Université d'Ottawa | University of Ottawa

Automating the Preparation of Curriculum Vitæ Documents

John C. Nash

WORKING PAPER
WP.10.07

September 2011
ISSN 0701-3086

École de gestion
TELFER
School of Management

Accredited by Association of MBAs

AACSB ACCREDITED EST. 1916

Automating the preparation of Curriculum Vitae documents

John C. Nash
Telfer School of Management, University of Ottawa

October 6, 2011

Abstract

This article describes one approach to the perennial administrative task faced by academic staff who must update their curriculum vitae. In particular it discusses some ways in which the required format and layout may be achieved more or less automatically, including the necessary counts of lifetime and "past seven years" research and supervisory work.

1 Motivations

The primary motivation for the work and tools described in this paper are the requirements of the Ontario Council on Graduate Studies (hereafter OCGS) that the appraisal process for graduate programs at Ontario universities requires submission of staff curricula vitae (CV) in a common format, of which there is an apparently prescribed format. This is mentioned in the Report of the Working Group on CV Format, October 22, 1999, which can be downloaded from `http://ocgs.cou.on.ca/_bin/councilCommittees/workingGroups.cfm`. It is notable that this document mentions that "The OCGS format is always acceptable, and is the default format." but does not define nor provide a reference to this format. (The document seems to have escaped copy-editing also.) Neither an automated search nor a careful manual examination of the OCGS web site revealed a definition of this format. Indeed, a Google search of "OCGS format" revealed a number of versions, of which one that originated at the University of Ottawa seems to be popular.

One can sympathise with an agency that must review many instructional programs and therefore demands some standardization of reports it received. However, such is the leverage of OCGS that its CV format appears to have become a requirement for many applications at the University of Ottawa and other Ontario universities, even though the prescribed format specifically omits undergraduate teaching, which is one of our primary, likely dominant, activities.

OCGS CVs are required to report lifetime activity in graduate supervision as well as specifying the students and their work for the last 7 years. Again there is a glaring omission of activities related to undergraduates, including important roles such as case competition mentoring and similar tasks, as well as administrative or community activities that are not related to graduate students. A similar breakdown is required for publications, and the categories for publications show an almost extreme bias towards the kinds of publications extant in the 1970s. Possibly this is when the developers of the 1999 Working Group were active in research. Software, electronic publications, and other "live" works that have continuing peer review do not have an appropriate place.

Preparing counts of activity for a lifetime and for the last 7 years is a chore that is tedious and highly error-prone. Automation of that aspect of CV preparation is therefore of interest.

A second motivation is intellectual: Can it be done relatively easily? The answer is, I contend, "Yes!", though some of my colleagues may want a definition of "relatively easily".

2 Choices in approach

Most of my colleagues maintain their CV is a word processing file, with Microsoft Word being the dominant format and tool. Indeed I maintained my CV master file for many years as a ".doc" file, with comment line to indicate the seven year boundary that I would move manually. Material was kept in reverse date order in the file within each section. To get a current "Last 7" CV, I would delete the tags and material below them.

Indeed I maintained this approach from the time OCGS format was introduced (I believe sometime in the early 1990s) until after I retired in 2008. In wanting to prepare a lifetime CV and add scans of my lifetime work to put on an optical disk (CD or DVD), I started to think how I would ideally proceed.

The requirements for automating the process of producing a "last 7" CV are:

- A publication database in a form that can be searched and sorted on year and category.
- A base text for the CV that can be segmented by year.
- Tools to carry out the requisite searching and sorting of the above file or files.
- Tools to render the resulting data into a presentation form.

The ".doc" file approach that most of us have been or still are using relies on our own activities with a word processor (Microsoft Word, or in my case OpenOffice or LibreOffice) to essentially manually do all the work associated with the first three items.

There must be a better way!

Indeed, we could consider putting information into some form of database. I believe this was tried in the Telfer School (when still the Faculty of Administration) using a Borland Paradox database in the mid-1980s. I recall a great many errors of both typography and classification, as well as a number of missing entries. The detail work was crushingly tedious, and the MSDOS environment of the time was not attractive to the secretarial staff.

The principal objection to using a generalized database is that there must be an agreed set of variables and formats for its use. Generalized databases are used for all sorts of applications, of which academic histories and bibliographies of publications are a tiny minority.

There is, however, a well-established system for managing academic reports and bibliographic information. This is the family of document management tools based on Donald Knuth's TeX system [Knu86]. Leslie Lamport in 1985 introduced LaTeX, a set of commands and macros to significantly enlarge the TeX system [Lam94]. Hefferon (`http://www.ctan.org/what_is_tex.html`) provides a short overview. The BibTex system, created by Oren Patashnik and Leslie Lamport in 1985, adds a specialized system for references and citations. For our purposes here, the particular advantaqe of using BibTeX is that many publishers and on-line systems for academic references provide pre-prepared citations in BibTeX formaT, for example, CiteULike (`www.citeulike.org`) and CiteSeer (`http://citeseerx.ist.psu.edu/`). These can be imported directly into a professor's list of publications.

3 Implementation

The solution presented here consists of the following components:

- A TeX formatted file with a lifetime record of academic activity. This has added tags in the form of LaTeX comment lines that provide time references so that material can be removed as needed for the "last 7 years" résumé. This file does NOT contain bibliographic material, since the specialized BibTeX tools are used for that information. For the sake of discussion, suppose this file is called `jn.tex`.

- A BibTeX file containing lifetime publications information. For example, call this, `jn.bib`. For the sake of orderly management of this material, I have chosen to organize it in reverse date order. However, that is not strictly necessary, as BibTeX tools can organize material in many ways. Moreover, the style of presentation is managed via sets of style and class files (.sty and .cls types), as well as bibliographic styles (.bst files). Frankly, I do not myself fully understand many of the nuances in such files, but have found that I can make them function.

- To allow the lifetime CV to be abridged for the "last 7 years", we need to add information to the TeX file `jn.tex`. I chose to do this with the one-line comment structure in LaTeX, which allows that a line beginning with the % symbol is not processed as part of the active document, but serves only for information. We use this to provide tags that can be processed by a separate program. These tags work as follows:

  - Any block of material which may need to be abridged begins with a comment line

    %% YYTOP

    and ends with a comment line

    %% YYEND

  - Material within such blocks is arranged in reverse date order and prefaced by a tag to indicate the year of its occurrence, for example,

    %% YY2009

- A Perl program `lifeplus7.pl` that is used to

  - Put together the activities and bibliographic information into a lifetime CV;
  - Scan the source CV file for tags indicating different types of student supervision and examination committee participation and build the table of counts (in TeX form) to insert into a copy of the CV file. (This task was done manually for the present exercise, but tags have been inserted in the CV source .tex file to allow its automation.)
  - Using the Bibtool (http://www.gerd-neugebauer.de/software/TeX/BibTool/index.en.html) program, extract publication types and count them for both lifetime and last x years;

- Process the original CV file and incorporate the publication counts into a new file, e.g., `jnL.tex`, to a finished publication (.pdf) form using the appropriate `latex`, `bibtex`, and `pdflatex` programs that are called from Perl via the `system()` command;
- Abridge the lifetime file for the "last x years" (the program can be adjusted as needed for shorter or longer periods), to yield a new file e.g., `jn2004.tex`, where we use the starting year of the "last x" in the filename;
- Process the abridged (.tex and .bib) files to a finished publication (.pdf) form using the appropriate programs.

4 Tags and tools

We have already seen the tags to be used for dating items in the CV. At present, this leaves only tags needed to allow for counts of student supervision, for which we use

- `GSD` - supervision of doctoral studies;
- `GSM` - supervision of Master's level studies;
- `GSR` - supervision of a directed reading or project at a graduate level;
- `EXD` - participation in doctoral examination committee work;
- `EXM` - participation in Master's level examination committee work; and
- `USR` - supervision or participation in an examination committee for an undergraduate project, thesis, directed reading or major (defense-type) examination.

To process the .tex and .bib files we have described, a number of tools have been mentioned. However, it is important to note that these are really families of software that run on different hardware and operating system platforms and sometimes have minor differences in appearance or presentation. The processing of the tags was done with a Perl program because the Perl language is available in multiple implementations on different platforms. However, in this exercise we have only used the Texlive system (http://www.tug.org/texlive/), which has been developed since 1996 by many members of different TeX User Groups and the Perl version 5.10.1 that is available with Ubuntu Linux 10.04 "Lucid Lynx". TeX files were edited using plain text editors gedit or leafpad, and processed with small script files written by the author to minimize command keyboard entry.

5 Comments and conclusions

The main conclusion of this proof-of-concept of automated CV processing is "It works!". This is not to say that the effort is an unqualified success. First, there are legion annoyances with TeX and Latex:

- They are awkward and non-intuitive to use.

- The lack of a "What you see is what you get" (WYSIWYG) environment is for many people a situation they cannot deal with. There exists a wrapper for Latex called Lyx (www.lyx.org) for which version 2.0.1 has just been releases in early September 2011. However, the setup of this tool is non-trivial, and it is not necessarily similar to traditional word processing software.
- Using TeX or Latex effectively means using packages, of which there are frequently several that perform similar functions. The main burden of the learning cost of such tools falls on the user.
- While Latex automates most formatting, fine adjustments require learning how to use a number of codes and tags.
- BibTeX has many variations in styles. Again there is a learning cost.
- Using bibliographies requires multiple passes of Latex to be performed so that the citations can be linked to the correct BibTeX entry.

Offsetting the learning costs associated with the use of Latex are the possibilities for extensive reuse of both the tools and the content. Moreover, if others need the same tools, we can share the learning cost.

The Bibtool program, while powerful, is one of the least friendly to use, and appears to have some documented features that I, for one, could not get to work.

Tagging issues are a nuisance, but tagging is done once as content is entered. We do not "unpublish" material, or "unsupervise" as student, so editing is rarely necessary.

A final class of annoyance is the fit or lack thereof between TeX and OCGS format requirements. The OCGS entry classifications that add the qualifier "in refereed journals" etc. have no direct parallel in BibTeX. Sometimes note fields in BibTeX entries are not displayed (this depends on the style file as well as other BibTeX rools.

Despite these complaints, I wish I had taken the time to carry out this exercise many years ago – it would have saved me a great deal of time and energy, and provided a less error-prone result.

Bibliographic note

This article is referenced as

Nash, J.C. (September 2011). Automating the Preparation of Curriculum Vitae Documents. Telfer School of Management Working Papers, WP.2011.07: 1-20.

References

[Knu86] Donald E. Knuth. *The TeXbook*. Addison-Wesley Professional, 1986.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System (2nd Edition)*. Addison-Wesley Professional, 2 edition, July 1994.

Appendix: The Perl program `lifeplus7.pl`

This program is no doubt going to change and evolve. This is the version as at October 2, 2011.

```
#!/usr/bin/perl
#
# lifeplus7.pl - Produce lifetime and last x (default 7) year CV
#   i.e., remove material in latex resume so only have last x years
# Version 110909
#
# Start:
#  - extract the arguments
#    @ARGV[0]
#
#      This program written by Prof. J C Nash
#         nashjc@uottawa.ca
#      This program is Copyright (C) 2011  J C Nash
#      It is distributed under the Gnu Public Licence
#       http://www.gnu.org/licenses/gpl.txt
#
use strict; # ensure all variables declared etc.
# use File::Find;  # module that lets us build a full tree.
# use File::Basename; # module that parses filenames into path, name, extension etc.
use File::stat; # module to allow access to file characteristics by name
# use File::Compare; # module to compare
# use File::Copy;
use Cwd;
# use Time::localtime; #!!!!! BAD -- does not work right
use POSIX;

my $dirchar = "\/";     # use / for all? "\\"; # for Windows. For Linux use /

# need to adjust for the actual variables we have
my ($aline, $firstyear, @flines, $i, $ifn, $ifnroot, $j, $kblock);
my ($lastyear, $nlines, $numclargs, $nyear, $sname, $t1, $t2, $titleline);
my ($tilife, $tiyear, @yline, @yy, $lfname, $yfname, $ofn, $ltexfr, $ytexfr);
my (@lcount, @ycount, $summline, $dttex, $dtbib, @tmp, @pubname, @summblock);
my ($day, $month, $year, $wkdir);
my (%lhash, %yhash, $k, $v, $ltotal, $ytotal);

$wkdir=getcwd(); # Must have current working dir
# get the year so we can compute which years to use
($day, $month, $year) = (localtime)[3,4,5];
# printf("The current date is %04d %02d %02d\n", $year+1900, $month+1, $day);
$year=1900+$year;
print "Current year is $year \n";
```

```
# check invokation and arguments
my $numclargs = @ARGV; # get the number of arguments on the command line
if ($numclargs==0) { #if no command line parameters, then display a message about the program
   print "Program  lifeplus7.pl -- lifetime and abridged CVs";
   print "This program was written by John C. Nash\n";
   print "     373 Laurier Ave E, #903, Ottawa, Ontario, K1N 8X6\n";
   print "       2011-9-13\n";
   print "       nashjc\@uottawa.ca\n";
   print "\n";
   print "Usage:\n";
   print "     lastxyr.pl mycvname [lastyear_to_include]\n";
   print "\n";
   $ifn = <>;
} else {
   if ($numclargs > 2) {
      die("Too many arguments to program lifeplus7.pl");
   } else {
      $ifnroot = $ARGV[0]; # filename root is first parameter
      if ($numclargs == 1) {
         $firstyear = $year - 7;
      } else {
         $firstyear = $ARGV[1];
      }
   }
}

print "First year of data to include is $firstyear\n";
print "ifnroot = $ifnroot\n";
# print "Continue?";
# $t1 = <STDIN>;

# clean up the CV file name
if ($ifnroot =~ '\.tex') { # leave filename as is if . present
   $ifn = $ifnroot;
   $ifnroot =~ s/\.tex//; # and get rid of extension for root
   print "Filename with LaTEX of CV is (unchanged) $ifn \n";
} else {
   $ifn=$ifnroot.".tex";
   print "Filename with LaTEX of CV is (with .tex appended) $ifn \n";
}

# read resume tex file
# find start and end of each Yearblock
# find "startignore" in each Yearblock based on last year
```

```
#  This assumes we have things in reverse date order, which may be OK
#  Therefore could simply reset start in Yearblock to start of ignore
# Process stored tex file, ignoring lines in ignore blocks
#   One way to do this may be to have logical vector of "OK lines" and
#   set ignored lines false, and just print the OK ones

open (INF, $ifn); # open the file
@flines = <INF>;
close(INF);

$nlines=@flines;
print "There are $nlines lines in the file $ifn\n";
# for ($i=0; $i<$nlines; $i++) {
#   chomp($aline=@flines[$i]);
# chomp seems to do the right thing
#   print "$aline \n";
# }
# print "Continue?";
# $t1 = <STDIN>;

# Find the title line
$i=0;
do {
 $i++;
 $aline=$flines[$i];
} until (($aline =~ /\\title/) || ($i >= $nlines));
$titleline=$i;
print "Titleline is $titleline: $flines[$titleline] \n";
chomp($aline=$flines[0]);
$j=index($aline, "NAME=");
$sname=substr($aline,$j+5);
print "The subject of this CV is named :$sname:\n";
$tilife="\\title\{$sname: Lifetime Curriculum Vitae\}";
$tiyear="\\title\{$sname: Curriculum Vitae from $firstyear\}";
# print "$tilife\n";
# print "$tiyear\n";
$lfname=$ifnroot."L.tex";
$yfname=$ifnroot.$firstyear.".tex";
print "filenames: $lfname, $yfname\n";
# print "Continue?";
# $t1 = <STDIN>;

# Find the summary count start point
$i=0;
do {
```

```
 $i++;
 $aline=$flines[$i];
} until (($aline =~ /^\\textit\{Lifetime summary\}/) || ($i >= $nlines));
$summline=$i;
# print "summline = $summline\n";
# print "Continue?";
# $t1 = <STDIN>;


# get tex date
$t1 = ( stat("$ifnroot.tex")->mtime );
# print "stat == $t1\n";
$dttex = localtime($t1);
# print "TEXfile $ifnroot.tex was updated at $dttex\n";
$t1 = ( stat("$ifnroot.bib")->mtime );
$dtbib = localtime( $t1 );
# print "BibTEXfile $ifnroot.bib was updated at $dtbib\n";


# Find file dates and enter them
$i=$nlines;
do {
   $aline = $flines[$i];
   if ($aline =~ /\%\% FILEDATES/) {
      $flines[$i]="$ifnroot.tex $dttex; ~~~~~~~ $ifnroot.bib $dtbib\n";
      $i = 0; # to finish loop
   } else {
      $i--;
   }
} until ($i == 0);
# print "Continue?";
# $t1 = <STDIN>;


my @ytop=();
my @yend=(); # inialize start and end arrays for year blocks
my @lineok=();
$t1=0;
my @lok=();
for ($i=0; $i<$nlines; $i++) {
   push @lineok, 1; # set all lines OK to start with
   @lok[$t1]=1;
   $t1++

}
```

```
# print "Counted $t1 lines\n";
$t2=@lok;
# print "Size of lok is $t2 \n";
# print "nlines = $nlines\n";

#  for ($i=0; $i<$nlines; $i++) {
#     print "$i  ";
#  }
#  print "\n";


#  for ($i=0; $i<$nlines; $i++) {
#     print "$i   @lok[$i] \n";
#  }
#  print "\n";


#  for ($i=0; $i<$nlines; $i++) {
#     print "@lineok[$i]  ";
#  }
#  print "\n";




# find the start and end lines of YY blocks
my $numblock = 0;
for ($i=0; $i<$nlines; $i++) {
   chomp($aline=@flines[$i]);
   if ($aline =~ '^%% YYTOP') {
       @ytop[$numblock]=$i;
       $numblock++;
   }
}

my $numb2 = 0;
for ($i=0; $i<$nlines; $i++) {
   chomp($aline=@flines[$i]);
   if ($aline =~ '^%% YYEND') {
       @yend[$numb2]=$i;
       $numb2++;
   }
}

# print "Start block positions \n";
```

```
# for ($j=0; $j<$numblock; $j++){
#   print "$ytop[$j]  ";
#}
#print "\n";


if ($numb2 != $numblock) {
  print "Number of year-block starts = $numblock\n";
  print "Number of year-block ends   = $numb2\n";
  die("Block starts != Block ends");
}

print "End  block positions \n";
for ($j=0; $j<$numb2; $j++){
   print "$yend[$j]  ";
}
print "\n";
print "Number of year-blocks = $numblock\n";

# need to check for correctness
for ($i=0; $i<$numblock; $i++){
   if ($ytop[$i] >= $yend[$i]) {
      $t1=$ytop[$i];
      $t2=$yend[$i];
      print "Block no. $i: $t1  >= $t2 \n";
      die("Bad start-end sequence of blocks");
   }
}

# Now find years within each block
for ($kblock=0; $kblock<$numblock; $kblock++) {
  $nyear=0; # number of years in block
  @yline=();
  @yy=();
#  print "Block $kblock \n";
  for ($i=$ytop[$kblock]+1; $i<$yend[$kblock]-1;$i++) {
     chomp($aline=@flines[$i]);
     if ($aline =~ '^%% YY') {
        $t1=substr $aline, 5, 4;
        @yy[$nyear]=$t1; # and the year
        $nyear++;
        if ($t1 < $firstyear) {
#           for ($j=$i; $j<$yend[$kblock]-1; $j++){  # May fail to catch last line
           for ($j=$i; $j<$yend[$kblock]; $j++){
              @lineok[$j]=0; # don't want that line
```

```
          }
        }
#        print "Line $i has year $t1 \n";
     }
  }
} # end loop over blocks


#for ($i=0; $i<$nlines; $i++) {
#   print "@lineok[$i]  ";
#}
#print "\n";

# Now need to do the runs of bibtool and latex
# Get counts for entire bibliography
$ofn = "lcount.rsc";
$ltexfr=$ifnroot."L";
open (OFN, ">", $ofn);
print OFN "output.file = {$ltexfr.bib}\n";
print OFN "pass.comments = ON\n";
print OFN "count.all = ON\n";
print OFN "count.used = ON\n";
close OFN;

print "$ofn :\n";
system("cat $ofn");
 print "Continue?";
# $t1 = <STDIN>;


# process for lifetime counts
system("cp $ifnroot.bib $ltexfr.bib");
system("bibtool -r lcount.rsc -i $ltexfr.bib 2>lcount.txt");
system("cat lcount.txt");
# print "Continue?";
# $t1 = <STDIN>;


# Now build the rsc file for bibtool like:
#
#  output.file = {atemp.bib}
#  input = {ax.bib}
#  select.by.string = {year "2006"}
#  select.by.string = {year "2007"}
#  select.by.string = {year "2008"}
```

```
#  select.by.string = {year "2009"}
#  select.by.string = {year "2010"}
#  select.by.string = {year "2011"}

my $rscfn = $ifnroot . $firstyear . ".rsc";
my $ybibfn=$ifnroot . $firstyear . ".bib";
print "rscfn = $rscfn; ybibfn = $ybibfn \n";
# print "Continue?";
# $t1 = <STDIN>;

if (-e $ybibfn) {
   print "Try to delete $ybibfn\n";
   if (unlink($ybibfn) == 0) { die("File $ybibfn NOT deleted!")};
   open (OFN,">",$ybibfn); # need to create an empty fn
   print OFN "";
   close(OFN);
}
# restricted output for reduced set of years
open (RFN, ">", $rscfn);
print RFN "output.file = {$ybibfn}\n";
print RFN "input = {$ifnroot.bib}\n";
print RFN "pass.comments = ON\n";
print RFN "count.all = ON\n";
print RFN "count.used = ON\n";
for ($i=$firstyear; $i<=$year; $i++){
   print RFN "select.by.string ={year \"$i\"}\n";
}
close RFN;

print "$rscfn :\n";
system("cat $rscfn");
# print "Continue?";
# $t1 = <STDIN>;

system("bibtool -r $rscfn -i $ybibfn 2>ycount.txt");
# Note 2> to get STDERR to counts
system("cat ycount.txt");
# print "Continue?";
# $t1 = <STDIN>;

# Build the lifetime summary table
open (IFN, "lcount.txt");
@lcount = <IFN>;
close(IFN);
open (IFN, "ycount.txt");
```

```
@ycount = <IFN>;
close(IFN);
$t2=@lcount; # number of lines in count file
# $aline = @lcount[7];
# print $aline;
@pubname=();
for ($i=7; $i<=20; $i++){
  $aline=@lcount[$i];
#  print "$aline\n";
  $aline =~ s/--- //;
  $j=index($aline,"  ");
  $t2=substr($aline,0,$j);
  $t2=~s/ //g;
  push(@pubname, $t2);
  $j=index($aline," read", $j);
  $t2=substr($aline, $j-4, 4);
  $t2=~s/ //g;
  @lcount[$i-7]=$t2;
  $aline=@ycount[$i];
#  print "$aline\n";
  $aline =~ s/--- //;
  $j=index($aline,"  ");
  $t2=substr($aline,0,$j);
  $t2=~s/ //g;
  if (@pubname[$i-7] != $t2) {die("pubname does not match $t2")};
  $j=index($aline," read", $j);
  $t2=substr($aline, $j-4, 4);
  $t2=~s/ //g;
  @ycount[$i-7]=$t2;
  print "$i   ".$pubname[$i-7]."  ". $lcount[$i-7]."  ". $ycount[$i-7]."\n";
#  print "Continuu?";
#  $t1<-<STDIN>;
}
#foreach (@lcount) {
#   print "$_ \n";
#}
# print "Continue?";
# $t1 = <STDIN>;
for ($i=0; $i<7; $i++) {
   delete @lcount[$i+14];
   delete @ycount[$i+14];
}
#foreach (@lcount) {
#   print "$_ \n";
#}
```

```
# print "Continue?";
# $t1 = <STDIN>;

#0
#1---  STRING              0 read      0 written
#2---  PREAMBLE            0 read      0 written
#3---  COMMENT             2 read      2 written
#4---  ALIAS               0 read      0 written
#5---  MODIFY              0 read      0 written
#6---  INCLUDE             0 read      0 written
#7---  Article            18 read     18 written
#8---  Book                2 read      2 written
#9---  Booklet             0 read      0 written
#10---  Conference          0 read      0 written
#11---  InBook              0 read      0 written
#12---  InCollection        0 read      0 written
#13---  InProceedings       8 read      8 written
#14---  Manual             22 read     22 written
#15---  MastersThesis       0 read      0 written
#16---  Misc               38 read     38 written
#17---  PhDThesis           0 read      0 written
#18---  Proceedings        12 read     12 written
#19---  TechReport         10 read     10 written
#20---  Unpublished         0 read      0 written

# Build the right count structure
%lhash = (); # initialize hash
$i=0;
$ltotal=0;
$ytotal=0;
foreach (@pubname){
   $lhash{$pubname[$i]}=$lcount[$i];
   $ltotal+=$lcount[$i];
   $yhash{$pubname[$i]}=$ycount[$i];
   $ytotal+=$ycount[$i];
   $i++;
}
# @tmp = %lhash;
# print "@tmp\n";
# print "Continue?";
# $t1 = <STDIN>;

$lhash{"Thesis"} = $lhash{"MastersThesis"}+$lhash{"PhDThesis"};
$yhash{"Thesis"} = $yhash{"MastersThesis"}+$yhash{"PhDThesis"};
```

```
$lhash{"alltech"} = $lhash{"Thesis"}+$lhash{"TechReport"};
$yhash{"alltech"} = $yhash{"Thesis"}+$yhash{"TechReport"};


$lhash{"other"} = $lhash{"Misc"}+$lhash{"Unpublished"};
$yhash{"other"} = $yhash{"Misc"}+$yhash{"Unpublished"};

$lhash{"allproc"} = $lhash{"Proceedings"}+$lhash{"InProceedings"};
$yhash{"allproc"} = $yhash{"Proceedings"}+$yhash{"InProceedings"};

$lhash{"EdBook"} = 0; # JN has none -- need to fix for others. ? Wikis?
$yhash{"EdBook"} = 0;

$lhash{"Total"} = $ltotal;
$yhash{"Total"} = $ytotal;

 while ( ($k,$v) = each %lhash ) { print "|$k| => |$v|\n"; }
# print "Continue?";
# $t1 = <STDIN>;


push @summblock, "\\begin\{tabbing\}";
push @summblock, "~\\=  Category ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \\=Lifetime ~~~~~~~
push @summblock, "\\\>Books authored  \\\>".$lhash{"Book"}." \\\>".$yhash{"Book"}."\\\\";
push @summblock, "\\\>Books edited  \\\>".$lhash{"EdBook"}." \\\>".$yhash{"EdBook"}."\\\\";
push @summblock, "\\\>Chapters in books  \\\>".$lhash{"InBook"}." \\\>".$yhash{"InBook"}."\\\
push @summblock, "\\\>Papers in refereed journals  \\\>".$lhash{"Article"}." \\\>".$yhash{"Ar
push @summblock, "\\\>Refereed conference proceedings  \\\>".$lhash{"allproc"}." \\\>".$yhash
push @summblock, "\\\>Software packages  \\\>".$lhash{"Manual"}." \\\>".$yhash{"Manual"}."\\\
push @summblock, "\\\>Technical reports  \\\>".$lhash{"alltech"}." \\\>".$yhash{"alltech"}."\
push @summblock, "\\\>Booklets  \\\>".$lhash{"Booklet"}." \\\>".$yhash{"Booklet"}."\\\\";
push @summblock, "\\\>Other works  \\\>".$lhash{"other"}." \\\>".$yhash{"other"}."\\\\";
push @summblock, "\\\>~~  \\\> -------\\\> -------\\\\";
push @summblock, "\\\>Totals:  \\\>".$lhash{"Total"}." \\\>".$yhash{"Total"}."\\\\";
push @summblock, "\\end\{tabbing\}";

foreach (@summblock) {
    print "$_ \n";
}

# print "Continue?";
# $t1 = <STDIN>;


# Output the lifetime TEX file
```

```
my $otexfn="$ltexfr.tex"; # for lifetime CV
open (OFN, ">", $otexfn);
for ($i=0; $i<$nlines; $i++) {
   if ($i == $titleline) {
        print OFN "$tilife\n"; # \n needed?
   } else {
      if ($i==$summline) {
         foreach (@summblock) {
            print OFN "$_ \n";
         } # output the summary block
      } else {
#        if ($lineok[$i]) {
          print OFN "$flines[$i]";
#        }
      }
   }
}
close OFN;


print "summline = $summline\n";
# Output the abridged TEX file
$ytexfr=$ifnroot.$firstyear;
my $otexfn="$ytexfr.tex"; # for abridged CV
open (OFN, ">", $otexfn);
for ($i=0; $i<$nlines; $i++) {
   if ($i == $titleline) {
        print OFN "$tiyear\n"; # ? \n?
   } else {
      if ($i==$summline) {
#         print "Found summline";
#         $t1=<STDIN>;
         foreach (@summblock) {
            print OFN "$_ \n";
         } # output the summary block
      } else {
        if ($lineok[$i]) {
          print OFN "$flines[$i]";
        }
      }
   }
}
close OFN;

# print "Continue?";
```

```
# $t1 = <STDIN>;


# ? Note -- do not handle all the types well esp. Books Edited
# subset bibliography functions for lifefime CV
system("cd $wkdir");
system("bibtool -- select{\@inproceedings} $ltexfr.bib >inproc.bib");
system("bibtool -- select{\@proceedings} $ltexfr.bib >proc.bib");
# How to extract just the inproceedings entries
system("bibtool -- select{\@book} $ltexfr.bib >book.bib");
system("bibtool -- select{\@inbook} $ltexfr.bib >inbook.bib");
system("bibtool -- select{\@article} $ltexfr.bib >article.bib");
system("bibtool -- select{\@manual} $ltexfr.bib >manual.bib");
system("bibtool -- select{\@techreport} $ltexfr.bib >techreport.bib");
system("bibtool -- select{\@misc} $ltexfr.bib >misc.bib");
system("bibtool -- select{\@unpublished} $ltexfr.bib >unpublished.bib");
system("bibtool -- select{\@booklet} $ltexfr.bib >booklet.bib");
system("rm thesis.bib");
system("bibtool -- select{\@phdthesis} $ltexfr.bib >thesis.bib");
system("bibtool -- select{\@mastersthesis} $ltexfr.bib >>thesis.bib");

# Process the lifetime TEX file
system("latex $ltexfr.tex");
system("bibtex $ltexfr".".aux");
system("bibtex $ltexfr"."1.aux");
system("bibtex $ltexfr"."2.aux");
system("bibtex $ltexfr"."3.aux");
system("bibtex $ltexfr"."4.aux");
system("bibtex $ltexfr"."5.aux");
system("bibtex $ltexfr"."6.aux");
system("bibtex $ltexfr"."7.aux");
system("bibtex $ltexfr"."8.aux");
system("bibtex $ltexfr"."9.aux");
system("latex $ltexfr.tex");
system("lado $ltexfr");

 print "Continue after lifetime CV output?";
 $t1 = <STDIN>;


# subset bibliography functions for abridged CV
system("bibtool -- select{\@inproceedings} $ybibfn >inproc.bib");
system("bibtool -- select{\@proceedings} $ybibfn >proc.bib");
# How to extract just the inproceedings entries
system("bibtool -- select{\@book} $ybibfn >book.bib");
```

```
system("bibtool -- select{\@inbook} $ybibfn >inbook.bib");
system("bibtool -- select{\@article} $ybibfn >article.bib");
system("bibtool -- select{\@manual} $ybibfn >manual.bib");
system("bibtool -- select{\@techreport} $ybibfn >techreport.bib");
system("bibtool -- select{\@misc} $ybibfn >misc.bib");
system("bibtool -- select{\@unpublished} $ybibfn >unpublished.bib");
system("bibtool -- select{\@booklet} $ybibfn >booklet.bib");
system("rm thesis.bib");
system("bibtool -- select{\@phdthesis} $ybibfn >thesis.bib");
system("bibtool -- select{\@mastersthesis} $ybibfn >>thesis.bib");



# Process the abridged TEX file
system("latex $ytexfr.tex");
system("bibtex $ytexfr".".aux");
system("bibtex $ytexfr"."1.aux");
system("bibtex $ytexfr"."2.aux");
system("bibtex $ytexfr"."3.aux");
system("bibtex $ytexfr"."4.aux");
system("bibtex $ytexfr"."5.aux");
system("bibtex $ytexfr"."6.aux");
system("bibtex $ytexfr"."7.aux");
system("bibtex $ytexfr"."8.aux");
system("bibtex $ytexfr"."9.aux");
system("latex $ytexfr.tex");
system("lado $ytexfr");

#  remove files!!
print "\n\n Remove files $ytexfr?\n";
system("ls $ytexfr?.*");
system("rm $ytexfr?.*");
system("ls $ytexfr?.*");
print "\n\n";

print "\n\n Now lifetime files $ltexfr?\n";
system("ls $ltexfr?.*");
system("rm $ltexfr?.*");
system("ls $ltexfr?.*");
print "\n\n";

unlink("jncvx2004.bib");
unlink("jncvxL.aux");
unlink("jncvxL.blg");
unlink("jncvx2004.aux");
```

```
unlink("jncvx2004.blg");
unlink("jncvx.blg");
unlink("jncvxL.bbl");
unlink("jncvxL.dvi");
unlink("jncvx2004.bbl");
unlink("jncvx2004.dvi");
unlink("jncvx2004.ps");
unlink("jncvx.bbl");
unlink("jncvx.dvi");
unlink("jncvxL.bib");
unlink("jncvxL.ps");
unlink("jncvx.ps");


die("Done!");
# ============ end of lastxyr.pl ==================
```