

# コンピュータ・サイエンス2

## 第6回 OSインストール実習

人間科学科コミュニケーション専攻  
白銀 純子

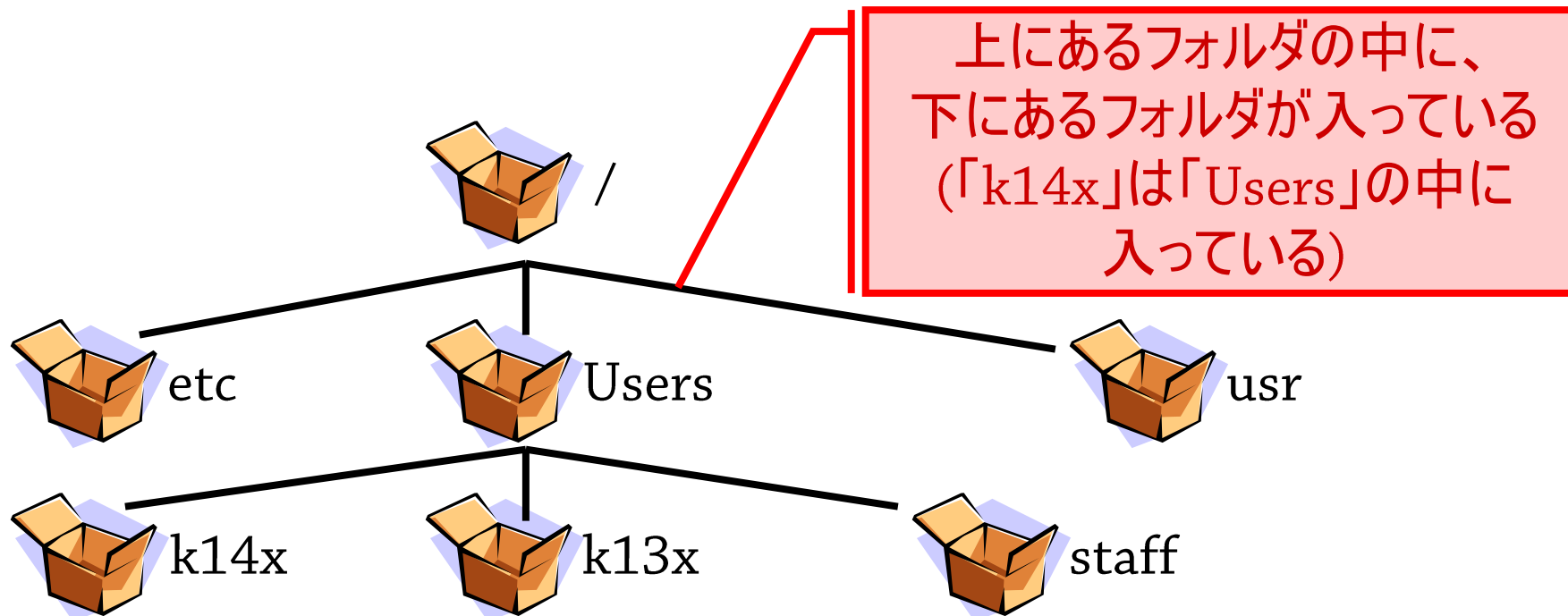
# 今回の内容

- OSのインストール実習

# 前回の復習

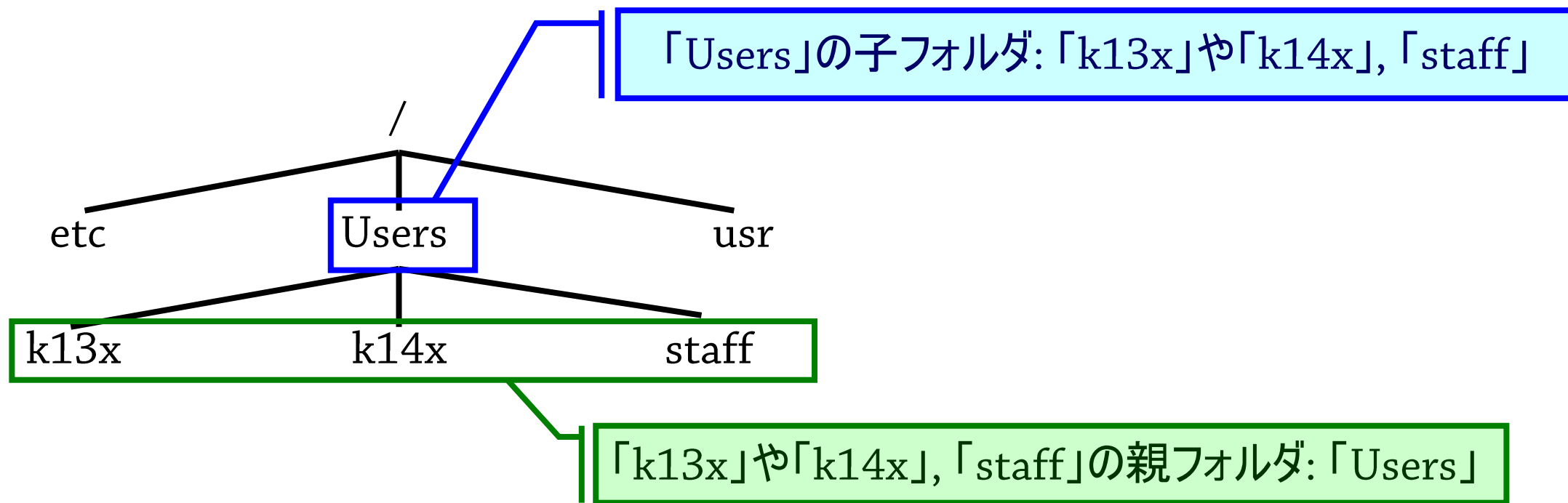
# フォルダの階層構造

- 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



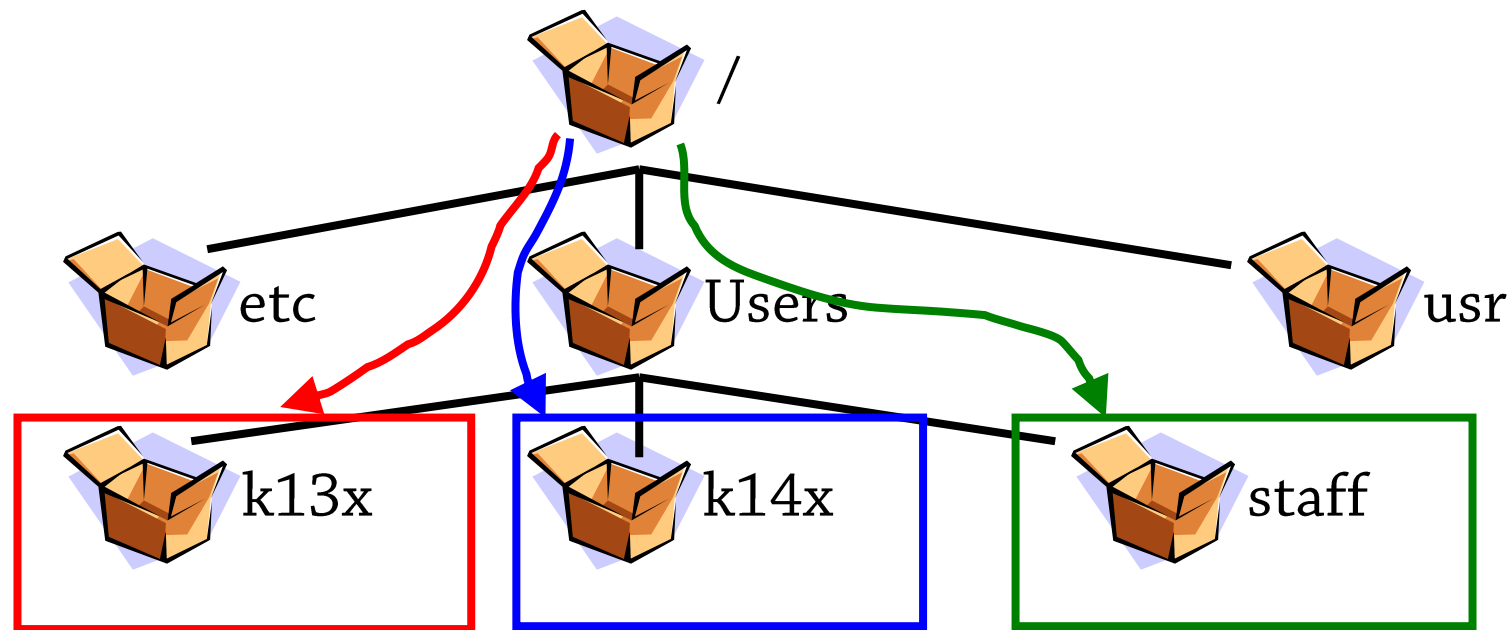
# 親フォルダ, 子フォルダ

- 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

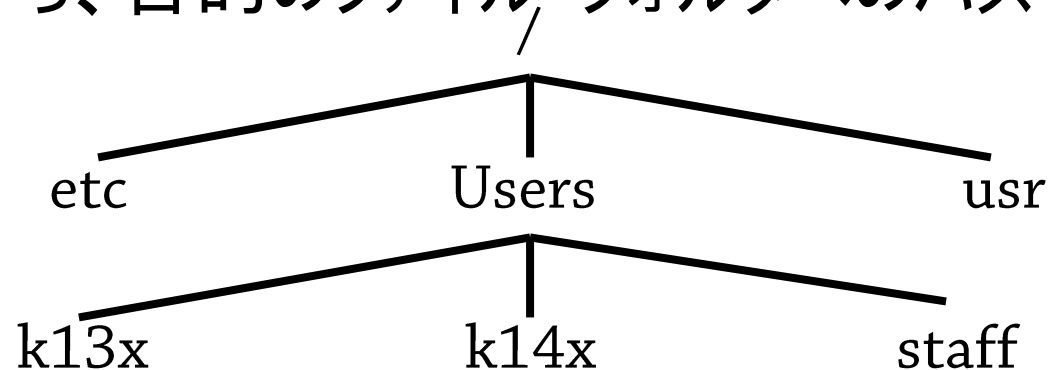
- あるファイルやフォルダが、どの位置にあるかを表すもの
  - どのようにフォルダをたどれば、目的のファイルやフォルダにたどり着くかを表すもの



k13xやk14x, 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



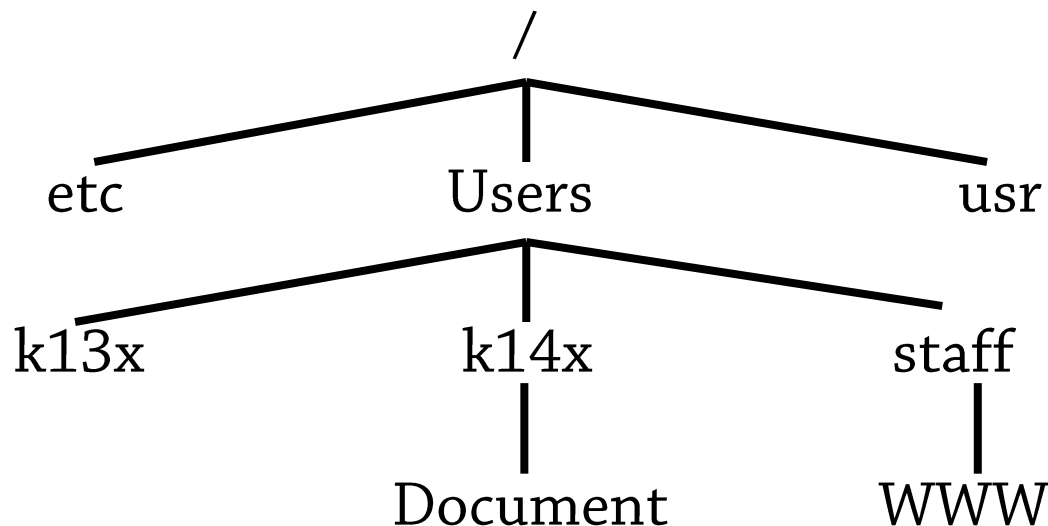
- k13xやk14x, staff: 「Users」の中にある
  - 「/」で表す(Windowsでは「¥」)
- ルートフォルダ: 「/」で表す



「k13x」の絶対パス: /Users/k13x  
「k14x」の絶対パス: /Users/k14x  
「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



k13xからWWWを見たとき: k13x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「k14x」の中の「Document」



# 相対パス[2]

k13xからWWWを見たとき(k13x→WWWへの経路):

カレントフォルダ → k13x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 → k14x → Document

1つ上のフォルダ:「..  
」で表す



k13xからWWWを見た相対パス: ../staff/WWW

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

※カレントフォルダ「./」は省略可能

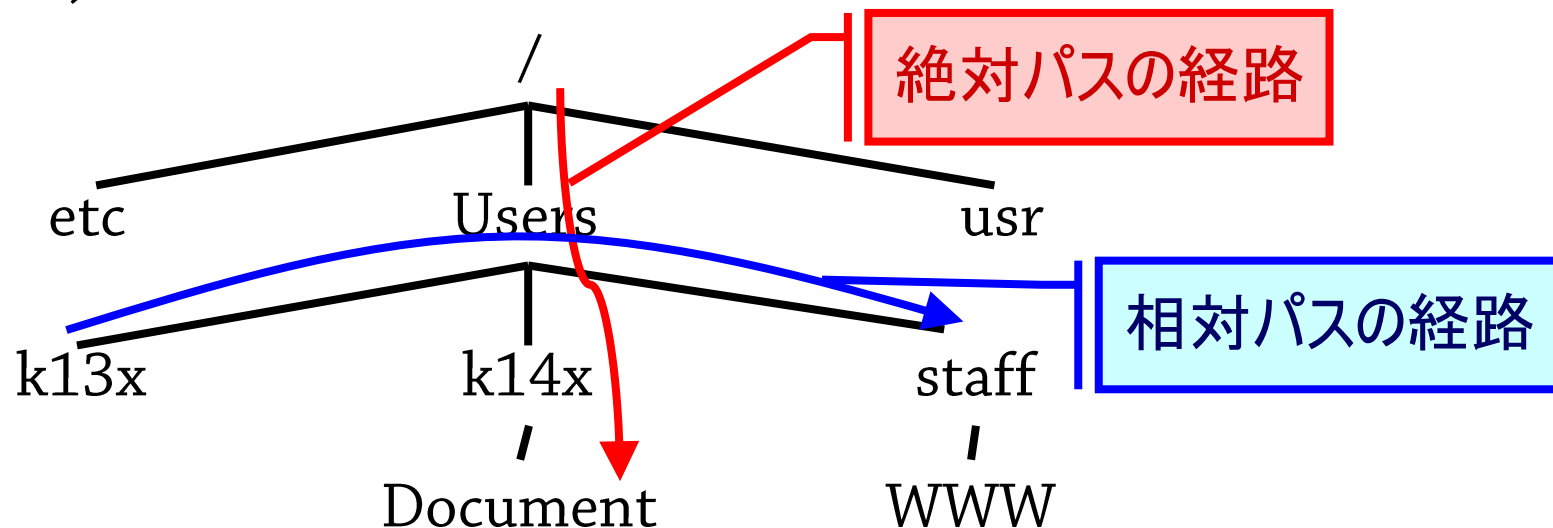
# パス(まとめ)

- 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:  $/ \rightarrow \text{Users} \rightarrow \text{k14x} \rightarrow \text{Document}$

EX2 - k13xからWWWまでの相対パス:  $\text{k13x} \rightarrow \text{Users} \rightarrow \text{staff} \rightarrow \text{WWW}$

# 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

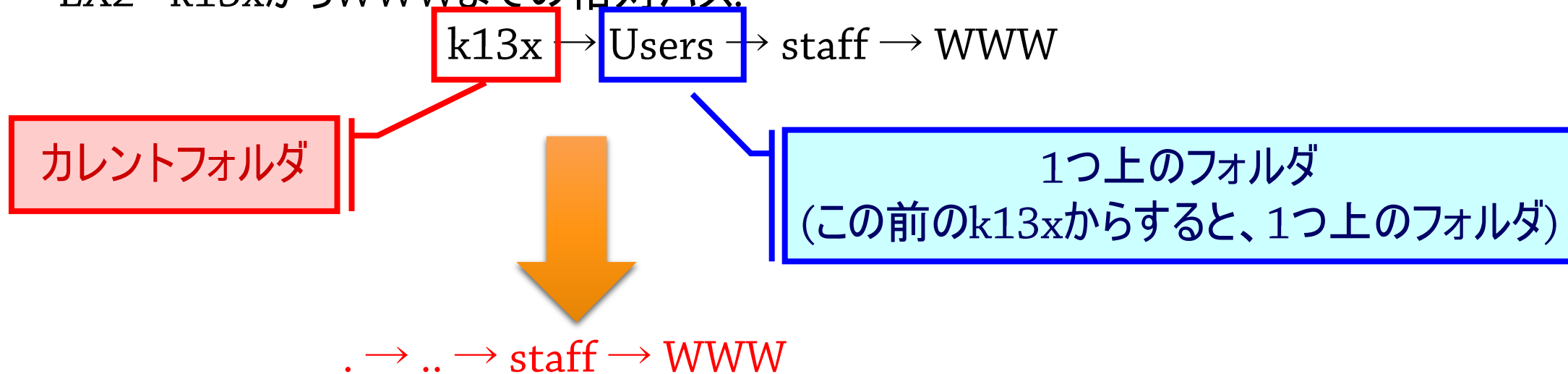
- カレントフォルダ(相対パスでの出発点): ./
- 1つ上のフォルダ: ..

Ex1 - Documentまでの絶対パス:

/ → Users → k14x → Document

EX2 - k13xからWWWまでの相対パス:

k13x → Users → staff → WWW



# 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 → k14x → Document

➡ **/Users/k14x/Document**

EX2 - k13xからWWWまでの相対パス: . → .. → staff → WWW

➡ **../staff/WWW**

# OSインストール実習

# PC準備

- HDD取り付け
  - HDDとマザーボードをケーブルで接続
  - HDDと電源を接続
- キーボード・マウス・ディスプレイとPCを接続

# 電源投入

- キーボードの「Delete」キーを連打して、マザーボードのBIOSを表示
  - BIOS
    - マザーボードに接続されている機器の確認やマザーボードの動作の設定をするためのソフトウェア
    - マザーボードに搭載されているソフトウェア
- BIOSで、HDDやDVDドライブの認識を確認



# ブートデバイスの検索順序を設定

- ブートデバイス: OSが保存されている部品
  - HDD・SSDやDVD-ROMなど
- ブートデバイスの起動順序
  - マザーボードは、PCの電源が入ると、ブートデバイスがどれかを検索
  - 検索し、初めてOSが見つかったデバイスでOSを起動
    - DVD→HDD→... のようにを探し、見つければOSを起動
  - 検索順序を設定可能
    - Ex. HDDが故障したとき
      - HDDを最初に探すことになっている場合、PCが起動しなくなる
      - DVDドライブを最初に探すように設定すると、OSのCDやDVDから起動ができ、故障の原因調査や修復を試みることができる

# OSのインストール

- OSのCDを入れてPCを起動
  - BIOSが動作している段階では、PCの電源をそのまま切ってOK
- 今回インストールするOS: Linux
  - Vine Linuxというディストリビューション

# OSの起動

- インストール終了後、CDを取り出してPCを再起動する
- ぶどうのマークが画面に表示されたら、「F1」キーを押す
  - OSの起動時に何をしているかが表示される

# コンピュータの状況の観察

- 「アプリケーション」→「システムツール」→「システム・モニタ」でコンピュータの状況を観察
  - 「リソース」タブで、CPUやメインメモリの使用状況
  - 「プロセス」タブで、各プロセスの状況
    - CPUやメインメモリの使用量など

# PCの再起動

- DVDを入れて起動してみる
  - どうか??

# 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

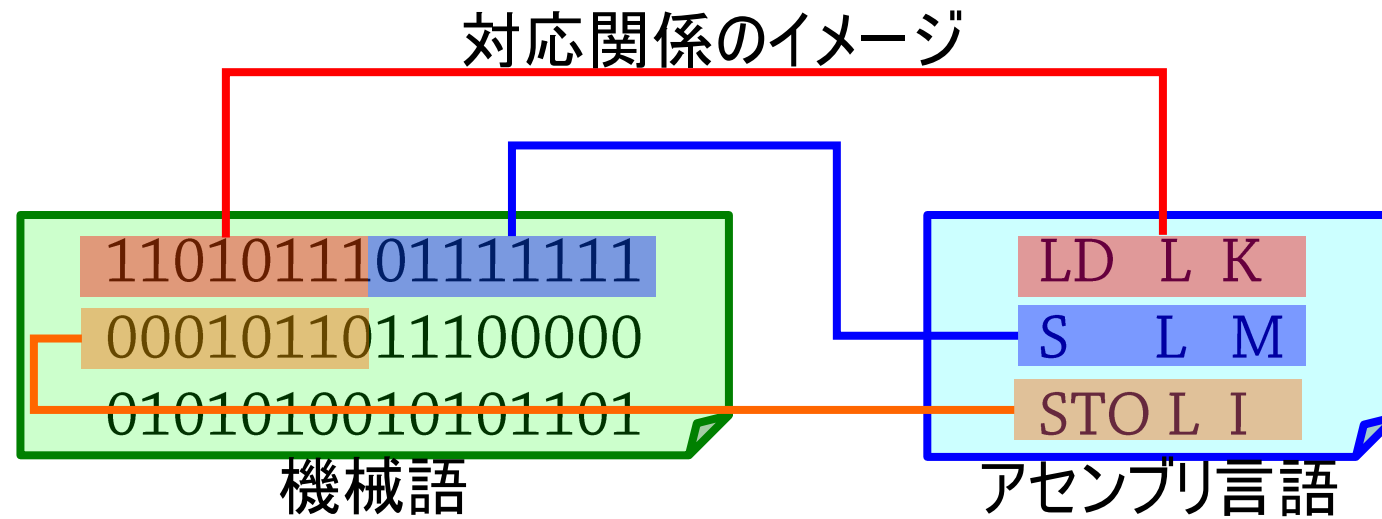
➤ 機械語: 0と1の2進数の形式の言葉

➡ 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

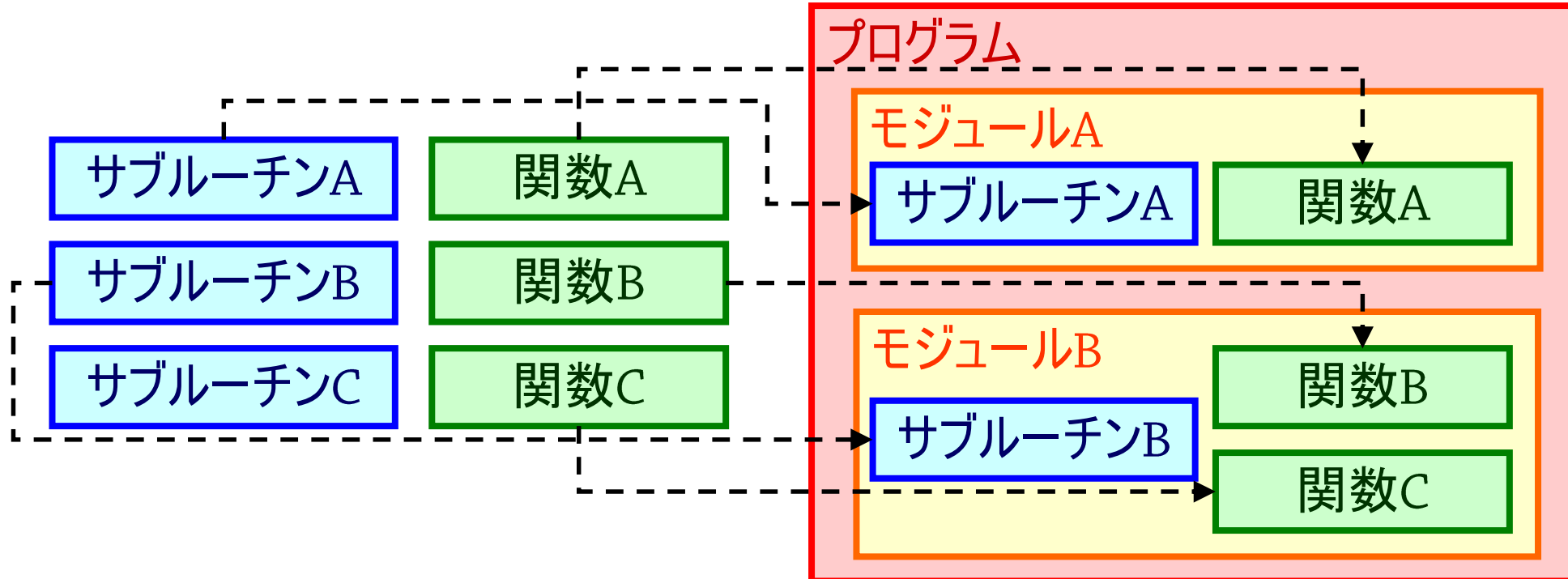
# プログラミング言語の変遷[3](p. 76)

- 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
  - ゼロックス社のバ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

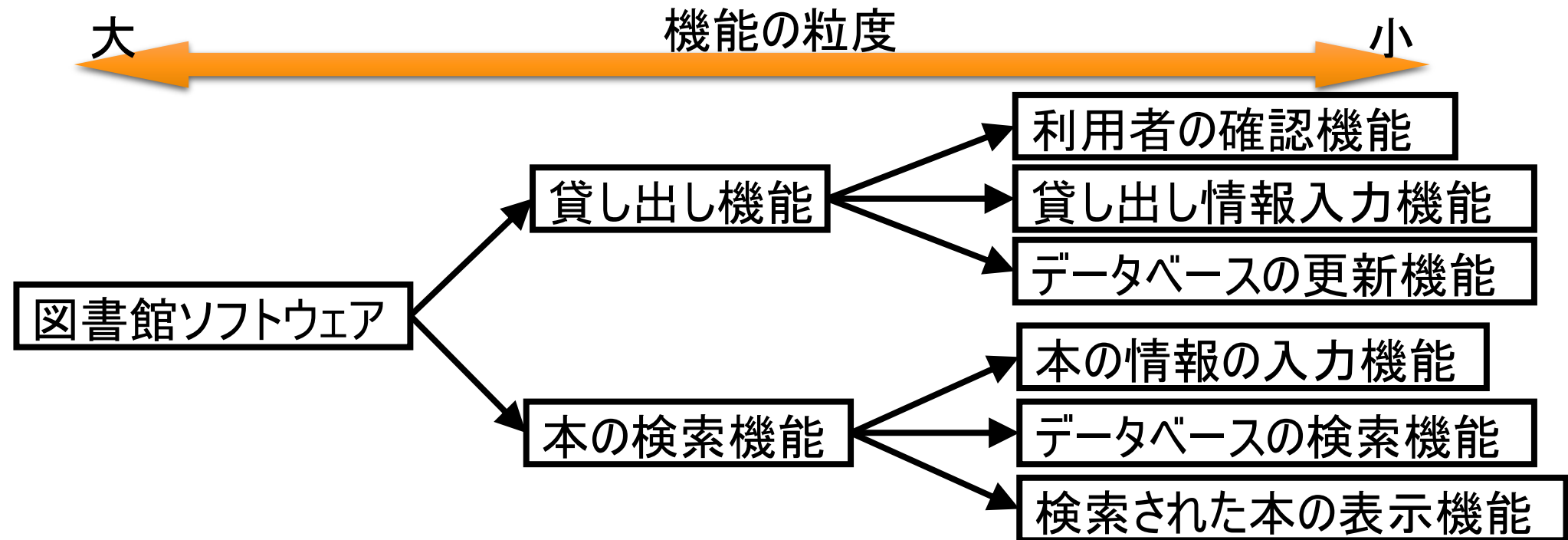
- プログラムを処理の単位ごとに分割(モジュール)
- サブルーチンと関数でモジュールを構成
  - サブルーチン: あらかじめ用意された、よく使う処理のまとめ
  - 関数: 自分で定義する、よくつかう処理のまとめ



# 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

