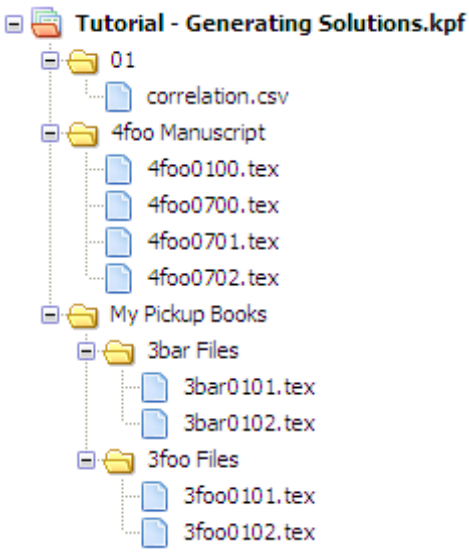
Generating a Solutions Manual

This tutorial shows how to create a solutions manual. To get started, copy the "Generating Solutions" directory to a temporary directory in your computer (for example, C:\working or "My Documents"). Then create a new Komodo project file in this directory. Also, open the file "Flowchart-Generating Solutins" PDF file under the Docs directory of the Tarp distribution.

In this tutorial, we will be creating chapter 01, sections 01 and 02 of the "4foo" book (fourth edition of the "Foo" book) using files from the "3foo" and "3bar" books (third edition of the "Foo" and "Bar" books, respectively).



The files at right will be visible in Komodo's "Projects" tab. The foler "01" is where we will create the 4foo*.tex files. It contains a file called "correlation.csv", which maps the pickup exercises and the new edition exercises. The remaining folders, "4foo Manuscript", "My Pickup Books" (and its contents), are where the current edition manuscript files and previous edition solution files are stored. In this tutorial, they are bundled together with the 4foo solutions directories; however in your set up these will be located somewhere else in your machine or maybe on another computer accessible through the network. This is important so I willl say it again:

> The folders "My Pickup Books" and "4foo Manuscript" are not hard wired. You will be using something like \\SERVER1\Pickups\4foo\ or C:\Manuscript Files\Foo\4Foo Manuscript\ instead.

How It Works

The file "Flowchart: Generating Solutions" shows the general steps in generating a solutions manual. In general, the task is to get the old files (aka "pickup" files), and following the instructions in the list, rearrange, add or remove exercises as needed in order to produce our new files. A new .tex file may come from several pickup files, and can also include new exercise content inserted from a template.

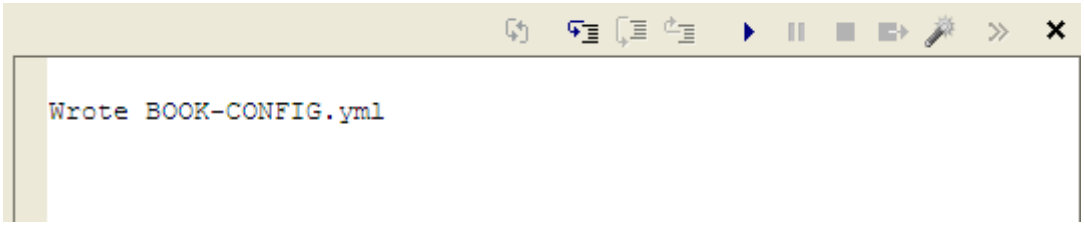
The list at the left of the chart shows what we need in order to create a solutions manual. The Komodo project we have opened contains three of these items: a CSV called correlation.csv, 3foo and 3bar pickup files, and 4foo manuscript files. We are missing two .tas files, "solutions-starter.tas" and "TASfile.tas". More on these later. For now, notice that there are four black boxes in the diagram, showing the four programs we will use to generate the new .tex files. These are "talatextract", "talatexcombine", "tagentex", and "solutions-starter.pl" The first three are installed with Tarp and can be used from the command line or from Komodo. The last one, "solutions-starter.pl", is a Perl script that we need to copy over to our working directory.

In Tarp, there are two kinds of programs: applications and *.pl scripts. Applications are used by opening a .tex or .pklist file in Komodo and clicking on the icon in Tarp Toolkit.kpr. *.pl scripts are used by copying them over from the Scripts folder of the Tarp disbritubion to your working directory, opening them in Komodo and running them.

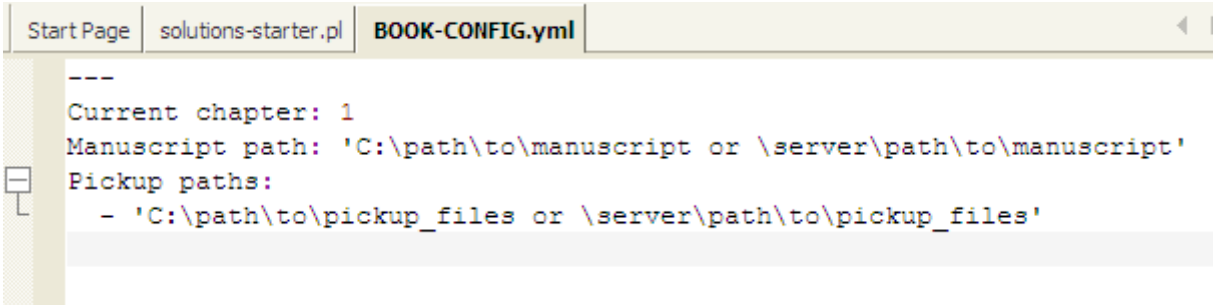
To complete the list of requirements, go to the Scripts folder in the Tarp distribution and copy "solutions-starter.tas" and "solutions-starter.pl" to the root directory of the "Tutorial – Generating Solutions.kpf" project.

Now, in Komodo, double click on solutions-starter.pl. This opens it in the main editing window. You will not need to edit anything in this script, but may want to read the instructions.

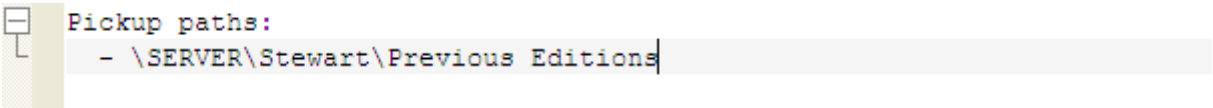
Select "Run without debugging" from the "Debug" menu or press Ctrl+F7. The following message appears:



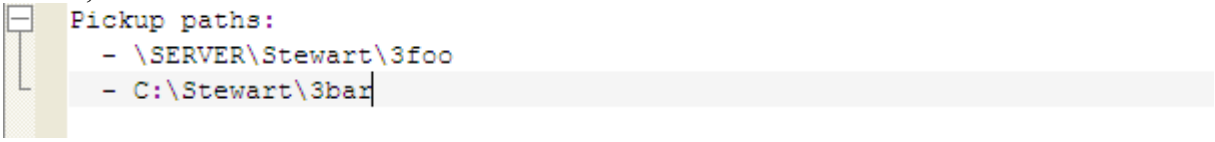
You will also see BOOK-CONFIG.yml on the left hand "Projects" pane. Open BOOK-CONFIG.yml. The contents will be as follows:



This step will be performed only once per book. Every time you start a new chapter, set "current chapter" to the subdirectory corresponding to that chapter. The manuscript path and pickup paths are also set once per book, and should correspond to the manuscript and pickup locations **for the entire book, not for one particular chapter.** The pickup paths may be a single entry, for example:

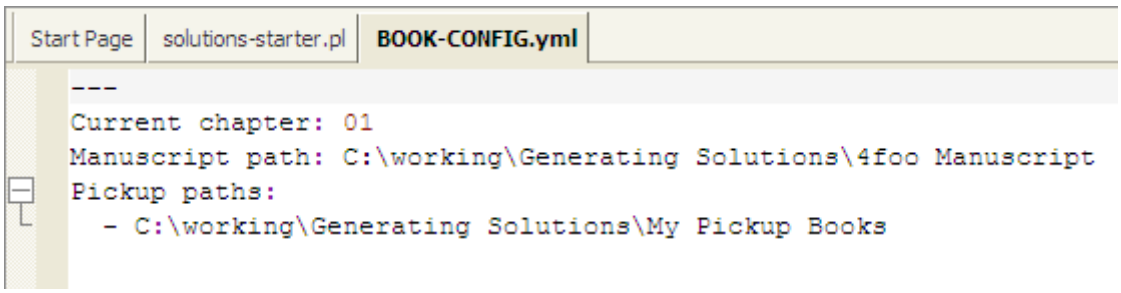


corresponding to the directory that contains all pickup .tex files for all he Stewart books ever produced, or

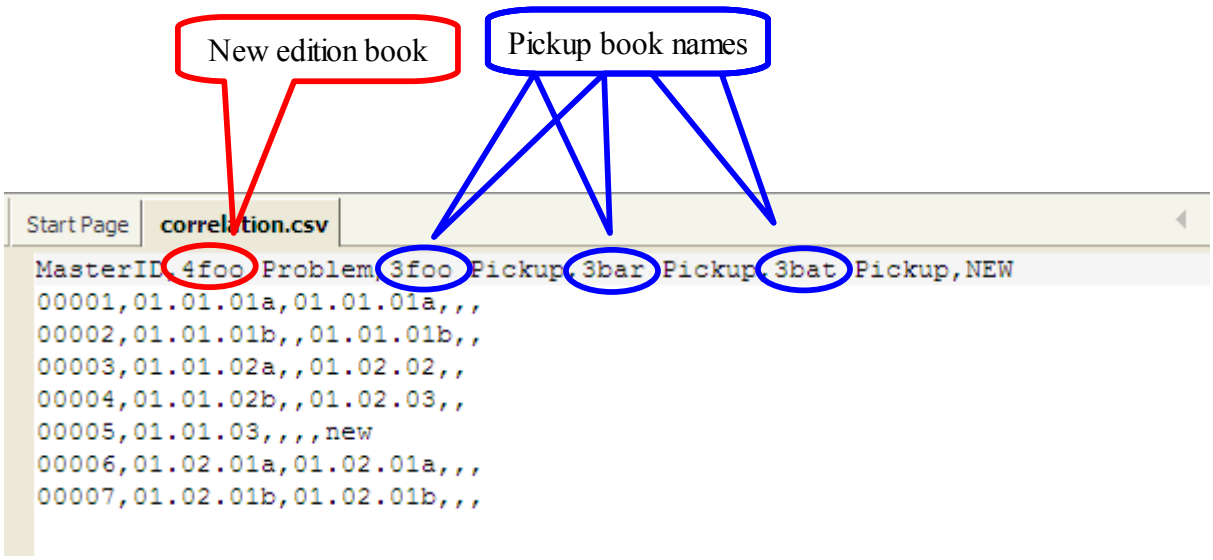


if the pickups directories do not share a parent directory. Network shares are OK.

Edit BOOK-CONFIG.yml until as shown below, replacing "C:\working" with the directory where you copied the tutorial files.



Now we are ready to start on chapter "01". Close BOOK-CONFIG.yml and open up the "correlation.csv" file in the "01" directory.



The heading of the correlation .csv file must be in this format. The new edition problem column ends in "Problem", the pickup columns all end in "Pickup", and the NEW and MasterID columns must have exactly this heading. The order of the columns, however, is irrelevant. The file may contain other columns, like TEC or GCALC but these are ignored.

By examining correlation.csv we can see that we will produce two new sections, 4foo0101 and 4foo0102, from the contents of 3foo0101, 3bar0101, 3bar0102 and 3foo0201. This information is difficult to figure out in the csv file, however, so we will create "pickup lists" (.pklist files) for each section instead.

Pickup list (.pklist) files

A pickup list is a series of instructions that will tell some of the programs in the Techarts Toolkit (we will get to which ones in a minute) where each exercise comes from. Pickup lists contain exactly four fields, as follows:

```
EX PKFILE PKEX MASTER
```

The first column, `EX`, contains the new file exercise number. The second, `PKFILE`, contains an ID for the pickup file where the new edition exercise is to be picked up from. The next column, `PKEX`, contains the exercise within the pickup file to be copied over. Finally, the `MASTER` column contains a master number corresponding to that exercise.

Generating .pklists

To create the pickup lists, simply run solutions-starter.pl again. The following message appears:

```
Reading 'correlation.csv'...

Identified the following columns (zero indexed):
NEW column: 5
New edition column: 1
Pickup column(s): 2 3 4
MasterID column: 0

Wrote 4foo0101.pklist
Wrote 4foo0102.pklist
Wrote CHAPT-CONFIG.yml
```

Two pickup lists, 4foo0101.pklist and 4foo0102.pklist, now exist in the "01" directory. A file called CHAPT-CONFIG.yml was also created. Open 4foo0101.pklist. It contains:

Start Page | correlation.csv | **4foo0101.pklist**

```
01a 3foo 01a 1
01b 3bar 01b 2
02a 3bar_2 02 3
02b 3bar_2 03 3
03 new .. 4
```

It says that exercises 01a and 02b are picked up from two different files, one from the "3foo" book and one from the "3bar" book, 02a and 02b from another file from the "3bar" book, and 03 is a new exercise. There is no mention of which files "3foo", "3bar", and "3bar_2" correspond to. This information is in CHAPT-CONFIG.yml. This file contains the following:

Start Page | correlation.csv | **CHAPT-CONFIG.yml**

```
---
Manuscript master file: 00.tex
Pickup files:
  01:
    3bar: 3bar0101.tex
    3bar_2: 3bar0102.tex
    3foo: 3foo0101.tex
    new: '(virtual)'
  02:
    3foo_2: 3foo0102.tex
```

Here we see that "3foo" is "3foo0101.tex", "3bar" is "3bar0101.tex", and "3bar_2" is "3bar0102.tex". You can verify this information in the correlation.csv file. CHAPT-CONFIG.yml also contains an entry called "Manuscript master file". This is the section 00 file for this chapter of the new edition book (inside the manuscript path specified in BOOK-CONFIG.yml) that contains the chapter name. As well, it tells solutions-starter.pl to look for other .tex files in the same directory as the 00 file that may contain other titles. This is further explained below.

The next step is to get these files and put them in the current directory. Solutions-starter.pl does this for us as well. We just need to verify that the filenames in CHAPT-CONFIG.yml are correct, and solutions-starter.pl will traverse the subdirectories of the pickup paths specified in BOOK-CONFIG.yml, copying every pickup file into the 01 directory.

Run solutions-starter.pl again. Some messages will appear about files being copied, ending with the following.

```
Books/3foo Files/3foo0102.tex' to '3foo0102.tex'
4 file(s) copied
Getting titles...
Found manuscript master file '4foo0100.tex' in
'C:\working\Generating Solutions\4foo Manuscript'
4foo0100.tex: CHAP_TITLE
4foo0101.tex: SECT_TITLE
4foo0102.tex: SECT_TITLE

Pickups Per Section:
        01: OK
        02: OK
Have 4 / 4 (100%) of the pickup files needed for this
chapter.
```

After finding "4foo0100.tex" in the "4foo Manuscript" directory, solutions-starter.pl used this directory to look for titles.

Getting Titles from Manuscript Files

The titles are searched as follows. First, filenames are matched against the "ms_filename" entry in solutions-starter.tas. This entry contains the following:

Start Page | **solutions-starter.tas**

```
# Manuscript file name

ms_filename = \
    $book$$chapt$$sect_0$\.tex    \ # e.g. 4c0100.tex
    $book$$ch$$sect$\.tex         \ # e.g. 4c0101.tex
    $book$$ch$$proj$\.tex         \ # e.g. 4c01ap01.tex
    $book$$ch$$special$\.tex        # e.g. 4c01r
```

There are four entries, each one matching a different type of filename. Each entry contains variables, which are couched between dollar signs. Variable definitions appear below the ms_filename entry. Each variable definition is a regular expression. Solutions-starter.pl matches the name of each file in the manuscript directory, starting with the section 00 file. This file matches the first ms_filename entry. Then, the contents of the section 00 file are then matched line-by-line with the values of an entry with the same name as each variable in the matching entry with "_search" appended. Got it? Maybe not.

Let's take the 4foo0100 file from this tutorial as an example. This file matches the first ms_filename entry, $book$$chapt$$sect_0$\.tex. Each line in the 00 file is then matched against the entries book_search, chapt_search, or sect_0_search. book_search and sect_0_search do not exist; however, chapt_search is defined as follows:

Start Page | **solutions-starter.tas**

```
chapt_search = \\chapter{\d\d?\\quad\s$CHAP_TITLE$}

    chapt_search::CHAP_TITLE = .*
```

If a line matches this regular expression, any variable contents are saved. The only variable in this expression is CHAPT_TITLE, and the value it intends to catch happens to be the chapter title

shown below.

Start Page | solutions-starter.tas | **4foo0100.tex**

```
\chapter{7\quad Differential Equations}

%TCIMACRO{\TeXButton{set page 1}{\setcounter{page}{1}}}%
%BeginExpansion
\setcounter{page}{1}%
```

So CHAPT_TITLE gets "Differential Equations". Then the search continues with 4foo0101.tex, which matches the second entry in ms_filename, $book$$ch$$sect$\.tex. The content of this file is matched against book_search, ch_search and set_search, of which only sect_search exists and is defined as follows:

Start Page | **solutions-starter.tas** | 4foo0100.tex | 4foo0101.tex

```
sect_search = \\section{.*\\quad\s$SECT_TITLE$}

    sect_search::SECT_TITLE  = .*
```

This matches the section title, "Modeling with Differential Equations", as shown below.

Start Page | solutions-starter.tas | 4foo0100.tex | **4foo0101.tex**

```
\chapter[7\quad Differential Equations]{}

\section{7.1\quad Modeling with Differential Equations}

%TCIMACRO{\TeXButton{setCCC}{\renewcommand{\CCC}{1}}}%
%BeginExpansion
\renewcommand{\CCC}{1}%
```

The same process is repeated for 4foo0102.tex, getting another value for SECT_TITLE. So where are these variables actually saved? The answer is, they are inserted into CHAPT-CONFIG.yml.

Start Page | **CHAPT-CONFIG.yml**

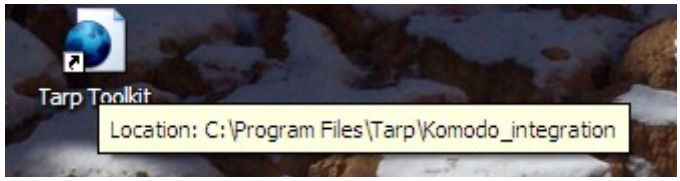
```
titles:
  01:
    CHAP_TITLE: 'Differential Equations'
    SECT_TITLE: 'Modeling with Differential Equations'
  02:
    CHAP_TITLE: 'Differential Equations'
    SECT_TITLE: 'Direction Fields and Euler''s Method'
```

One thing should jump out at you: out of sections 00, 01, and 02, only sections 01 and 02 show up in the list. Also, CHAPT_TITLE is duplicated in both 01 and 02. This is because we identified 4foo0100 as the "master" file, and all variables in the master file are "promoted" to appear in all sections. The master file does not have an entry of its own, though.

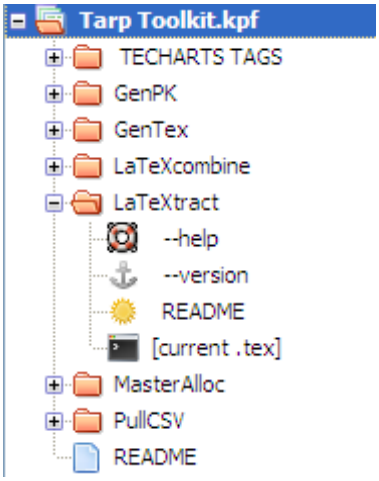
So now we have all we need in order to proceed: two pickup lists, four pickup .tex files, and

CHAPT-CONFIG.yml containing mappings between the pickup lists and these .tex files plus a few titles. Now we are at the end of the "set up" phase for chapter 01 and are ready to follow the steps in the "Generating Solutions" flowchart.

The first step shown in the flowchart is to run LaTeXtract on all input .tex files. This ensures that Tarp can identify the exercises in these files correctly. To run LaTeXtract, open the "Tarp Toolkit" Komodo project, which is accessible through a link on your Desktop.

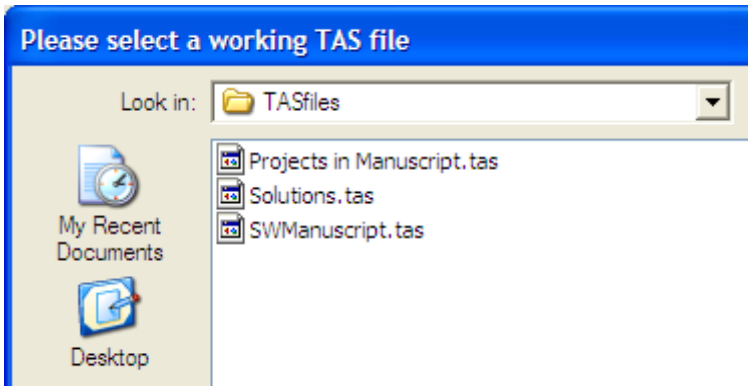


Double clicking on this link opens the project in Komodo.

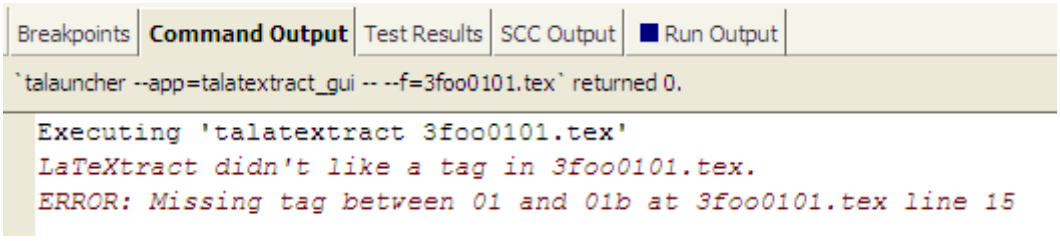


The project contains folders for all the main Tarp applications. Each application contains a few shortcuts: "help", "version", "README", and finally at least one shortcut to run the program. In the case of LaTeXtract, it is labeled [current .tex]. This indicates that we are going to open a .tex file, then double click on [current .tex] to latextract it. Open 3foo0101.tex and click [current .tex].

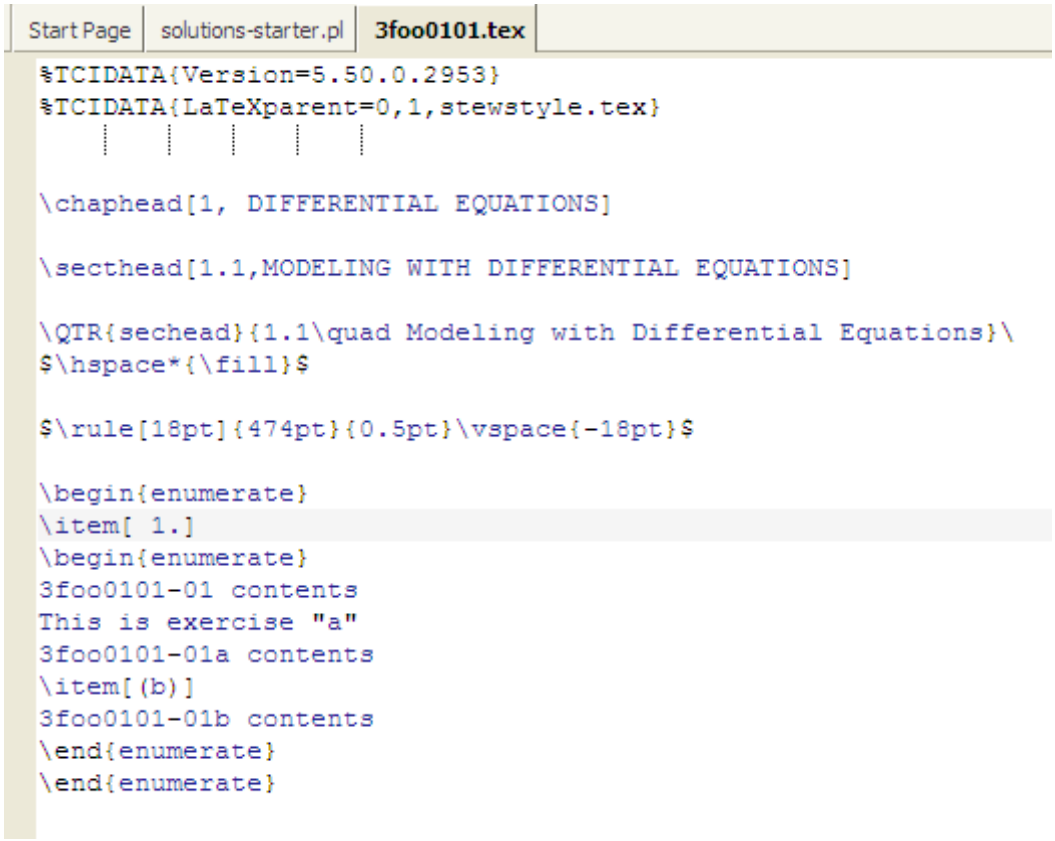
A dialog box will appear, asking you to select a working TAS file.



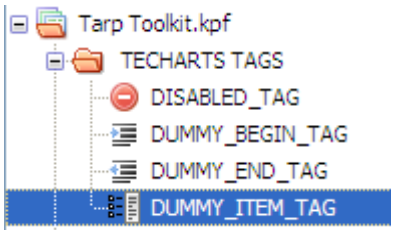
Select Solutions.tas and click OK. This will copy Solutions.tas to TASfile.tas in the current directory. Then the following message will appear.



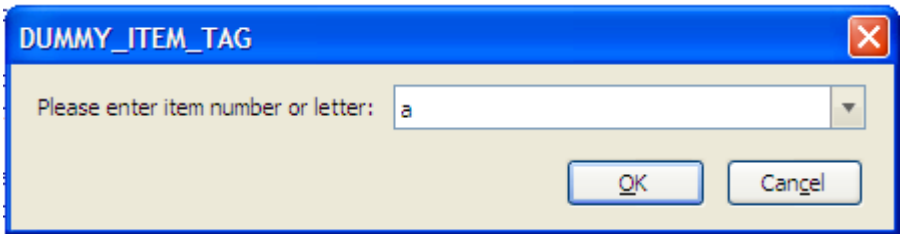
Double clicking on the last message will take you to line 15, where there is an \item tag corresponding to exercise 01.



The problem is that the tag for “a” is nonstandard: it says 'This is exercise “a”'. To allow Tarp to recognize this exercise, we will add a dummy tag just before this line. Dummy tags are in “Techarts Toolkit.kpr”, in a folder called “Techarts Tags”.



Insert a new line before 'This is exercise “a”', and double click on DUMMY_ITEM_TAG. The following message will appear. Type in an “a” in the box, and click OK.



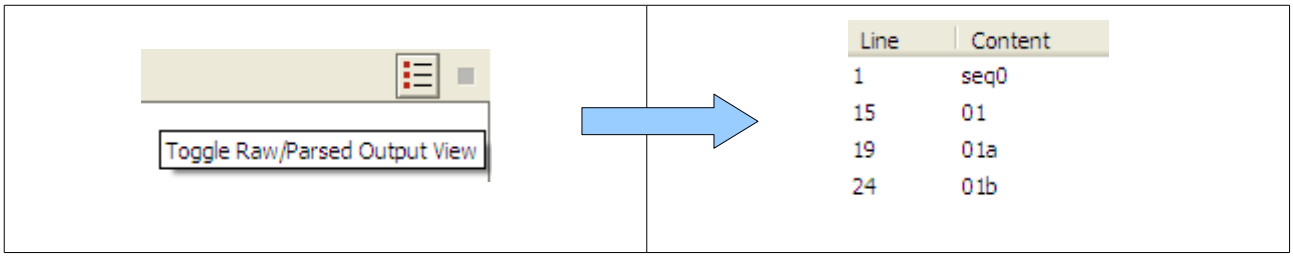
Now the exercise list will look as follows:

```
\begin{enumerate}
\item[ 1.]
\begin{enumerate}
3foo0101-01 contents
TECHARTS_DUMMY_ITEM_TAG-(a)
This is exercise "a"
3foo0101-01a contents
\item[(b)]
```

Clicking [current .tex] under LaTeXtract will result in the following.

```
Executing 'talatextract 3foo0101.tex'
seq0: 1 30
01: 15 29
01a: 19 23
01b: 24 28
```

The message that appears tells us where each exercise begins and ends. On the far right of the "Command Output" pane, there is a little icon with bullet points, for "Toggle Raw/Parsed Output View". Click on it. Now the output is displayed in columns:



Double clicking on any line will take you to where each exercise starts. Do the same with all of the 3foo and 3bar files (the pickup files). They should pass without any modification. Now we will briefly look at the content of TASfile.tas, wich is the Tarp Style file for this project.

Tarp style (TAS) files

The other file in the tut01 folder a Techarts Style (TAS) file called TASfile.tas. This file defines the way certain elements of a TeX file are written. With the TeX file above, a list containing a single exercise, 01ai, would be written as follows:

```
\begin{enumerate}
\item[\hfill 1.]
\begin{enumerate}
\item[(a)]
\begin{enumerate}
\item[(i)]
```

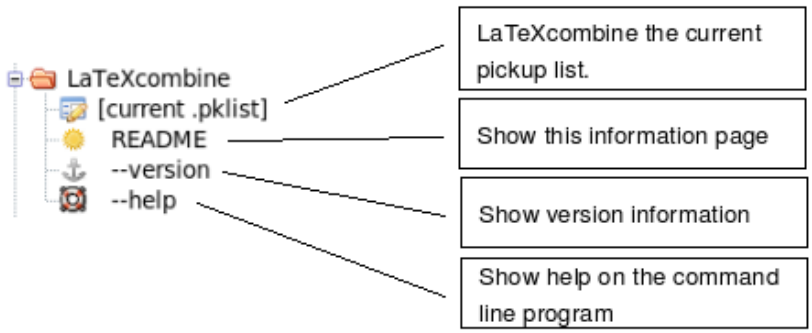
\end{enumerate}
\end{enumerate}
\end{enumerate}

The EX variable in the TAS file gets replaced with the correct exercise number according to the formatting rules that are also in the file (in this case, exercises are written as numbers, parts as letters in the latin alphabet, and subparts in roman numerals). There is more information about TAS files in the Tarp::Style manual page.

For generating solutions files, we will be using two programs: LaTeXcombine and GenTex. The former merges and rearranges exercise contents in one or more TeX files and creates a "chunk" file. A "skeleton" file is also produced, containing the exercise list for the new .tex file but without any of the contents for each exercise. The latter gets a "chunk" and a "skeleton" file and merges them into a .tex file.

Mix and match: LaTeXcombine

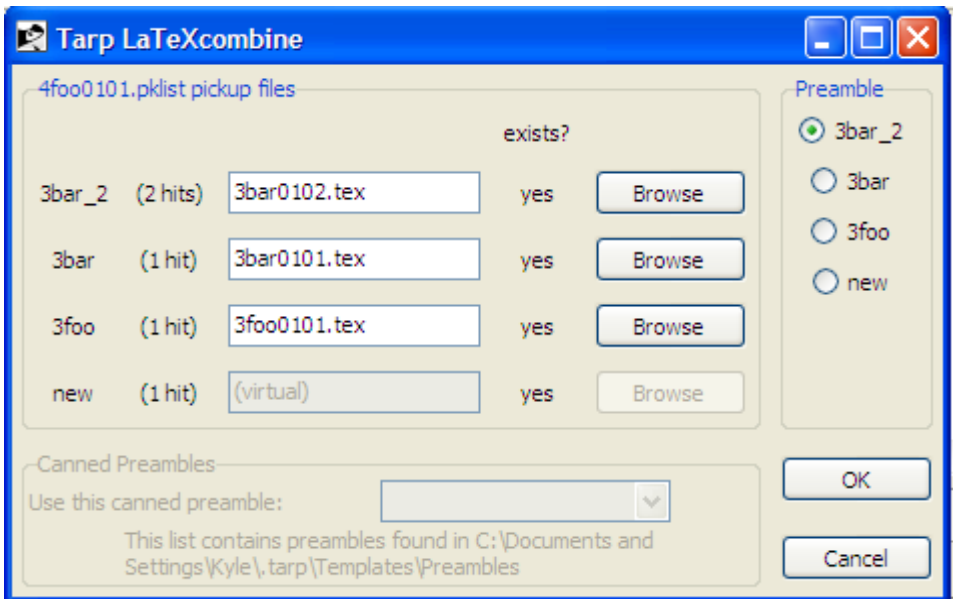
The first script we will use is LaTeXcombine. If the LaTeXcombine folder in the Techarts Toolkit.kpr is not expanded, click on the plus sign next to it to expand it. The folder has the followig items:



In order to learn what each of these options is, double click the README item. With the file 4foo0101.pklist open in the Komodo workspace, double click the [current .pklist] shortcut. The "command output" tab will show the following,

```
Got section '01' from filename '4foo0101.pklist'
Found section '01' pickups in CHAPT-CONFIG.yml
```

and this dialog will appear.



The main area to the left allows you to specify the pickup files. These were read from CHAPT-CONFIG.yml by looking up the "01" section. The "01" was extracted from the filename 4foo0101.pklist by matching it against the "filename" entry of the .tas file and using the contents of the last variable:

Start Page | TASfile.tas

```
filename = \                    #    e.g...
    \w+$CHAP$$SECT$ \           # 4foo0101.pklist
    \w+\d{2}$REVIEW$ \          # 4foo01r.pklist
    \w+\d{2}$PROJ$              # 4foopr1a.pklist

    filename::CHAP   = \d{2}
    filename::SECT   = \d{2}
    filename::REVIEW = [rR]
    filename::PROJ   = pr\d[a-z]
```

In this case, \w+$CHAP$$SECT$ matched, and the contents of the $SECT$ variable, "01", was taken to be the section.
To the right of the dialog there is a "Preamble" frame that asks you where the preamble of the output file comes from. The preamble is everything before the exercise list. By default it is taken from the pickup file with the most hits, but you can also specify a "new" pramble. Click on "new", and you will see that a third area of the dialog becomes active. This contains a drop down list of all installed preambles in your home directory. Select "standard" from the list and click OK.
The "Command Output" pane will show this message:

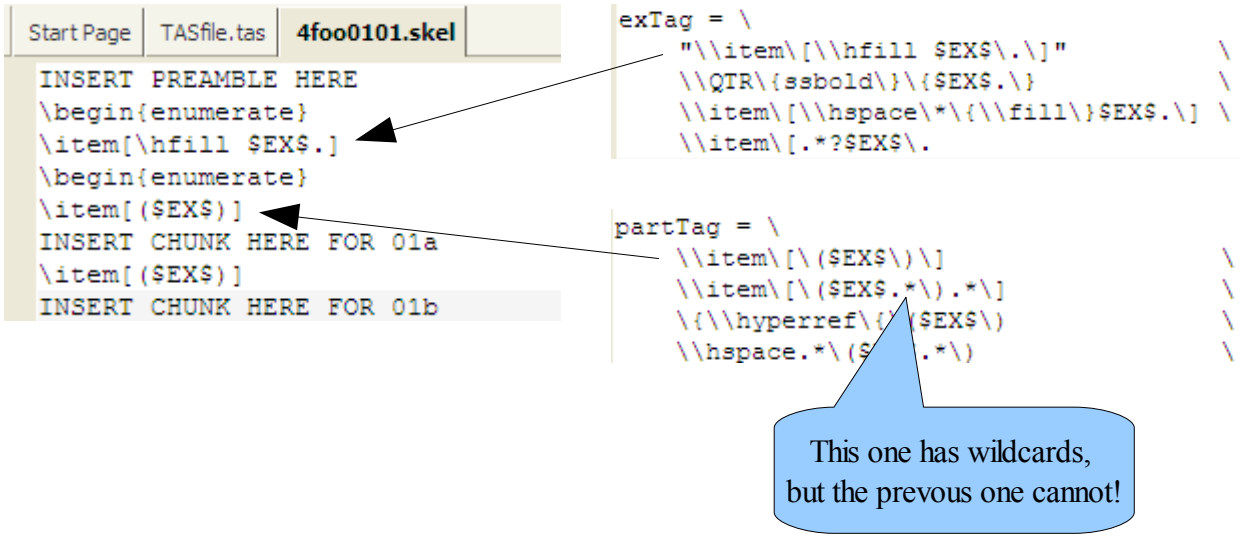
Breakpoints | Command Output | Test Results | SCC Output

`talauncher --app=talatexcombine_gui -- 4foo0101.pklist` returned 0.

```
Chunks of sequential input:
01a - 01a
01b - 01b
02a - 02a
02b - 02b
03 - 03
Wrote 4foo0101.chunk and 4foo0101.skel.
```

Two files, 4foo0101.chunk and 4foo0101.skel, have been created. The .chunk file contains the bits of the pickup files we want, while the .skel file contains a skeleton exercise list with slots where these chunks are to be inserted later.

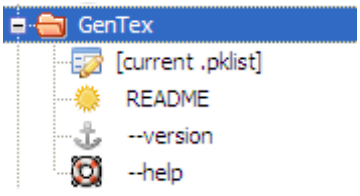
The \item tags in the .skel file are printed using the first entry in the exTag, partTag and subPartTag entries in TASfile.tas. So these entries have two functions: first, they match the exercise lists in the source .tex files, and secondly they act as templates for new content written to the .skel file. The upshot of this approach is that the first entry cannot contain any wildcards such as * or ?, because these are ambiguous when it comes to printing out. If you want wildcards (and you should want them), these have to be appended to exTag, partTag or subPartTag as additional values.



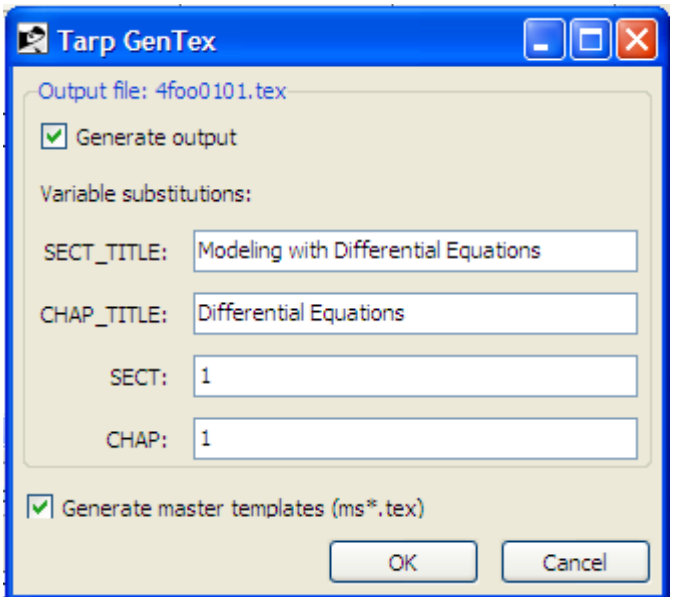
The .skel value has EX variables instead of item numbers, letters or roman numerals. The .chunk file, if a chunk encompasses more than one exercise, will also have \item tags with an EX variable, which is stripped from the original input. The remaining task is therefore to combine the .chunk and .skel files and replace EX with the actual number or letter that is specified in the pickup list. This is the job of Tarp's GenTex program.

Enter GenTex

As with LaTeXcombine, the GenTex folder in the Techarts Toolkit.kpr project contains shortcuts for commonly used features of the underlying command-line program; in this case, tagentex. The following shortcuts are provided:



With the pickup list open in the Komodo workspace, double click the first shortcut in the GenTex folder. This will pop up the following dialog.



And the command output pane will say:

```
Running 'talauncher --app=tagentex_gui -- 4foo0101.pklist'...
Got section '01' from filename '4foo0101.pklist'
Loaded CHAPT-CONFIG.yml
Got section '01' titles from CHAPT-CONFIG.yml
```

Again, the section was guessed from the filename by matching agains the 'filename' entry, and then the titles for that section were looked up in CHAPT-CONFIG.yml.

Click OK.

A file called 4foo0101.tex will now exist. Also, a directory called “new” with templates for any new solution files will have been created.

You can run LaTeXcombine and GenTex to obtain 4foo0102.tex.