The homework class and style*

Matt Bauman
mbauman@gmail.com

March 7, 2012

Abstract

This package contains a both a class and a style designed to simplify the authoring of schoolwork, homework and assignments. They may be used independently of each other; the class provides some slight modifications to the article class, while the style adds commonly used packages and functionalities.

Contents

1 Introduction

The homework package aims to put together a thorough and modern solution for the authoring of schoolwork, homework and assignments. I (Matt Bauman) am by no means a LaTeX expert. I have, however, made my best effort in creating

*This document corresponds to homework v0.1, dated 2012/03/01.

a package that uses the current best-practices in LaTeX authoring. This means that homework relies on some relatively new packages (like ε-TeX) that may not be available on older systems and some web-based compilers.

2 The homework class

The homework class is provides minor enhancements and modifications to the article base class. Only alterations which cannot be reliably achieved across a variety of document classes are implemented here.

2.1 Arbitrary section numbering

Most notably, the homework class allows for the user to define arbitrary section numbers. As homework assignments are very closely tied to the *number* of the problem, relying on automatic sequential numbering can be problematic. Additionally, problem numbers are not always sequential or even sensible. Thus, the homework class augments the standard `\section`, `\subsection`, etc, syntax to optionally specify its number. For simplicity, I will describe everything in terms of `\section`, but this modification applies to all sectioning commands.

The optional argument of the original syntax `\section[toc-name]{sec-name}` is changed to allow a prefix `[number|toc-name]`. Recall that the `toc-name` is how the section will be reported to the table of contents and headers, and that when omitted, it is the same as `sec-name`. This addition attempts to be as compatible as possible with the original syntax. If a `|` character appears within the optional argument, then everything before it is considered the 'number' and everything after is the section name for the table of contents. Note that a `|` character may be 'hidden' by enclosing it within a double `{{}}` group (Todo: is it possible to achieve this with just one group?), in which case it is no longer considered the separator.

Note that 'empty' parts of the optional argument are handled differently, depending upon which part was omitted. If the `toc-name` is omitted, e.g., `\section[number|]{sec-name}`, then the section name is used as the name for the table of contents. If, however, the `number` is omitted and the `|` remains, e.g., `\section[|toc-name]{sec-name}` then the section number is set to be empty.

2.2 Class options

In addition to the standard options provided by article, the homework class adds a screen/print option pair. These mutually exclusive options do not have very much functionality in the class currently. They do, however, change the default sidedness of the article class (screen defaults to oneside and print defaults to twoside). In addition, the homework style uses this switch to configure some options for the hyperref package.

2.3 Implementation

1 ⟨*class⟩

2.3.1 Setup

The etoolbox package is required for some of the operations within this class file, including `\newtoggle`, `\ifcsundef` and `\ifstrempty`.

```
2 \RequirePackage{etoolbox}
```

2.3.2 Class option handling

```
3 \newcommand{\hw@sidedness}[1]{\def\hw@side{#1side}}
4 \newtoggle{hw@print}
5
6 \DeclareOption{print}{\toggletrue{hw@print}   \hw@sidedness{two}}
7 \DeclareOption{screen}{\togglefalse{hw@print} \hw@sidedness{one}}
8 \DeclareOption{oneside}{\hw@sidedness{one}}
9 \DeclareOption{twoside}{\hw@sidedness{two}}
10
11 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
12
13 \ExecuteOptions{11pt,screen}
14 \ProcessOptions\relax
15
16 \LoadClass[\hw@side]{article}
```

2.3.3 Arbitrary section numbering

To implement this, first, save the kernel `\@sect` command as `\@@sect`.

```
17 \let\@@sect\@sect
```

Then, redefine `\@sect` to call the function that will handle the parsing and implementation of the new syntax. Add two `|` at the end of the optional argument to ensure that there will *always* be at least three parts separated by `|`.

```
18 \def\@sect#1#2#3#4#5#6[#7]#8{\hw@sectsplit{#1}{{#2}{#3}{#4}{#5}{#6}}[#7||]{#8}}
```

The `\hw@sectsplit` macro is the meat of the implementation of arbitrary numbering. It parses the optional argument into three parts, `#3`, `#4`, and `#5`.

```
19 \def\hw@sectsplit#1#2[#3|#4|#5]#6{
```

As the `\thesection` (or `\thesubsection`, etc, but for simplicity, I will describe this in terms of `\section`) macro is overwritten whenever a custom number is used, we need to ensure that the original value is saved. The first time a sectioning command is called, we save this value into `\hw@theorigsection`.

```
20   \ifcsundef{hw@theorig#1}
21     { \csletcs{hw@theorig#1}{the#1}\csletcs{hw@theprev#1}{the#1} }
22     { \relax }
```

To allow the user to redefine `\the*` in the middle of a document, compare it to `\hw@theprev*`. If they are different, then the user has changed it.

```
23   \ifcsequal{the#1}{hw@theprev#1}
24     { \relax }
25     { \csletcs{hw@theorig#1}{the#1}\csletcs{hw@theprev#1}{the#1} }
```

Now we must parse the optional argument. Argument `#5` simply absorbs any extra `|`s. If it is empty, then that means that there were no `|`s in the input, and only a `toc-name` was specified. In this case, simply ensure that `\thesection` is defined as its original definition and call the kernel's `\@sect` using the defined `toc-name`.

```
\ifstrempty{#4#5}
  {
    \csedef{the#1}{\csexpandonce{hw@theorig#1}}
    \@@sect{#1}#2[{#3}]{#6}
  }
```

If, however, argument `#5` was not empty, then the user is calling the new custom syntax. We define the `\thesection` as argument `#3` and then call the kernel's `\@sect` command. If argument `#4` is empty, use the default `toc-name`. Otherwise, use the input provided by the user in argument `#4`.

```
  {
    \csedef{the#1}{#3}
    \csletcs{hw@theprev#1}{the#1}
    \ifstrempty{#4}
      {\@@sect{#1}#2[{#6}]{#6}}
      {\@@sect{#1}#2[{#4}]{#6}}
  }
}
```

2.3.4 Document titling

I find the amount of whitespace above the title excessive in most cases. This attempts to patch the kernel's `\@maketitle` command to omit the initial `\vskip`.

```
\patchcmd{\@maketitle}{\null\vskip 2em}{\null}{}{}
```

Define more convenient synonyms for the small font sizes

```
\let\Small\footnotesize
\let\SMALL\scriptsize
```

⟨/class⟩

3 The homework style

As the homework style is mostly composed of other packages, the implementation serves nicely as its only documentation section.

⟨*package⟩

Load fixes to LaTeX 2_ε right away.

```
\usepackage{fixltx2e}
```

3.1 Font and encoding setup

Fonts are convoluted and tricky, and they have changed a lot since TEX and METAFONT first appeared. A lot of information online is outdated. One major change is that fonts used to be bitmaps, and now they are almost all vectorized – font sizing shifted from discrete to continuous. In addition, the world has gotten much better at supporting non-English languages through different font and text encodings. LATEX still defaults to using bitmap fonts and the old encoding schemes.

There are three major advantages for an English language writer to switch to the newer font encoding `T1`.

1. Accented characters in the output may be selected, copied and searched. Contrast Gödel with Gödel. (This only simulates its rendering, but I believe it should be accurate. Try it!)

2. Words with accented characters are properly hyphenated. See: Süpercálîfragilísticëxpialïdoçious.

3. The characters `<>|` in the text source appear properly as <>| in the output, instead of ¡¿—. (They have always worked properly in math mode.)

I highly recommend it.

This does, however, have an impact on the available fonts; they must match the encoding. Not all fonts are available in all encodings. Generally, if you wish to use a font other than Knuth's Computer Modern, you may have greater success using XeLATEX to typeset your document. It vastly simplifies font selection, and enables system fonts to be used in the document.

As I understand it, there are three major descendants of Knuth's original Computer Modern font:

Blue Sky Computer Modern A private company, Blue Sky, took Knuth's original source and spent many hours hand-tuning the hinting, and were selling it. AMS found the improvements very worthwhile and bought the rights to the font, allowing them to freely distribute it in their package. This is the default font in most (all?) modern LATEX distributions, but it is unfortunately not encoded in T1.

CM-Super was the first effort in converting Computer Modern to the T1 encoding. It is based on the EC, or European, variant, with many many more available symbols. This is the default font when using a T1 encoding.

Latin Modern is derived from both cm and cm-super. It is generally regarded as superior to cm-super, as its vectorization was done by hand and it includes some much needed fixes to the font metrics. It has many more available glyphs and has continued development. See "An exploration of the Latin Modern fonts" by Will Robertson (pdf) for many more details.

When using cm or cm-super, you should load the fix-cm package *before* you call the document class. This enables continuous scaling of the fonts and applies some

other fixes. But homework shall default to Latin Modern and T1 for the reasons listed above (when not using XeLaTeX).

```
\usepackage{ifxetex}
\ifxetex
  \usepackage{amssymb} % Load this first due to an incompatibility
  \usepackage{xltxtra}
\else
  \usepackage{lmodern}
  \usepackage[T1]{fontenc}
```

Just as font encodings have changed in the last 20 years, so has the encoding of text files. TeX assumes an ASCII source file. However, most every plain text document today is unicode. While the new encoding is backwards compatible and is generally readable as ASCII, explicitly telling LaTeX that the document is unicode allows for more complicated characters in the source. For example, it enables the writing of accents directly: `\verb|àéîöüñ|`. Even crazier characters may be easily typeset (provided you know how to input them): `\verb|¡£¢§¶•ªº´®ßµ|`. XeLaTeX supports this with the xltxtra package loaded above.

```
  \usepackage[utf8]{inputenc}
```

As a final modification to TeX's font handling, load the microtype package. In order to justify the text, TeX adjusts the spacing between words to make the right column properly aligned. Microtype applies a few minor adjustments. The option `stretch=10` means that TeX will stretch the font itself, up to 10 % (the default is 20 %, but I find that a bit excessive). This is barely noticeable, and often results in smaller (and more acceptable) inter-word spacing. The `protrusion=true` option allows punctuation to slightly hang over the right column. This makes text appear to align *better* along the right margin. This package is currently under development for full XeTeX compatibility.

```
  \usepackage[stretch=10,protrusion=true]{microtype}
\fi
```

The textcomp package allows easy access to some nice glyphs, such as `\textmu` (µ), bullets (• † ∗), arrows (↓ →), currency symbols and other symbols that are traditionally mathmode-only. See the venerable `symbols-a4.pdf` for a full listing.

```
\usepackage{textcomp}
```

3.2 Maths

The canonical way of doing Mathematics in LaTeXis with amsmath and its associated packages. That said, there are a few bugs that they are unable to fix; mathtools provides these fixes as well as a few much-needed enhancements. This includes fixing `\underbrace` and `\overbrace`, providing alignment options to `\matrix*`, and centering the colon in := (contrast with :=), and more.

```
\usepackage{amsmath,amsthm,amssymb}
\usepackage{mathtools}
```

Redefine the equation environment to be the same as gather for compatibility with hyperref:

```
\let\equation\gather
\let\endequation\endgather
```

Another augmentation for amsmath is the empheq package. It allows for many sorts of decorations on equation blocks. Most notably, this includes bracketed equations with `\begin{empheq}[left=\empheqlbrace]{align*}` and boxed equations with `\begin{empheq}[box=\fbox]{align}`.

$$\begin{cases} x + y + 5 = 0 \\ x - 3y + 2 = 0 \end{cases}$$

$$\boxed{\begin{aligned} x &= \sqrt{2} \\ y &= 3/2 \end{aligned}}$$

```
\usepackage{empheq}
```

The dsfont package is my preferred choice for typesetting the standard number sets: `\mathds{R...}` $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{N}$.

```
\usepackage{dsfont}
```

Additionally, I find the default script fonts lacking. I find Ralph Smith's Formal Script font, rsfs much more appealing in some cases. Compare `\mathcal` with `\mathscr`: $\mathcal{F}$, $\mathcal{L}$ $\mathscr{F}$, $\mathscr{L}$. Unfortunately, the mathrsfs package does not load the font for variable sizing, even though it supports it. See http://tex.stackexchange.com/q/10698 for details; thanks to user Leo Liu for this solution. (This could potentially be split off into a simple fix-rsfs package, mirroring fix-cm.)

```
\DeclareFontFamily{U}{rsfs}{\skewchar\font127 }
\DeclareFontShape{U}{rsfs}{m}{n}{ % Allow continuous sizing
   <-6>  rsfs5
   <6-8> rsfs7
   <8->  rsfs10
}{}
\usepackage{mathrsfs}
```

The xfrac package provides very high quality slanted fractions, provided through the `\sfrac` command: $\sfrac{1}{2}$. It is the generally accepted replacement to nicefrac. There are lots of configuration options, but the defaults work very well.

```
\usepackage{xfrac}
```

I prefer the variants for the Greek ϑ, φ, and ε. This makes the variants the default.

```
\newcommand{\sch@swap}[2]{\let\sch@tmp#1 \let#1#2 \let#2\sch@tmp}
\sch@swap{\theta}{\vartheta}
\sch@swap{\phi}{\varphi}
\sch@swap{\epsilon}{\varepsilon}
```

3.3 Science

The siunitx package is a marvelously designed and very flexible way to display numbers with units attached. It does spacing properly, and formats all the units consistently throughout the document. Some examples:

- `\SI{3}{\meter\per\second}` → $3\,\mathrm{m/s}$
- `\SI{1.2}{\micro\ampere\per\centi\meter\squared}` → $1.2\,\mu\mathrm{A/cm^2}$
- `\SI{\sqrt{2}}{kW/m^2}` → $2\,\mathrm{kW/m^2}$
- `\SI{95.4}{\percent}` → $95.4\,\%$
- `\SI{37.5}{\degreeCelsius}` → $37.5\,^\circ\mathrm{C}$

You can also make it automatically detect quotients in the value and use `\sfrac` to format them:

```
\sisetup{quotient-mode = fraction, fraction-function = \sfrac}
```

```
\usepackage{siunitx}
\sisetup{per-mode = symbol}
```

Similarly, mhchem provides very smart chemical formulas and equations. `\ce{(NH4)2S}` → $(\mathrm{NH_4})_2\mathrm{S}$.

```
\usepackage[version=3]{mhchem}
```

3.4 Graphics and Figures

```
\usepackage[svgnames]{xcolor}
\usepackage{graphicx}

% Figure handling
\usepackage{float}    % Allow "unfloating" with the H placement specifier
\usepackage{wrapfig}
% \floatstyle{boxed}
% \restylefloat{figure}
\usepackage[small,labelfont=bf]{caption}
% \DeclareCaptionFont{singlespacing}{\setstretch{1}}
% \captionsetup{font=singlespacing}

\usepackage{placeins} % Allow \FloatBarrier

% Package for including code in the document
\usepackage{listings}
% For faster processing, load Matlab syntax for listings
\lstloadlanguages{Matlab}
\newcommand*{\matlabuserfunctions}[1]{
  \lstset{language=Matlab, morekeywords=[3]{#1}} }
\lstset{language=Matlab,
        frame=single,
```

```
        basicstyle=\footnotesize\ttfamily,
        keywordstyle=[1]\color{Blue}\bfseries,
        keywordstyle=[2]\color{Purple},
        keywordstyle=[3]\color{Blue}\underbar,
        identifierstyle=,
        commentstyle=\footnotesize\ttfamily\itshape\color{Green},
        stringstyle=\color{Purple},
        showstringspaces=false,
        tabsize=5,
        % Put standard MATLAB functions not included in the default
        % language here
        morekeywords={xlim,ylim,var,alpha,factorial,poissrnd,normpdf,normcdf},
        % Put MATLAB function parameters here
        morekeywords=[2]{on, off, interp},
        % Put user defined functions here
        % morekeywords=[3]{},
        morecomment=[l][\color{Blue}]{...},
        numbers=left,
        firstnumber=1,
        numberstyle=\footnotesize\color{Blue},
        stepnumber=5
        }
\newcommand*{\matlabscript}[2]
  {\begin{itemize}\item[]\lstinputlisting[caption={\texttt{#1.m}. #2},label={lst:#1}]{#1.m}\end
\usepackage[marginpar]{todo}
% \iftoggle{hw@print}
%   {\usepackage{hyperref}}
\usepackage[colorlinks,linkcolor=blue]{hyperref}
\newcommand*{\magicref}[2]{\hyperref[#2]{#1 \ref{#2}}}
\newcommand*{\hwClass}[1]{\def\@hwClass{#1}}
\newcommand*{\hwTitle}[1]{\def\@hwTitle{#1}}
\title{\textbf{\@hwClass:} \@hwTitle}
\usepackage{tikz}
\usepackage{pgfplots}
\pgfplotsset{compat=1.4}
```

⟨/package⟩