Automating Document Compilation using LATEX

Typesetting is the ancient art of creating beautifully formatted text. In olden times typesetting was done by meticulously arranging little metal blocks with embossed characters on them – called types or sorts – to create lines of text. Nowadays typesetting is usually done digitally on a computer. There are many tools available to typeset documents. LATEX is a popular and free typesetting tool.

LATEX is basically a markup language in which we describe the structure of the document. The software uses this information to generate the document according to predefined typesetting rules. The result is the creation of a high-quality document with minimal knowledge of the formatting rules. The user need only worry about the contents while LATEX will take care of all the formatting. LATEX can do things such as create a table of contents, create a bibliography or an index – often with just a line of command.

One of the advantages of using LATEX is that being a text-based markup language, the commands required to generate a document can be generated using any programming language. That is, there is a potential for automating the entire document compilation process. For example, it would be possible to create smart documents that can automatically retrieve and update its revision number from a centralized database that contains all the document revision histories.

The purpose of this article is to illustrate the process of automation using LATEX. Please be aware that we would just skim the surface of what is possible with LATEX. There are many resources online to explore LATEX further. In this article we shall try to develop an engineering document using LATEX and a scripting language. The objective is to automate the entire process right from performing the calculations to compiling the document. We will try to achieve this using a single Linux command. Let us see how far we can get.

We will go through the following steps in the process:

- Read Data from a Database.
- Create the Calculation Writeup.
- Run the Design Software.
- Generate a Summary of the Output.
- Attach the Output Files.

LATEX commands are written in a file with a .tex extension. A command has the following syntax:

```
\commandname [optional arguments,…] {arguments}
```

The file is split into two main parts: the main contents is between a pair of \begin{document} and \end{document} commands; and the preamble. The preamble is the portion before the \begin command and is used to define the document class, import third party packages, or define user defined commands called macros. A detailed explanation of LATEX commands and packages is outside the scope of this article, but we shall try to explain those commands that are required for the understanding of our topic.

```
\documentclass[a4paper,11pt]{article}
%import the required packages
\usepackage[a4paper, inner=1.5cm, outer=1.5cm, top=2cm, bottom=3cm,
headheight=15pt]{geometry}
\usepackage{graphicx}
\usepackage[export]{adjustbox}
\usepackage{hyperref}
\usepackage{amsmath}
\usepackage{pdfpages}
\usepackage{fancyhdr}
\usepackage{array}
\setlength\parindent{0pt}
%import the file containing macros that define information such as
%document number, title and revision history
\input{macros.tex}
\input{calc_number.tex}
%start of document
\begin{document}
    %header definition using predefined macros
    \pagestyle{fancy}
    \fancyhead{}
    \fancyhead[L]{\calctitle}
    \fancyhead[R]{Rev:\calcrevisionone}
    %the title page
    \include{title.tex}
    %table of contents
    \tableofcontents
    %insert writeup
    \include{writeup.tex}
    %insert result table
    \newpage
    \section{DESIGN RESULTS}
    \input{result.tex}
    %insert attachments
    \newpage
    \section{ATTACHMENT 1: OUTPUT FILES}
    \input{attachment.tex}
\end{document}
```

One of the good things about LATEX is that is we can breakup the main document into parts, each in its own .tex file. This means that we can modularize our code breaking it up into smaller and simpler parts. These parts can be included in the main tex file using the \include or \input commands. The difference between the two is that \include will start a new page in the document while \input will not.

In out main tex file we demonstrate the power of LATEX. We create a table of contents using a single \tableofcontents command and we define headers using the \fancyhead command which is available in the fancyhdr package. Notice we used two macros: \calctitle and \calcrevisionone, as arguments to the \fancyhead command. These are defined in the macros.tex file and is populated with data obtained from the database. We will see how to do this in just a second.

To compile our calc.tex file to pdf we use the pdflatex tool. In a Linux terminal type in the following command:

```
Pdflatex calc.tex
```

But LATEX will complain because we have not defined the other tex files that we have referred to in the calc.tex file. Let us correct this.

Reading Data from a Database:

Most organization maintain some record of their documents, whether it be simple excel sheets or big enterprise databases. We would like our document to tap into these systems and retrieve data such as the title and revision history. The user provides the document number based on which all the other details are retrieved. The document number is provided in a tex file called calc_number.tex. It has only one line in it.

```
\newcommand{\calcnumber}{540001-CAL-STR-0002}
```

The \newcommand command is used to define a macro. It has two required arguments: the first argument is the name of the macro which should start with a backslash, and the second argument is the output. Henceforth when ever LATEX encounters the command \calcnumber it will replace it with the value '540001-CAL-STR-0002'.

Assume we have a Mysql database called Calculations with the document details. It comprises of three tables:

```
mysql> show tables;
+-------------------------+
| Tables_in_calculations |
+-------------------------+
| Calculations           |
| Projects               |
| Revisions              |
+-------------------------+
3 rows in set (0.01 sec)
```

Let us look at the contents of these tables.

```
mysql> select * from Projects;
+------------+----------------+
| Project_Id | Project_Name   |
+------------+----------------+
|          1 | Big Steel Plant |
+------------+----------------+
1 row in set (0.00 sec)

mysql> select * from Calculations;
+---------------------+---------------------------------------+------------+
| Calculation_Number  | Calculation_Title                     | Project_Id |
+---------------------+---------------------------------------+------------+
| 540001-CAL-CIV-0001 | Foundation Design for Piperack PR-01 |          1 |
| 540001-CAL-STR-0001 | Steel Design for Piprack PR-01       |          1 |
| 540001-CAL-STR-0002 | Connection Design for Piperack PR-01 |          1 |
+---------------------+---------------------------------------+------------+
3 rows in set (0.00 sec)

mysql> select * from Revisions;
+-------------+---------------------+---------------+-----------------+-------------------------+-------------+-------------+-------------+--------+
| Revision_Id | Calculation_Number  | Revision_Date | Revision_Number | Purpose                 | Prepared_By | Reviewed_By | Approved_By | Status |
+-------------+---------------------+---------------+-----------------+-------------------------+-------------+-------------+-------------+--------+
|           1 | 540001-CAL-STR-0001 | 2023-07-14    |               0 | Issued for Review       | Roshn       | Aravind     | Samir       | ISSUED |
|           2 | 540001-CAL-STR-0001 | 2023-08-21    |               1 | Issued for Construction | Roshn       | Aravind     | Samir       | ISSUED |
|           3 | 540001-CAL-STR-0002 | 2023-07-25    |               0 | Issued for Review       | Jim         | Aravind     | Samir       | ISSUED |
|           4 | 540001-CAL-STR-0002 | 2023-08-13    |               1 | Re-issued for Review    | Jim         | Aravind     | Samir       | ISSUED |
|           5 | 540001-CAL-STR-0002 | 2023-09-01    |               2 | Issued for Construction | Jim         | Aravind     | Samir       | WIP    |
|           6 | 540001-CAL-CIV-0001 | 2023-06-04    |               0 | Issued for Review       | Prasad      | Kevin       | Samir       | ISSUED |
|           7 | 540001-CAL-CIV-0001 | 2023-07-10    |               1 | Issued for Construction | Ravi        | Kevin       | Samir       | ISSUED |
+-------------+---------------------+---------------+-----------------+-------------------------+-------------+-------------+-------------+--------+
7 rows in set (0.00 sec)
```

```
import mysql.connector
import sys
calc_number = [sys.argv[1]]
out = open('macros.tex','w')
db = mysql.connector.connect(
        host = 'localhost',
        user = 'roshn',
        password = 'Y00h00mysql',
        database = 'calculations'
        )
cur = db.cursor()
stmt= """SELECT Calculation_Title, Project_Name
FROM Calculations INNER JOIN Projects ON Calculations.Project_Id = Projects.Project_Id
WHERE Calculation_Number = %s ;"""
cur.execute(stmt,calc_number)
res = cur.fetchone()
out.write('\\newcommand {\\calctitle}{'+ res[0] + '}\n')
out.write('\\newcommand {\\calcproject}{' + res[1] + '}\n')
stmt = """SELECT Revision_Number, Revision_Date, Purpose, Prepared_By, Reviewed_By,
Approved_By
FROM Calculations INNER JOIN Revisions ON Calculations.Calculation_Number =
Revisions.Calculation_Number
WHERE Calculations.Calculation_Number = %s AND Status = 'WIP';"""
cur.execute(stmt,calc_number)
res = cur.fetchone()
if res:
    out.write('\\newcommand {\\calcrevisionone}{' + str(res[0]) + '}\n')
    out.write('\\newcommand {\\calcdateone}{' + res[1].strftime('%d-%m-%Y') + '}\n')
    out.write('\\newcommand {\\calcpurposeone}{' + res[2] + '}\n')
    out.write('\\newcommand {\\calcpreparedbyone}{' + res[3] + '}\n')
    out.write('\\newcommand {\\calcreviewedbyone}{' + res[4] + '}\n')
    out.write('\\newcommand {\\calcapprovedbyone}{' + res[5] + '}\n')
stmt = """SELECT Revision_Number, Revision_Date, Purpose, Prepared_By, Reviewed_By,
Approved_By
FROM Calculations INNER JOIN Revisions ON Calculations.Calculation_Number =
Revisions.Calculation_Number
WHERE Calculations.Calculation_Number = %s AND Status = 'ISSUED'
ORDER BY Revision_Number DESC
LIMIT 2;"""
cur.execute(stmt,calc_number)
res = cur.fetchall()
if res[0]:
    out.write('\\newcommand {\\calcrevisiontwo}{' + str(res[0][0]) + '}\n')
    out.write('\\newcommand {\\calcdatetwo}{' + res[0][1].strftime('%d-%m-%Y') + '}\n')
    out.write('\\newcommand {\\calcpurposetwo}{' + res[0][2] + '}\n')
    out.write('\\newcommand {\\calcpreparedbytwo}{' + res[0][3] + '}\n')
    out.write('\\newcommand {\\calcreviewedbytwo}{' + res[0][4] + '}\n')
    out.write('\\newcommand {\\calcapprovedbytwo}{' + res[0][5] + '}\n')
else:
    out.write('\\newcommand {\\calcrevisiontwo}{-}\n')
    out.write('\\newcommand {\\calcdatetwo}{-}\n')
    out.write('\\newcommand {\\calcpurposetwo}{-}\n')
    out.write('\\newcommand {\\calcpreparedbytwo}{-}\n')
    out.write('\\newcommand {\\calcreviewedbytwo}{-}\n')
    out.write('\\newcommand {\\calcapprovedbytwo}{-}\n')
if res[1]:
    out.write('\\newcommand {\\calcrevisionthree}{' + str(res[1][0]) + '}\n')
    out.write('\\newcommand {\\calcdatethree}{' + res[1][1].strftime('%d-%m-%Y') + '}\n')
    out.write('\\newcommand {\\calcpurposethree}{' + res[1][2] + '}\n')
    out.write('\\newcommand {\\calcpreparedbythree}{' + res[1][3] + '}\n')
    out.write('\\newcommand {\\calcreviewedbythree}{' + res[1][4] + '}\n')
    out.write('\\newcommand {\\calcapprovedbythree}{' + res[1][5] + '}\n')
else:
    out.write('\\newcommand {\\calcrevisionthree}{-}\n')
    out.write('\\newcommand {\\calcdatethree}{-}\n')
    out.write('\\newcommand {\\calcpurposethree}{-}\n')
    out.write('\\newcommand {\\calcpreparedbythree}{-}\n')
    out.write('\\newcommand {\\calcreviewedbythree}{-}\n')
    out.write('\\newcommand {\\calcapprovedbythee}{-}\n')

out.close()
cur.close()
db.close()
```

To extract the required data from this database we write a python script. The document number is passed in as an argument when the script is called. To do this we import the sys module. The connection to the database is achieved using the mysql.connector module. We used our username and password to connect to the database 'calculations' using the connect method. If the database is present on the current computer, the value for host will be localhost. Using SQL, we get the details of the required document number where the status is 'WIP' (Work In Progress). This would be the current revision of the document. Next, we get the details of two more 'ISSUED' revisions to populate in a revision history table. For this we sort the results in descending order using DESC and LIMIT the output to 2 rows.

We use this data to generate the macros.tex file. Assuming that we are currently in the folder where the script is save, to run the python script we type the following command in the linux terminal.

```
python3 read_database.py "540001-CAL-STR-0002"
```

Which will generate the following macros.tex file.

```
\newcommand {\calctitle}{Connection Design for Piperack PR-01}
\newcommand {\calcproject}{Big Steel Plant}
\newcommand {\calcrevisionone}{2}
\newcommand {\calcdateone}{01-09-2023}
\newcommand {\calcpurposeone}{Issued for Construction}
\newcommand {\calcpreparedbyone}{Jim}
\newcommand {\calcreviewedbyone}{Aravind}
\newcommand {\calcapprovedbyone}{Samir}
\newcommand {\calcrevisiontwo}{1}
\newcommand {\calcdatetwo}{13-08-2023}
\newcommand {\calcpurposetwo}{Re-issued for Review}
\newcommand {\calcpreparedbytwo}{Jim}
\newcommand {\calcreviewedbytwo}{Aravind}
\newcommand {\calcapprovedbytwo}{Samir}
\newcommand {\calcrevisionthree}{0}
\newcommand {\calcdatethree}{25-07-2023}
\newcommand {\calcpurposethree}{Issued for Review}
\newcommand {\calcpreparedbythree}{Jim}
\newcommand {\calcreviewedbythree}{Aravind}
\newcommand {\calcapprovedbythree}{Samir}
```

Once we have these macros defined it is only a matter of using these macros in our tex file like we did in the header defined in the calc.tex file. Let us also go ahead and build a title page with the information that we have gathered.

```
\begin{titlepage}
    {\centering
        {\Large Project: \calcproject \\[1in]}
        {\Huge \calctitle\\[0.25in]}
        {\large Calculation No: \calcnumber \\}
    }
    \vfill
    \begin{table}[h]
    \centering
    \begin{tabular}{|c|c|c|c|c|c|}
        \hline
        Rev & Date & Purpose & Prepared By & Reviewed By & Approved By\\
        \hline
        \calcrevisionone & \calcdateone & \calcpurposeone & \calcpreparedbyone &
\calcreviewedbyone & \calcapprovedbyone\\
        \hline
        \calcrevisiontwo & \calcdatetwo & \calcpurposetwo & \calcpreparedbytwo &
\calcreviewedbytwo & \calcapprovedbytwo\\
        \hline
        \calcrevisionthree & \calcdatethree & \calcpurposethree & \calcpreparedbythree &
\calcreviewedbythree & \calcapprovedbythree\\
        \hline
    \end{tabular}
    \end{table}
\end{titlepage}
```

Here we make use of the macros we defined to create a table with the revision history. The resulting pdf would look like the following.

Project: Big Steel Plant

Connection Design for Piperack PR-01

Calculation No: 540001-CAL-STR-0002

Rev	Date	Purpose	Prepared By	Reviewed By	Approved By
2	01-09-2023	Issued for Construction	Jim	Aravind	Samir
1	13-08-2023	Re-issued for Review	Jim	Aravind	Samir
0	25-07-2023	Issued for Review	Jim	Aravind	Samir

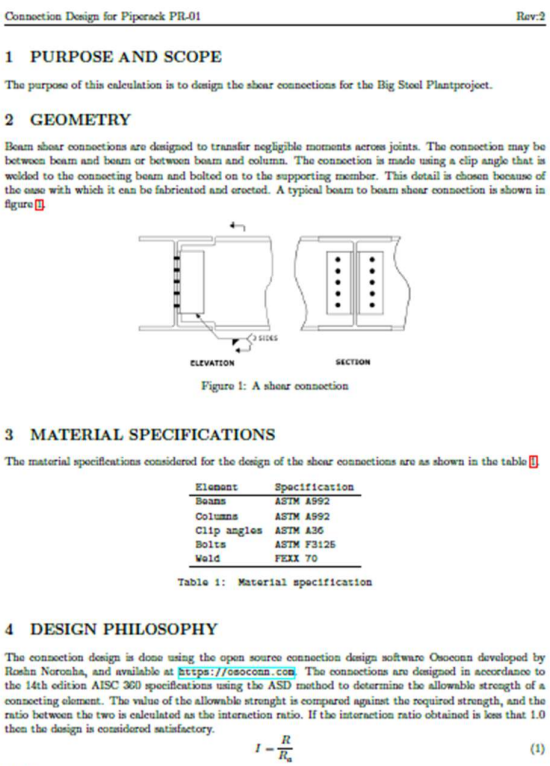
Create the Calculation Writeup

We keep the writeup of the calculation in a separate writeup.tex file. In this way if the user wants to make any modifications to the writeup, only this file needs to be edited. Our writeup gives a brief

description of the calculation purpose and methodology. The contents of this file is static and is not modified by any code.

```
\section{PURPOSE AND SCOPE}
The purpose of this calculation is to design the shear connections for
the \calcproject  project.

\section{GEOMETRY}
Beam shear connections are designed to transfer negligible moments across
joints. The connection may be between beam and beam or between beam and column.
The connection is made using a clip angle that is welded to the connecting beam
and bolted on to the supporting member. This detail is chosen because of the
ease with which it can be fabricated and erected. A typical beam to beam shear
connection is shown in figure \ref{fig:shear_conn}.

\begin{figure}[h]
    \centering
    \includegraphics[width=0.5\textwidth]{./images/sc001_1}
    \caption{A shear connection}
    \label{fig:shear_conn}
\end{figure}

\section{MATERIAL SPECIFICATIONS}
The material specifications considered for the design of the shear connections
are as shown in the table \ref{tab:mat_spec}.

\begin{table}[h]
    \centering
    \ttfamily
    \begin{tabular}{ll}
        Element         &Specification\\
        \hline
        Beams           &ASTM A992\\
        Columns         &ASTM A992\\
        Clip angles     &ASTM A36\\
        Bolts           &ASTM F3125\\
        Weld            &FEXX 70\\
        \hline
    \end{tabular}
    \caption{Material specification}
    \label{tab:mat_spec}
\end{table}

\section{DESIGN PHILOSOPHY}
The connection design is done using the open source connection design software Osoconn
developed by Roshn Noronha, and available at \url{https://osoconn.com}. The connections
are designed in accordance to the 14th edition AISC 360 specifications using the ASD
method to determine the allowable strength of a connecting element.
The value of the allowable strenght is compared against the required strength, and the
ratio
between the two is calculated as the interaction ratio. If the interaction ratio obtained
is
less that 1.0 then the design is considered satisfactory.
\begin{equation}
    I = \frac{R}{R_a}
\end{equation}
where,

\null\quad\quad \(I\),is the interaction ratio\\
\null\quad\quad \(R\), is the required strength\\
\null\quad\quad \(R_a\), is the allowable strength

The output of the connection design software is provided in Attachment 1.
```

The resulting pdf would look like this.



Connection Design for Piperack PR-01 Rev:2

1 PURPOSE AND SCOPE

The purpose of this calculation is to design the shear connections for the Big Steel Plantproject.

2 GEOMETRY

Beam shear connections are designed to transfer negligible moments across joints. The connection may be between beam and beam or between beam and column. The connection is made using a clip angle that is welded to the connecting beam and bolted on to the supporting member. This detail is chosen because of the ease with which it can be fabricated and erected. A typical beam to beam shear connection is shown in figure 1.

Figure 1: A shear connection

3 MATERIAL SPECIFICATIONS

The material specifications considered for the design of the shear connections are as shown in the table 1.

Element	Specification
Beams	ASTM A992
Columns	ASTM A992
Clip angles	ASTM A36
Bolts	ASTM F3125
Weld	FEXX 70

Table 1: Material specification

4 DESIGN PHILOSOPHY

The connection design is done using the open source connection design software Osoconn developed by Roshn Noronha, and available at https://osoconn.com. The connections are designed in accordance to the 14th edition AISC 360 specifications using the ASD method to determine the allowable strength of a connecting element. The value of the allowable strenght is compared against the required strength, and the ratio between the two is calculated as the interaction ratio. If the interaction ratio obtained is less that 1.0 then the design is considered satisfactory.

$$I = \frac{R}{R_a} \quad (1)$$

where,

2

As we can see LATEX took care of all the formatting for us. It also inserted cross referencing links and hyperlinks to websites automatically for us. This was because we had imported the hyperref package.

Run the design software:

To perform the calculation for our hypothetical connection design calculation we shall make use of the free and open-source connection design software (created by the author) Osoconn. This utility has a command line interface which can be called from the Linux terminal. Assume we have the connection input data saved in .cin files in a folder called design_files. A sample input file is shown here.

```
CONNECTION_ID SBC1
TRANSFER_FORCE 15000
SHEAR_FORCE 90000
SUPPORT_TYPE BEAM
BEAM_SECTION W410X67
SUPPORT_SECTION W410X67
COPE_LENGTH 150
TOP_COPE_DEPTH 50
BOTTOM_COPE_DEPTH 50
BACKING_BEAM YES
CLIP_ANGLE_TOP 60
```

The standard connection details are saved in a file called standard.txt. The program iterates through these standard details and picks the first detail that results in a safe design. To run the program (which is located in ./osoconn/ folder) we can enter the following command in the Linux terminal.

```
./osoconn/osoconn -s ./design_files/standard.txt ./design_files/
```

The program will save the output in a .cou file generated for each .cin input file. A sample output file will look like the following.

```
------------------------------------------------------------------------+----------------------
Design Summary                                                          |
------------------------------------------------------------------------+----------------------
   Connection is OK                                                     |
   Maximum utility ratio for connection                                 | 0.608
------------------------------------------------------------------------+----------------------
Design Inputs                                                           |
------------------------------------------------------------------------+----------------------
   Design method                                                        | ASD
   Young's modulus of elasticity                                        | 200000.000 MPa
   Poisson's ratio                                                      | 0.300
                                                                        |
Connection forces:                                                      |
   Transfer force (TF)                                                  | 15000.000 N
   Shear force (SF)                                                     | 90000.000 N
…
```

To download the program or to read the complete documentation see https://osoconn.com.

Generate a Summary of the Output

Now that we have the output of the design calculations, our next step is to create an executive summary of the design results. Since the input and output files of Osoconn are nicely formatted we can use a scripting language to parse through these files and extract the information that we need.

For this purpose, we create a Bash script. Bash scripts can be run directly from the Bash shell and we don't require a programming language like python. It is very convenient to work with files using Bash scripts. The first line of the script states the path to Bash shell. Then we use a series of echo commands – which normally sends a string to the standard output – but we redirect the output to the result.tex file using the > and >> redirect operators. The first operator creates a new file, while the latter appends to a file if it exists.

To iterate through the input and output files we use a for loop with an in statement, which loops through all the .cin files present in the design_files directory. We parse each file using the sed command. Sed stands for stream editor, and it is a power command line tool used to search and edit text files without having to open them. How it works is that we give a pattern to look for, and sed finds these patterns and performs the operation that we specify on them. These patterns and the operations to perform are provided as a string in a particular format.

Sed is one of those commands that you could spend a lifetime mastering as there are so many options available. For this article I just made use of the substitution functionality. The basic format of the string to perform substitution and print the output is shown below.

```
Sed -n 's/search_string/replace_string/p' filename
```

The first s stands for substitution, and the last p stands for print the output. Sed reads the file with name filename line by line and when it finds a line that matches the pattern defined in *search_string* it replaces it with the *replace_string*. The search string can be a regular expression. For example '.' Will match any character or number and '*' says match the previous value 0 to many times. So '.*' would say match 0 to many of any character or number. So the pattern 'CONNECTION_ID .*' will match a line containing the string CONNECTION_ID followed by a space and then followed by 0 or

many characters. If we look at the Osoconn input file shown above this would match the first line ‘CONNECTION_ID SBC1’.

```
#!/usr/bin/bash
echo "\newcommand{\cellwidth}{1.75cm}">result.tex
echo "\begin{table}[h]" >> result.tex
echo "\centering">> result.tex
echo "\ttfamily">>result.tex
echo
"\begin{tabular}{m{\cellwidth}m{\cellwidth}m{\cellwidth}m{\cellwidth}m{\cellwidth}m{\cellw
idth}m{\cellwidth}m{\cellwidth}}">> result.tex
echo "ID &Transfer force (N) & Shear force(N) &Support type &Beam &Support &Max Ratio
&Result\\\\">> result.tex
echo "\hline">> result.tex
for input_file in ./design_files/*.cin
do
    #get connection ID
    conn_id=$(sed -n 's/CONNECTION_ID \(.*\)/\1/p' $input_file)
    #get connection forces
    tf=$(sed -n 's/TRANSFER_FORCE \(.*\)/\1/p' $input_file)
    #get connection details
    sf=$(sed -n 's/SHEAR_FORCE \(.*\)/\1/p' $input_file)
    #get support type
    supp_type=$(sed -n 's/SUPPORT_TYPE \(.*\)/\1/p' $input_file)
    supp_type=${supp_type/_/ }
    #get beam section
    beam_section=$(sed -n 's/BEAM_SECTION \(.*\)/\1/p' $input_file)
    #get support section
    supp_section=$(sed -n 's/SUPPORT_SECTION \(.*\)/\1/p' $input_file)
    #get the corresponding output file name
    o_ext=".cou"
    output_file=${input_file/.cin/}${o_ext}
    if [ -e $output_file ]
    then
        #get maximum interation ratio
        ic=$(sed -n 's/Maximum utility ratio for connection \(.*\)| \(.*\)/\2/p'
$output_file)
        #get design result
        res=$(sed -n 's/Connection is \(.*\)|.*/\1/p' $output_file)
    fi
    echo "$conn_id" >> result.tex
    echo "&$tf" >> result.tex
    echo "&$sf" >> result.tex
    echo "&$supp_type" >> result.tex
    echo "&$beam_section" >> result.tex
    echo "&$supp_section" >> result.tex
    echo "&$ic" >> result.tex
    echo "&$res\\\\" >> result.tex
done
echo "\hline">> result.tex
echo "\end{tabular}" >> result.tex
echo "\caption{Design Results}">> result.tex
echo "\label{tab:res_tab}" >> result.tex
echo "\end{table}">>result.tex
```

Sed can be told to mark out a particular portion of the line. This portion is marked using the \(and \) pair. So if we want to mark the portion of after the space in the preceding example, we enclose it with these. Hence, ‘CONNECTION_ID \(.*\)’ tells sed to mark out the ‘.*’ portion of the line, which in our case will match ‘SBC1’. Later we can refer to the marked portion using the \1 character sequence. This we will use in the *replace_string* portion to tell sed to replace the line with the portion we have marked out and print the output. One thing to note here is that sed does not actually edit the original file unless explicitly told to do so.

This way we can iterate through all the input and output files extracting the information that we require and generating the LATEX file in the process. The output looks like the follows.

5 DESIGN RESULTS

ID	Transfer force (N)	Shear force(N)	Support type	Beam	Support	Max Ratio	Result
SBC1	15000	90000	BEAM	W410X67	W410X67	0.608	OK
SBC2	13000	34000	BEAM	W310X52	W360X51	0.472	OK
SBC3	11000	45000	BEAM	W260X32.7	W250X67	0.822	OK
SBC4	34000	15000	COLUMN FLANGE	W310X60	W360X64	0.680	OK
SBC5	45000	90000	COLUMN WEB	W410X67	W360X91	0.889	OK

Table 2: Design Results

Attach the Output Files

Our calculation looks great. One final step is pending in our calculation preparation process. It is typical to attach the output of design software as an attachment to calculations for record. We already have the output files from the Osoconn run that we did. These are in text format. We will first have to convert them to pdf format.

To do this we use the lpr command to print the text file to pdf format. This assumes that a pdf printer is installed and is the default printer. We iterate through all the output files in the the design_files directory printing each in turn. The command sends the file to a print queue, so it is essential to tell the script to wait for some time till the printing is done. We do that with the sleep command, with which we induce a pause of 2 seconds.

One thing to note is that my pdf printer saves the output files in a folder called PDF in my home folder so a bit of house keeping is required here. Before printing we move all existing files in the PDF folder to a temporary folder. Then after printing be move all the pdf files generated to our design_files directory. Then we restore all the original files deleting the temporary folder.

```
#!/usr/bin/bash
#backup all existing pdf files in ~/PDF
mkdir ~/PDF/temp
mv ~/PDF/*.pdf ~/PDF/temp/
#iterate through all output files and print them
for file in ./design_files/*.cou
do
    if [ -f "$file" ]; then
        echo "Printing $file..."
        lpr -p -o media=Custom.9.5x11in $file
    fi
done
#wait till printing is done
sleep 2
#move all printed files from default folder to a folder under the current folder
if [ -d ./design_files/script_output_pdf_files ]
then
    rm ./design_files/script_output_pdf_files/*.pdf
else
    mkdir ./design_files/script_output_pdf_files
fi
mv ~/PDF/*.pdf ./design_files/script_output_pdf_files/
#restore backup files and delete the temporary folder
mv ~/PDF/temp/* ~/PDF/
rm -d ~/PDF/temp

#crear contents and update attachement.tex
> ./attachment.tex
for file in ./design_files/script_output_pdf_files/*.pdf
do
    echo "\includepdf[pages=-]{$file}" >> ./attachment.tex
done
```

To attach the newly created pdf files we use the \includepdf command in LATEX. As the name suggests, it includes the given file to the current document. This is command is from the pdfpages package. We use the echo command to write a series for \includepdf commands for each pdf file into the file attachement.tex.

Bringing it All Together

We are almost done now. Only thing left is to accomplish what we set out to do – do everything that we just did with one command. For this we create one more Bash script. The only thing this file does is calls all the other script files that we have created so far and run pdflatex.

```
#!/usr/bin/bash
#extract the document number from calc_number.tex file
calc_num=$(sed -n 's/.*calcnumber}{\(.*\)}/\1/p' calc_number.tex)
#Read the database and generate macro.tex
echo "Reading database..."
python3 ./read_database.py "$calc_num"
#Run the design software
echo "Running osoconn..."
./osoconn/osoconn -s ./design_files/standard.txt ./design_files/
#Summarize the results in results.tex
echo "Summarizing results..."
./summarize_results.sh
#Print the output and create attachement.tex
echo "Printing output..."
./print_output.sh
#Compile the calculation. This is required to be done twice for the
#the table of contents to be displayed properly.
echo "Compiling calculation..."
pdflatex calc.tex
pdflatex calc.tex
echo "DONE!"
```

And we are done. Now we can do everything we just did using the single command.

```
./compile_calc.sh
```

All the code and the final output file is available on my github page. Hope you enjoyed this tutorial as much as I enjoyed making it.