コンピュータ・サイエンス2

第7回
プログラミングとアプリケーション

人間科学科コミュニケーション専攻
白銀 純子

今回の内容

- プログラミング
- プログラミング言語
- アプリケーションの種類
- 情報ネットワーク

設問1

- クラウドコンピューティングの例として思いつくサービス(これはクラウドかな? と思うサービス)を1つ以上挙げなさい。

解答:

- Googleが提供するサービス(Gmail, Googleドライブなど)
- Dropbox
- iCloud
- etc.

前回の質問の回答

クラウドコンピューティング

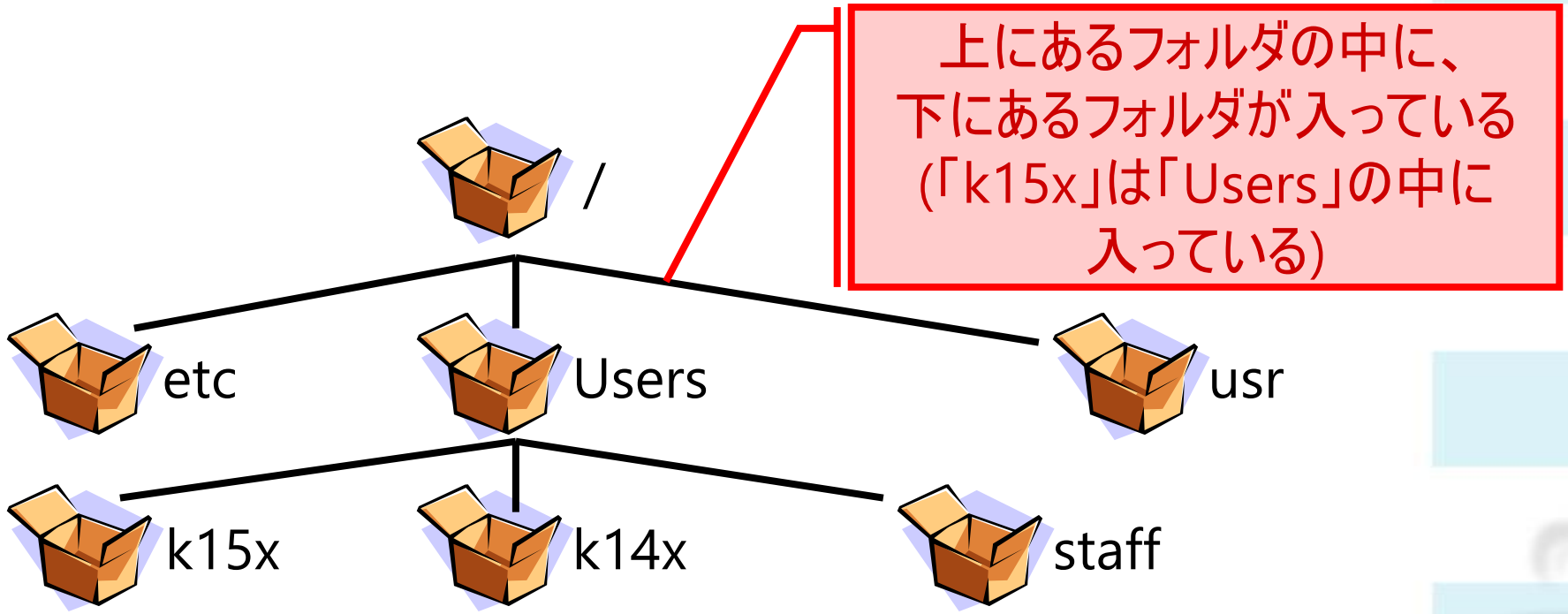
- 身近なところから考えると…
  - 従来はコンピュータに自分でインストールして使っていたアプリケーションがインターネットを通じて使える
    - オフィスソフト, メールソフトなど
  - 作ったファイルをインターネットの世界に保存できる

> これらができるものは、最近はクラウドのサービスであることが多い
> ショッピングなどのサービスは、通常はクラウドとは呼ばない

前回の復習

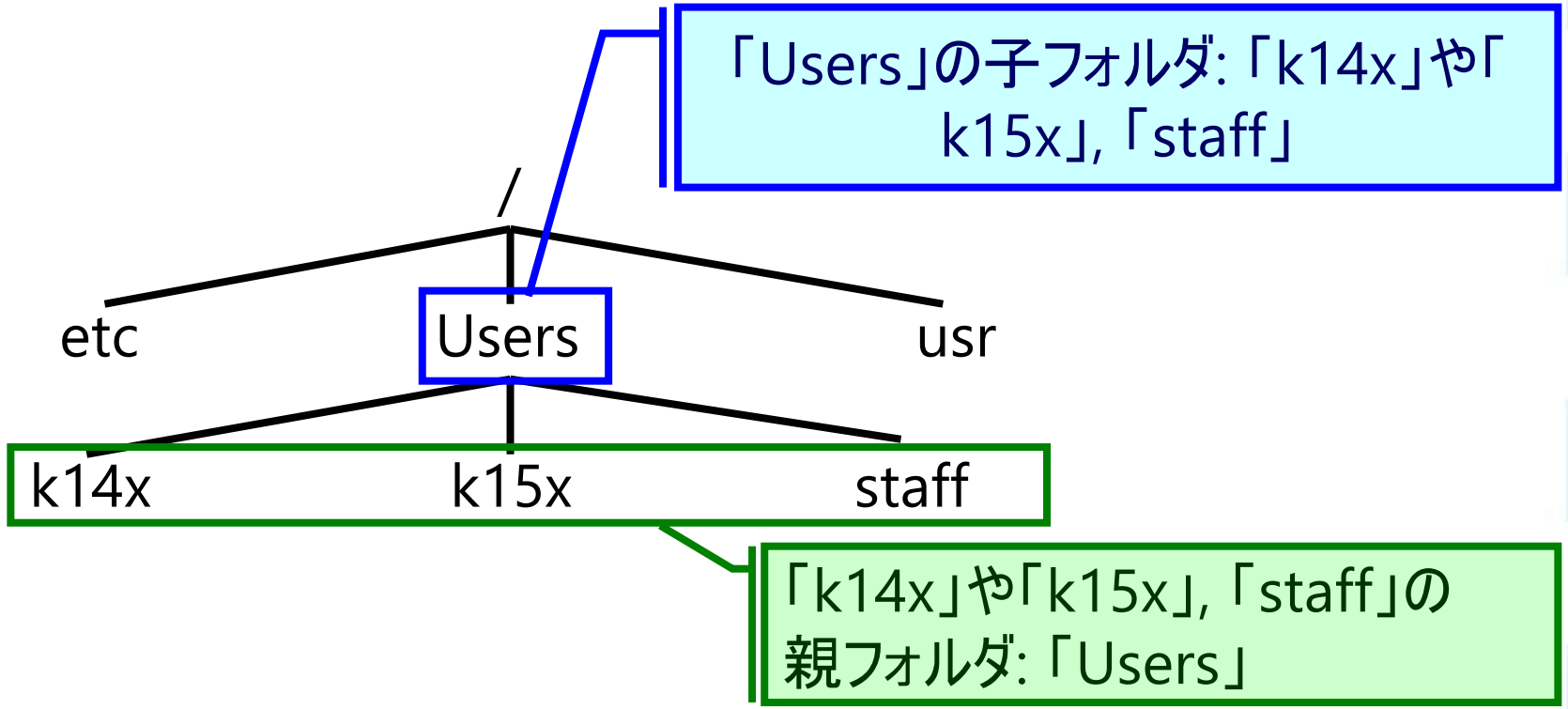
フォルダの階層構造

- どのファイルやフォルダが、どのフォルダの中に入っているか、ということを表した構造
- /: コンピュータ上で最も大きなフォルダ(ルートフォルダ)
  - 全てのフォルダはこのフォルダの中に入っている



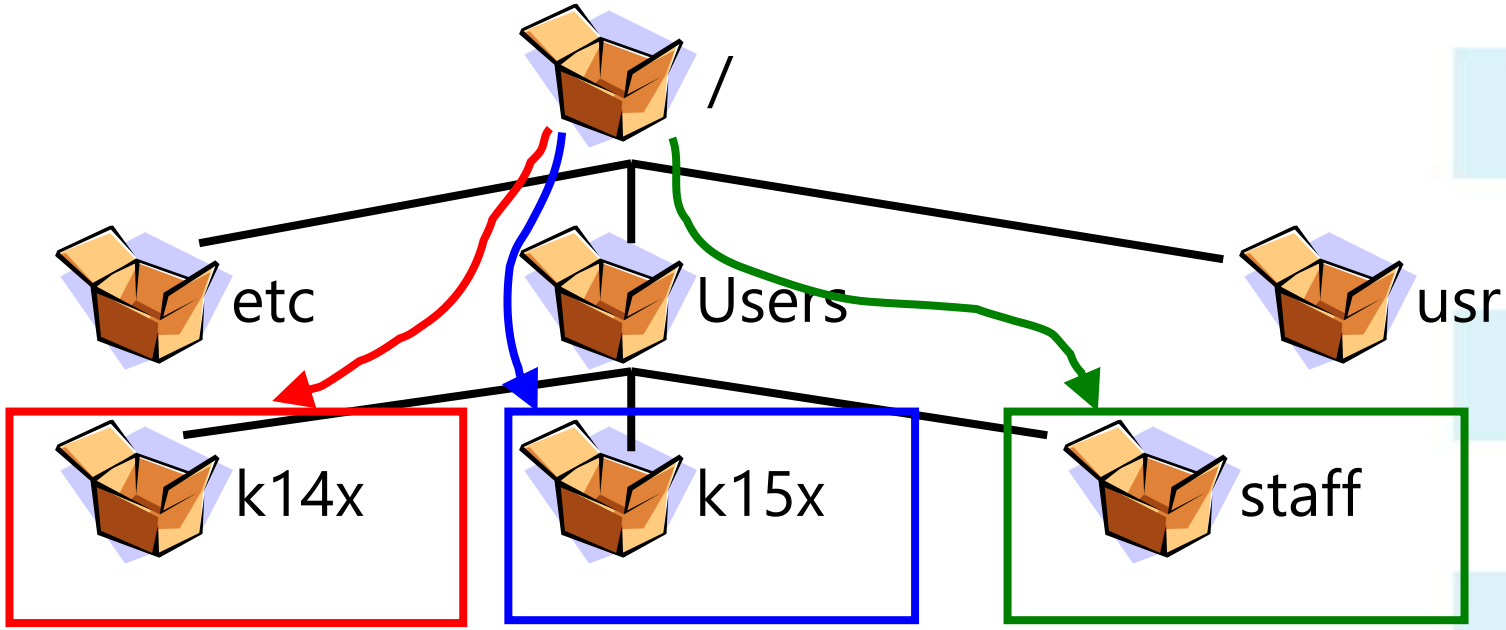
親フォルダ, 子フォルダ

- 親フォルダ: フォルダAの中にフォルダBが入っている場合、フォルダAをフォルダBの「親フォルダ」と呼ぶ
- 子フォルダ:フォルダAの中にフォルダBが入っている場合、フォルダBをフォルダAの「子フォルダ」と呼ぶ



パス

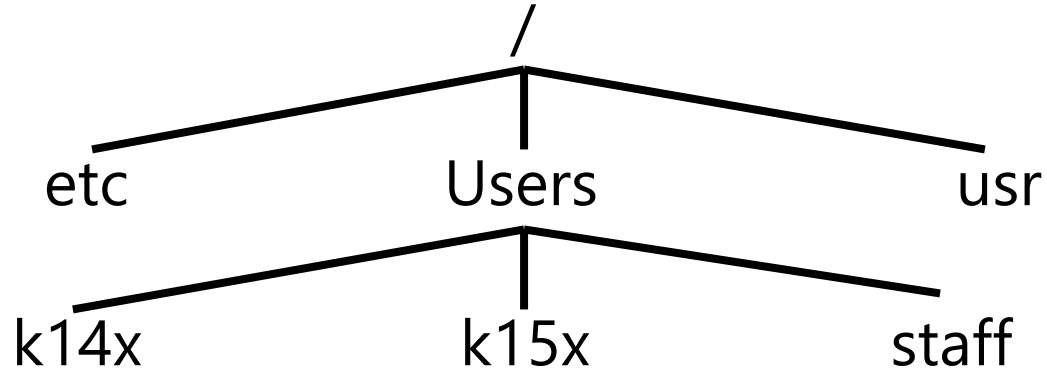
- あるファイルやフォルダが、どの位置にあるかを表すもの
  - どのようにフォルダをたどれば、目的のファイルやフォルダにたどり着くかを表すもの



k14xやk15x, staff:
ルートフォルダ(「/」)から見て、「Users」の中にある

絶対パス

- ファイルやフォルダの、ルートフォルダ(「/」)から見た位置
  - ルートフォルダから、目的のファイル・フォルダへのパス



- k14xやk15x, staff: 「Users」の中にある
- ルートフォルダ: 「/」で表す

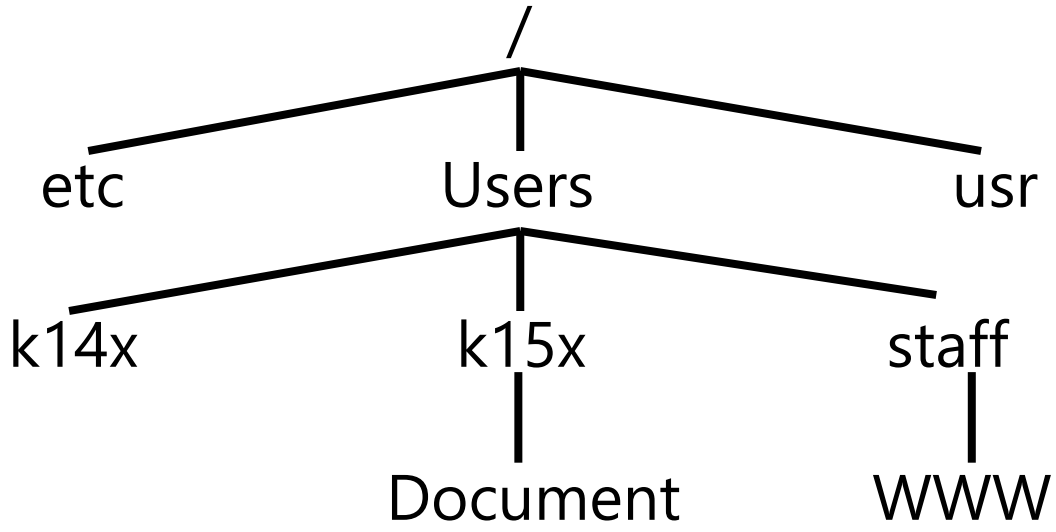
「/」で表す(Windowsでは「¥」)

「k14x」の絶対パス: /Users/k14x
「k15x」の絶対パス: /Users/k15x
「staff」の絶対パス: /Users/staff

相対パス[1]

- あるファイル・フォルダから見た、別のファイル・フォルダの位置
  - ルートフォルダ以外のフォルダから、目的のファイル・フォルダへのパス

カレントフォルダ



k14xからWWWを見たとき: k14xと同じフォルダの中の「staff」の中の「WWW」

WWWからDocumentを見たとき:
WWWの1つ上の「staff」の1つ上の中の「k15x」の中の「Document」

相対パス[2]

k14xからWWWを見たとき(k14x→WWWへの経路):
カレントフォルダ → k14xの1つ上のフォルダ → staff → WWW

カレントフォルダ:
「.」で表す

WWWからDocumentを見たとき (WWW→Documentへの経路):
カレントフォルダ → WWWの1つ上のフォルダ → staffの1つ上のフォルダ → k15x → Document

1つ上のフォルダ:
「..」で表す

k14xからWWWを見た相対パス: ../staff/WWW
WWWからDocumentを見た相対パス: ../../k15x/Documents

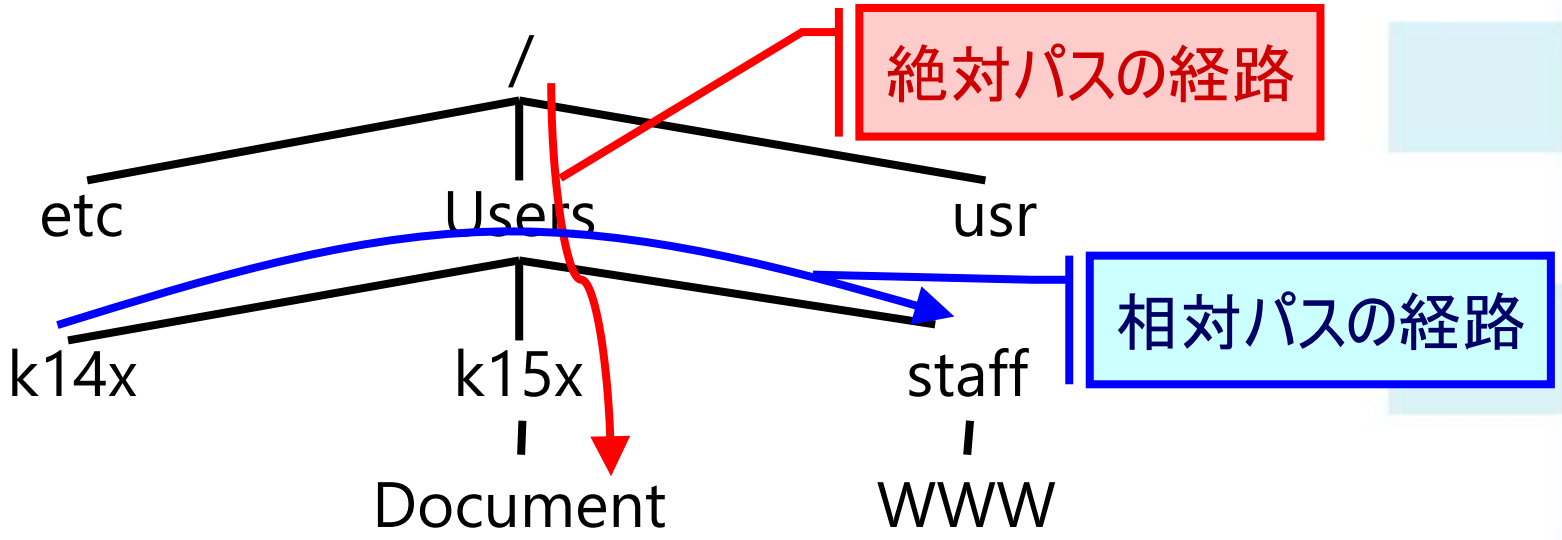
※カレントフォルダ「./」は省略可能

パス(まとめ)

- あるフォルダAにフォルダBが入っている:「A/B」と表す
  - 「/」は「¥」とも表す
  - フォルダのことを「ディレクトリ」とも呼ぶ
- 絶対パス
  - ルートフォルダは「/」と表す
- 相対パス
  - 1つ上のフォルダ:「..」で表す
    - 「カレントフォルダの1つ上」に限らず、「1つ上のフォルダ」は必ず「..」と表す
  - カレントフォルダ:「.」で表す

パスの考え方[1]

1. 出発点から、線をたどって目的地までの経路を書く
   - 出発点(絶対パス): ルートフォルダ
   - 出発点(相対パス): カレントフォルダ



Ex1 - Documentまでの絶対パス: / → Users → k15x → Document

Ex2 - k14xからWWWまでの相対パス: k14x → Users → staff → WWW

パスの考え方[2]

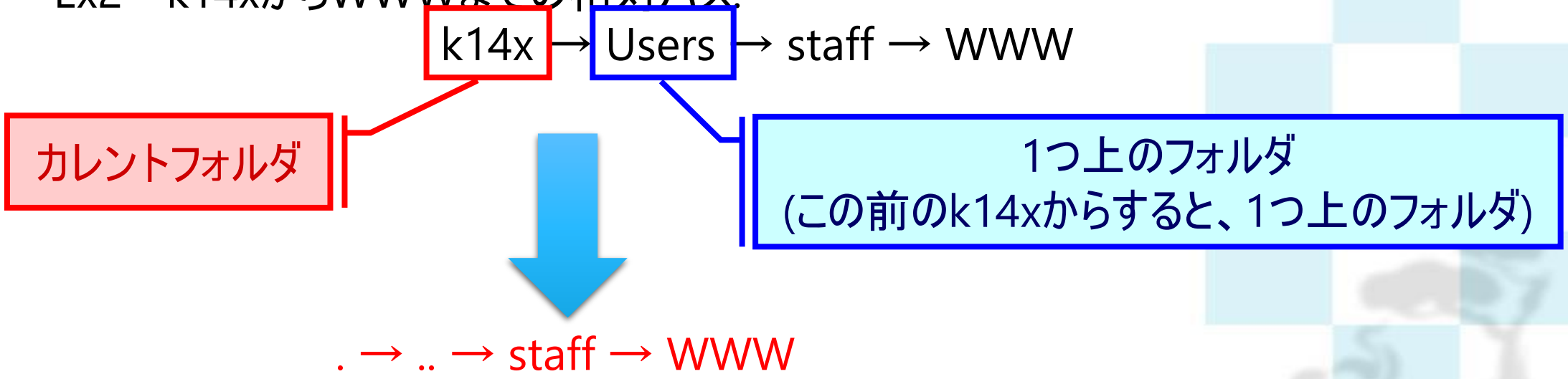
2. カレントフォルダと1つ上のフォルダの名前を置き換え

- カレントフォルダ(相対パスでの出発点): ./
- 1つ上のフォルダ: ..

Ex1 - Documentまでの絶対パス:

/ → Users → k15x → Document

Ex2 - k14xからWWWまでの相対パス:



パスの考え方[3]

3. 「→」を「/」または「¥」に置き換え
   - パスの中に1つ上のフォルダ「..」が含まれる場合は、カレントフォルダも省略
   - 絶対パスの先頭は「//」とは書かずに「/」

Ex1 - Documentまでの絶対パス: / → Users → k15x → Document

➡ /Users/k15x/Document

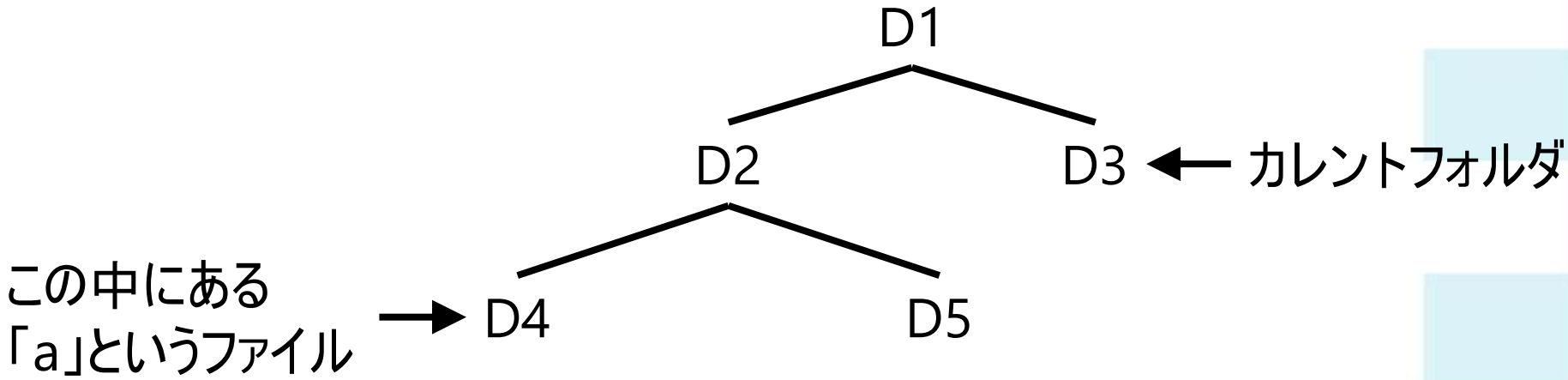
Ex2 - k14xからWWWまでの相対パス: . → .. → staff → WWW

➡ ../staff/WWW

やってみよう![1]

1. D3をカレントフォルダとして、D4の中にあるファイルaを指定する相対パスを考えること

※2009年度ITパスポート 春期試験より



2. 1.と同じファイルの階層構造で、D5までの絶対パスを考えること

やってみよう![2]

- ファイルシステムに関する次の記述中のa～cに入れる字句を考えること

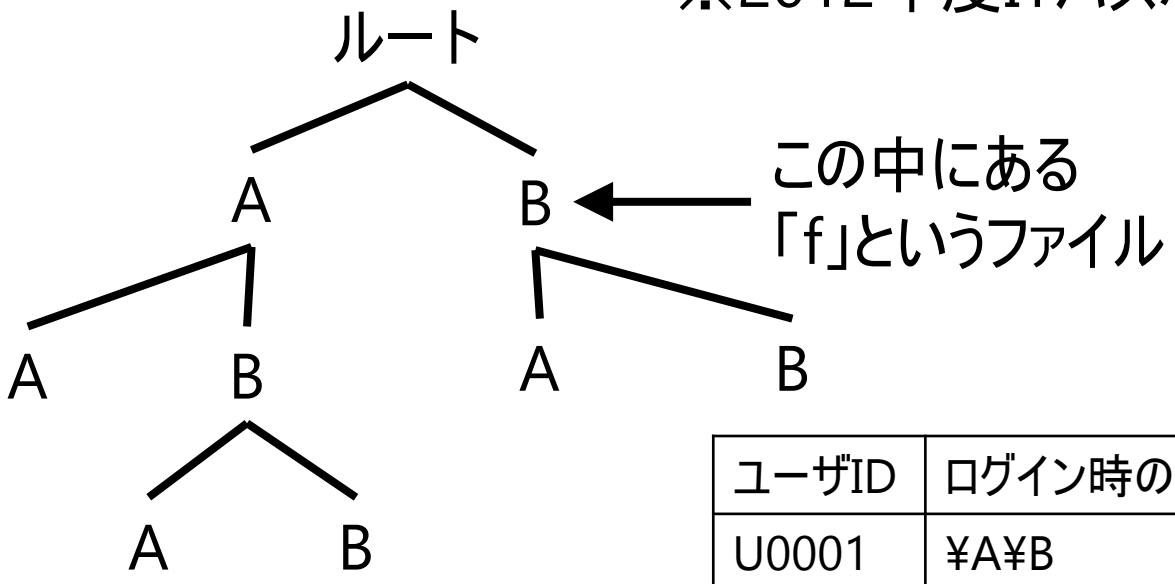
※2011年度ITパスポート 春期試験より

PCでファイルやディレクトリを階層的に管理するとき、最上位の階層に当たるディレクトリを［　a　］ディレクトリ、現時点で利用者が操作を行っているディレクトリを［　b　］ディレクトリという。［　b　］ディレクトリを基点としてファイルやディレクトリの所在場所を表す表記を［　c　］パスという。

やってみよう![3]

- 図に示す階層構造で、複数個の同名ディレクトリA、Bが配置されており、ユーザIDごとにログインしたときのカレントディレクトリが異なる。U0002がログインした直後に矢印が示すディレクトリBに存在するファイルfを指定するパスを考えること
  - ディレクトリ間、ディレクトリとファイル間の区切りは「¥」で表す

※2012年度ITパスポート 秋期試験より



ユーザID	ログイン時のカレントディレクトリ
U0001	¥A¥B
U0002	¥A¥B¥A

Question!

プログラミングとプログラミング言語

プログラミング言語の変遷[1](p. 76)

- プログラム: コンピュータに命令を伝えるための文書
- プログラミング言語: プログラムを記述するための言葉

初期: 機械語でプログラムを記述

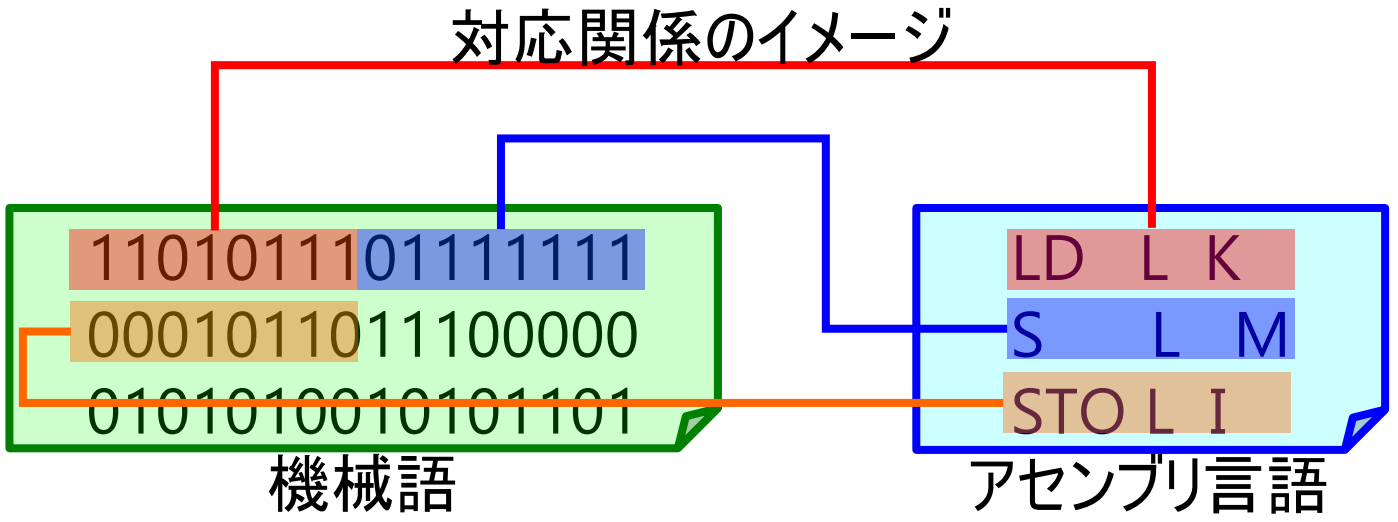
- 機械語: 0と1の2進数の形式の言葉

→ 不便!

→ アセンブリ言語が登場

アセンブリ言語

- 英語に似せた言語
- 機械語と1対1で対応
- アセンブリ言語のプログラムを機械語に翻訳
  - 翻訳ソフトウェア: アセンブラ



プログラミング言語の変遷[2](p. 76)

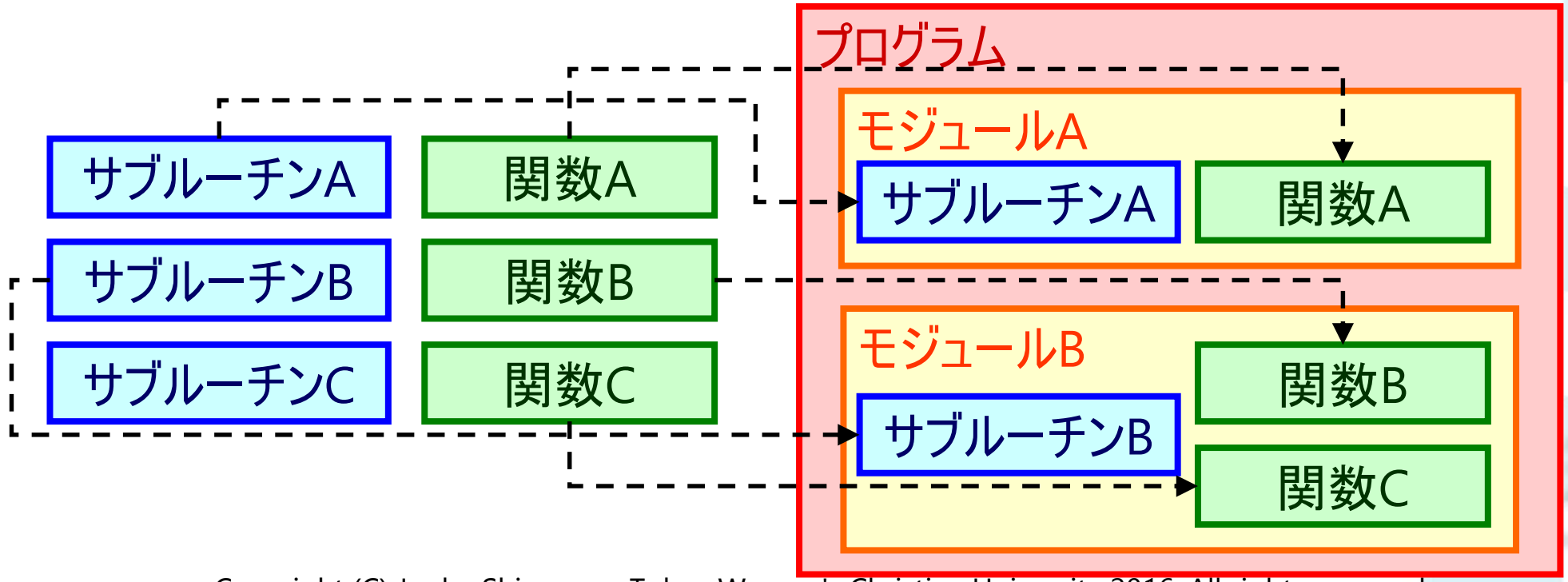
- 1954年にFORTRAN(FORmula TRANSlator)
  - IBM社が科学技術用言語として提唱
- 1959年にCOBOL(Common Business Oriented Language)
  - アメリカ国防省が商用言語として提唱
- 1962年にBASIC(Beginner's All purpose Symbolic Instruction Code)
  - ダートマス大学で初心者でも使える言語として提唱
  - ビル・ゲイツがよく利用し、Microsoft社が開発に注力

プログラミング言語の変遷[3](p. 76)

- 1972年にC言語
  - ベル研究所がOSなどの基幹ソフトウェアの開発用言語として開発
  - UNISがOSとして初めてC言語で記述
- 1972年にSmalltalk
  - ゼロックス社のパロアルト研究所でオブジェクト指向言語として開発
- 1995年にJava
  - サン・マイクロシステムズ社(現オラクル社)でネットワーク対応言語として開発

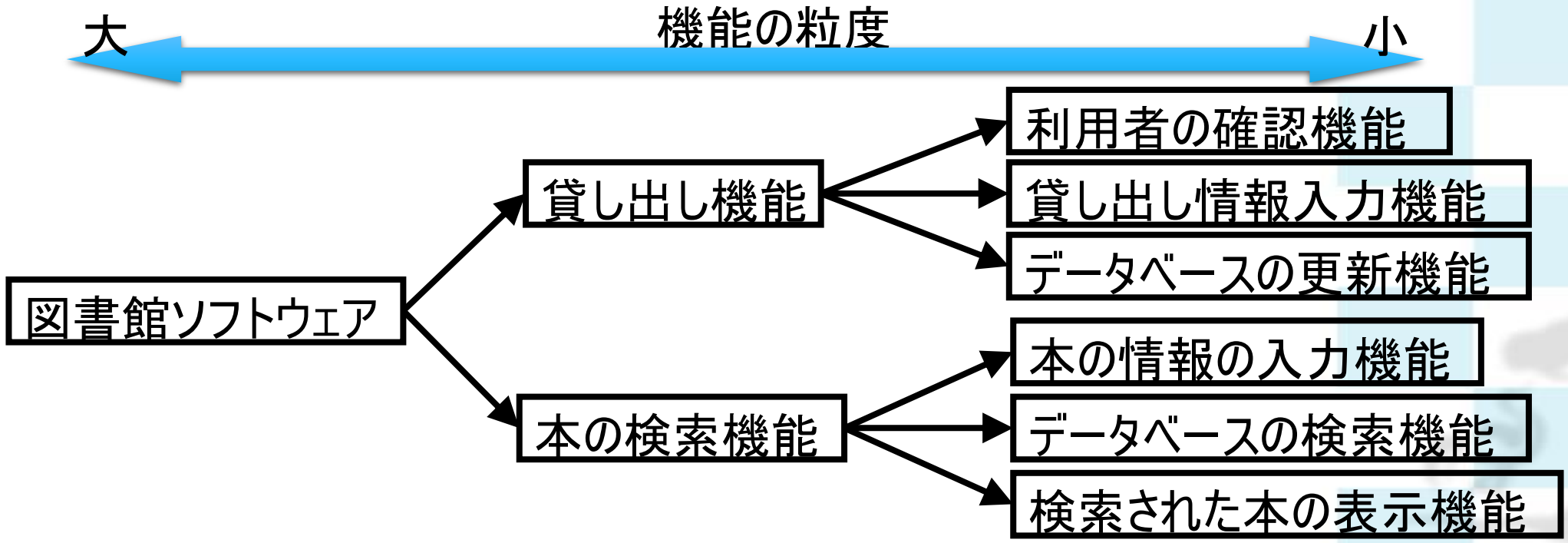
プログラミングの方法[1](p. 77)

- 手続き型プログラミング
  - プログラムを処理の単位ごとに分割(モジュール)
  - サブルーチンと関数でモジュールを構成
    - サブルーチン: あらかじめ用意された、よく使う処理のまとまり
    - 関数: 自分で定義する、よくつかう処理のまとまり



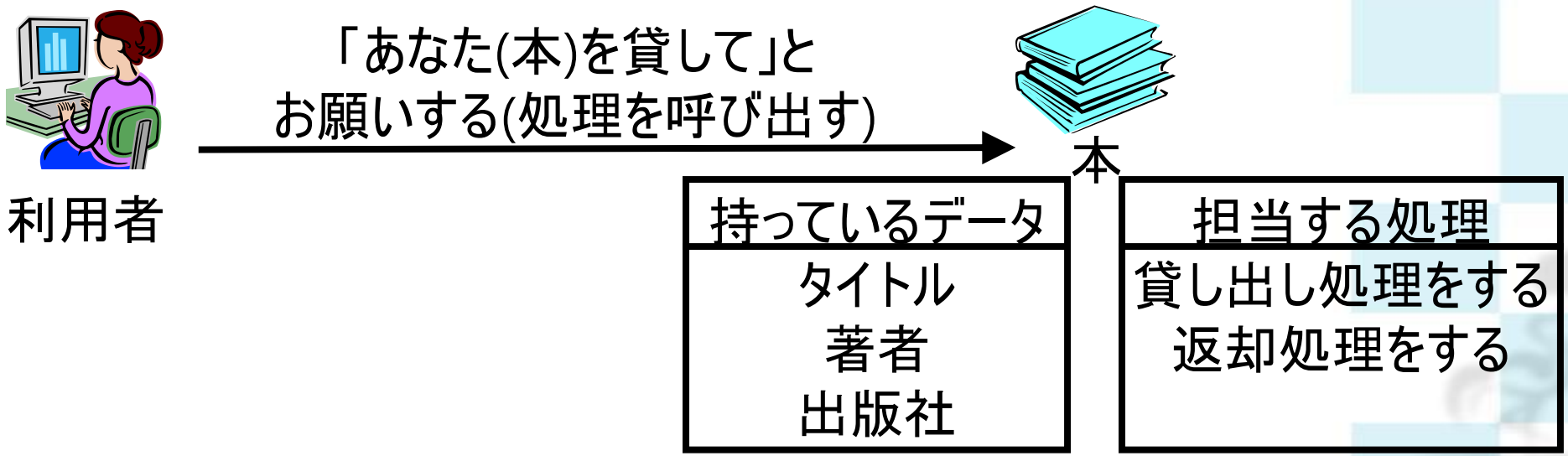
プログラミングの方法[2](p. 77)

- 構造化プログラミング
  - プログラムを小さな機能に分解し、それを組み合わせると完成するという考え方
  - 連続(順番に処理する)・判断(状況によって異なる処理をする)・反復(同じ処理を繰り返す)の組み合わせ



プログラミングの方法[3](p. 77)

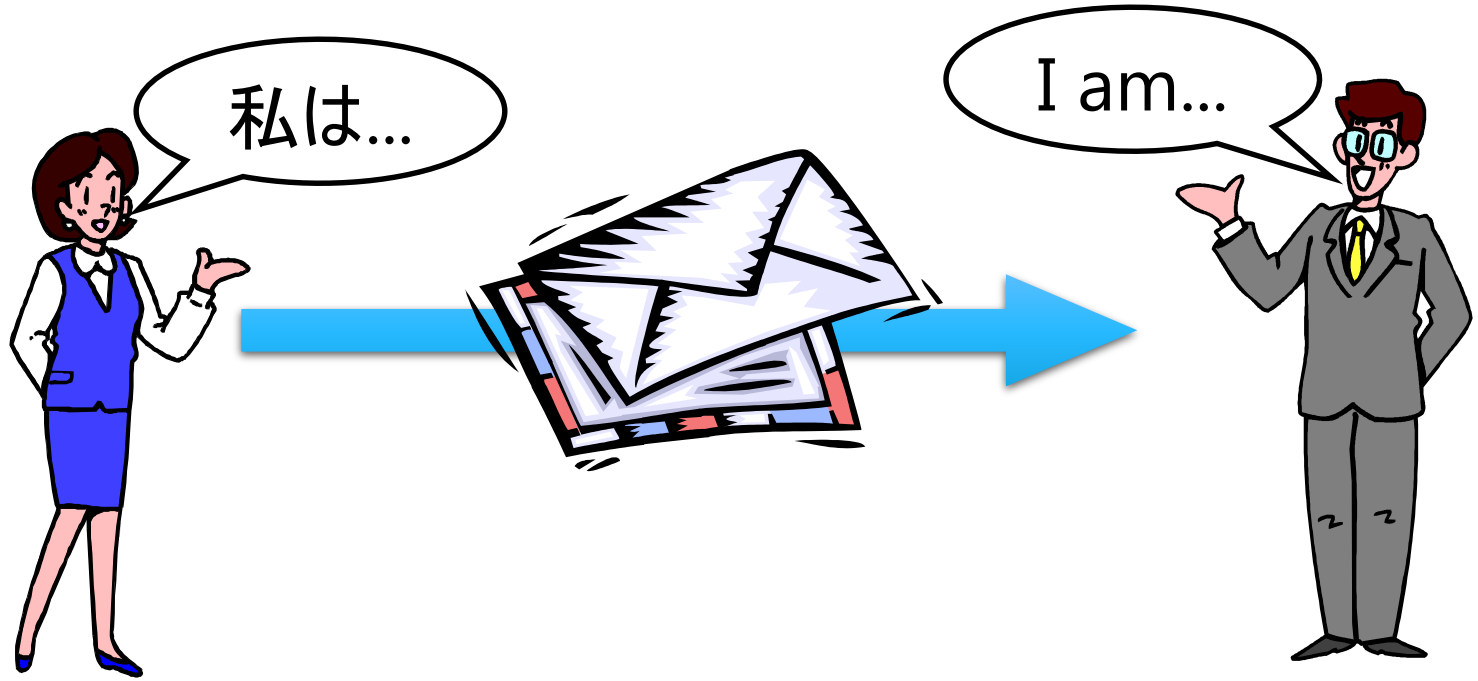
- オブジェクト指向プログラミング
  - 構造化プログラミングの欠点を克服することを目指して考案
  - 現実世界の「もの(オブジェクト)」に着目したプログラムの作成方法
    - 図書館システムであれば「本」や「利用者」など
  - 処理は、他のオブジェクトから依頼されて、オブジェクト自身が実行する、という考え方



プログラミング言語

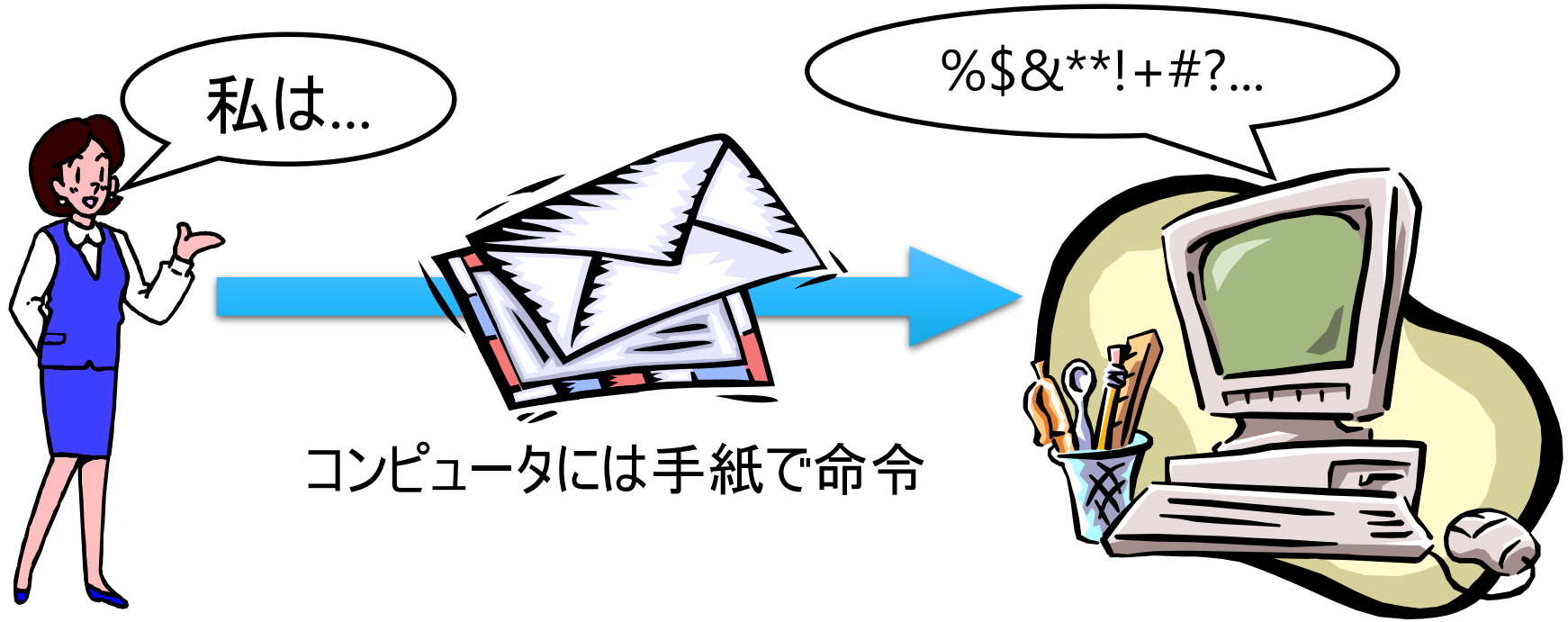
外国人に手紙を書く場合どうする??

- 相手がわかる言葉で手紙を書く
  - 相手が理解できる言葉を覚えるのは大変!!
- コンピュータには、手紙(命令書)で命令
  - コンピュータが理解できる言葉で手紙(命令書)を書く



外国人に手紙を書く場合どうする??

- 相手がわかる言葉で手紙を書く
  - 相手が理解できる言葉を覚えるのは大変!!
- コンピュータには、手紙(命令書)で命令
  - コンピュータが理解できる言葉で手紙(命令書)を書く

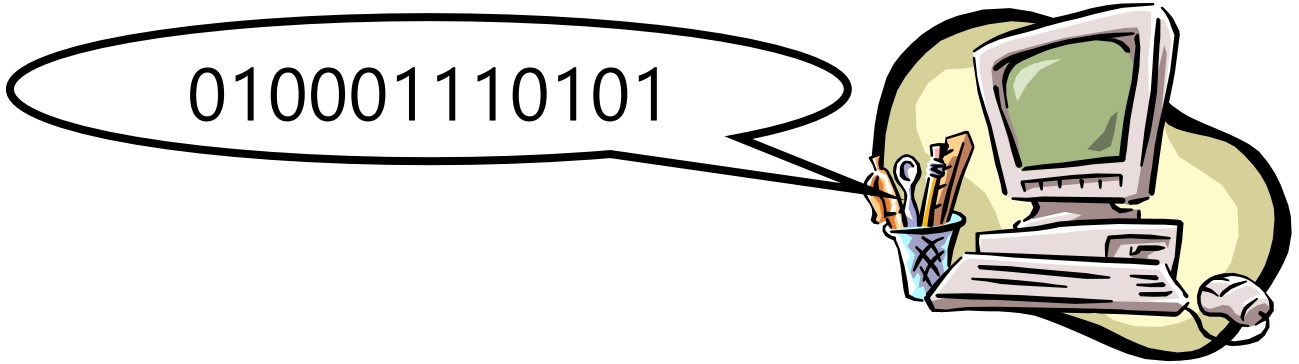


コンピュータが理解できる言葉は?

- コンピュータが理解できる言葉: 機械語
  - コンピュータは、2進数しか理解できない

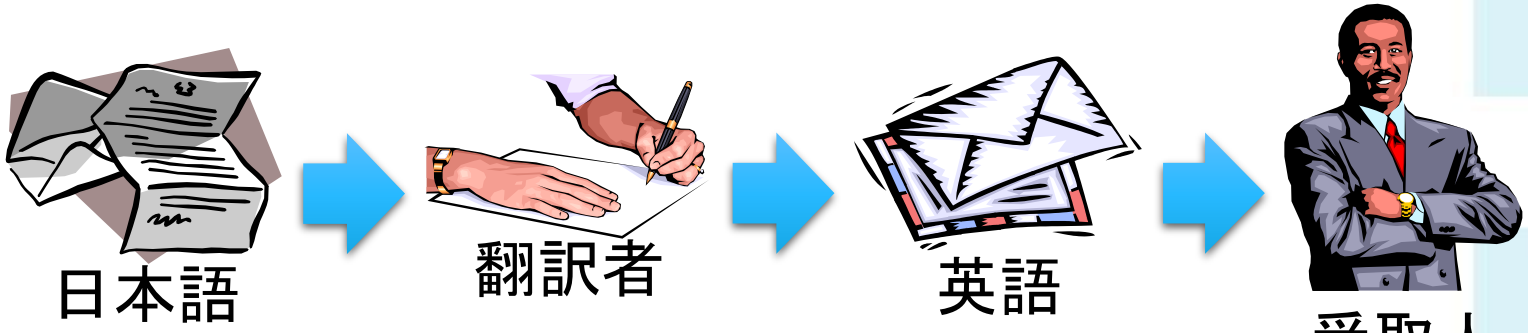
- 人間が理解するのは難しい
- 命令書を人間が理解できる言葉で書き、それを訳したものをコンピュータに渡す

プログラミング言語

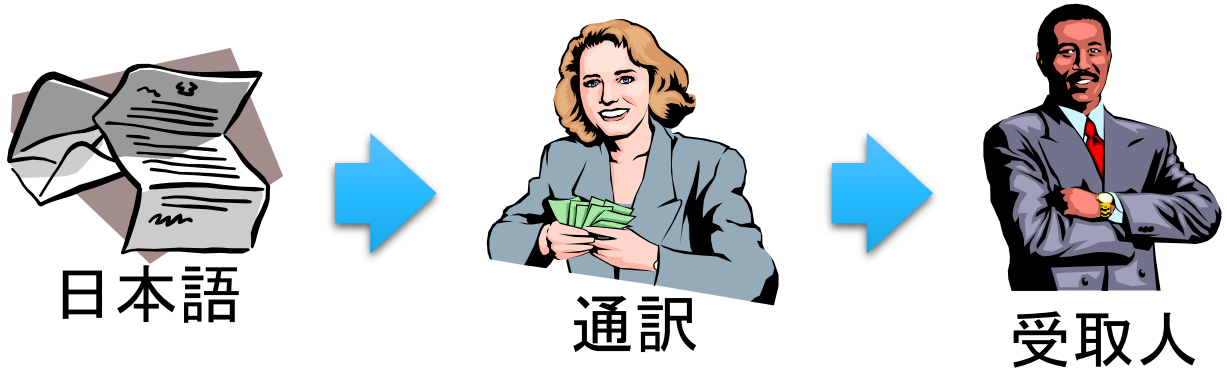


手紙を訳すには?

- 手紙を翻訳する



- 手紙を通訳する



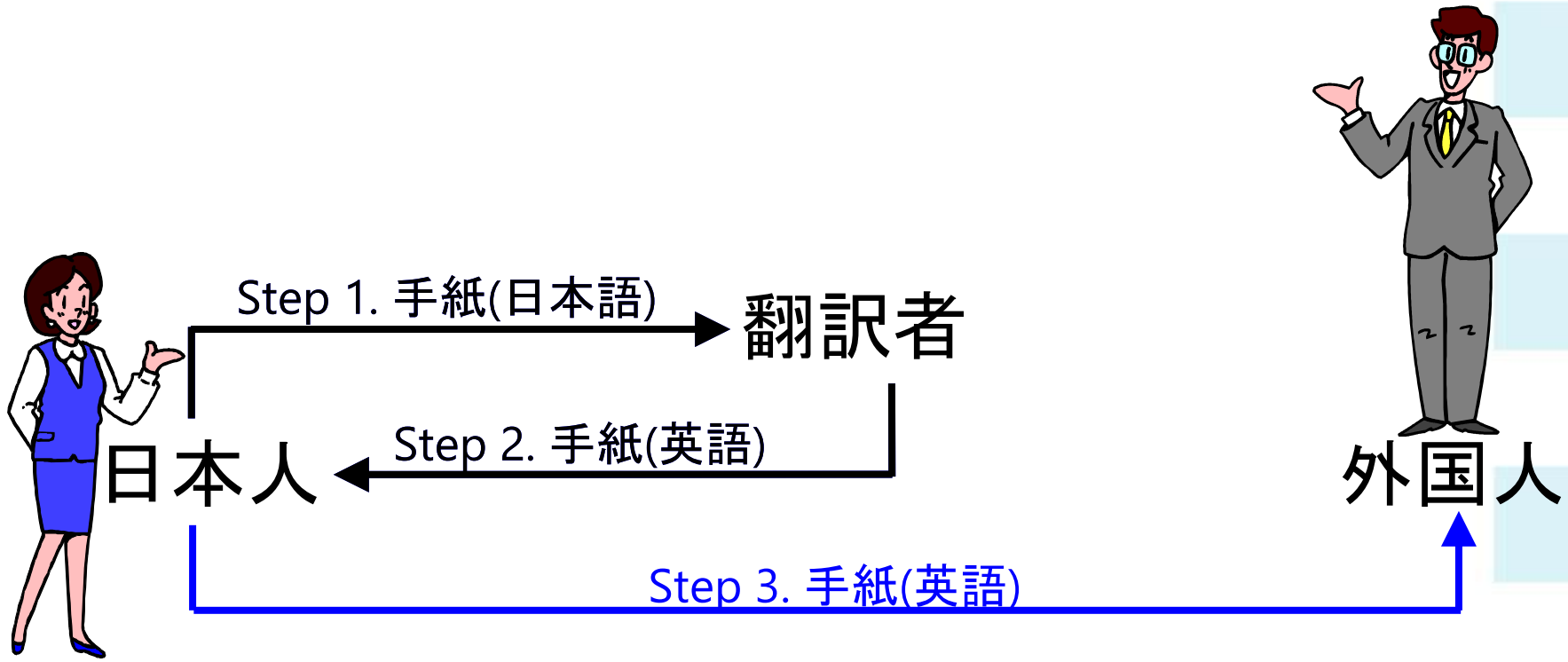
プログラムも、同じように機械語に訳す

変換方式

- プログラミング言語で書かれた命令書: 機械語に変換しなければ、コンピュータは実行不可能
- 変換方式は大きく分けて2種類
  - コンパイラ型:　翻訳
  - インタプリタ型: 通訳

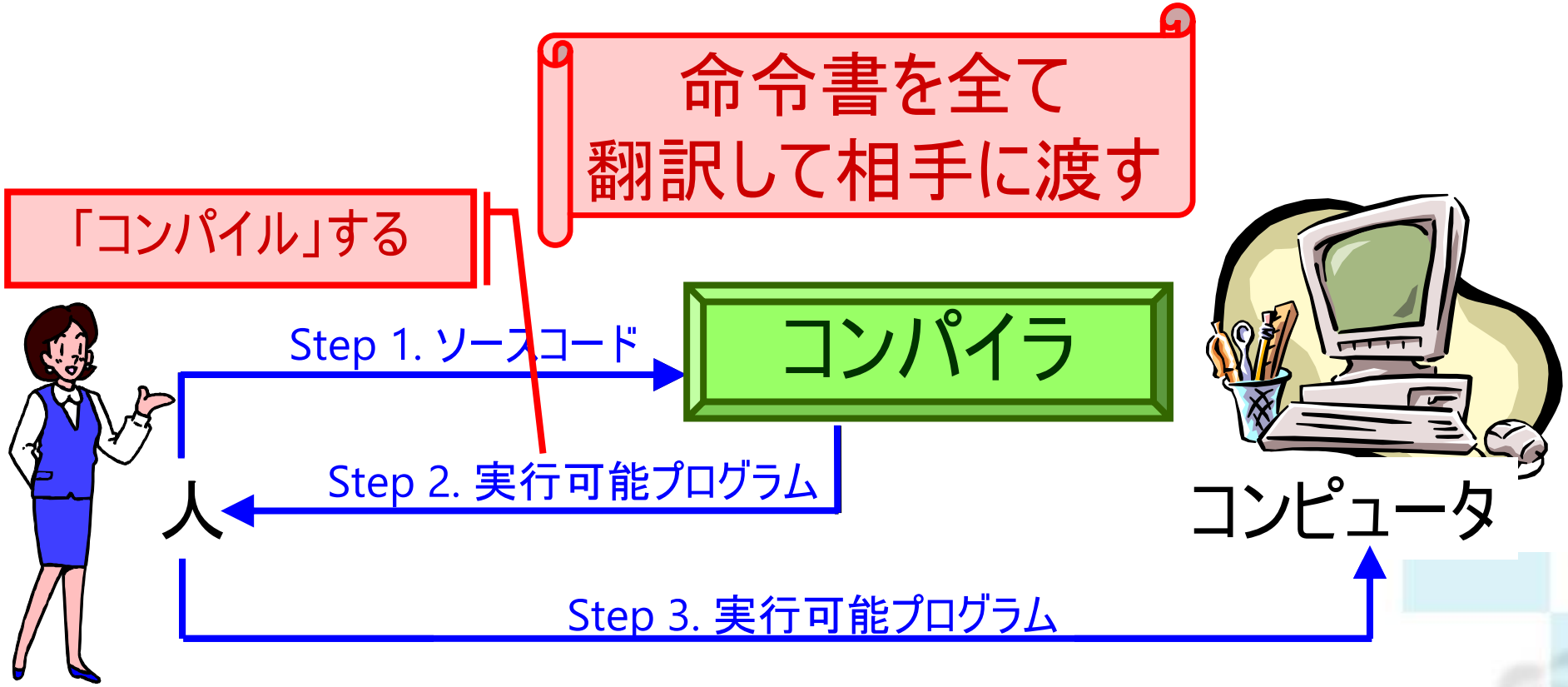
コンパイラ[概要](p. 78)

- コンパイラ: 命令書を機械語に翻訳し、コンピュータで実行可能にするためのソフトウェア

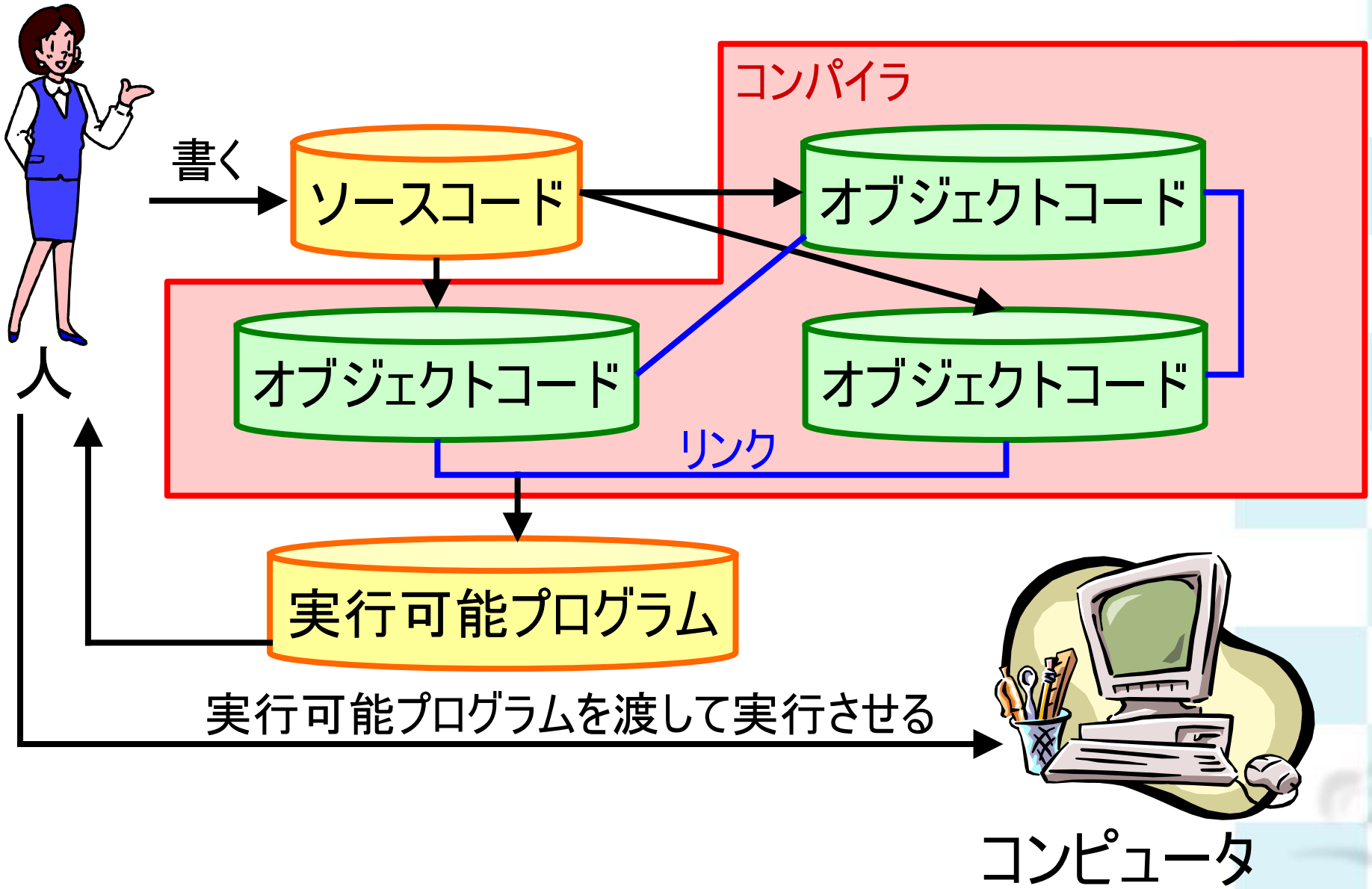


コンパイラ[概要](p. 78)

- コンパイラ: 命令書を機械語に翻訳し、コンピュータで実行可能にするためのソフトウェア



コンパイラ[詳しく](p. 78)

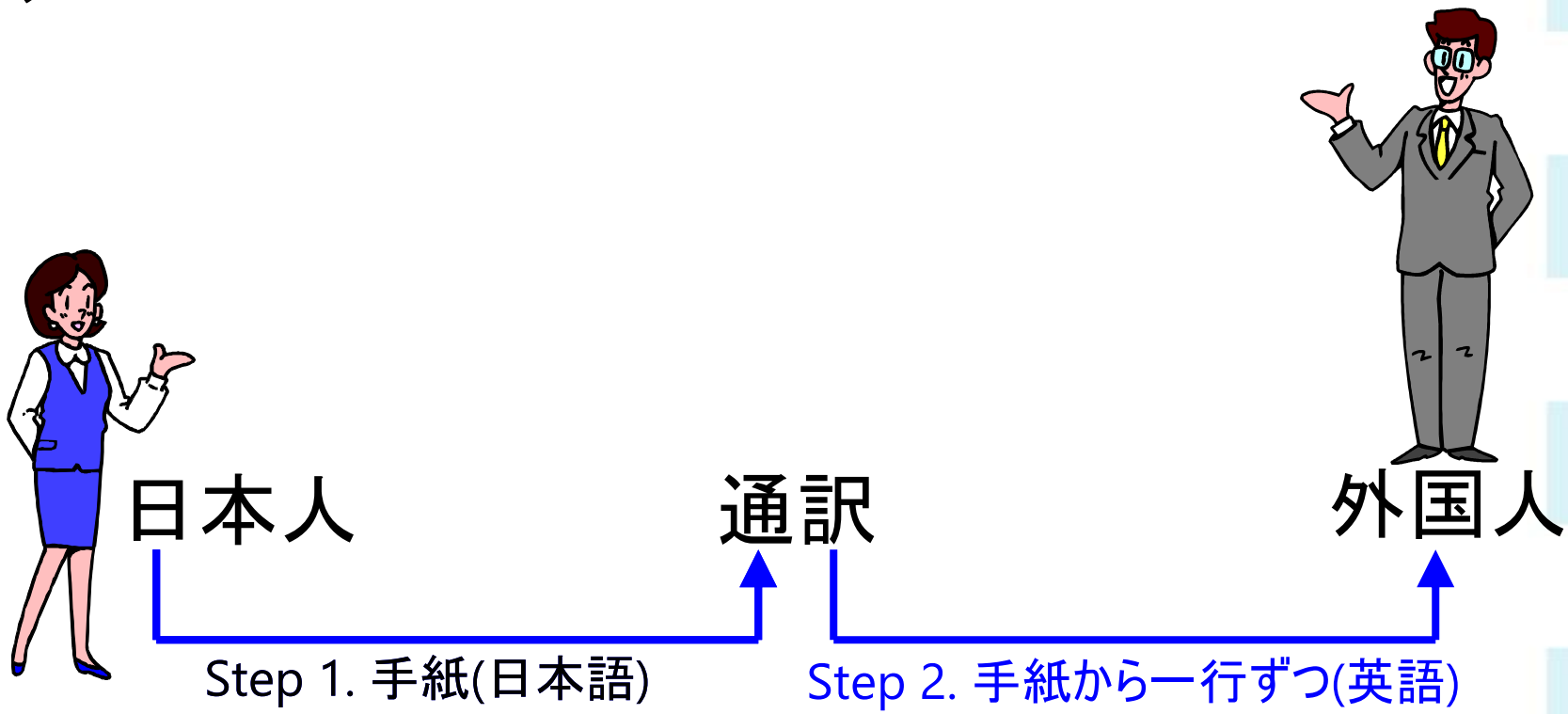


用語[1](p. 78)

- ソースコード: プログラミング言語で記述した命令書
- オブジェクトコード: ソースコードを翻訳したもの
- 実行可能プログラム: オブジェクトコードを連携させて、動作可能な形にしたもの
- プログラム: ソースコードと実行可能プログラムの双方の意味
  - どちらの意味で使われるかは、その時々で異なる

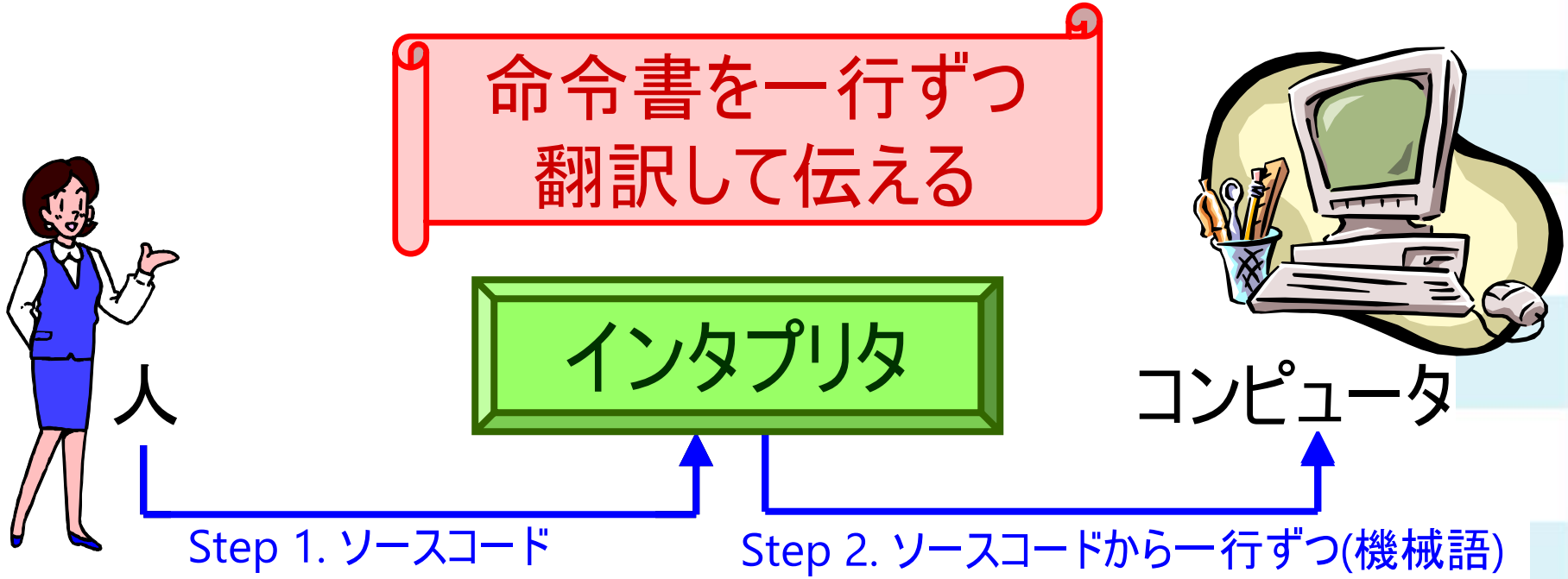
インタプリタ

- インタプリタ: 命令書を最初から1行ずつ読んで機械語に通訳するためのソフトウェア



インタプリタ

- インタプリタ: 命令書を最初から1行ずつ読んで機械語に通訳するためのソフトウェア



Question!

アプリケーション

アプリケーションの種類(p. 79)

- アプリケーション: 人間が直接操作して、様々な処理をするためのソフトウェア
  - OSはソフトウェアではあるが、アプリケーションではない
- 種類
  - 専用アプリケーション
  - オフィススイート
  - 画像・音楽・映像用アプリケーション
  - コミュニケーションツール
  - セキュリティ対策アプリケーション
  - eラーニング用アプリケーション
  - 組込みソフトウェア

専用アプリケーション(p. 79)

- 企業や組織、大学などの業務に特化したもの
  - 自分のところで独自に開発や、開発会社に開発を依頼
  - 業務の形態が変わると、開発のしなおし
    - 銀行の統合、大学の入試システムの変更、etc.
  - ものによっては、汎用の機能を持つものをカスタマイズ
    - Ex. 大学の履修登録システムや図書館システムなど
- 例
  - 銀行のオンラインシステム
  - POS(Point Of Sales)システム
    - コンビニやスーパーなどの販売管理システム
  - 大学の業務システム
    - 東女のCampus Squareなど

オフィススイート[1](p. 80)

- 事務作業でよく使われるアプリケーションをまとめたもの
  - ワープロソフト
    - 文書を作成するためのアプリケーション
    - 以前はワープロ専用機を利用、現在はコンピュータ上のアプリケーションを利用
  - 表計算ソフト
    - データの集計や計算をするためのアプリケーション
    - グラフ作成なども可能
  - データベースソフト
    - 業務に利用するデータをため込んで管理し、容易に集計・検索・抽出するためのアプリケーション
  - プレゼンテーションソフト
    - アイデアや調査内容などの発表をするためのアプリケーション

オフィススイート[2](p. 80)

- 例
  - Microsoft Office
  - OpenOffice.org
  - etc.

画像・音楽・映像用[1](p. 81)

- グラフィックスソフト
  - 写真やイラストなどの画像の作成・加工のためのアプリケーション
  - ペイント系ソフト
    - ビットマップ画像を処理するためのアプリケーション
    - ビットマップ画像: 1つ1つの点が何色かで内容を表現する画像
      - 画像の内容についての情報(○とか□とか)を持たないので、拡大・縮小すると、画質が劣化するが、写真などの規則性を持たない画像に向く
  - ドロー系ソフト
    - ベクトル画像を処理するためのアプリケーション
    - ベクトル画像: 内容を方程式の形で表現する画像
      - 画像の内容についての情報(○とか□とか)を持つので、拡大・縮小しても、画質は維持されるが、写真などの規則性を持たない画像には向かない

画像・音楽・映像用[2](p. 81)

- 3D画像の処理アプリケーション
  - 平面図・立面図・側面図を書く機能: モデリング
  - 2次元の画面に投影する機能: レンダリング
  - CPUやビデオカードに高い性能が必要
- キャプチャソフト
  - TVやビデオカメラなどの映像をコンピュータに取り込むアプリケーション
- 音声の処理アプリケーション
  - 音楽や声をマイクなどからコンピュータに取り込み
- CAD(Computer Aided Design)
  - 回路や建築などの設計図を作成するためのアプリケーション

コミュニケーションツール[1](p. 81)

- コンピュータを通じて他者とのコミュニケーションをするためのアプリケーション
  - メーラ(メールソフト)
    - 現在は、ブラウザ上で利用するものが多い
  - ブラウザ
    - Webページを管理しているコンピュータと通信し、必要なファイルを入手し、内容を解析して表示するアプリケーション
    - 1990年ごろにCERN(欧州原子核研究機構の前身)の研究者が研究情報のやりとりのためにHTMLを考案
      - HTML(Hyper Text Markup Language): Webページの内容を記述するための言語
    - この研究者が、W3C(World Wide Web Consortium)を設立
    - W3CがHTMLの文法など、WWWに関する国際規格を策定

コミュニケーションツール[2](p. 81)

- 現在は、様々なアプリケーションがブラウザ上で実現
  - BBS: 電子掲示板
  - ブログ(Weblog): WWW上での雑記帳
    - Webページの紹介や覚書、論評などを記録
  - RSS(Realty Simple Syndication): Webサイトの更新情報を簡単にまとめて配信するための文書フォーマット
    - RSSを読むためのアプリケーション: RSSリーダ
    - BBSやブログと連携

セキュリティ対策アプリケーション(p. 82)

- コンピュータをウィルスや不正アクセスの被害から守るためのアプリケーション
  - ウィルス: コンピュータを病気のような状態にするソフトウェア
  - 不正アクセス: コンピュータの利用権限を持たない人が勝手にコンピュータを利用すること
    - ソフトウェアの不具合や、利用権限を持つ人のパスワードの流出などにより行われる
  - 頻繁なアップデートが必要
    - ウィルスは毎日のように新種が出現
    - 不正アクセスは、様々な手口が考案

eラーニング用アプリケーション(p. 83)

- コンピュータでの学習を支援するためのアプリケーション
  - 大学での語学の授業
  - 企業での研修, etc.

- LMS(Learning Management System)
  - eラーニングのためのコンテンツの作成支援
  - 個人の学習履歴の記録や学習者間、教員と学生間のやりとり
  - 例: 東女のWebClass

組込みソフトウェア

- 家電製品などの特定のハードウェアに特化したアプリケーション
  - 小規模なものは、OSとアプリケーションが一体化
  - 大規模なものは、PCのOSなど、汎用のOS上で動作するものも
- 例
  - クーラー
  - 冷蔵庫
  - カーナビ, etc.

オープンソースソフトウェア

オープンソース(p. 84)

- オープンソース: ソースコードをインターネットで公開して、誰でも編集できるようにしたもの
  - 様々な人の知恵を集めて良いものを作る、という考えのもとにできた仕組み
    - オープンソースの場合、様々な人がソースコードを見るので、不具合の修正が早い(自分で修正することも可能)
    - オープンソースでない場合、ソフトウェアの作成者が不具合を修正するのを待つ
  - ソフトウェアの著作権は保護
  - 多くの場合、複数のプラットフォームに対応
    - プラットフォーム: OSやハードウェアなどのコンピュータの環境

オープンソースのOS(p. 84)

- Linux
  - 1991年に開発されたOSのカーネル
    - カーネル: OSの中でも核となる機能をあつめたもの
  - UNIXに似たOS(UNIX like OS)
  - インターネット上でのサービスを提供するためのコンピュータで多く利用
  - 様々な人が手を加えて、現在では様々な形のもの(ディストリビューション)が存在
    - 操作性や管理方法などがそれぞれ異なるものが存在
    - この授業のOSインストール実習でも利用(Vine Linux)

オープンソースのアプリケーション(p. 85)

- ブラウザ上で実現されているもの
  - SNS(Social Networking Service)
    - Mixiのようなコミュニケーションツール
    - オープンソースソフトウェアを組み合わせて開発
  - Wiki
    - ブラウザを利用して文書を書き込めるシステムで、意見交換やデータの共有などに利用
    - Wikipediaが代表的な例
  - etc.
- ブラウザ上以外で実現されているもの
  - gimp(グラフィックスソフト), OpenOffice.org, Mozilla Firefox(ブラウザ), etc.

Question!