# otl2aft

Todd Coram ( http://maplefish.com/todd)

September 9, 2009

## Contents



Version 1.3



---

## 1 Introduction

Otl2aft converts VimOutliner files into AFT http://www.maplefish.com/todd/aft.html documents. This file can be run with nawk, mawk or gawk.

This tool was created upon the request/inspiration of David J Patrick (of http://linuxcaffe.ca fame).

You can downloaded the most up to date source here http://www.maplefish.com/todd/otl2aft.awk. A PDF version of this file resides here http://www.maplefish.com/todd/otl2aft.pdf.

## 2 Code

In the beginning we want to print out the AFT title and author. We expect the otl file to contain two top level headlines. The first will be used as the title and the second as the author.

We also print out some control bits and a table of contents placeholder.

```
BEGIN {
  VERSION="v1.3 9/04/2009";
  getline; # expect title
  print "#---SET-CONTROL tableparser=new"
  print "*Title: " $0;
  getline; # expect author
  print "*Author: " $0;
  print "\n*TOC\n"
}
```

Scan for one or more tabs followed by a colon (:). This is the outliner's markup for *body text (wrapping)*. If we are not nested inside a list (subheaders), then reset (§ 2.2) before doing any work. This makes sure we properly terminatel tables and other modes.

Our work here involves simply killing tabs and removing the colon. We then continue reading the rest of the file.

```
/^[\t]+:/ {
  if (!list_level) reset();
  gsub(/\t/,"");
  sub(/[ ]*:/, "");
  print $0; next;
}
```

Scan for *user defined text block (wrapping)*. If we get this, we kill the tabs, remove the > and if we discover a crosshatch #, we start a list. If we are already in a list, we continue the list. Both starting and continuing is handled by handlelist (§ 2.1).

```
/^[\t]+>/ {
  if (!list_level) reset();
  gsub(/\t/,"");
  sub(/>/, "");

  if (list_level || $0 ~ /[ ]*[#*]/) {
    handlelist();
  }
  print $0; next;
}
```

Scan for ; or < which indicate *preformatted body text* and *user-defined preformatted text block* respectively. Both of these are non wrapping but we ignore that (for now). We handle lists just like the previous scan action.

```
/^[\t]+[;<]/ { # Handle ";" and "<" (preformated text)
  if (!list_level) reset();
  gsub(/\t/,"");
```

```
  sub(/[;<]/, "");

  if (list_level || $0 ~ /[ ]*#/) {   # Convert "< #" into numbered lists.
    handlelist();
  }
  print $0; next;
}
```

Scan for a table. This is tricky. We want to cast the Outliner table into the AFT *new table* format. AFT tables (especially as rendered by LaTeX) really want to have captions/headers. We fake that for now by using a - place holder. This should be fixed!

```
/^[\t]+\|/ {
  if (!in_table) reset();
  in_table = 1
  gsub(/\t/,"");
  if ($1 ~ /\|\|/) {
    print "\t!   _      !";
    print "\t!----------!"
  }
  gsub(/\|\|/,"!");
  gsub(/\|/,"!");
  print "\t"$0
  print "\t!----------!"
  next;
}
```

The default scan matches anything not matching the above scan. We simply go through and set the known indent level based on the number of tabs seen.

```
{ match($0,/^[\t]+/); indent = RLENGTH; if (indent == -1) indent = 0; }
```

Given the iden level set by the default scan (above), we now determine what type of AFT output to do.

Indent levels lower than 7 are represented directly using AFT sections.

```
indent < 7 { gsub(/\t/,"*"); print "";}
```

Indent levels greater than 6 are represented by AFT bullet lists. This is done by first killing some tabs (we don't want to start off nesting too deeply), and using the remaining tabs to adjust to the appropriate list nesting level.

```
indent > 6 {
```

```
  gsub(/\t\t\t/, "");
  match($0,/^[\t]+/);
  remtabs = substr($0,RSTART,RLENGTH);
  text = substr($0,RSTART+RLENGTH);
  $0 = remtabs"* "text;
  print "";
}
```

After adjusting indentation, just print out the line.

```
{ print $0 }
```

### 2.1 handlelist

Look at the indentation and produce lists accordingly.

```
function handlelist() {
  if (!list_level) {
    list_indent = length(indent) + 1;
  }
  list_level = list_indent - length(indent);

  if ($0 ~ /[ ]*#/) {    # Convert " #" into numbered lists.
    for (i=0; i < list_level; i++)
      printf("\t");
    gsub(/[ ]*\#/,"#.");
  } else if ($0 ~ /[ ]*\*/) { # Convert " *" into bullet lists.
    for (i=0; i < list_level; i++)
      printf("\t");
    gsub(/[ ]*\*/,"*");
  } else if (list_level) {
    for (i=0; i < list_level; i++)
      printf("\t");
  }
}
```

### 2.2 reset

Reset various parameters to get us out of various modes.

```
function reset() {
  if (list_level) {
    print "  ";
    list_level = 0;
  }
```

```
  if (in_table) {
    print "\t!----------!\n"
    in_table = 0;
  }
}
```

That's all folks!