電子組版による
科学技術文書の作成

2023年度2Q 5c/6c（IL2） 木曜日

担当: 地引

TA: 増井

コンピュータにおけるデータの表現形式

コンピュータにおける数値や文字の表現（1）

- データの電子的表現
  - コンピュータ上では、あらゆるデータを二進法で表現
    ≫ 0/1 → つまりは、電気を流すか/流さないか
  - 文字データも二進法で表現される。

- 二進法の利用
  - データの実体は二進表現された数 → 10110010 など
  - 人間が直接扱うのは少々困難 → 人間は“文字”を使う。
  - 人間が扱う“文字”と電子的な二進表現との対応付けが必要
    → どんな対応表を作ればよいか？

コンピュータにおける数値や文字の表現(2)

- 二進法の利用(続き)

  - 二進法 4 桁の最大値 = $2^4 - 1 = 15$
    → 二進法 4 桁までの数字は、0 ～ 15 まで 16 個の数字が存在する。
  - つまり、二進法 i 桁の数字は 2^i 個ある。
  - N 種類のデータと対応させる場合は、
    $2^{i-1} < N \leqq 2^i$
    を満たす二進法 i 桁の数が必要
  - 欧米文字は 100 文字程度
    ≫$2^7 = 128$ より、二進法 7 桁で表現可能
    ≫例えば、大文字 A は、二進表現 01000001 で表わされる。

コンピュータにおける数値や文字の表現(3)

- 二進法の利用(続き)

  - 0/1 だけだと見づらいので、
    二進法の 4 桁を 1 桁にまとめた十六進法がよく使われる。

  - 例えば、文字 A を表わす二進法 01000001 に対して、4 桁ずつ区切ると
    | 0100 | 0001 | となり、この 4 桁を 1 桁にまとめて | 4 | 1 | と表わす。

  - 但し、このままだと十進法と紛らわしいので、
    十六進法は先頭に “0x” という記号を付けて、0x41 と表現する。

文字コードの導入

- 文字コード
  - 人間が扱う“文字”と電子的な二進法表現との対応付けが必要
    → これが文字コード
  - **欧米文字は二進法 7 桁**で表現可能（ASCII コード）
    ≫欧米文字 = 100 文字程度，$2^7 = 128$
  - 欧米以外の文字 → 二進法で何桁必要か不明
    ≫文字データを特殊な制御コードで囲む。
    ≫囲まれた文字データに固有の文字コードを適用

- 文字だけではなく、画像/音声など、様々なデータと二進法表現との対応が定義されている。

文字コードの種類

- ASCII コード：最初に米国で定義された文字コード
  - アルファベット/数字等を二進法 7 桁と対応付け
- JIS コード（ISO-2022-JP）：インターネット上で日本語を表示する標準
  - 制御コードを導入，複数バイト コード（二進法 16 桁以上，$2^{16} = 65536$）
- シフト JIS コード：マ社が独自に作成した文字コード
- EUC コード：UNIX 上で利用される文字コード
- Unicode：全文字に同一の符号化方式（複数バイト コード，最近のDe-facto Standard）
  - 「符号化文字集合（各文字固有の二進法表現）」+「文字符号化方式（UTF-8 等）」
  - 当初は二進法 16 桁に抑えようとした（似ている文字は同じ符号で表現 → 大きな反発）

電子データの単位

- 電子データは、微細なものから巨大なものまで存在するので、独特な単位が用意されています。
  - 二進法表現の 1 桁分を 1 bit と呼びます。
    - ≫ 1 bit が表わせる数字は、0/1 の 2 個だけです。
  - 二進法表現の 8 桁 = 8 bit を 1 byte と呼びます。
    - ≫ byte は、電子データの基本単位です。
    - ≫ 1 byte が表わせる数字は、$2^8 = 256$ 種類です。
    - ≫ byte は、単に B で表わすことがあります。
    - ≫ 1000 byte を 1 KB, 1000 KB を 1 MB, 1000 MB を 1GB, 1000 GB を 1 TB と呼びます。
  - 例えば、1 秒当たりのデータ転送量を、bit/s, bps（bits per second）等と表現します

電子的な文書作成の基本

文書データの形式

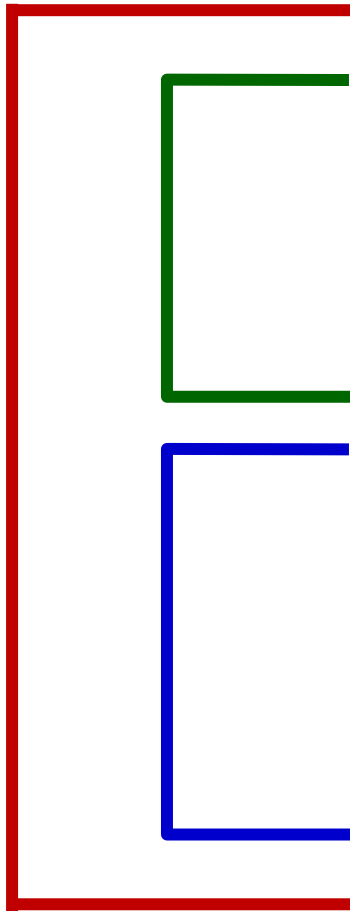
- 文字データそのものは二進法表現だとして、文書として扱う場合に必要な体裁などのデータは、どのように扱われるのか？

- 文字データに必要な属性/命令などを付加
  - 例えば、ホームページ（HTML）では‥
    - ≫＜と＞で囲まれた HTML タグを用いて記述
    - ≫**＜ほげほげ＞ ‥‥‥ ＜/ほげほげ＞**
    - ≫**ほげほげ**の部分で、‥‥‥ に記述されている部分の処理方法を指定する。

- このような形式をマーク アップ形式と呼びます。
  - マーク アップ形式を用いた文書作成の基本として、まずはホーム ページの中身を見てみましょう。

ホーム ページ（HTML）の記述例

入れ子構造



```
<!DOCTYPE html>
<!-- 記述に用いる自然言語を明記する -->
<html lang="ja">
    <head>
        <!-- 文字エンコーディングの種別を明記する -->
        <meta charset="utf-8" />
        <!-- ウェブページのタイトル -->
        <title>地引昌弘のホームページ</title>
    </head>
    <body>
        <!-- ウェブページの大見出し -->
        <h1>地引昌弘のホームページ</h1>
        <!-- 段落 -->
        <p>ほげほげ</p>
        <p><em>強調する文章</em></p>
    </body>
</html>
```

HTML タグを入れ子にすることで、複雑な表現が可能

なぜ簡単なツールを使わないの？

- コンピュータ/ネットワーク技術の進展に伴い、これまでは解決の非常に難しかった技術的問題も解決できるようになった。
- 情報環境は完成された技術分野ではなく、今後も次々と姿を変えながら進化していく分野
- 難問に対して情報環境の観点から新しいアプローチをするには、情報環境の仕組みを理解し、それを応用できる必要がある。
  - それをやらない ⇒ 所詮は他人の用意したメニューをなぞるだけの 2 番煎じ
- 以下は私個人の意見
  - 多くのソフトウェア：工業製品としては最悪の品質
  - ブラック ボックス化（or 自動化）→ 低レベル化（Creative な作業に向かない）
  - 手厚い機能 vs. 信頼/性能の悪化

文書作成ツール

実は Word にも、フィールド コードと呼ばれるコマンドが埋め込まれている。

- ワープロ：
  - 基本的には画面に表示されている通りに印刷される。
    ≫WYSWYG：What You See is What You get
  - 書体の変更や文字以外のデータ取り込みなど機能が豊富
    ≫内部で動的にマークアップを作成 → 処理の洪水で遅い/バグが多い。

- エディタ：
  - 基本的には文字の入力のみ。
    ≫Windows のメモ帳はエディタの一つ。
  - シンプルなので速い/バグが少ない ⇒ プロ用
    ≫複雑な文書作成も**コマンドの埋め込みで可能 ⇒ マークアップ系**

自動/手動の境界：システムの勘所

blog

- Weblog（強いて言えば、Web 記録）の略
- 宣伝では、誰でも簡単にホーム　ページを作成できるとのことであるが‥
- 当初の位置付けは日記的な Web サイト
  - Web サーバ上に、ホーム ページ作成支援環境を用意し、定型のページを作成
    - ≫ コメント機能/時系列機能
    - ≫ ユーザは、基本的に内容のみに専念できる。
  - トラック バックによる箔付け
    - ≫ 自分のページが他からリンクされると、リンクされたことが通知されるので、情報発信者としての重みが増す。

Wiki

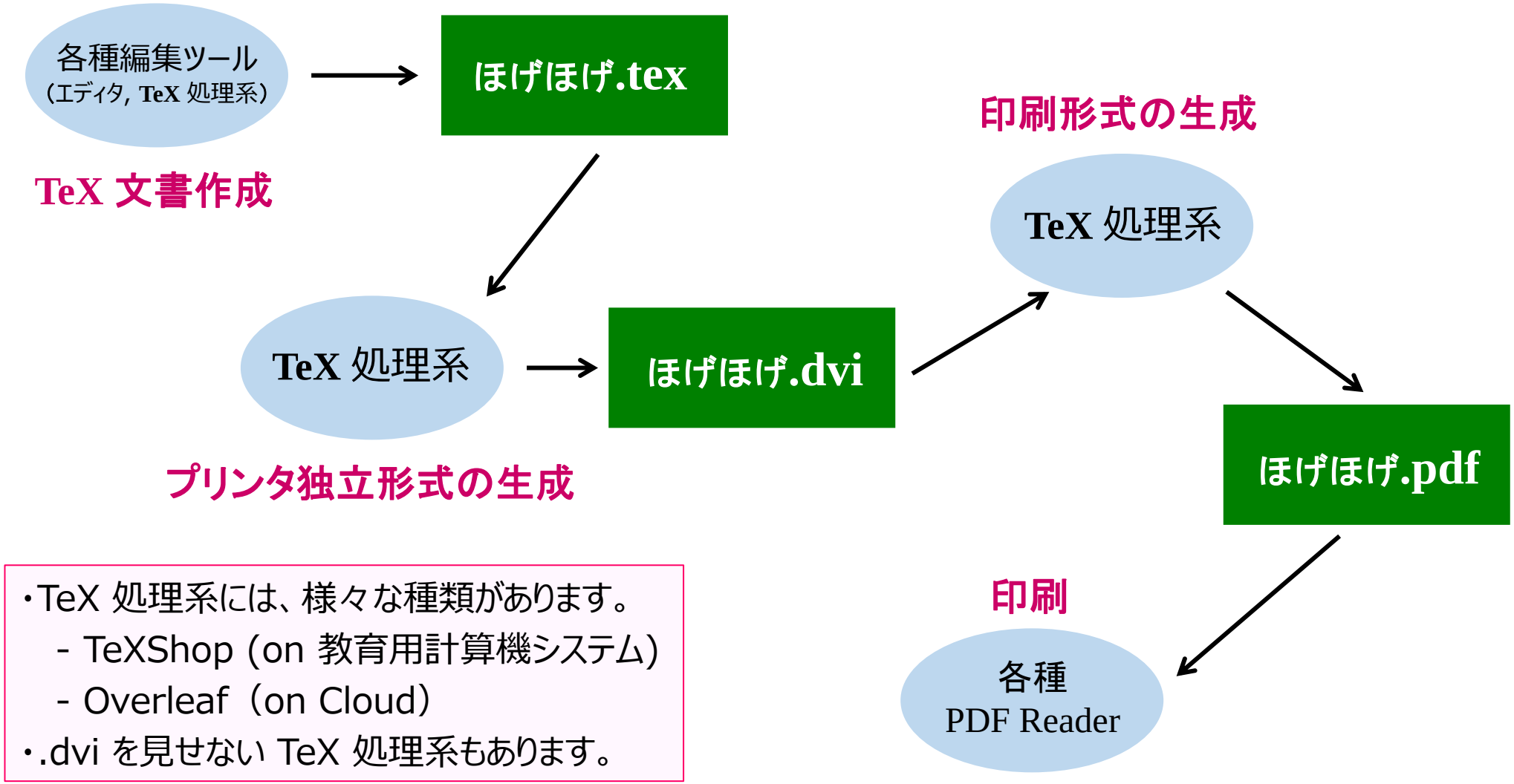
- Web ブラウザを用いて、
  Web サーバ上のハイパー テキストを書き換えるシステム
  - ネットワーク上のどこからでも コンテンツを書き換え可能
  - 物理的に離れたメンバー間で情報共有が可能
- Wiki を構築する様々な派生システムが存在
- 基本動作：
  - **ブラウザ上で、Wiki 用マークアップ言語**により文章を記述
  - **Wiki システムが** Wiki 用マークアップ言語を **HTML へ変換**
- マークアップ言語の扱いに慣れよう

本格的な電子組版

TeX 文法の初歩

- TeX とは
  - 出版物と同程度の品質を備えた本格的な電子組版システム
- TeX のタグ
  - ¥タグ名{命令} という形
  - タグ名/命令は、必ず半角英数字
    ≫例）¥hspace*{1em} → 1 文字分空ける
- 具体的な命令は、共通ページを参照のこと。
  - pLaTeXを用いた論文執筆実習のサポート サイト (https://titechcomp.github.io/y22-il2j/latex/)
  - 不明な部分はインターネットで検索しよう。

TeX による文書生成サイクル



・TeX 処理系には、様々な種類があります。
  - TeXShop (on 教育用計算機システム)
  - Overleaf（on Cloud）
・.dvi を見せない TeX 処理系もあります。

参照文献の記載(1)

- 技術文書では、記述内容の客観的な正当性を示す証拠として、
  多くの参照文献を提示します。
- 参照文献は、後から第三者が確認できるように、下記の項目を明記します。
  - 著者(通常は著者全員)
  - 題名(論文の場合、一般に論文集名と論文名は異なります)
  - 掲載箇所(論文集名, 第 x 巻, y ページなど)
  - 出版社/出版年度/出版地(会議の予稿などの場合は開催地も明記)
- 執筆する文書が増えてくると、上記の項目を毎回準備するのは大変です。
- TeX には、文献データベースを構築/利用する機能が用意されています。

参照文献の記載(2)

- 文献データベース: ほげほげ.bib
  - 今回の演習では、文献 DB となる .bib ファイルが用意されていますが、本来は自分で作ります。
  - .bib ファイルは、エディタでも作成できますが、専用のアプリも存在します(共通ページ参照, BibDesk など)。
  - 複数の .bib ファイルから必要な文献情報だけを抽出して、. tex ファイルに引用することができます。
- 文献データベースの作成/利用については、演習用教材にある
  - paper.tex: 文献 DB の読込み(ファイルの最後部分)
  - references.bib: 各文献情報の記載方法

  を参考にしましょう。

参照文献の記載(3)

参考:

個人的には、TeX ファイルの一番最後に、
下記の thebibliography 環境を直接記述することが多いです。

```
¥begin{thebibliography}{参照文献概数}
    ¥bibitem{参照キー1} 著者1, ``文献名1'', 出版情報1
    ¥bibitem{参照キー2} 著者2, ``文献名2'', 出版情報2
        ...
    ¥bibitem{参照キーN} 著者N, ``文献名N'', 出版情報N
¥end{thebibliography}
```

参照文献概数には、文献数が *i* 桁であれば、適当な *i* 桁の数字が入ります。
文献名の囲みに使う ` は、バッククォート(backquote)と呼ばれ、' の反対向きの形をしています。
この記号は、JIS キーボード(日本語キーボード)であれば、Shift+@キーで入力できます。

クラス ファイル/スタイル ファイル

- TeX では、マーク アップ形式で文書データを直接作成するので、版面サイズ/使用字体/行間/余白/図表の位置ほか、文書の体裁を自由に作成することができます。

- しかし、論文など定型文書において同じ文書データを毎回作成するのは効率的ではないため、通常は、事前にある程度の体裁を定義した雛形用ファイルを用意しておき、.tex 内でこれを読み込んで利用します。

- この雛形用ファイルを、クラス ファイル(.cls)/スタイル ファイル(.sty)等と呼び、大半の学会では、その学会が発行する論文誌に合わせたファイルが用意されています。
  - 例えば今回の演習用データでは、literacy.sty が相当します。
  - ○○.cls, △△.sty は、それぞれ以下のコマンド読み込みます(.tex に記述します)。

    ≫ ¥documentclass[様々なオプション]{○○}, ¥usepackage {△△}

演習を始めるに当たり

- 今後、文書を構成する様々なファイルを作成／利用することになります。
  - 各データファイル（図，グラフ，画像）
  - 状況によっては、複数の TeX ファイル
  - 中間データファイル，印刷形式ファイル

- これらのファイルを整理するために、まずは作業フォルダを作成し、
  そこで作業を行ないましょう。

- 処理の中には、GUI 上でクリックするのではなく、キーボードから、
  コマンドを入力するものもあります。コマンドによる操作を覚えましょう。

例年であれば、教育用計算機システムに用意されている Mac 上の TeX 処理系である TeXShop を用いて演習を行ないます。

データ解析と同様、今年度は、クラウド上で提供されている Overleaf という TeX 処理系を利用します。

演習の進め方（1）

- Overleaf へ実習教材をアップロードします。
  - アップロードするファイルは、以下の 2 箇所からダウンロードしたファイルを展開した後、A： paper.tex, references.bib, literacy.sty の三つ、B： 全ファイルです。

    A）“情報リテラシ第二（5c/6c ページ）” →
    “科学技術文書(.tex)の作成に用いる文書ファイル/参考ファイル”
    ファイル名： theme2.zip

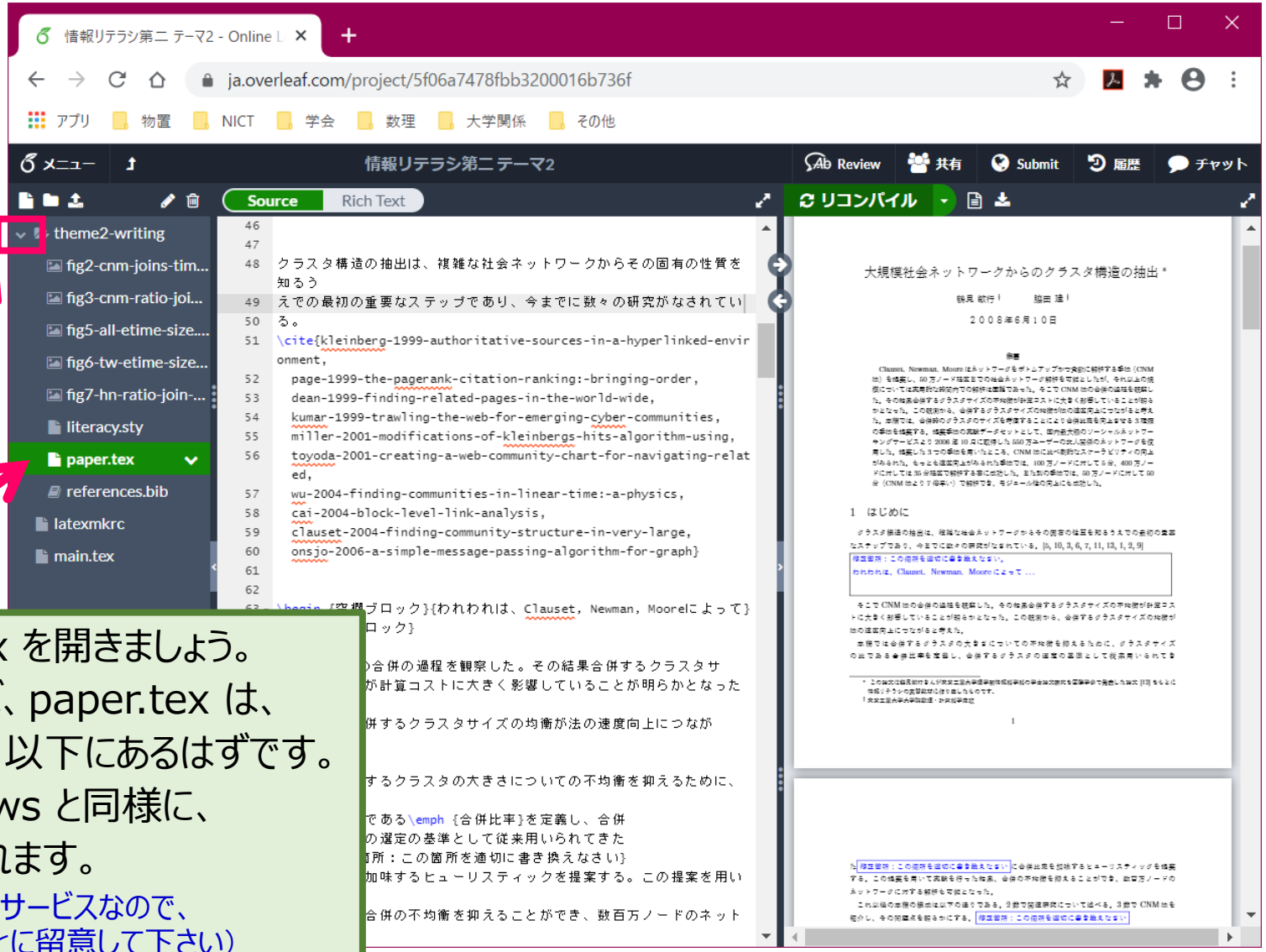
    B）“情報リテラシ第二（5c/6c ページ）” →
    “科学技術文書(.tex)の作成に用いるグラフ ファイル(by gnuplot)”
    ファイル名： images.zip

  - Overleaf へのアップロード方法は、補足資料として 5c/6c のページにある“準備： Overleaf の環境作成” を参照のこと。
  - 同、zip ファイルの展開方法は、“準備： zip ファイルの操作” を参照のこと。

演習の進め方（2）

- Overleaf 上にアップロードした paper.tex, references.bib を修正し、model.pdf と同じ文書を作成します（literacy.sty は修正しません）。
  - model.pdf は、前スライドにあるダウンロード A より展開したファイルに含まれています。
- paper.tex の修正については、同じくダウンロード A に含まれる hint.pdf を参考にして下さい（references.bib については別途説明します）。
  - hint.pdf 内の赤字部分を全て paper.tex 内に記述します。
  - 記述に必要な TeX コマンドについては、hint.pdf 内で修正箇所の近くにある旗マークの右側にある水色で囲まれた文字部分をクリックして下さい。該当する TeX コマンドを説明した Web ページが開かれるはずです。
  - 著者に、自分を追加して下さい。

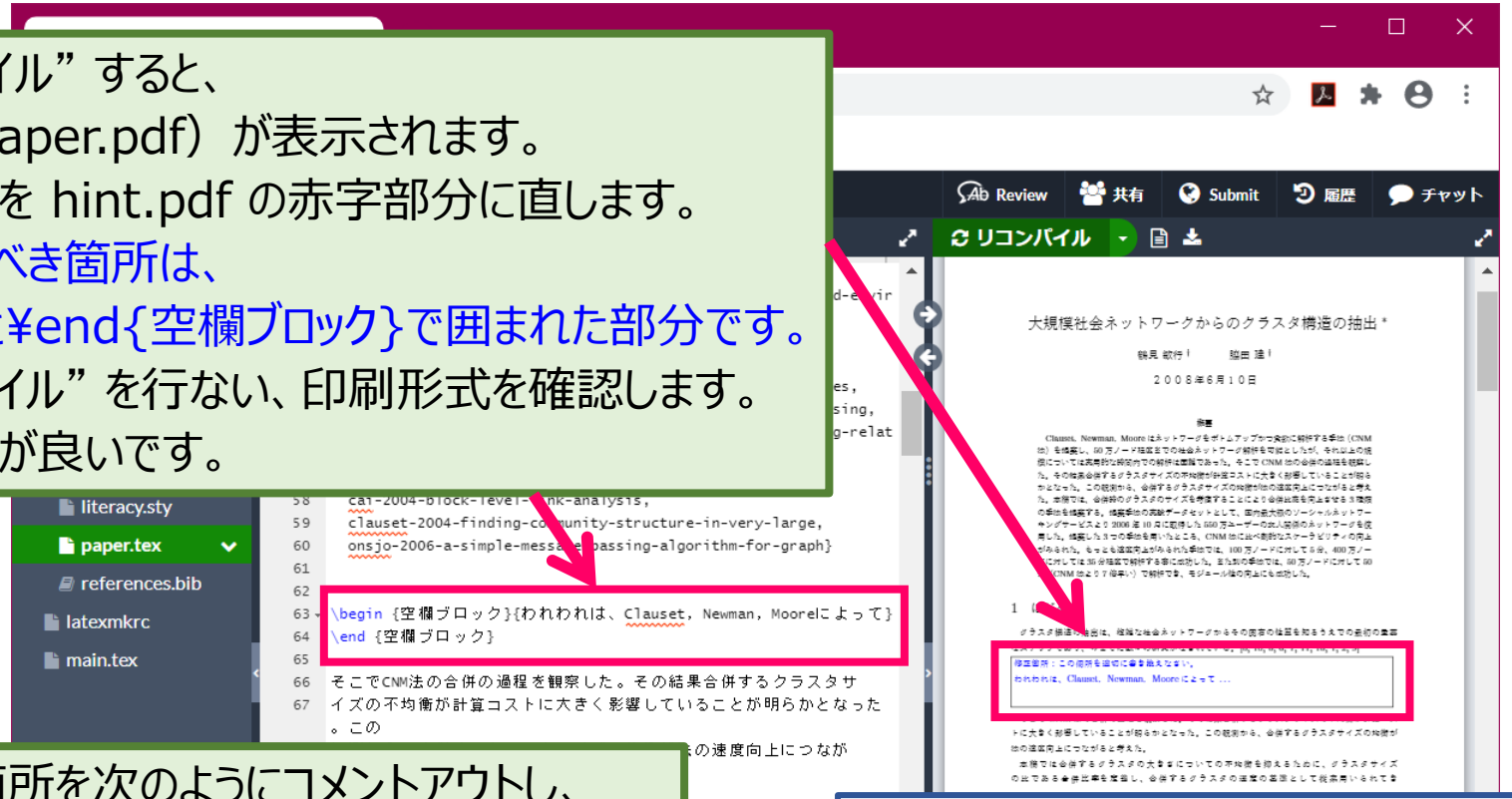
Overleaf 上で paper.tex を修正する（1）



まずは、Overleaf 上で paper.tex を開きましょう。
教材に沿って環境を整備してあれば、paper.tex は、
作業フォルダ “theme2-writing” 以下にあるはずです。
フォルダは、エクスプローラ@Windows と同様に、
右端の “ ” をクリックすると展開されます。
（この図では “V” になっていますが、オンライン サービスなので、
メニューのデザイン等は常時変更されていることに留意して下さい）

Overleaf 上で paper.tex を修正する（2）

① paper.tex を “リコンパイル” すると、
右のような印刷形式（paper.pdf）が表示されます。
paper.pdf の青字部分を hint.pdf の赤字部分に直します。
paper.tex 内で修正すべき箇所は、
¥begin{空欄ブロック}と¥end{空欄ブロック}で囲まれた部分です。
修正後は適宜 “リコンパイル” を行ない、印刷形式を確認します。
修正は少しずつ行なう方が良いです。



② 例えばこの部分では、修正箇所を次のようにコメントアウトし、
%¥begin{空欄ブロック}{われわれは、… によって}
%¥end{空欄ブロック}
hint.pdf にある下記の部分を追記します。
「われわれは、Clauset, Newman, Moore によって提案 …
… 実用的な時間での解析は不可能であった。」

TeX では “%” 以降の部分を単なる注釈として扱い、
印刷形式では表示しません。
これを利用して、TeX のコマンドや文章に % を付け、
注釈として無効化することを “コメントアウト” と呼びます。
コメントアウトは、編集経過を残すことを意味するので、
一時的な編集作業ではよく使われる手法です。
もちろん、その部分の編集が完了すれば（不要になれば）、
削除します（煩雑になるので、常時残しておくことはしません）。