2018 git based workflow *(draft)*

Peter Teuben, UMD

16th Dec 2018

# Introduction

These notes follow up on Keith’s excellent August 2017 description of the ADASS proceedings editing workflow. What we describe here are some proposed important modifications that were made to make things even easier.

The ADASS Proceedings Preparation Kit (APPK?) is now in github, and a git branch is used to isolate files the editing team is modifying to tailor to their particular situation. However, with the intention that they be merged into the master before the next year team takes over. Of course the 2019 team could also branch off the latest 2018 branch in case the 2018 team is late. Merging might now be a bit harder.

Keith’s workflow is MacOS oriented. The 2018 team is rather diverse: Linux (2), Mac (1) and Windows (1).

E.g. ImageMagick should be advertised for easy conversion into “eps” files if authors have “jpg” or “png”. Gimp also works.

To summarize, the main deviations from what I know as the 2015/16 workflow are:

1. We have an **ADASSProceedings** preparation kit in git (currently on github).
2. We have an **ADASSProceedings2018** implementation in git (currently in github). Editors will need both git trees to work efficiently.
3. We use a pre-processor (tex2inc.py) that turns a standalone paper (the “tex” file the author submits) into an “inc” file that the final aspvolume will then include. The “inc” file will never be edited by us (but potentially by the ASP team), we edit the author’s “tex” file. *I believe much of the work Finish.py does, is now done by tex2inc.py, but that needs to be reconciled.*
4. We have integrated the ASCL index, which much like the subject index, %\ooindex{} lines. This needs a change to asp2014.sty
5. We keep all the bib files by themselves, there was no need to merge them into one *(as far as we currently see).*
6. We highly recommend a “make” based system. For this a public Makefile and author based makedefs file is used. This adds one minor pain on the author to create this makedefs file, but at a huge benefit for the editors. We do have a semi-automated way to create a makedefs file.
7. On the top level, everything is directed by self-describing hierarchical Makefiles.
8. We use the TEXINPUTS environment variable to control where figures and photos are located, so no need to edit the path in “tex” file for this, as some years had to do
9. Conference photos go into the **photos/** directory, a number of papers have enough room below the bibliography to add some photos.
10. The author contributions go into the **papers/PID** directory (where PID is something like P1-12, note we didn’t use the AuthorLastName but there is no reason this cannot be used).
11. If authors really submit the correctly named tar or zip file, they can be dumped straight into the **papers/PID** directory, and the ingestion process really can be 10 seconds.
12. The preprocessed “inc” file plus “bib” and “eps” figures are later copied into one flat **authors/** directory for final book inclusion. This directory is a build directory, and no files in this directory are edited by humans.
13. To maintain the “make” environment the author uses, the editors need to use the “**../pmake**” script that emulates this environment from within each **papers/PID** directory.

# Some new scripts.

Some new scripts were developed for the 2018 ADASS:

ascl.py

tex2inc.py

The 2018 team hasn’t made use of the ADASS\_Configuration file. I propose it be a .dotfile though. Currently we use some environment variables (pmake and make use different ones), so this could be confusing and improved.

On Keith’s laptop, the configuration file looks like this:

#

#  A D A S S \_ C o n f i g u r a t i o n

#

#  Defines the location of various files used in the course of editing an

#  ADASS Proceedings volume.

#

#  Entries in this file are name/value pairs of strings, the name then the

#  value. Strings with spaces can be quoted, either using 'single' or "double"

#  quotes.

MainSubjectIndexFile "~/Trieste ADASS/Proceedings/Files/subjectKeywords.txt"

NewSubjectIndexFile "~/Trieste ADASS/Proceedings/Files/newKeywords.txt"

# Editing steps for 2018

Before editing starts, the editors needs to source the correct rc.sh or rc.csh script, to modify the TEXINPUTS, BIBINPUTS and BSTINPUTS variables. *Much like Keith’s configuration file, this is something that could be improved, e.g. modify the $PATH so it includes the directory where the python scripts are located.*

This is my new sequence, assuming the ingestion has been done, and you are located in one of the **papers/PID** directories. There will be the PID.tex and makedefs file, and optionally a PID.bib and some PID\_\*.eps files. Hopefully also a copyright file, but nothing more, and nothing less.

Open the copyright file and see if it really is a copyright file, and if it’s been signed properly (not all have).

../pmake check (runs PaperCheck.py)

../pmake fix (runs FixUnprintable.py)

../pmake pdf (re-make what is the authors’s PDF)

../pmake pdf2 (make the PDF that looks like the book: box + margin index)

xdg-open PID.pdf

*<below here should be the much improved list from Keith’s august 2017 version>*

Check that it has the right number of pages.

Check that the author list is correct and the presenting author is first.

See if there were any LaTeX errors (overfull \hbox etc, undefined references) or warnings (underful \hbox etc). (TexWorks will show you a separate sub-window with the LaTeX warnings and errors, which makes this easy.)

Make sure the running heads are correct. A lot of authors leave the title unchanged from the “Authors Final Checklist” used by the template, and many change it to the author list instead of the paper title. (The template will be improved for next year!). And make sure the title is not so long the page numbers run into the margin.

Read the paper looking for poor grammar and unclear text or illustrations.

Run a spelling checker (I just read the .tex file into Textedit and use that. I find I can ignore the errors from LaTeX directives easily enough, but a good LaTeX-aware checker would be nice. I’ve been experimenting with Excalibur, a LaTeX-aware spelling checker distributed with TexShop. but this needs to be configured for our set of LaTeX commands, and I found it slower to use than just looking at the file in Textedit.)

Fix any trivial problems, send non-trivial ones back to the author. (I define non-trivial as ones I can’t fix in less time than it takes to write a detailed e-mail explaining the problem, with an allowance for the time spent waiting for a reply etc.)

Check the names of the graphics files used – these are printed out by PaperCheck.py, – and make sure they follow the required convention. If necessary, change them.

Add %\aindex entries, one for each author, using Aindex.py

../pmake aindex

Add %\ssindex entries, as required using Index.py

../pmake index TERM=vla

Add %\ooindex entries, as suggested using ascl.py

../pmake ascl

# ASCL index entries

This section is new. This used to be done manually by Alice Allen and Peter Teuben, and typically was a ½ day of work at the end of the whole editing process. It is better done during the editing, since the procedure is close to the other two indices, so we added this to the workflow.

We use the object index, \ooindex{} for the ASCL index.

ascl.py P1-12.tex

would scan the paper for potential occurrences of ASCL codes, but be forewarned, there are a LOT of false positives. The [line number] and line that matches the occurrence will be printed as well, making it easy to decipher if the entry is warranted. So have an editor open in another window.

The output might look as follows:

#[260] TOPCAT has many visualisation modes enabling   
  
%\ooindex{TOPCAT, ascl:1101.010}    
  
#[268] for {\it Gaia\/} data:   
  
%\ooindex{GAIA, ascl:1403.024}

In this case line 260 had a valid match, but line 268 did not. This particular case might be more appropriate for an %\ssindex{} entry.

Editors would use

../pmake ascl

since the ascl.py script is not in their directory or search path. This is something to be considered to be improved.