this capability.

2 Use of Color

Chapter 9 of [3] is devoted to a discussion of color in the context of LATEX. The color package provides support for both RGB and CMYK models and a "named" model which provides access to a collection of 68 colors specified with names like `SkyBlue` and `Goldenrod`; the following page shows all possibilities.

To use the color package, include the following statement in the preamble of your LATEX document:

```
\usepackage[pdftex,dvipsnames,usenames]{color}
```

The "named" model is particularly convenient. For example,

```
\textcolor{RoyalBlue}{The quick brown fox}
```

produces The quick brown fox.

Using `color{`*colorname*`}` will switch the color to the specified color. This color will remain in effect until a subsequent `color{`*colorname*`}` command appears. For example,

```
\color{RoyalBlue}
The quick brown fox jumps over
\color{Magenta}
the lazy dog.
```

produces The quick brown fox jumps over the lazy dog. Like fonts, colors can be overused; however, some use of colors in classroom presentation materials can be useful for emphasis.

Named colors in `dvipsnam.def`

3 Symbols: Mathematical and Otherwise

Scott Pakin [8] has documented 2,266 symbols and their corresponding LaTeX commands that produce them. It is probably safe to say that any symbol needed by a mathematician appears somewhere within this collection.

In addition to the wealth of mathematical symbols, there are many other types of symbols which can be used. For example, the Zapf Dingbat fonts are available via the pifont package. Symbols such as ✎, ❖, ➠, and ✓ are made available from this package. These symbols are specified within a document by `\ding{48}`, `\ding{118}`, `\ding{224}`, and `\ding{51}`, respectively. Symbols like these can be used to "dress up" foils and presentations. For complete details, refer to [8].

4 Graphics

4.1 Screen captures and PNG files

A simple way to include a graphic image in a document involves a screen capture. Of all the techniques possible, this offers the poorest resolution, since the image is a bitmap. Scaling a bitmap may result in further lack of image quality. On the other hand, it is sometimes essential to show *exactly* what appears on the screen. We choose the png format since it is one of the native formats supported by pdfLaTeX.

There are just two steps involved: first, obtain the screen capture, saving it as a png file and, second, include this file in a LaTeX document.

There are many ways to obtain a screen capture, including Gimp and KSnapshot. Of these, KSnapshot is easiest to use; its use is self-explanatory! If you intend to use this often, you might consider placing an icon on your desktop; the executable name is `ksnapshot`.

The inclusion of the png file makes use of the graphicx package. (Note carefully the spelling; the package name ends with `x`, not `s`.) See [10] for more details than you ever thought possible regarding graphics!

As a simple example, include the line

```
\usepackage[pdftex]{graphicx}
```

in the preamble of the LaTeX document. To insert the png image, use the `\includegraphics` command at the desired location. For example, to insert the file maple.png at 50% its original size, use the command:

```
\includegraphics[scale=0.5]{maple}
```

Observe that the file extension (.png in this case) does not need to appear within the argument of the `\includegraphics` command. Figure 1 shows the result of such an inclusion.

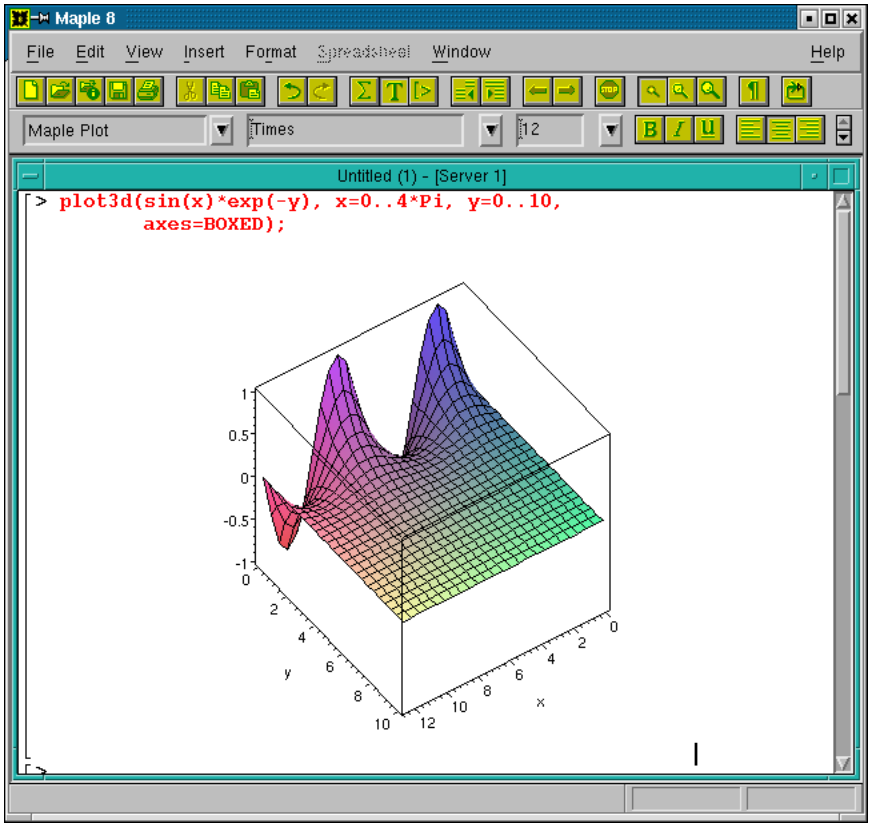

Figure 1: A screen capture of a Maple session, stored as a png file.

4.2 Using Xfig with PDF files

Among the many tools available for drawing figures is xfig. The "native" file format for this program is fig, although this format cannot be directly used by a LaTeX document. xfig can export to many different file formats however; among these is the pdf format, which can be processed directly by pdfLaTeX.

The file link.pdf can be inserted into a document using the command

```
\includegraphics[scale=0.5]{link}
```

Figure 2 shows the result of including this file. Again, notice the file extension, .pdf, is not needed.

4.3 Using EPS files

The eps file format is very popular, and for good reason. It scales well, providing excellent print quality. It is also well supported by the graphicx package and the TeX-related drivers. If the ultimate aim is to produce Postscript, it is probably the graphic format of first choice.

However, eps is not among the formats supported by pdfLaTeX. Consequently, if the goal is to produce high-quality pdf documents, an alternative must be found. One approach is to use latex (including the eps graphic), generate a dvi file, convert this file to Postscript with dvips and, finally, convert the Postscript to pdf using a tool such as Ghostscript's ps2df. This works well, but it requires multiple steps.

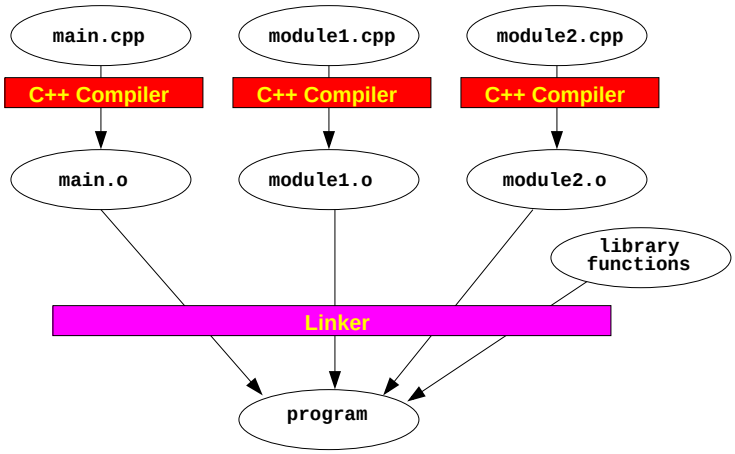

Figure 2: A drawing from xfig, exported as a pdf file.

An alternative approach uses purifyeps[9], which takes fewer steps, involves fewer files, and is generally simpler. To consider a concrete example, suppose we wish to include a Maple plot in a pdf document. The steps are as follows.

1. Generate the plot within Maple.
2. Select the plot with a left mouse click.
3. Use the Maple **Export As** menu (right mouse click) to save the image as an eps file.
4. Use purifyeps to transform this eps file to a format acceptable to pdfLaTeX.
5. Include the graphic in the LaTeX document using `\includegraphics`.

Suppose a Maple plot has been exported to maple-plot.eps. The command to transform this file is:

```
purifyeps maple-plot.eps maple-plot.eps
```

The two arguments which appear on the command line are the input and output files, respectively. As this example shows, the output file name is allowed to match the input file name, causing the original file to be replaced with its purified form.

When using purified eps files, the preamble of the LaTeX document should include these statements:

```
\ifx\pdfoutput\undefined
\else
  \DeclareGraphicsExtensions{.png, .pdf, .jpg, .mps,.tif, .eps}
  \DeclareGraphicsRule{.eps}{mps}{*}{}
\fi
```

Figure 3 shows the result of including an exported eps file in this manner.

As a caveat, it is possible that purifyeps may not produce an acceptable image file. Should this occur, epstopdf can be used to produce an equivalent image in pdf format which can then be included in the document.

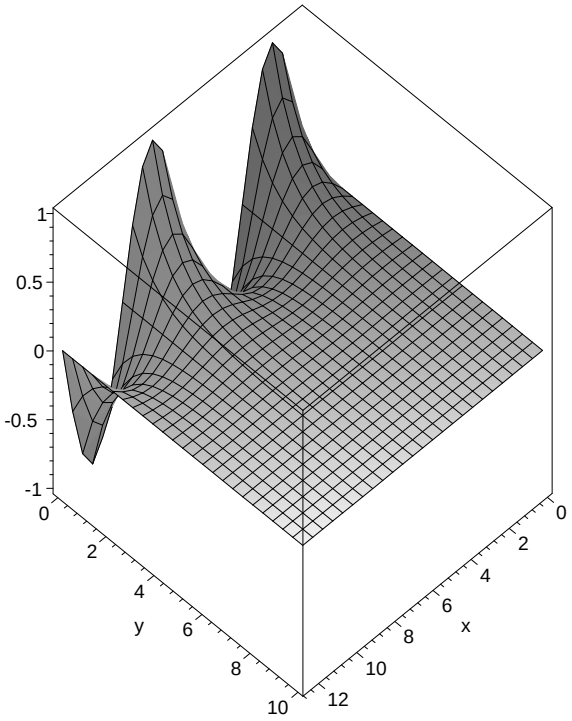

Figure 3: A Maple plot, stored as an eps file. This plot was exported by Maple to eps, then "purified." A scale factor of 0.5 was used.

4.4 Annotated EPS files

Sometimes it is desirable to include LaTeX symbols or equations within an eps file. For example, one might want to annotate a graph which has been exported by Maple.

One way to do this involves the LaTeX package psfrag. However, this package was designed with Postscript in mind and does not work directly with pdfLaTeX. This deficiency is noted in recurring discussions posted on the newsgroup comp.text.tex. The conventional wisdom on this topic is that some preprocessing steps are needed to allow the appropriate transformations.

Although the results can be quite good, some trial and error is sometimes needed. The basic idea is as follows. First, an eps file is created with additional placeholder symbols. These placeholders are used to mark the locations of LaTeX text which will ultimately appear in their place. Second, a file is prepared describing what text substitutions should take place. Finally, the script psfrag2pdf is invoked, which applies the substitutions and creates a pdf file containing the graphic, which can then be incorporated into the document using `\includegraphics`.

To consider a concrete example:

1. Using Maple, produce and export a graph, producing graphs.eps. For best results, use the Maple command `psfrag`. When this graph is produced, include several textual items—such as `y=f(x)` and `y=g(x)`. Figure 4 shows this graph with original annotations.

2. Prepare a file of psfrag substitutions. This file should match the name of the eps file, with a second extension of .psfrag. Figure 5 illustrates the substitution file for this example.

3. Perform the transformation by invoking the psfrag2pdf command:

   ```
   psfrag2pdf graphs.eps
   ```

   This script examines graphs.eps and graphs.eps.psfrag, producing graphs.pdf. Figure 6 shows the result of this processing.

4. Insert the graphic file:

   ```
   \includegraphics[scale=1.0]{graphs.pdf}
   ```

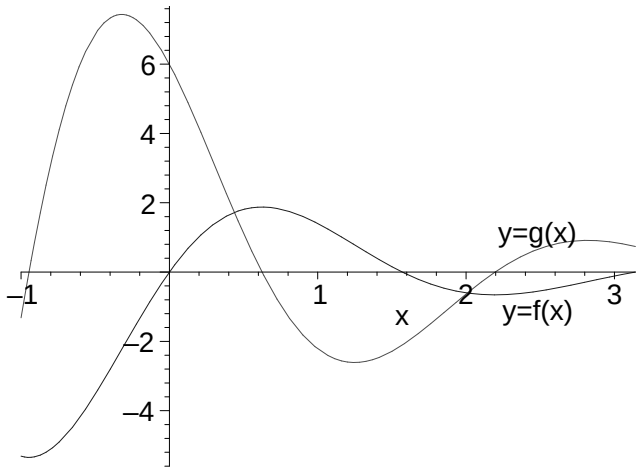

Figure 4: A Maple plot, stored as an eps file, with original annotations.

As an aside, it is worth noting that the Maple command psfilter and the shell script psfrag2pdf are non-standard localizations. For those interested in shell scripts, psfrag2pdf employs some interesting techniques.

```
\psfrag{y=f(x)}{$y=f(x)$}
\psfrag{y=g(x)}{$y=f'(x)$}
\psfrag{x}{$x$}
```

Figure 5: psfrag substitutions which appear in the file graphs.eps.psfrag.

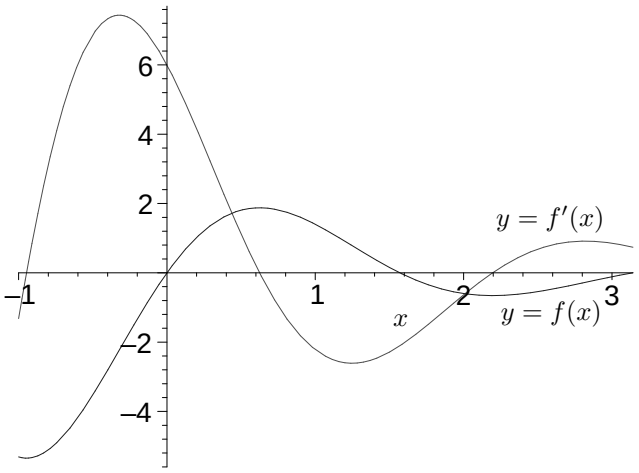

Figure 6: A Maple plot, exported to eps and processed by psfrag2pdf.

5 Foils

5.1 Foils for presentation

The foilTEX package[5] allows for the production of "slides" or "foils." Its use of larger sans serif fonts makes it ideally suited for classroom presentation. When used with the color package[1] and pdfLATEX, the result is a pdf file which can be used in a classroom lecture; additionally, it may be posted on a course web site. Some additional "special effects" may also be obtained with the use of Ppower4[4].

Figure 7 shows a typical preamble used in the production of foils. In the preamble, a number of packages are first specified. The pifonts package appears in order to redefine the symbols used in itemized environments. Ppower4 allows incremental slides and two-color backgrounds, which accounts for the pause and background packages. Several other packages are declared here to allow for more sophisticated Ppower4 use. Also included in the preamble are statements which give convenient names to colors, information for a title page, and details about what should appear at the foot of each slide. Finally, a conditional statement appears which allows convenient use of purifyeps for graphics inclusion. Although the preamble is a bit complex, it does not need to be changed much from one set of foils to another.

Figure 8 illustrates what commands are placed in the document body to generate the desired foils. This tiny example produces just two foil: the title foil and one other. The title foil is produced by the information given in the abstract and title/author/date information provided in the preamble. As is customary in LATEX documents, the `\maketitle` command produces the title page.

Each foil begins with a `\foilhead` command, with the desired heading as its sole argument.

It is worth noting that there is nothing special about including graphics within foils: standard techniques apply. However, since pdfLATEX is being used, only graphics formats it can process must be used. These formats include png and pdf, but *not* eps. As previously noted, purifyeps can be used to convert many eps files to an acceptable form.

Web Tools for Mathematics and Computer Science

William Slough
Eastern Illinois University
Department of Mathematics and Computer Science
cfwas@eiu.edu

September 2002

Abstract
An overview of some TEX-based tools is provided.

Web Tools for Mathematics and Computer Science

- Based upon pdfLaTeX and TeX4ht
- Uses TeXLive 7
- PDF and HTML are the predominant file format
- Colors, symbols, graphics, and hyperlinks


```
\documentclass[20pt,landscape,footrule]{foils}
\usepackage[pdftex,dvipsnames,usenames]{color}
\usepackage{hyperref}
\usepackage{pause}
\usepackage{background}
\usepackage{tabularx}
\usepackage{pp4link}
\usepackage{mpmulti}
\usepackage[pdftex]{graphicx}
\usepackage{pifont}

% Redefine the itemize symbols with pifont symbols
\renewcommand{\labelitemi}{\textcolor{RoyalBlue}{\ding{118}}}
\renewcommand{\labelitemii}{\textcolor{RoyalBlue}{\ding{224}}}
\renewcommand{\labelitemiii}{\textcolor{RoyalBlue}{\ding{51}}}

% Color to use for the header/footer of each slide
\definecolor{Headcolor}{named}{RoyalBlue} %{cmyk}{0.92,0,0.59,0.25}
\renewcommand\normalcolor{\color{Headcolor}}

% Two-color vertical fade for background
\definecolor{Lopagecolor}{cmyk}{0.03,0,0.02,0}
\definecolor{Hipagecolor}{cmyk}{0.24,0,0.06,0}
\vpagecolor[Lopagecolor]{Hipagecolor}

% Define nicknames for colors to be used throughout the foils
\definecolor{Textcolor}{named}{Black}    \newcommand\Text{\color{Textcolor}}
\definecolor{Highlight}{named}{BrickRed} \newcommand\High{\color{Highlight}}
\definecolor{Dullness}{named}{Aquamarine}\newcommand\Dull{\color{Dullness}}
\definecolor{Emphcolor}{named}{Plum}     \newcommand\Emph{\color{Emphcolor}}

\definecolor{TwoColor}{cmyk}{0,0,0,1}
\pausecolors{TwoColor}{Textcolor}{Highlight}

\title{\textcolor{RoyalBlue}{Web Tools for Mathematics and Computer Science}}
\author{William Slough\\  Eastern Illinois University\\
        Department of Mathematics and Computer Science\\
        \texttt{cfwas@eiu.edu} }
\date{September 2002}

\MyLogo{\color{Headcolor}\copyright~2002, W.~Slough
  \rightfooter{\pauselevel{=1 +1}
    {\quad\textsf{\tiny\color{Headcolor}\thepage}}}}

\setlength{\parindent}{0pt}       % don't indent paragraphs

% Used in conjunction with "epspurify"-ied graphics
\ifx\pdfoutput\undefined
\else
  \DeclareGraphicsExtensions{.png, .pdf, .jpg, .mps,.tif, .eps}
  \DeclareGraphicsRule{.eps}{mps}{*}{}
\fi
```

Figure 7: The preamble used for preparation of foils

```
\begin{document}
\maketitle
\begin{abstract}
An overview of some \TeX-based tools is provided.
\end{abstract}

\foilhead{Web Tools for Mathematics and Computer Science}
\begin{itemize}
\item Based upon {\sf pdf}\LaTeX\ and \TeX4{\sf ht}
\item Uses \TeX Live~7
\item {\sf PDF} and {\sf HTML} are the predominant file format
\item Colors, symbols, graphics, and hyperlinks
\end{itemize}
\end{document}
```

Figure 8: The body used for preparation of foils

5.2 Foils for distribution

Foils are used primarily for classroom presentation and, as such, use oversized fonts for clarity. To make these available for distribution, it is desirable to reduce these. Among the many possibilities, "four-up" or "eight-up" are two ways to conserve paper.

The pdfpages package[7] provides a flexible and simple way to produce such N-up formats. To illustrate, suppose `week1.pdf` is the file that holds 35 foils to be used during classroom presentation. To produce slides 4 to a page, the file `week1-4up.tex` is prepared. Processing this file with pdflatex produces `week1-4up.pdf`.

Within `week1-4up.tex`, options to pdfpages are given to control:

- the number of virtual pages (`nup`)
- the orientation on the page (`landscape`)
- whether virtual pages should be framed (`frame`)
- what page offset to use (`offset`)
- the spacing to use between the virtual pages (`delta`)

In this example, we specify an offset to allow room for punching holes. Observe that `delta` takes a specification for each of the horizontal and vertical dimensions. The `\includepdf` command is used to specify the pages which should appear. The scaling factor allows for a margin to appear.

The full details are given in figure 9; page 15 shows the result.

```
\documentclass{article}      % Produces 2x2-up landscape slides
\usepackage{pdfpages}
\pagestyle{empty}

\includepdfset{nup=2x2,
               landscape=true,
               frame=true,
               offset=0.25in 0in,
               delta=0.1in 0.2in}

\begin{document}
  \includepdf[pages={1-35}, scale=0.9]{week1.pdf}
\end{document}
```

Figure 9: Code to produce 4-up foils

MAT 2670 - Computer Science II

1. organize a project using the Linux `make` utility
2. design and construct C++ classes
3. understand and utilize class inheritance
4. solve problems using recursion
5. manipulate basic data structures
   from the C++ Standard Library

 1

6. build simple data structures
   that use dynamic memory allocation
7. implement simple event–driven programs
   with the EzWindows API

 2

Week 1 Topics

1. Syllabus and General Course Guidelines
2. Linux Operating System, Commands
3. Compiling, loading, archiving
4. `make` and Makefiles
5. Recursion

 3

General Guidelines

- Syllabus
- Schedule (Tentative, but note exam dates)
- Labs
- Programming Projects - Academic Integrity
- Homework

 4

To obtain an 8-up format, we use 2 columns of 4 virtual pages in a portrait orientation. The `pagecommand` is also included to place page numbers and a running head on each page. As before, an offset is provided to allow room for punching holes. Full details are given in figure 10; page 17 illustrates the result.

```
\documentclass{article}                % Produces 2x4-up portrait slides
\usepackage[top=0.2in]{geometry}
\usepackage{pdfpages}

\pagestyle{myheadings}
\markright{\em MAT 2670 -- Week 1 Slides}

\includepdfset{nup=2x4,
               landscape=false,
               frame = true,
               offset=0.5in 0in,
               delta=0.2in 0.1in,
               pagecommand={\thispagestyle{myheadings}}
              }

\begin{document}
  \includepdf[pages={1-35}, scale=0.9]{week1.pdf}
\end{document}
```

Figure 10: Code to produce 8-up foils

MAT 2670 - Computer Science II

1. organize a project using the Linux `make` utility
2. design and construct C++ classes
3. understand and utilize class inheritance
4. solve problems using recursion
5. manipulate basic data structures from the C++ Standard Library

6. build simple data structures that use dynamic memory allocation
7. implement simple event–driven programs with the EzWindows API

Week 1 Topics

1. Syllabus and General Course Guidelines
2. Linux Operating System, Commands
3. Compiling, loading, archiving
4. `make` and Makefiles
5. Recursion

General Guidelines

- Syllabus
- Schedule (Tentative, but note exam dates)
- Labs
- Programming Projects - Academic Integrity
- Homework

Linux Operating System

- User accounts & passwd
- Commands:
  - `mkdir` — make a directory
  - `cd` — change directories
  - `ls` — list contents of directory
  - `pwd` — print working directory
  - `cp` — copy command
  - `mv` — move, a way to rename or relocate files or directories
  - `rm` — remove, to delete a file

  - `rmdir` — remove an empty directory
  - `man` — the manual pages
- Handout: `Linux Installation and Getting Started, Chapter 3`
- Unix Programming Tools Manual (Sections 1 & 2): online
- GNU `make` Manual, Section `Introduction to Makefiles`: online

Hello World!

```
// MAT 2670 --- Lab1
// A very simple  program
//

#include <iostream>

using namespace std;

int main() {
  cout << "Hello World!!" << endl;
  return 0;
}
```


timer.h

```
#ifndef TIMER_H
#define TIMER_H
#include <ctime>
class Timer {
 public:
  Timer();
  void Start();
  void Stop();
  clock_t ElapsedTime() const;
 private:
  clock_t Begin;    clock_t End;
};
#endif
```


6 Generating HTML files

There are many software products which can be used to generate html files. Some people might choose to edit html files "bare-handed" for fine control. For those who prefer to generate html with a LaTeX flavor, there is TeX4ht. Complete details can be found in Chapter 4 of [2].

To give a sense of the possibilities, Figures 11–12 illustrate a sample course web site and the corresponding source code. Using this example as a framework, it is possible to extend this to suit the needs of other courses.

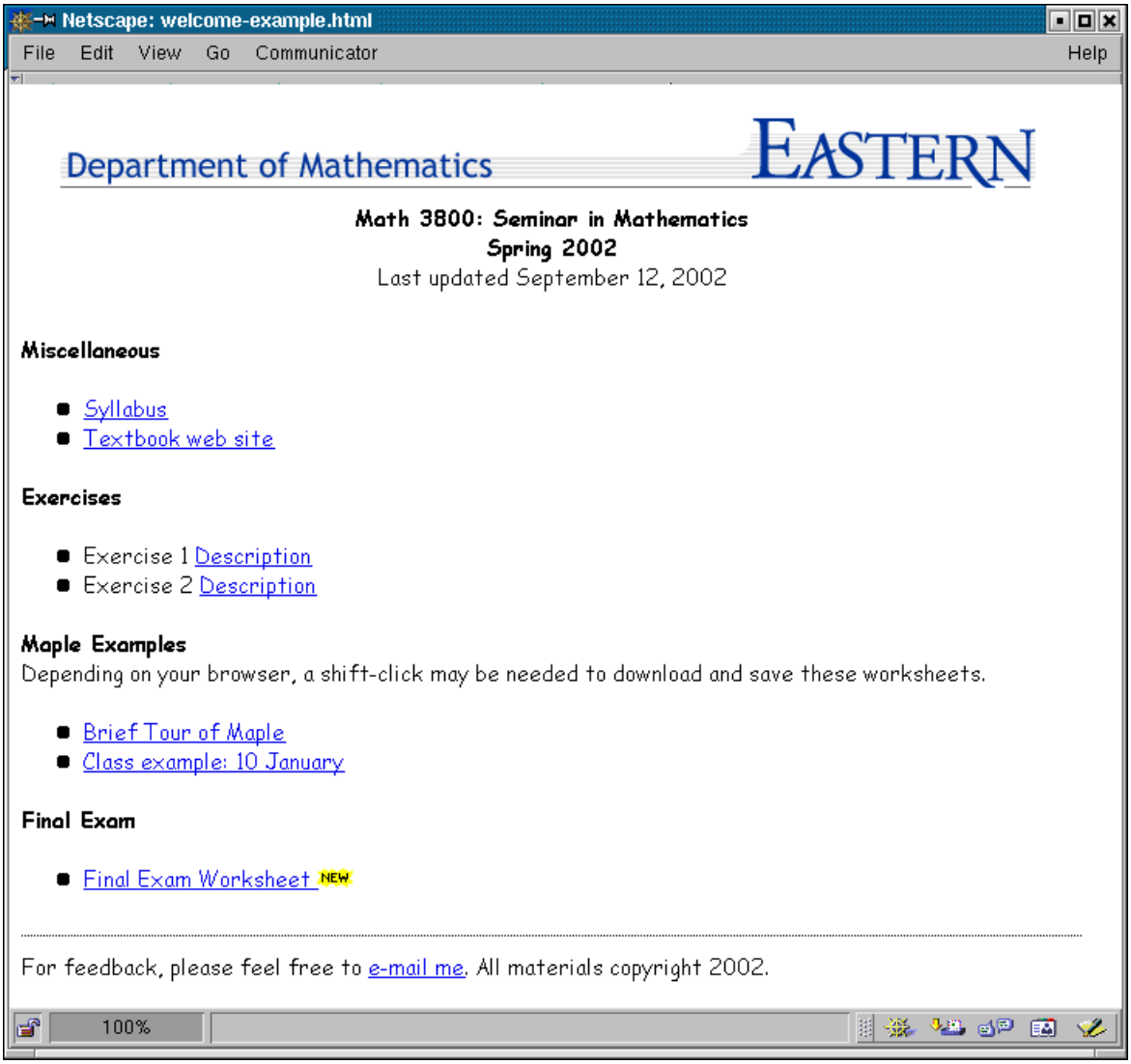

Figure 11: A sample html file, produced with TeX4ht.

```
\documentclass{article}
\usepackage[html]{tex4ht}
\begin{document}
\setlength{\parindent}{0in}
\Configure{section}
{}{}
{\bf}{\rm}
\HCode{<body BGCOLOR="FFFFFF">}

\begin{center}
\HCode{<IMG ALIGN=CENTER SRC="math.gif">}\\

{\Large \bf Math 3800: Seminar in Mathematics}\\
{\bf Spring 2002}\\
Last updated \today
\end{center}

\section{Miscellaneous}

\begin{itemize}
 \item \Link[Webview/syllabus.pdf]{}{}Syllabus\EndLink
 \item \Link[http://centaur.maths.qmw.ac.uk/CwM/]{}{}Textbook web site\EndLink
\end{itemize}

\section{Exercises}
\begin{itemize}
 \item Exercise 1
   \Link[Webview/Exercises/hw01.pdf]{}{}Description\EndLink
 \item Exercise 2
   \Link[Webview/Exercises/hw02.pdf]{}{}Description\EndLink
\end{itemize}

\section{Maple Examples}\\
Depending on your browser, a shift-click may be needed to download and
save these worksheets.
\begin{itemize}
 \item \Link[Webview/Examples/example01.mws]{}{}Brief Tour of
   Maple\EndLink
 \item \Link[Webview/Examples/example02.mws]{}{}Class example: 10
    January\EndLink
\end{itemize}

\section{Final Exam}\\
\begin{itemize}
 \item \Link[Webview/Examples/final.mws]{}{}Final Exam Worksheet
    \EndLink
    \Picture[new]{new.gif}
\end{itemize}

\HCode{<HR>}
For feedback, please feel free to
\Link[mailto:cfwas@eiu.edu]{}{}e-mail me\EndLink.\\[0in]
All materials copyright 2002.
\end{document}
```

Figure 12: Source file for html generation.

References

[1] David Carlisle, *Packages in the graphics bundle*, CTAN, January 1999.

[2] Michel Goosens and Sebastian Rahtz, *The LaTeX web companion*, Addison-Wesley, 1999.

[3] Michel Goosens, Sebastian Rahtz, and Frank Mittelbach, *The LaTeX Graphics Companion*, Addison-Wesley, 1997.

[4] Klaus Gunterman, *PPower4 Manual*, www-sp.iti.informatik.tu-darmstadt.de/software/ppower4.

[5] Jim Hafner, *The* FoilTeX *class package*, CTAN, August 15, 1998.

[6] Helmut Kopka and Patrick W. Daly, *A guide to LaTeX, 3rd edition*, Addison-Wesley, 1999.

[7] Andreas Matthias, *The* pdfpages *package*, CTAN, April 23, 2002.

[8] Scott Pakin, *The comprehensive LaTeX symbol list*, CTAN, July 2, 2001.

[9] Scott Pakin, purifyeps *manual page*, CTAN, April 2002.

[10] Keith Reckdahl, *Using imported graphics in LaTeX2e*, CTAN, December 15, 1997.