SOFTWARE ARCHITECTURE
4. TEXT FORMATTING SYSTEM

Tatsuya Hagino
hagino@sfc.keio.ac.jp

lecture URL

https://vu5.sfc.keio.ac.jp/slide/

Text Formatting System

- Text Formatting
  - Print out document nicely
  - Align the right-hand side of lines
  - Use nice fonts
  - Print out complicated mathematical formulae

- Two types of text formatting system
  - WYSIWYG
    - What You See Is What You Get
    - Directly manipulate and see the result of formatting
    - Easy to use
    - Heavy processing
    - Sometimes screen may not exactly match print out
  - Batch
    - Prepare text, run a format program to format
    - Need preview to check the result
    - Can handle large size document like books
    - Can generate contents, references and so on

Text Formatting System on UNIX

- roff
  - Stands for “to run off a copy”
  - For formatting UNIX manual.
  - UNIX version of Multics runoff.
  - Extended to nroff, troff, groff, etc.
  - Can be combined with tbl and eqn to handle tables and mathematical formulae.

- TeX
  - By Donald E. Knuth (Stanford University)
  - Donald Knuth is a famous researcher of computer algorithms
  - Author of “The Art of Computer Programming”
    - Difficult to format mathematical formula
    - Did not like to proofread and correct again and again
    - Printing company does not know mathematical formula
    - Wanted to control everything in typesetting
  - Created own fonts for mathematical formulae
  - A macro extension LaTeX is often used for writing scientific papers.

Features of TeX

- Macro extension mechanism
  - Customizable
  - Can write programs
- Fonts can be designed
  - METAFONT
  - Vector fonts (vs bitmap fonts)

- Developed by “Literate Programming”
  - Difficult to write documents of programs afterwards
  - Combine programs and documents together
  - WEB (Weave program and document)
  - Extract programs and documents from the same WEB

Example of WEB

- WEB

1. Central Algorithm
   do_something() is the main process.

```
<a routing> == item.do_something().
```

2. Main Loop
   Process items in a collection using **<a routing>.**

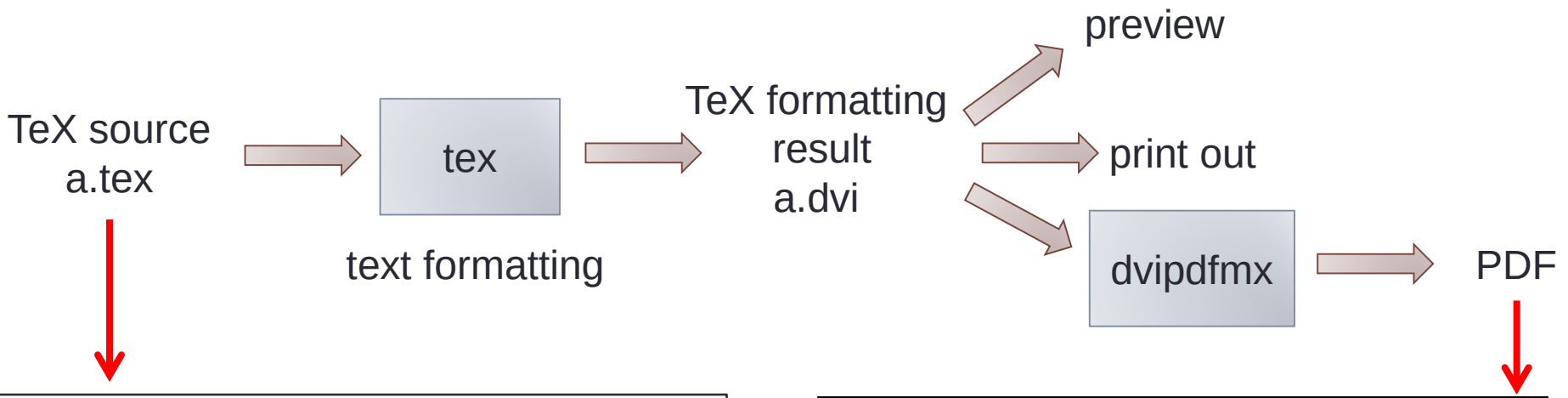
```
<main> == for item in collection
            <a routing>
```

- Extract a program using Tangle

```
for item in collection
    item.do_somthing().
```

- Using WEB, programs are written one by one with their explanation.
  - Program Refinement
  - Top Down Programming

Example of TeX Processing




```
This is a {\it very} {\bf simple} \TeX{}
source file. We can write equation like
$a x^2 + b y + c = 0$ and
$$1+2^2+3^2+\cdots+n^2 = \sum_{i=1}^{n} i^2
= {n (n+1)(2n+1) \over 6}$$
easily.
```

This is a *very* **simple** TEX source file. We can write equation like $ax^2 + by + c = 0$ and

$$1 + 2^2 + 3^2 + \cdots + n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

easily.

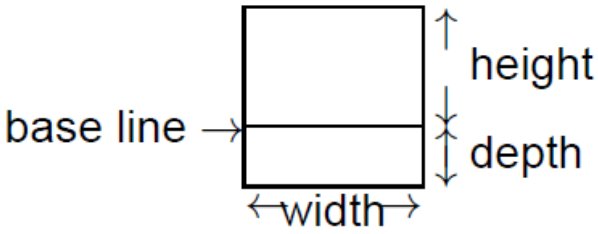
- A source consists of text, macros and formulae.
- A macro starts with '\' (backslash).
- A formula is surrounded by '$'.
- '{' and '}' are used to create groups.
  - Can be nested.

Text Formatting Features

- Features necessary for text formatting:
  - Specifying fonts (roman, gothic, italic, etc.)
  - Specifying the size of fonts.
  - Centering a line, flushing to right, proportional spacing.
  - Breaking lines which are longer than the text width (hyphenation)
  - Formatting mathematical formulae.
  - Breaking into pages.
  - Inserting tables and figures.
  - Putting page numbers, headers and footers.
  - Numbering chapters, sections and so on.
  - Creating a table of contents.
  - Creating references, indexes and so on.

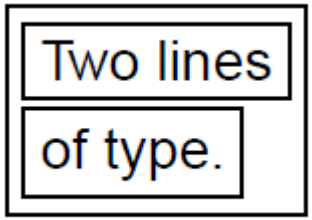
Box Model of TeX

- Box
  - Boxes are connected horizontally and vertically.
  - A character is the smallest box.
  - Each box has width, height and depth.
  - Boxes are connected horizontally at base line.
  - Depth
    - Kanji and Hiragana have depth 0.
    - Alphabets have depth.
    - Zenkaku ‘(’ and hankaku ‘(’ have different depth.



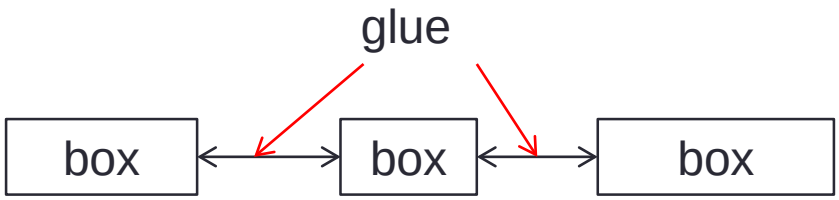
- hbox and vbox
  - hbox's are connected horizontally.
  - vbox's are connected vertically.
  - Example

```
\vbox{\hbox{Two lines}\hbox{of type.}}
```

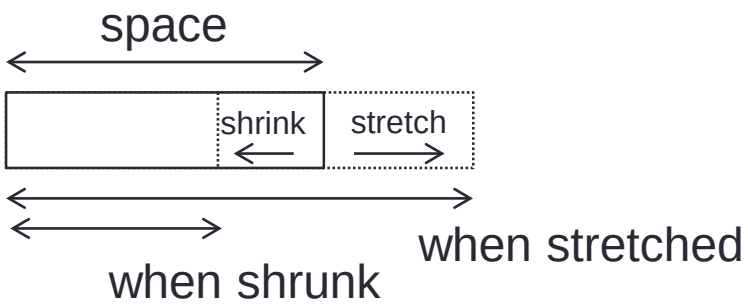


Glue

- Boxes are connected using glue
  - Bond, gum
  - Stretch and shrink

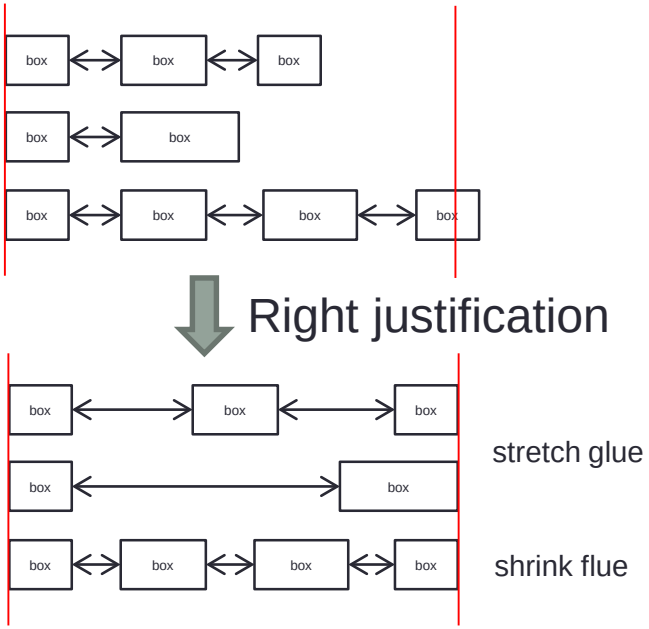


- Each glue consists of:
  - space: normal width
  - stretch: how much it can stretch
  - shrink: how much it can shrink



- Example
  - **`\hskip 10mm plus 8mm minus 3mm`**
  - The normal size is 10mm.
  - May have size of 7mm～18mm.

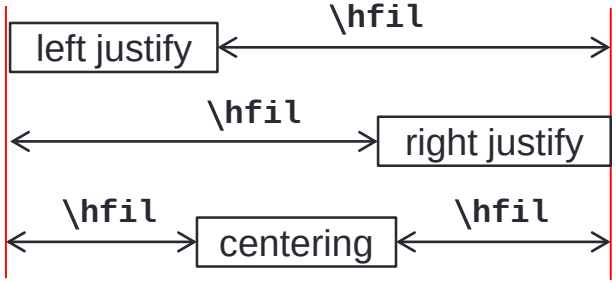
- Right justify all the lines:
  - Each line has the same width.
  - Too long lines are broken into multiple lines.
  - Glue may be stretched or shrunk.
    - Spaces are glue.



Glue of Infinite Stretch

- Glue may be stretched infinitely.
  - **`\hfil`**
  - **`\hskip 0pt plus 1fil`**
  - Can be stretched infinitely in the horizontal direction.

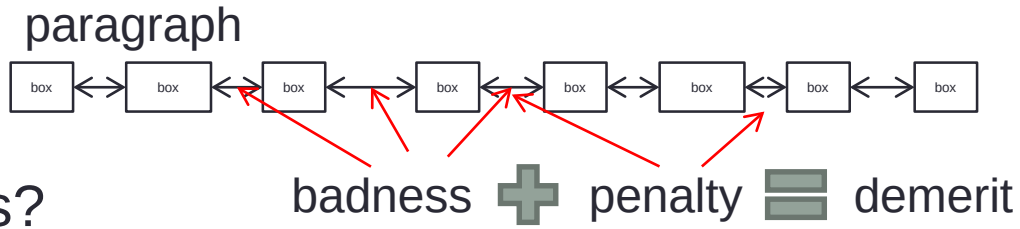
- Can be used for centering lines, flushing lines to right or left.
  - **`\line{left justify\hfil}`**
  - **`\line{\hfil right justify}`**
  - **`\line{\hfil centering\hfil}`**



- Multiple infinite
  - **`fil, fill, filll`**
  - Larger infinite ignores smaller infinite.

- Infinite shrink
  - **`\hss (\hskip 0pt plus 1fil minus 1fil)`**

Breaking lines



- When to start breaking lines?
  - All the lines are connected from left to right
  - Carriage returns are treated as spaces.
  - Blank line separates paragraphs.
  - When there is a paragraph, it is divided into lines.

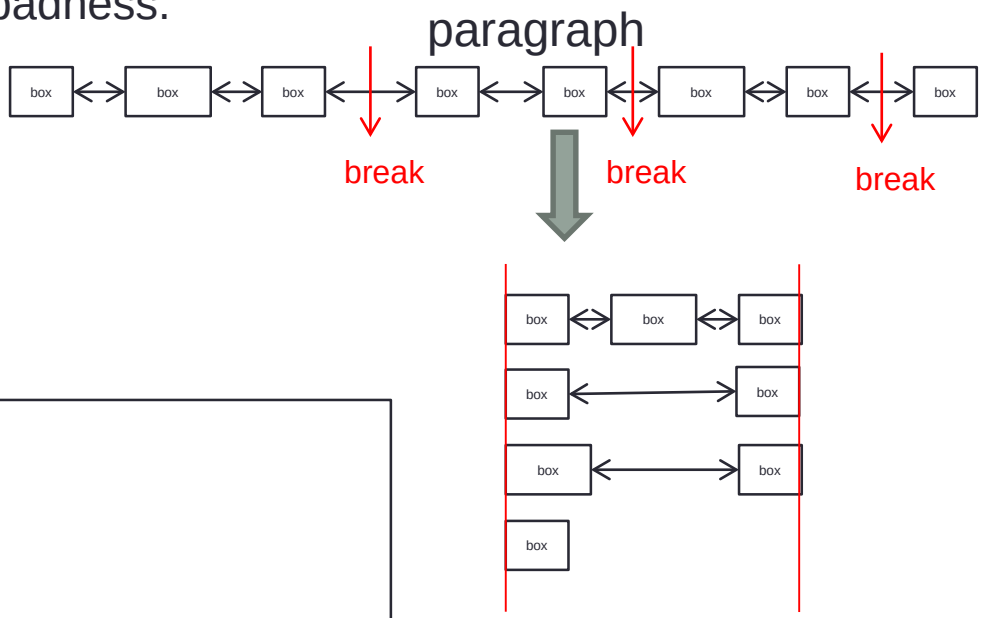
- Determine line breaks:
  - Minimize the demerit（d）of a paragraph.
  - Line badness（b）
    - Cube of shrink and stretch ratio times 100 (max is 10000).

$$d = \begin{cases} (l+b)^2 + p^2 & (0 \le p < 10000) \\ (l+b)^2 - p^2 & (-10000 < p < 0) \\ (l+b)^2 & (p \le -10000) \end{cases}$$

    - Break point penalty（p）
    - Default line badness（l）, initial value 10, can be set by **`\linepenalty`**

Line Break Algorithm

- Breaking lines to minimize the sum of badness.
- Minimization problem
  - Cannot try all the possible breaks.
  - Use Dynamic Programming
- Dynamic Programming
  - Find shortest path

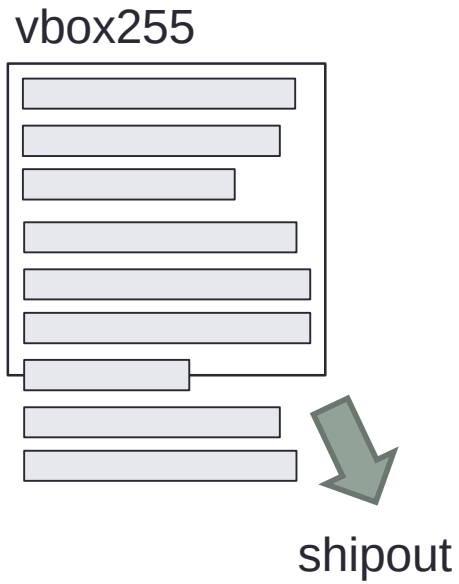



```
for (i = 1; i <= n; i++) {
  for (j = 1; j <= n; j++) {
    a[0,i,j] = w[i,j];
  }
}
for (k = 1; k <= n; k++) {
  for (i = 1; i <= n; i++) {
    for (j = 1; j <= n; j++) {
      a[k,i,j] = min(a[k-1,i,j], a[k-1,i,k] + a[k-1,k,j]);
    }
  }
}
```

w[i, j] distance between **i** and **j**

a[k, i, j] shortest distance from **i** and **j** using nodes less then or equal to **k**

Page break and Output

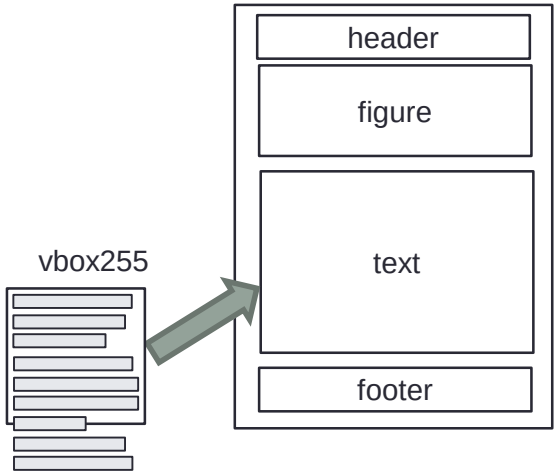


- Create pages from lines.
  - Calculate page cost from badness and penalty.
  - Calculate local minimization.
    - Global minimization too costly

- Output processing
  1. Specify the text height with **`\vsize`**
  2. Lines are put into **`\vbox255`**
  3. When the height of **`\vbox255`** becomes bigger than **`\vsize`**, **`\output`** is called.
  4. Use **`\shipout`** to actually output.

```
\def\output{\shipout\vbox255}
```

- **`\output`** adds headers, footers and figures.
- Some lines of **`\vbox255`** may not be shipped out.
  - The rest will be kept in **`\vbox255`** for the next page.



Macros

- Formatting process is complicated.
  - Generating chapter and section numbers.
  - Inserting headers, footers, figures and tables.

- Define macros:

```
\def\sfc{Shonan Fujisawa Campus}
\sfc
\def\sfc#1#2{SFC#1 building #2 floor}
\sfc A3
\sfc{High School}{2}
```

- Macro programming:
  - Call macros inside macros
  - Conditional using if macro
  - Use recursive call

```
\def\money#1{{\ifnum#1<0$\triangle$\count3=-#1\else\count3=#1\fi\count4=0\mloop}}
\def\mloop{{\count0=\count3 \divide\count3 by 10
    \advance\count4 by 1
    \ifnum\count4=3 \count4=0\fi
    \ifnum\count3>0 \mloop\ifnum\count4=0 ,\fi\fi
    \count2=\count3 \multiply\count2 by -10
    \advance\count0 by\count2 \number\count0}}
```

LaTeX

- LaTeX is TeX with macros.
  - Similar to text formatting system scribe
  - Closer to mark up language
  - Define environments

- Close to HTML tags

```
\begin{itemize}
\item The first item
\item The second item
\end{itemize}
```

```
\documentclass[a4]{report}
\begin{document}
\chapter{Introduction}
This is an example.
\begin{itemize}
\item The first item
\item The second item
\end{itemize}
\end{document}
```

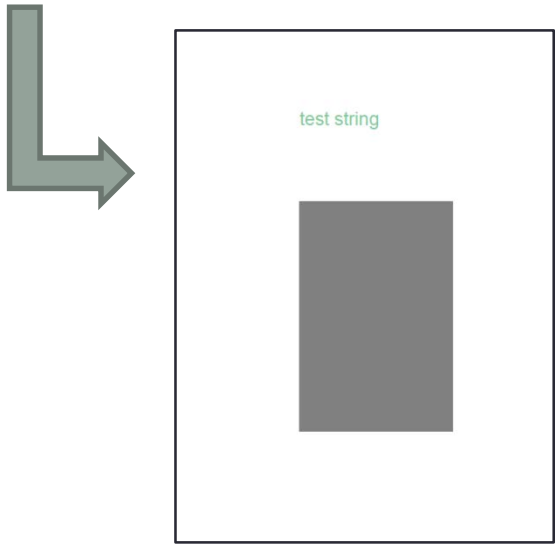
- Specify document style
  - **book**, **report**, **article**, etc. specified by **documentclass**
    - **article**: consists of sections
    - **report**: consists of chapters and sections
    - **book**: consists of parts, chapters and sections

Output Format

- DVI
  - TeX original output format
  - DeVice Independent
- PS
  - PostScript
  - Adobe page description language
  - A set of drawing operations
  - Stack base programming language
- PDF
  - Portable Document Format
  - Adobe defined
  - A set of objects

PS

```
%!
/RRECT { newpath 4 copy pop pop moveto dup 0 exch
rlineto exch 0 rlineto neg 0 exch
rlineto closepath pop pop } def
100 100 100 150 RRECT
.5 setgray
fill
100 300 moveto
/Helvetica findfont
12 scalefont
setfont
.5 0 .5 0 setcmykcolor
(test string) show
showpage
```



Summary

- Text Formatting System
  - WYSIWYG vs Batch
  - TeX and LaTeX
  - box and glue
  - line break optimization
  - macro

- References
  - “TeXbook, The (Computers & Typesetting)”, Donald E. Knuth, Addison-Wesley Professional
  - “Literate Programming (Center for the Study of Language and Information Publication Lecture Notes)”, Donald E. Knuth, Cambridge University Press