コンピュータ・サイエンス2

第5回
オペレーティングシステム(続き)

人間科学科コミュニケーション専攻
白銀 純子

第4回の内容

- オペレーティングシステム(続き)
- クラウドコンピューティング
- ファイルシステム

前回の出席課題の解答

設問1

- 「デバイスドライバ」とは何かを説明しなさい。

解答:
補助記憶装置や入出力装置などの制御を行うソフトウェア

設問2

- 下記について、正しいか間違いかを答えなさい。
  a. スマートフォンはプログラム内蔵方式ではない
  b. コンピュータの速さの性能は、CPUの性能のみで決定される
  c. 現在のノートPCで、Microsoft Wordのプログラムは、補助記憶装置(HDDまたはSSD)に保存されている
  d. プログラム内臓方式のコンピュータでは、プログラムは1度作れば良い

解答:

- 正しい: c, d
- 間違い: a, b

前回の質問の回答

前回の復習

プログラム内蔵方式[1](p. 56)

- 当初のコンピュータ: 現在プログラムが行っている処理をスイッチと配線の組み換えでその都度操作
  - スイッチと配線の組み換えで様々な命令を表現
  - 非常に手間がかかり、間違いが多発
    - たくさんのスイッチのONとOFFを逐一切り替える必要
    - 同じ処理をするために毎回プログラムを作成する必要

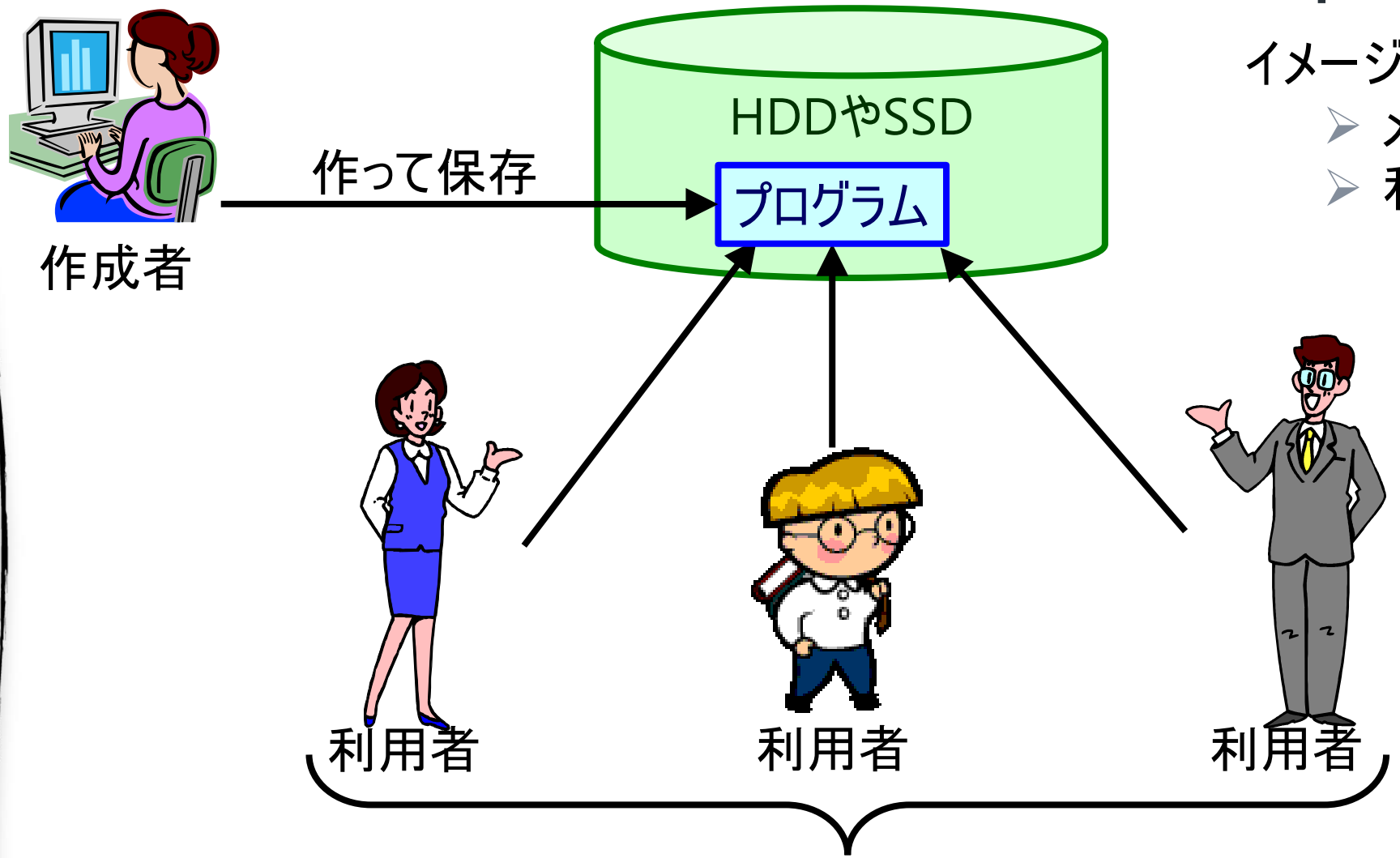
プログラム内蔵方式[2](p. 56)

- プログラム内蔵方式: 作成したプログラムを記憶装置に記憶させて利用する方式
  - プログラムを実行するときに命令を記憶装置から取り出して実行
    - 同じ処理をするためにプログラムは1度だけ作成
  - 用途に応じてプログラムを入れ替え、様々な処理を実行

「ノイマン」という人によって提案された方式

- ノイマン:「コンピュータの父」と呼ばれるアメリカの数学者
- プログラム内蔵方式で動くコンピュータを「ノイマン型コンピュータ」
  - ✓ 現在のコンピュータのほとんどはノイマン型コンピュータ

プログラム内蔵方式[3](p. 56)



イメージ例: 服

- メーカー: 作る
- 利用者:
  - ✓ 買ってきてたんすに入れる
  - ✓ 必要なときにたんすから出して着る

- 服: プログラム
- メーカー: 作成者
- たんす: HDDやSSD

- 作る必要はなし
- 必要なときに取り出して利用

逐次制御方式[1](p. 56)

- メインメモリから命令が1つずつ取り出されて実行される方式
  = メインメモリとCPUは頻繁にやりとり
  - プログラムカウンタによって、命令が格納されている番地を指定
  - その番地から命令を取り出して実行
  - プログラムカウンタの数値を1増やして次の命令の番地を指定

CPUとメインメモリの間の通信経路の転送速度のためにコンピュータ全体の処理の速度が遅くなる問題も存在

フォン・ノイマン・ボトルネック

逐次制御方式[2](p. 56)

利用者

利用したいと指定

HDDやSSD

プログラムA

1つ1つの命令を1つ1つの部屋にコピー

- プログラム: 様々な命令のかたまり

メインメモリ

部屋から1つずつ命令を取り出し、順番に実行

= 逐次処理方式

CPU

Question!

オペレーティングシステム

ソフトウェアの種類(p. 64)

- オペレーティングシステム(OS)
  - ハードウェアを動かすための基本的なソフトウエア
    - ハードウェアと応用ソフトウェアの間のやり取りを仲介
    - ハードウェアの制御
- アプリケーション(アプリケーションソフト、とも)
  - OSをもとに動くソフトウェア
  - 人間が直接操作するソフトウェア
    - 文書処理ソフト
    - 表計算ソフト
    - Webブラウザ
    - etc.

オペレーティングシステム(p. 64)

- Operating System(OS)
- 「基本ソフトウェア」とも呼ばれる
- コンピュータに必須の基本的なソフトウェア
  - 役割: ハードウェア資源(CPUや記憶装置など)とソフトウェア資源(プログラムやデータ)を効率的に管理

※プログラム: コンピュータに指示をするための命令の集まり

OSの発展[1](p. 65)

- 道具や機械で計算をさせるという考えは古くから存在
  - 紀元前: 算盤
  - 1640年代: 歯車式計算機
    - フランスのパスカル(Blaise Pascal)が考案
  - 1945年: ENIAC(Electronic Numerical Integrator and Computer)
    - ペンシルベニア大学で開発

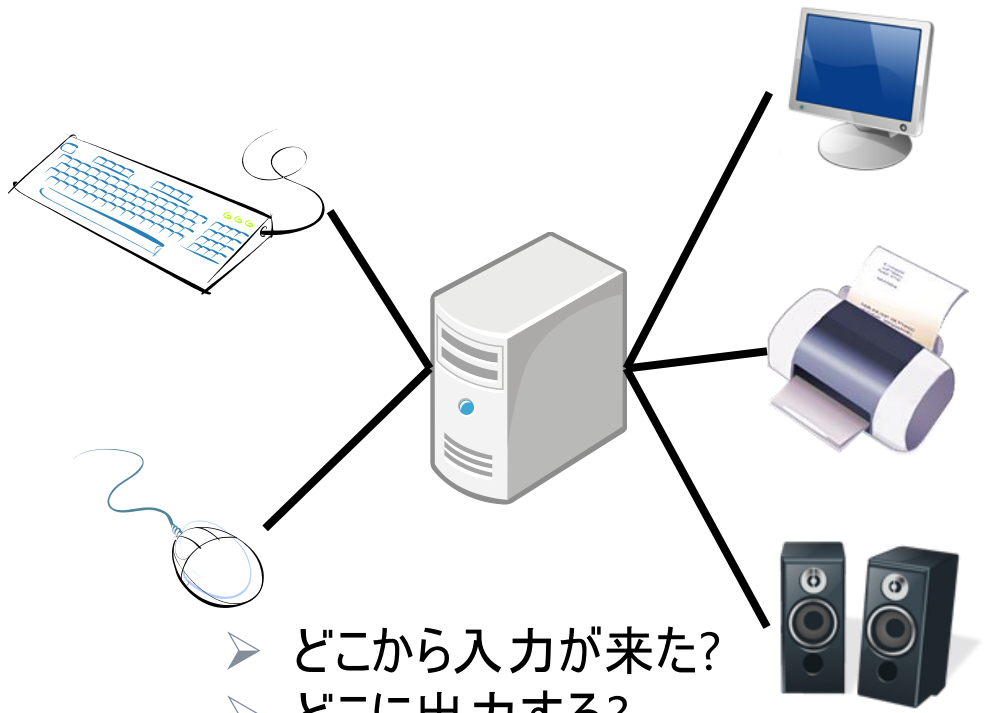
- 「OS」と呼べるほどのきちんとした形のソフトウェアはなかった
- 人間が線をつなぐことにより、計算していた
  - ✓ 計算速度は人間よりも高速
  - ✓ 異なる手順で計算をする場合には、配線をし直す必要

OSの発展[2](p. 65)

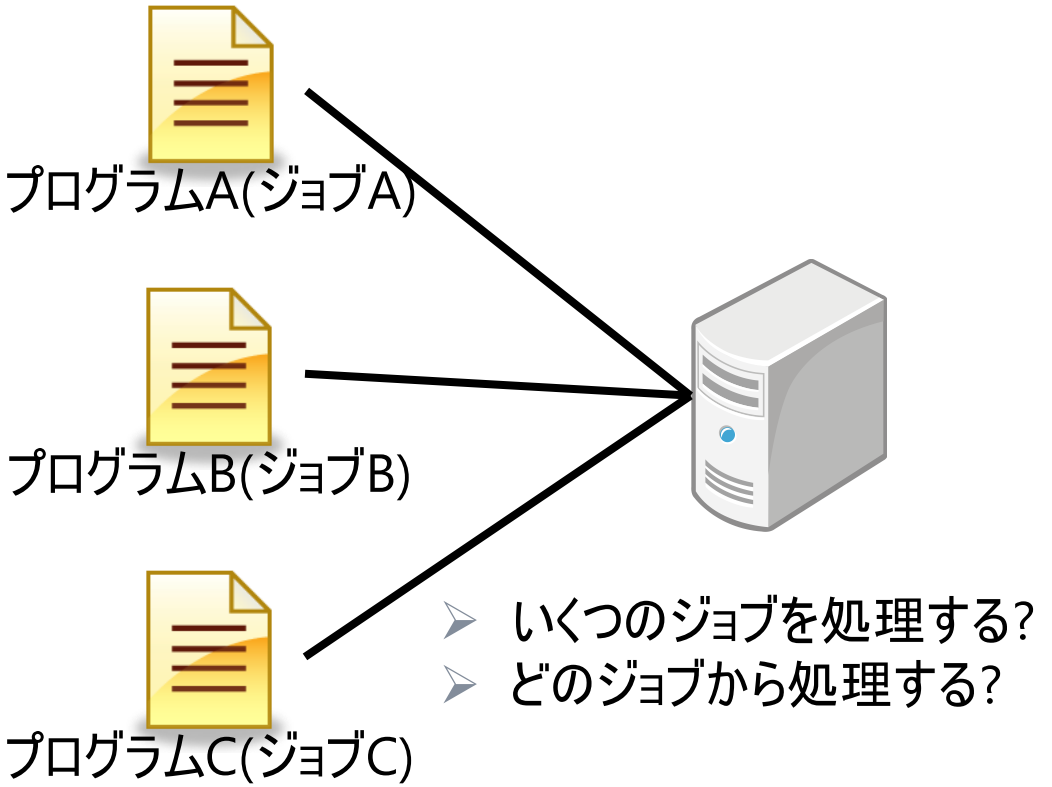
- ノイマン(John von Neuman)がプログラム内蔵方式を提唱
  - ハンガリー出身のアメリカの数学者
- 1949年: EDSAC(Electronic Delay Strage Automatic Calculator)
  - Maulice Wilkesらが開発
  - 実際に動くことはなかったが、プログラム可能なコンピュータの原型
- 初期(この頃)のコンピュータでは、実行できるプログラムは1つだけ

OSの発展~OSでの処理~[1](p. 66)

- 時代が進むと…
  - 1台のコンピュータに様々な入力・出力装置(入出力制御)
  - 複数のプログラム(ジョブ)を連続して処理(ジョブ管理)
    - ジョブ: 人間がコンピュータに命令する処理の単位



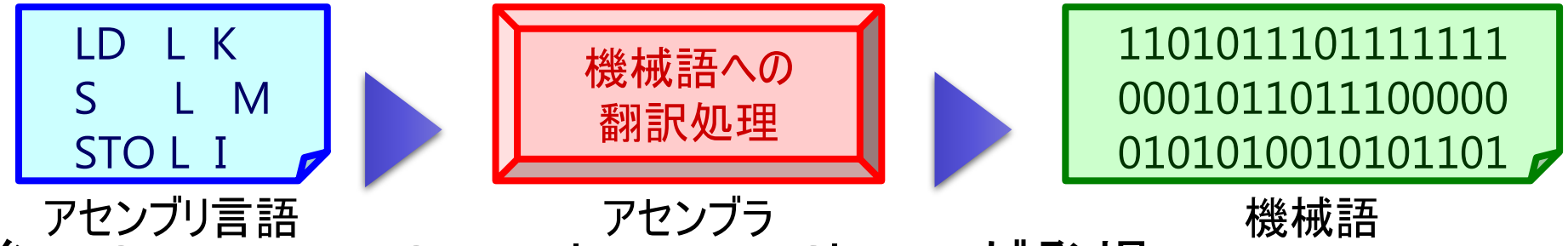
- どこから入力が来た?
- どこに出力する?



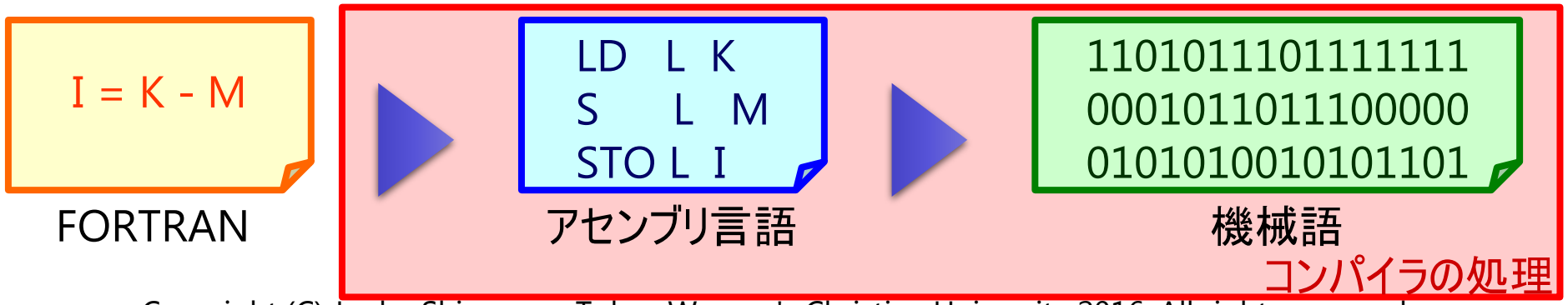
- いくつのジョブを処理する?
- どのジョブから処理する?

OSの発展~OSでの処理~[2](p. 67)

- プログラム(コンピュータへの命令書)を記述できる言語
  - 初期: 機械語(2進数だけの表現)で記述
  - 機械語は不便: アセンブリ言語が登場
    - アセンブリ言語でプログラムを書き、機械語に翻訳(アセンブラ)



  - その後: FORTRAN(FORmula TRANSlator)が登場
    - 英語に近い形で記述し、機械語に翻訳(コンパイラ)



OSの発展～OSでの処理～[3](p. 67)

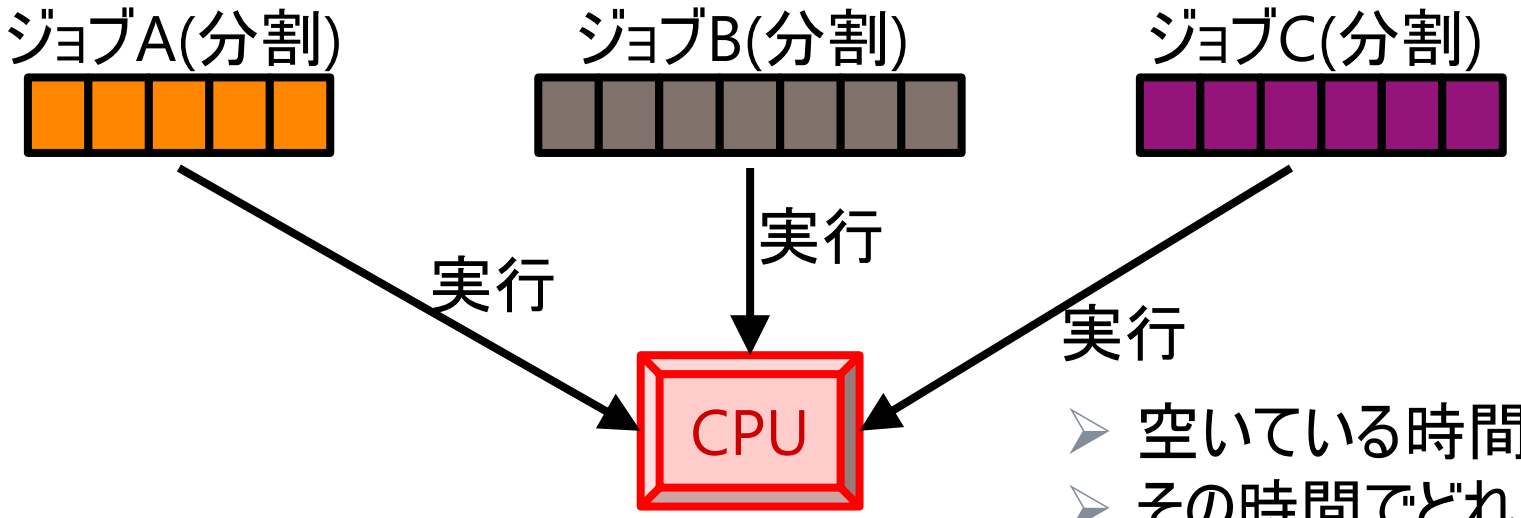
- 当時のコンピュータ: メインメモリの容量が少
  - メインメモリ: プログラム実行時に、プログラム内の命令や利用するデータを一時的に保存
    - 容量が少ないと、命令の多いプログラムやサイズの大きいデータを保存しきれない

➡ メインメモリが足りなくなったら、補助記憶装置(HDD)にメインメモリの役割をさせる(仮想記憶装置(Virtual Storage))

➡ メインメモリと補助記憶装置(仮想記憶装置)とのやりとりを管理するプログラムも必要

OSの発展~OSでの処理~[4](p. 68)

- 様々な処理をコンピュータで行う必要
  - CPUに、効率よく、各ジョブを実行させる必要
    (タイムシェアリングシステム (Time Sharing System, TSS))



- 空いている時間はいつ?
- その時間でどれを実行する?

➡ 多重プログラミング(同時に複数の処理を並行して行うこと)が可能に
- CPUが一度に行う処理は1つだが、CPUの処理時間を細かく分割し、複数の処理を同時に行っているように見せかける方法

➡ CPUの割り当て制御プログラムが必要

OSの発展～OSでの処理～[5](p. 68)

- 1台のコンピュータを遠隔地から使う必要

  ➡ リモートアクセスのプログラムが必要

OSの発展～OSでの処理～[6](p. 68)

- DOS/VS(Disk Operating System/Virtual Strage)にこれまでの必要なプログラムがすべて搭載
  - コンピュータがメインフレームと呼ばれていた頃に搭載されていたOS

- コンピュータを制御するための各種プログラムが開発
- さまざまなプログラムや処理方式を統合し、1つにまとめて提供するという方向性

OS(Operating System)として登場

OSの発展～その後～[1](p. 69)

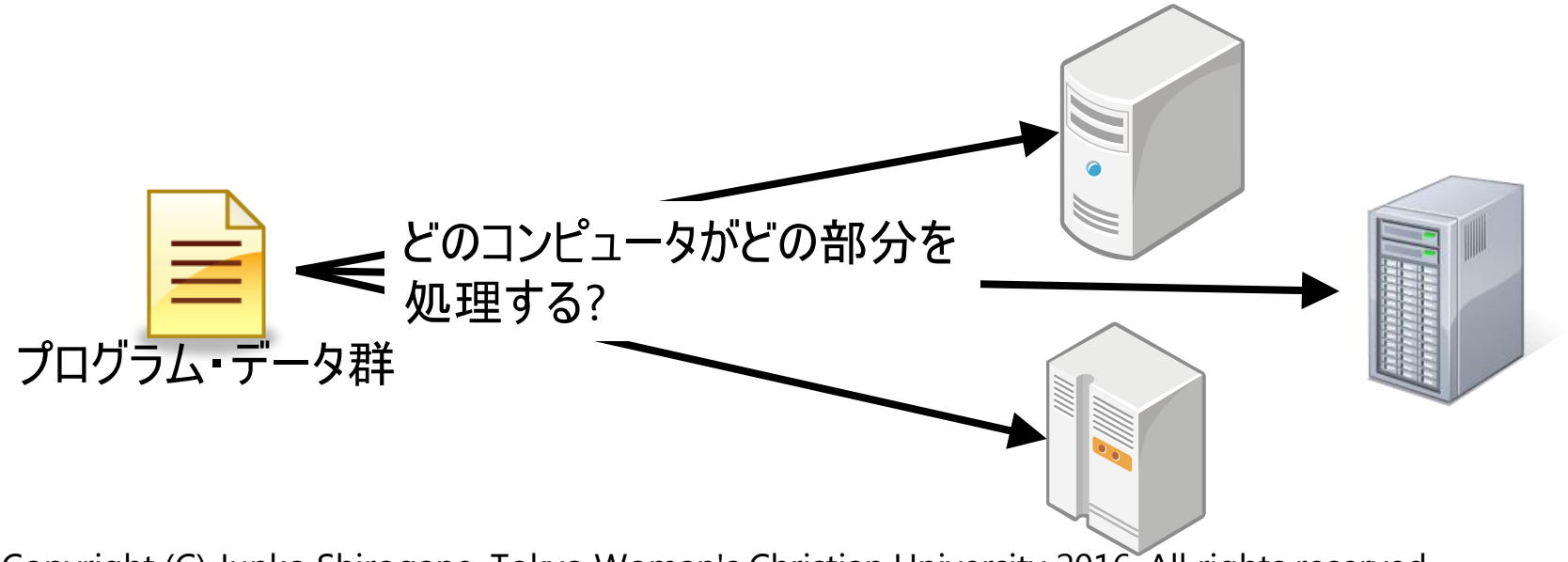
- 1台のコンピュータを数台分のコンピュータとして利用する考え方が登場(仮想化)
  - タイムシェアリングシステムの発展
  - CPUやメインメモリなどを仮想化

  ➡ マルチプロセッシングシステムや複数のOSを管理するシステムが必要
    - マルチプロセッシングシステム: 複数個のCPUなどを制御するためのシステム

- コンピュータとコンピュータをつないで利用する考え方が登場(コンピュータネットワーク)

OSの発展～その後～[2](p. 69)

- 離れたところにあるコンピュータ同士を専用回線でつなげて使う必要
  - ➡ 通信制御装置を管理するプログラムが必要
- コンピュータとコンピュータをつないで利用する考え方が登場
  (コンピュータネットワーク)
  - 1台に処理がかたより、処理速度が遅くなってしまうため
    - ➡ 分散処理システムが開発



現在(p. 70)

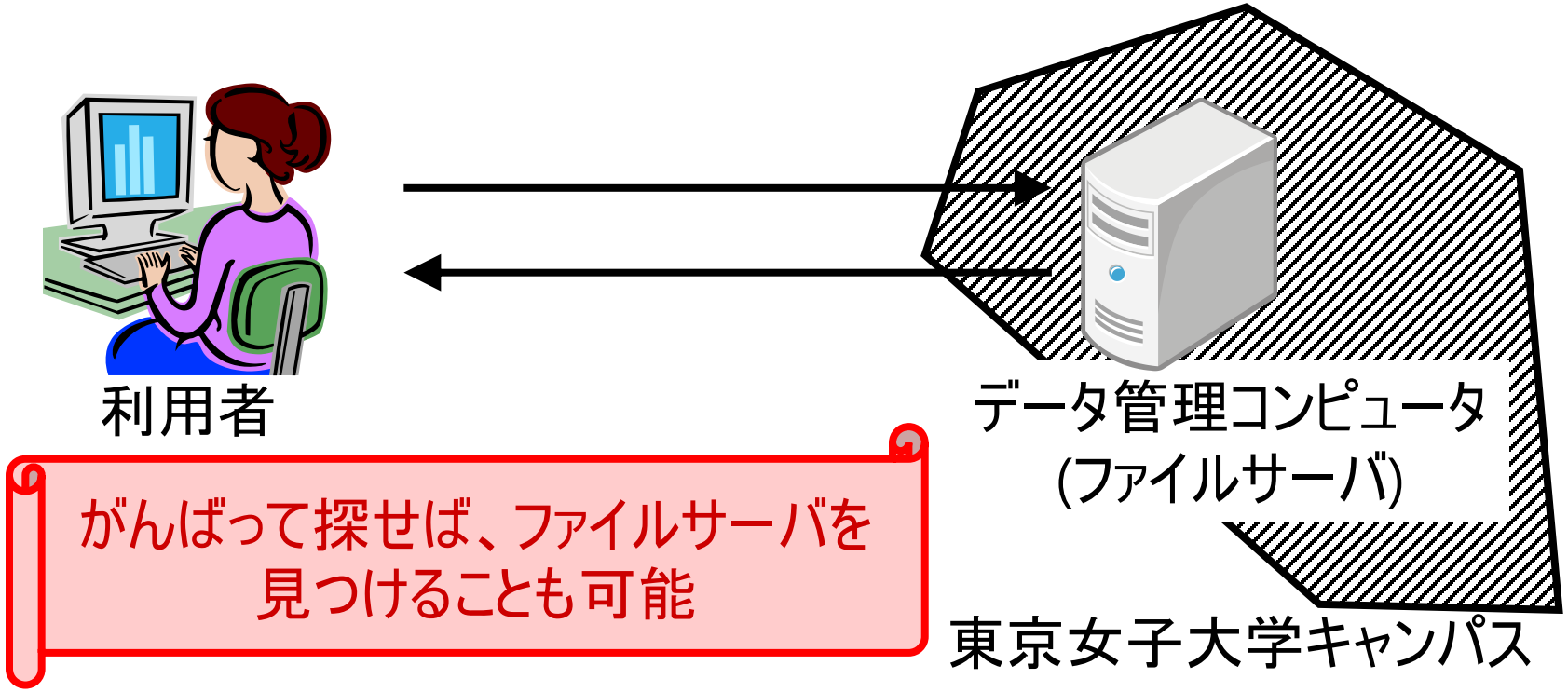
- メインフレームの時代のOSの機能は、現在でも利用
- クラウドコンピューティングの登場(「クラウド」とも)
  - ソフトウェアのオープンソース化や無償化
    - オープンソース: 機械語に翻訳前のプログラムを公開して誰でも編集可能にしたもの
  - 企業や組織などで利用するソフトウェア・ハードウェアの維持管理コストの軽減のニーズ

クラウドコンピューティング[1](p. 70)

- これまで: 利用者がアクセスするコンピュータは1台
  - データ管理, メールの処理, etc.
  - どこにあるどのコンピュータか、特定可能

Ex. 東京女子大学でのホームのデータ利用

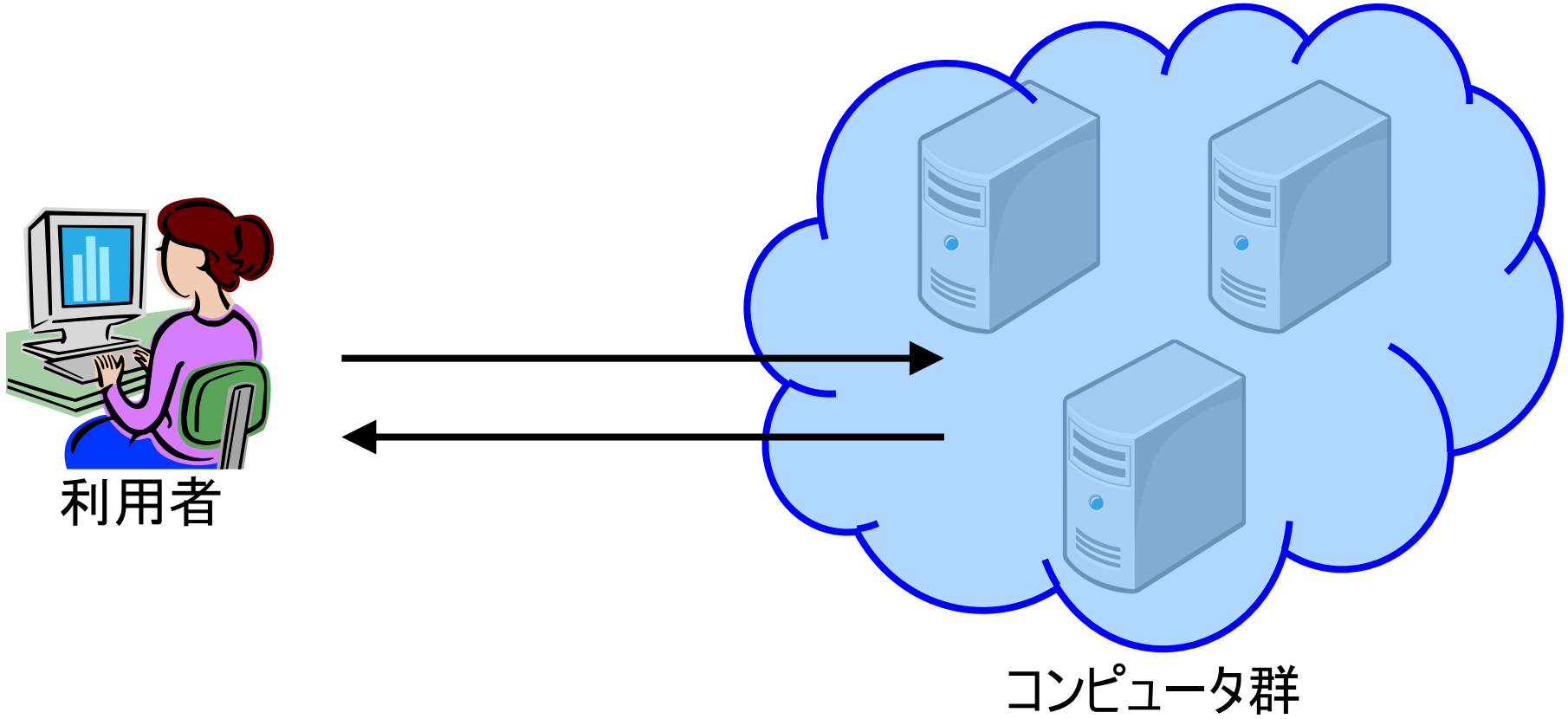
- データはファイルサーバに保存されているので、ネットワークを通じてどのコンピュータからも利用可能



がんばって探せば、ファイルサーバを
見つけることも可能

クラウドコンピューティング[2](p. 70)

- クラウド: たくさんのコンピュータのどれかとやりとり
  - たくさんのコンピュータを1つのコンピュータとして見せかけ



クラウドコンピューティング[3](p. 70)

- クラウドコンピューティングでは…
  - アクセスするコンピュータは、地理的に一か所に固まって置かれておらず、世界中の各地に置かれていることも

利用者側からすると…
自分が通信しているサーバが、日本にあるかアメリカにあるかもわからない

- アクセスするコンピュータの実体がどこにあるかわからない
- インターネットの概念図は、ネットワークを雲の形で表現することが多い

「クラウド(雲)コンピューティング」と呼ぶ

クラウドコンピューティングの形態(p. 71)

- SaaS(Software as a Service)
  - 必要なときにインターネットを通じてソフトウェアを利用できる仕組み
    - これまで: ソフトウェアは1台1台の利用者用のコンピュータにインストールして利用
  - 有料のものも無料のものも存在
- PaaS(Platform as a Service)
  - OSやデータベースなどの基本的なソフトウェアをインターネットを通じて利用できる仕組み
- IaaS(Infrastructure as a Service)
  - コンピュータの基盤を提供するサービス
    - 機材や回線、OSなどのシステムに必要なインフラ

Question!

仮想化

仮想化(p. 72)

- 仮想化: ハードウェアのリソースを複数台分に、複数台を1台分に見せかけて動作させること
  - リソース: CPUやメインメモリ、補助記憶装置、入出力装置などなどの利用可能な資源
  - 利用者それぞれの専用のリソースがあるように見せかけて利用者に使わせることが可能

仮想化の方法[1](p. 72)

- CPU
  - 複数の実行させるプログラム(プロセス)の制御
    - 起動の制御, 実行順序の管理, etc.
  - スレッドによる並行処理
    - スレッド: プロセスでの様々な処理を同時に実行させる方法
- メインメモリ
  - 仮想記憶装置の作成
  - メインメモリと仮想記憶装置の統合的な管理

仮想化の方法[2](p. 72)

- HDD/SSD
  - ファイルやフォルダ(ディレクトリ)の管理(ファイルシステム)
    - ファイルの作成・削除・ファイル名によるアクセス・ファイルの保護, etc.
- 入出力機器
  - 入力機器からの入力、結果の出力機器への出力の制御
    - プログラム制御方式: 入出力のデータ転送をCPUが管理
    - チャネル制御方式: 入出力のデータ転送を「チャネル」という専用装置が管理
      (CPUを介して転送)
    - DMAコントローラ方式:入出力のデータ転送を「DMAコントローラ」という専用装置が管理
      (CPUを介さないで転送)

OSの種類

以前のコンピュータ[1](p. 73)

- メインフレームの時代: 機種に依存(機種ごとに異なる)
  - 時代が進んでコンピュータ同士を接続して利用
    - ➡ OSが違うと、接続プログラムが非常に複雑化
- 現在: ある程度OSが標準化
  - 各種OSでサーバ用OSとクライアント用OSに分けられている
    - 同じハードウェアのコンピュータでも、どちらをインストールするかでサーバになったりクライアントになったり
    - 店で売られているPCはクライアント用OS

PCのOS[2](p. 73)

- MS-DOS: PCの初期の頃によく使われていたOS
  - Microsoft-Disk Operating System
  - 「コマンドプロンプト」が画面上に表示
    - コマンド(命令)を入力することで、OSに直接命令可能(マウスは使わず、キーボードから命令を入力するだけ)
  - CUI(Character User Interface)のOS
    - CUI: 人間とのやりとりの接点(User Interface)が文字だけ
- Mac OS: Apple社が開発したOS
  - GUI(Graphical User Interface)を初めて搭載したOS
    - GUI: マウスを使ってアイコンや、ボタン、入力フィールドなど、視覚的な操作が可能なユーザインタフェース(現在の形)

現在のOS[1](p. 74)

- Mac OS
  - 1978年にApple社から提供された、GUIを持つ最初のOS
  - マウス操作やアイコンの概念を初めて導入
  - 現在はMac OS X
- Microsoft Windows
  - 1985年にMicrosoft社から初めて提供されたGUIを持つOS
    - 初期のWindows(Windows3.0, Windows3.1)はMS-DOSの拡張
    - Windows95で現在の原型
    - 現在のWindow 8までバージョンアップをして様々な機能を追加
    - サーバ用OSも提供

現在のOS[2](p. 75)

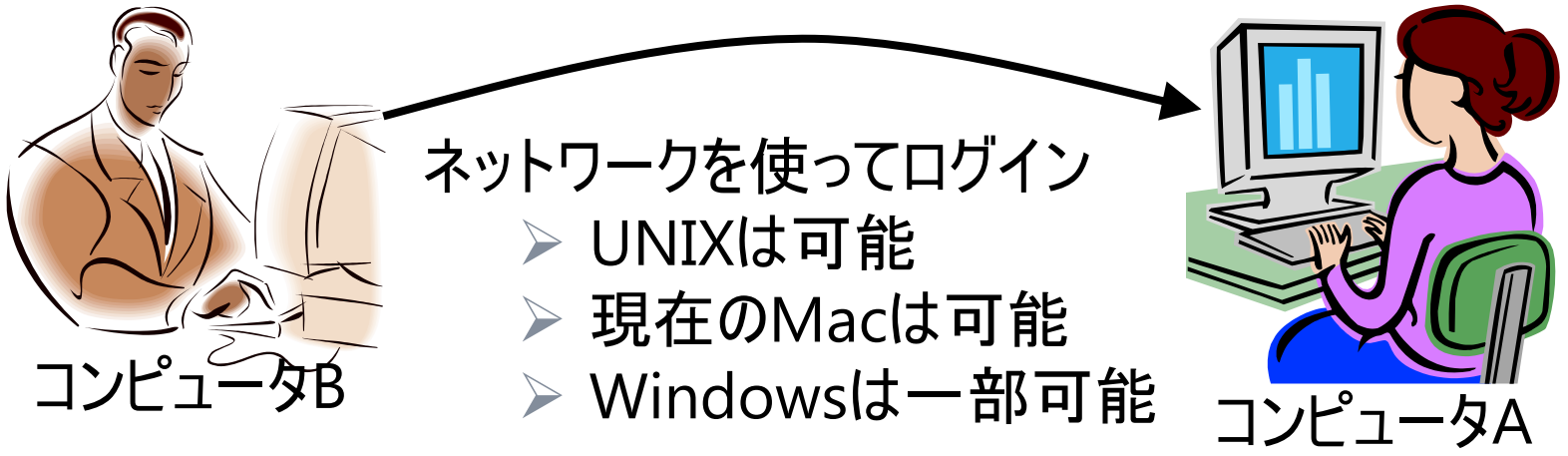
- UNIX
  - 歴史
    - 1969年にAT&T(アメリカの電話会社)から提供
    - 1970年から1980年初期まで大学や研究所などで利用
  - 現在はサーバとして多く利用
  - マルチユーザ環境
  - Solaris, AIX, SUN OSなど、様々な種類が存在(有料のものもフリーのものも存在)

現在のOS[3](p. 75)

- Linux
  - 1991年にリーナス・トーパスル(ヘルシンキ大学の学生)が開発
  - UNIX的なOS(使い方などがほぼUNIXと同じ)
  - オープンソースのフリーなOS
  - PCでも大型機でも利用(サーバとしても利用)

マルチユーザ環境

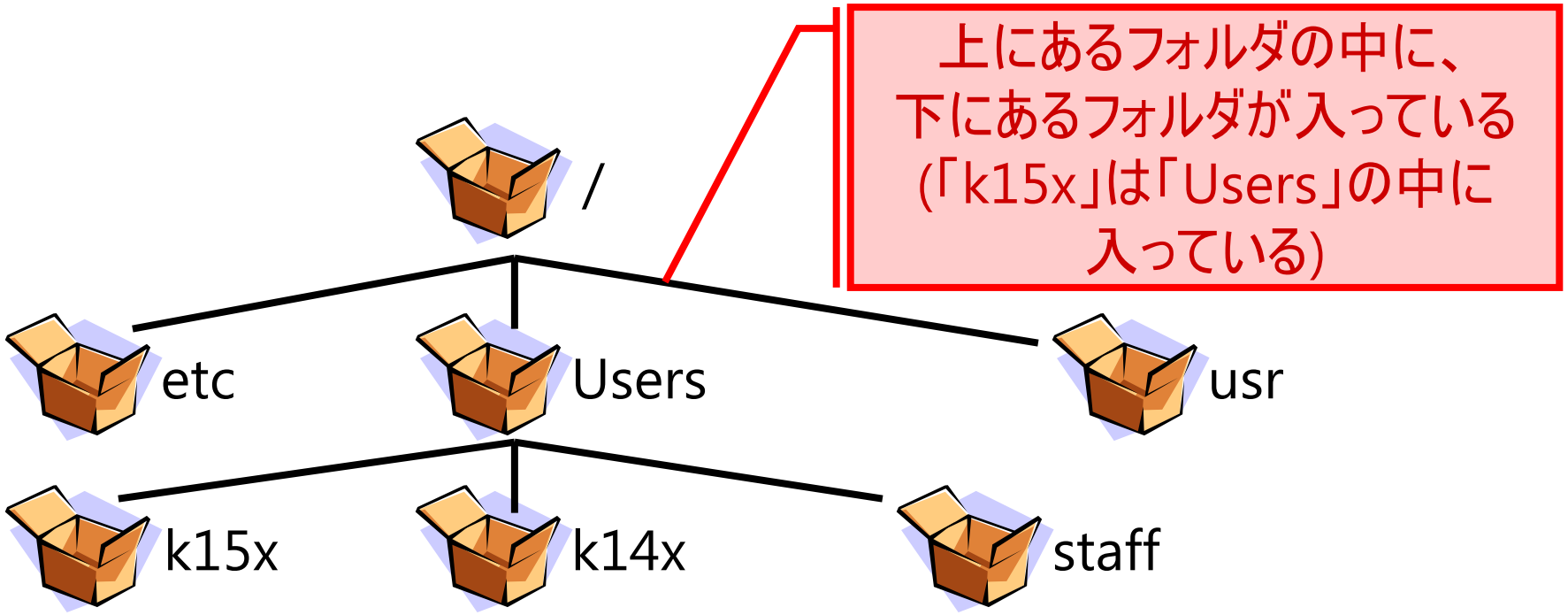
- 1つのコンピュータを複数の人が同時に利用可能
  - ネットワークを使って、あるコンピュータAから別のコンピュータBにログインすることができる
    - AにはなくてBにあるソフトウェアを利用したい場合(Aの前に別の人が座っている場合など)
    - 遠くからBを操作する必要がある場合(コンピュータの管理など)



ファイルシステム

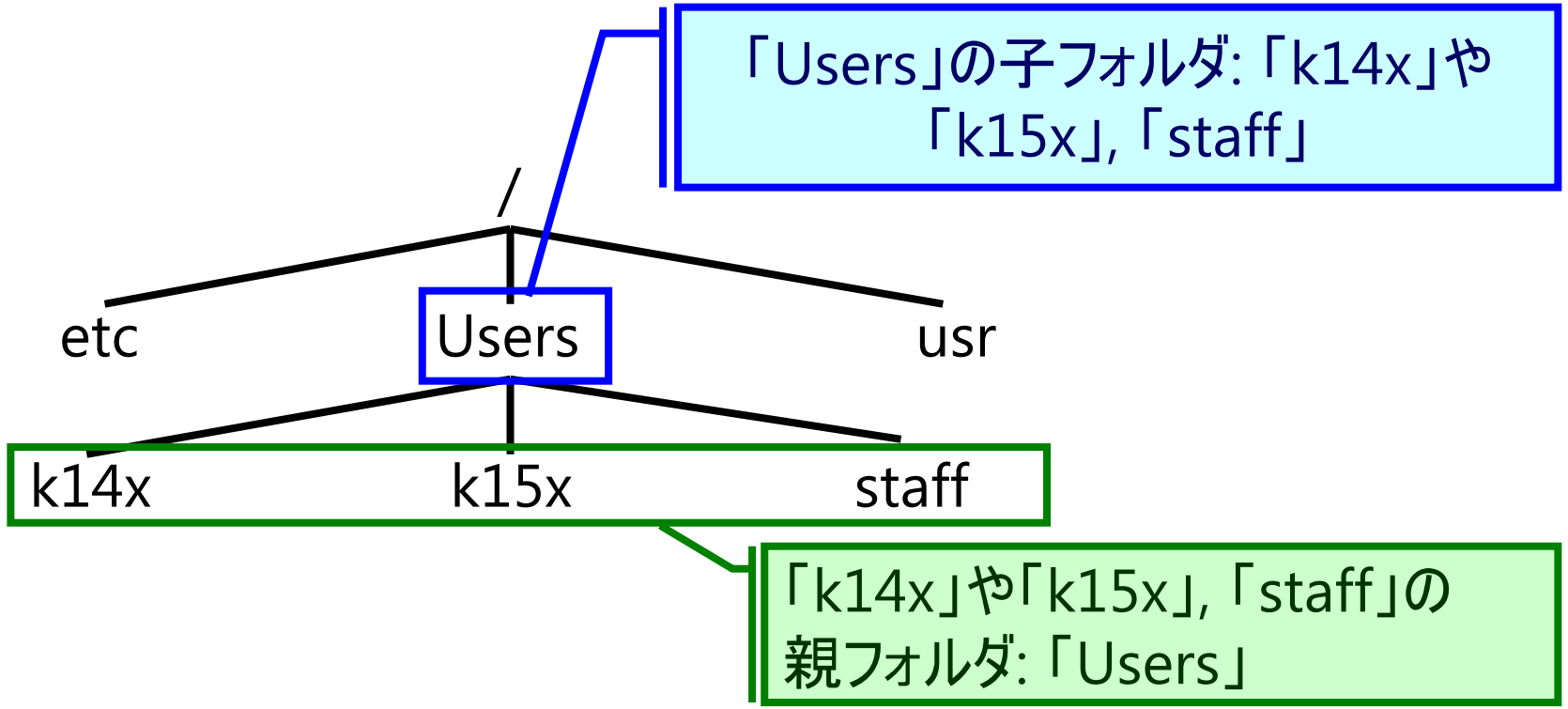
フォルダの階層構造

- 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



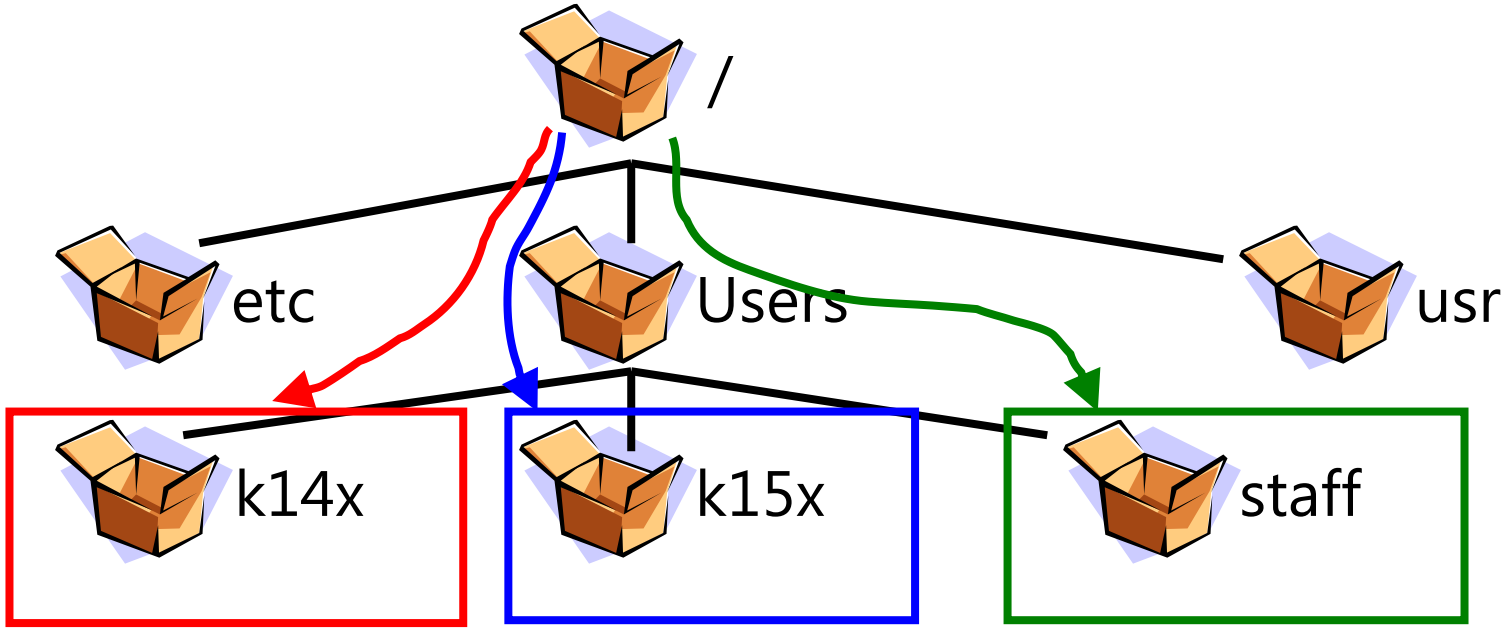
親フォルダ, 子フォルダ

- 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

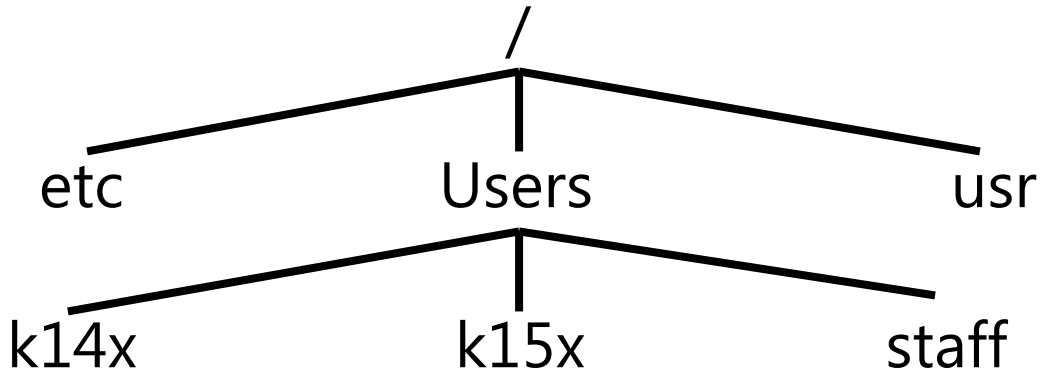
- あるファイルやフォルダが、どの位置にあるかを表すもの
  - どのようにフォルダをたどれば、目 的のファイルやフォルダにたどり着くかを表すもの



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

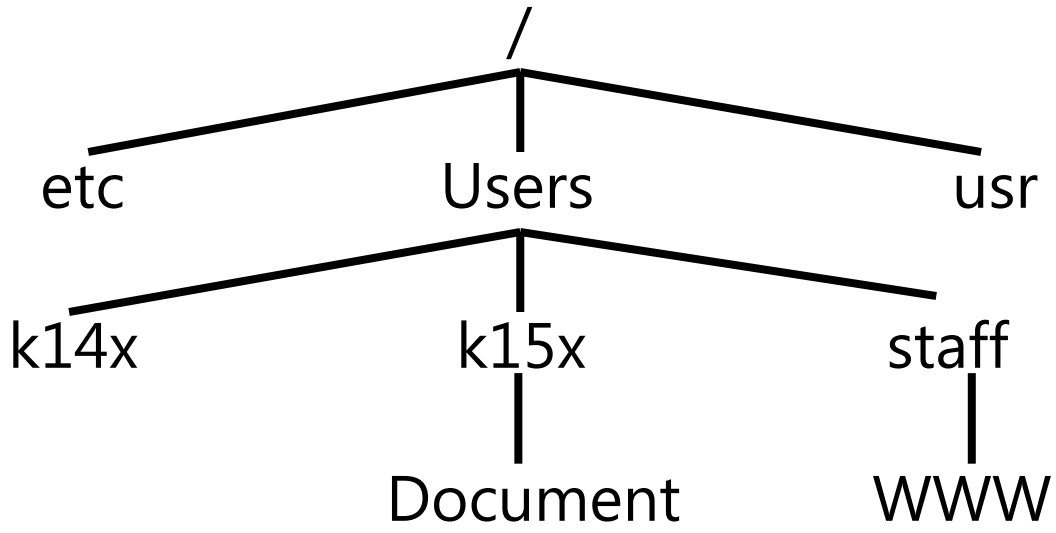
「/」で表す(Windowsでは「¥」)

「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

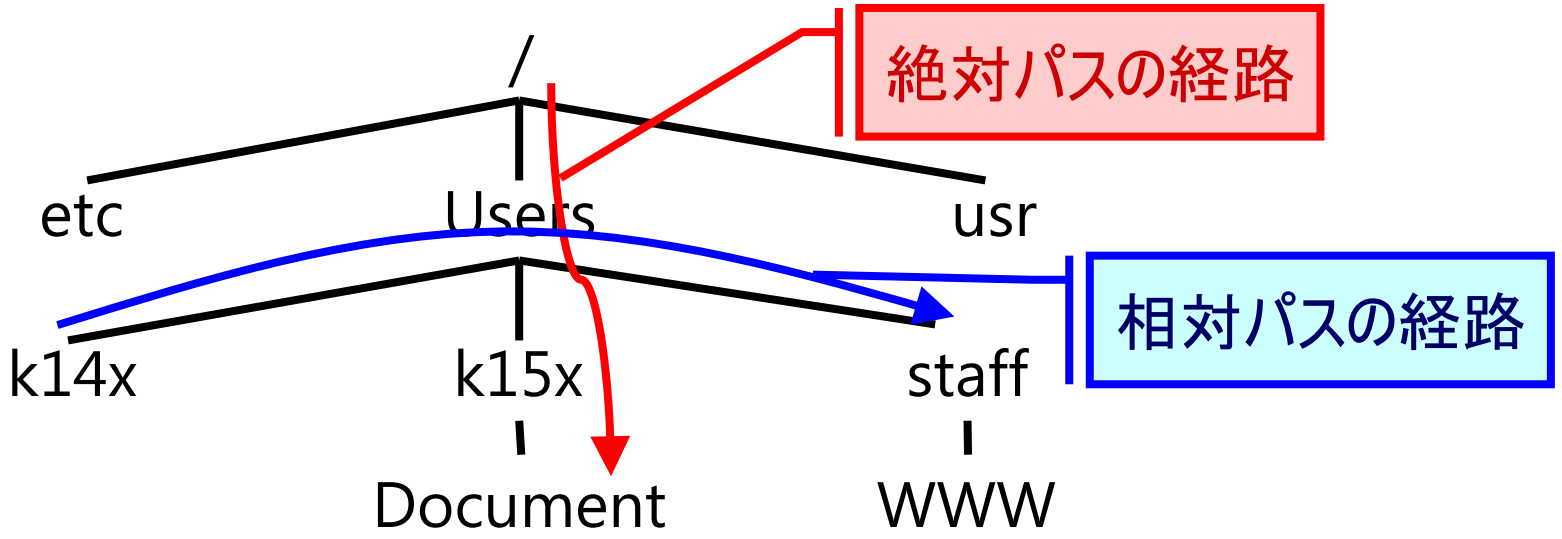
※カレントフォルダ「./」は省略可能

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

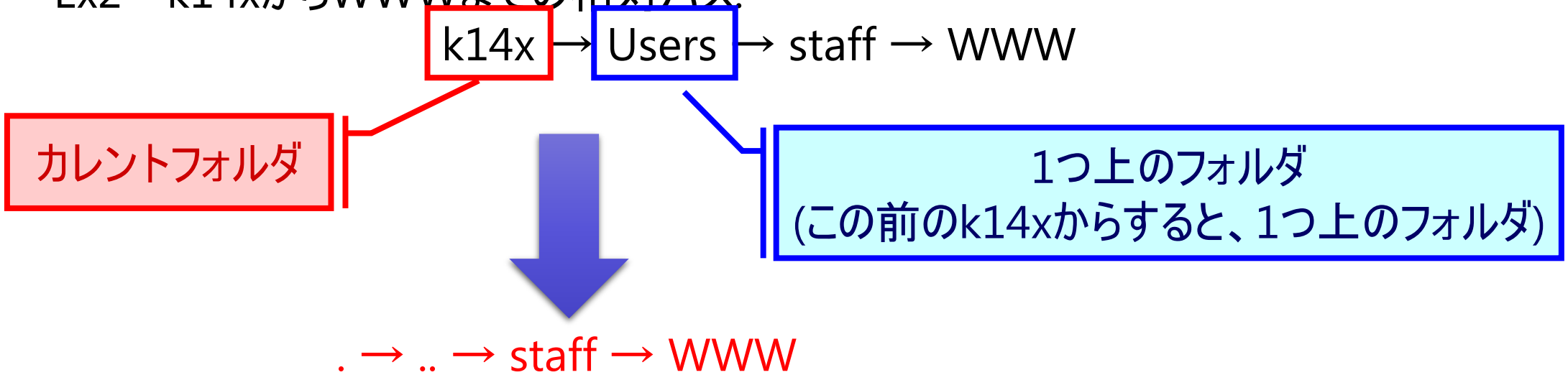
パスの考え方[2]

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

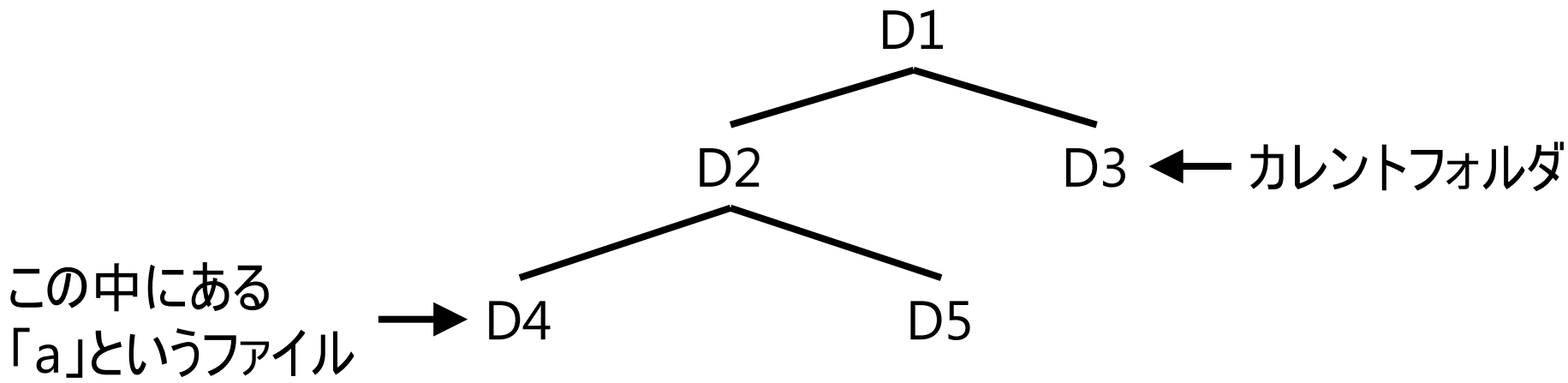
Ex2 - k14xからWWWまでの相対パス: . → .. → staff → WWW

➡ ../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

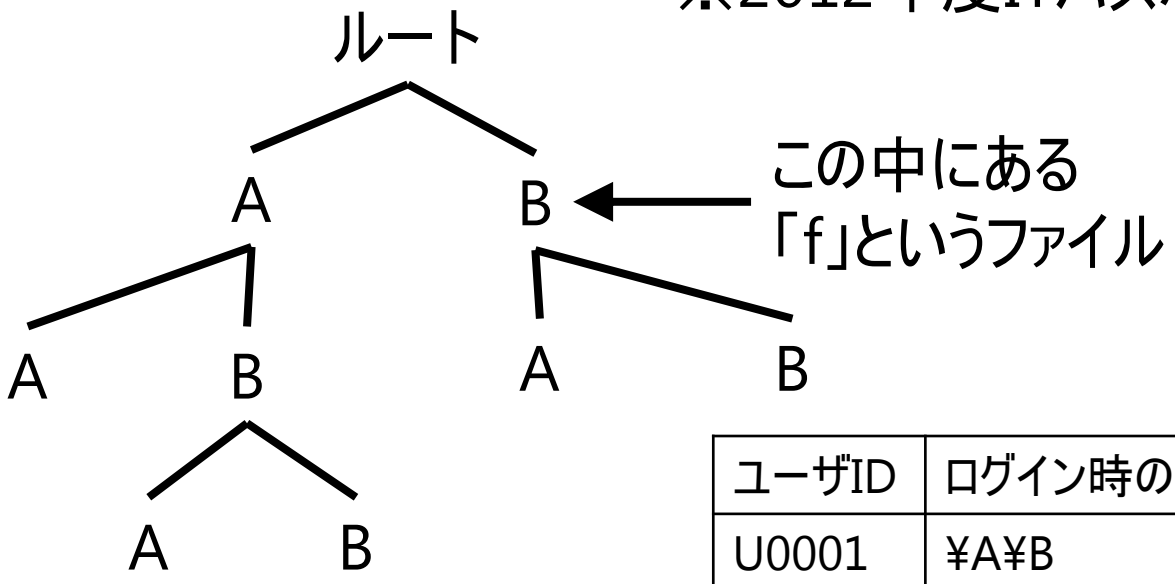
※2011年度ITパスポート 春期試験より

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!

次回

- 11月1日: 休講
- 11月8日: OSのインストール実習
  - 遅刻しないこと!
  - 荷物運びのお手伝いをしてくれる人を募集
    - してくれる人は、次週8:55頃に8号館4階8413号室前に来てください。