こちらは本講座で使用したテキストの一覧です。
復習用にご活用ください。

Linuxをはじめよう

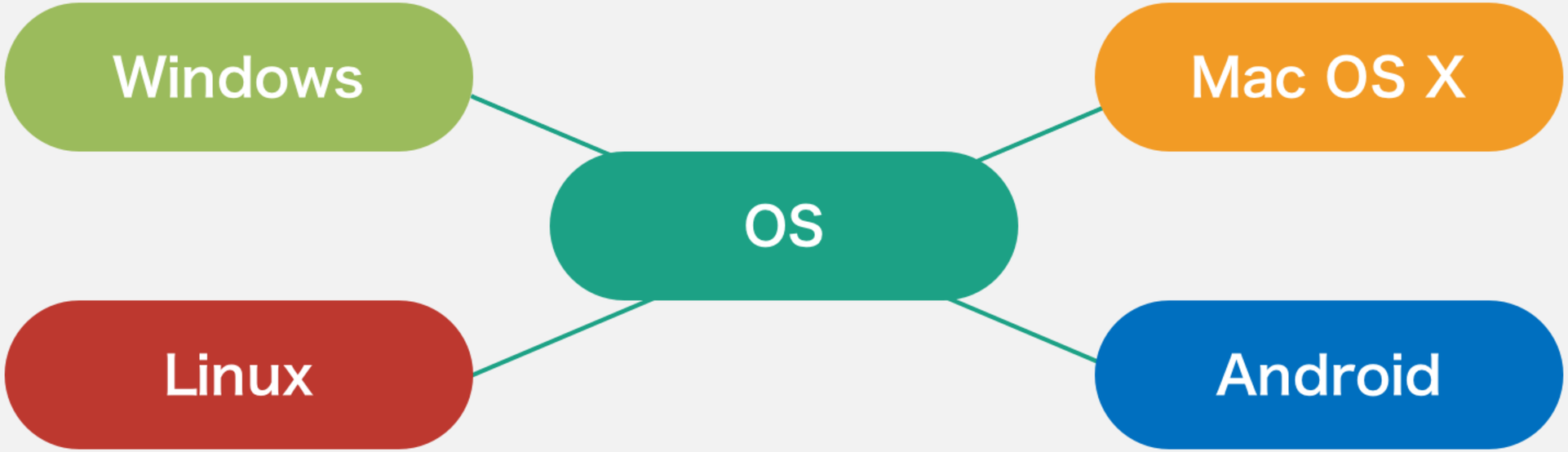
- Linuxってなに？
- なぜLinuxを学ぶの？
- Linuxディストリビューションってなに？
- シェルってなに？
- VirtualBoxをインストールしよう
- コマンドを打ってみよう
- シェルの便利機能

Linuxをはじめよう

- Linuxってなに？ ここ
- なぜLinuxを学ぶの？
- Linuxディストリビューションってなに？
- シェルってなに？
- VirtualBoxをインストールしよう
- コマンドを打ってみよう
- シェルの便利機能

Linuxって何？

LinuxはOSの一種で、コンピュータを動かすための基本ソフト



Linuxはどのように誕生したの？

1970年頃

- UnixというOSが誕生
- シンプルな小さいOSで大学や研究所を中心に普及
- 有料。ライセンス縛りも厳しい

1985年〜

- オープンソース版のUnixを開発するプロジェクトがスタート
- ソースコードも公開されていたので、様々な派生版のUNIXが誕生

1991年

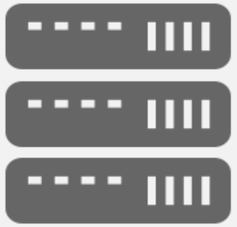
- 当時大学生のリーナス・トーバルズがLinuxを開発し、オープンソース化
- UNIXを真似て、一からOSを作る
- 最終的に最も普及

有料のUnixが、Linuxという形で、
無料で使えるようになった

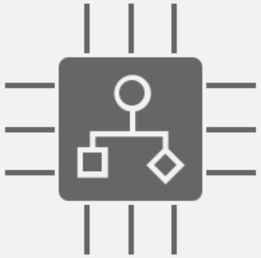
出展）　https://ja.wikipedia.org/wiki/%E3%83%AA%E3%83%BC%E3%83%8A%E3%82%B9%E3%83%BB%E3%83%88%E3%83%BC%E3%83%90%E3%83%AB%E3%82%BA

Linuxはどこで使われてるの？

サーバー、組み込み、デスクトップといった幅広い用途で使われている



サーバー



組み込み

デスクトップ

Linuxの特徴と欠点

特徴

- オープンソースなので誰でも自由に使える
- 世界中で使われていて信頼性が高い
- 品質の高い多くのソフトウェアが利用できる
- 自動化するための仕組みがあり、サーバーの運用が行いやすい

欠点

- 商用アプリケーションがあまり存在しない
  - Microsoft Officeが動作しない
  - 画像、音楽、動画などのマルチメディア系ファイルを編集するのが苦手
- 日本語対応されていない部分が多い

まとめ

- LinuxとはOSの一種で、コンピュータを動かすための基本ソフト
- 1991年にリーナス・トーバルズがUnixをベースに一から開発
- 現在では大変普及しており、サーバー、組み込み、デスクトップといった幅広い用途で使われている
- 特徴
  - オープンソースなので誰でも自由に使える
  - 品質の高い多くのソフトウェアが利用できる
- 欠点
  - 商用アプリケーションがあまり存在しない
  - 日本語対応されていない部分が多い

Linuxをはじめよう

- Linuxってなに？
- なぜLinuxを学ぶの？ ここ
- Linuxディストリビューションってなに？
- シェルってなに？
- VirtualBoxをインストールしよう
- コマンドを打ってみよう
- シェルの便利機能

なぜLinuxを学ぶの？

Linuxは枯れている技術であり、何年たっても使える技能を得ることができる

幅広い場所で使われている

- サーバー用
- 組み込み器用

幅広い場所で使われており、そういった場所で
マシンを操作する時に必要

長年使える

Linuxの技術は長い時間をかけて
多くのユーザーに検証された結果、
大きく変更する必要がなくなっている

▶ 技術が枯れている

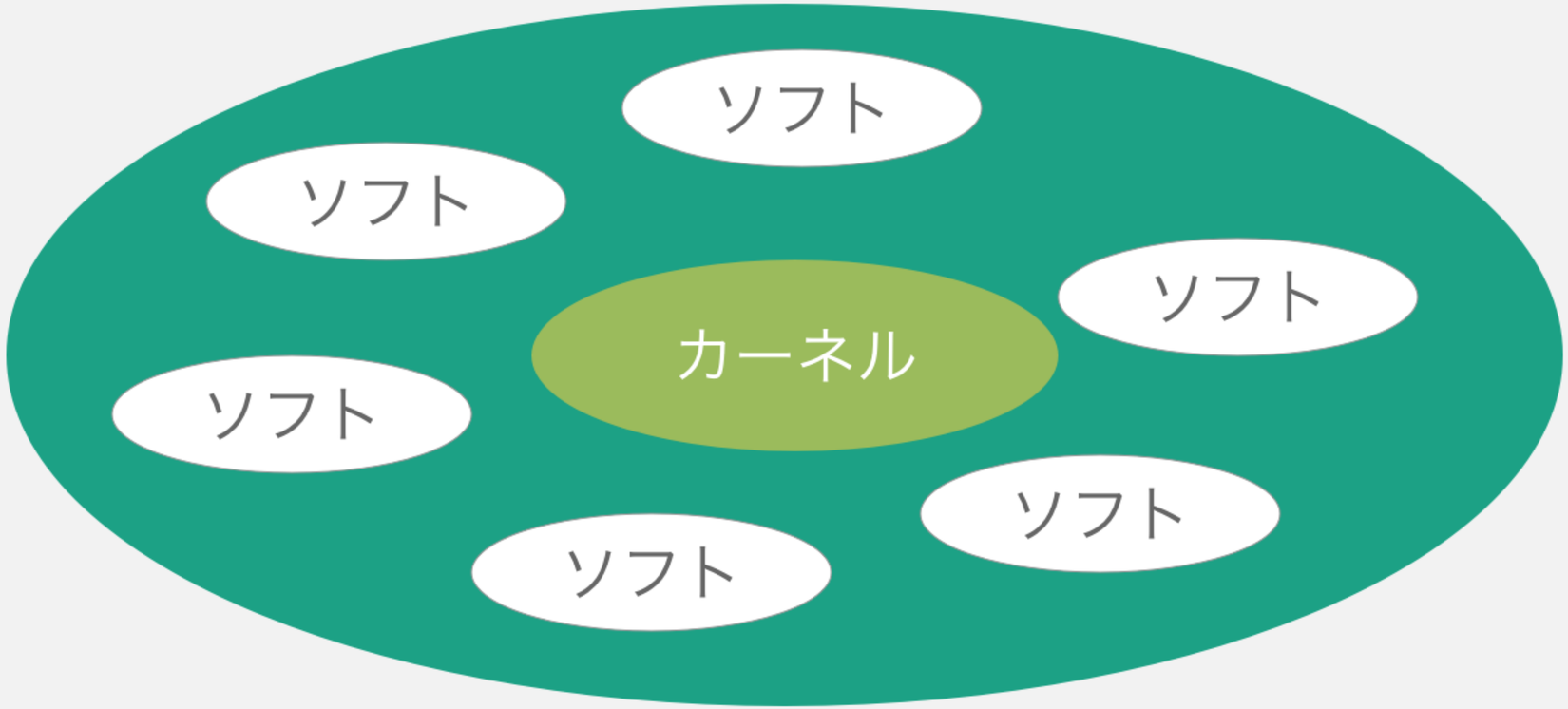
▶ 一度修得すれば
何年たっても使える！

Linuxをはじめよう

- Linuxってなに？
- なぜLinuxを学ぶの？
- Linuxディストリビューションってなに？ ここ
- シェルってなに？
- VirtualBoxをインストールしよう
- コマンドを打ってみよう
- シェルの便利機能

カーネルとディストリビューション

カーネルはOSの中核となるソフトウェア。ディストリビューションは
カーネルに色々なソフトウェアを組み合わせて使いやすくしたもの。



ディストリビューション

ディストリビューションには多くの種類がある

ディストリビューションは1000種類以上あり、独自のコミュニティが開発している。
Red Hat系とDebian系が主流。

Red Hat系

- Red Hat Enterprise Linux
- CentOS
- Fedora

本講座で使用する。
無償で手に入り、利用者が多く、情報も豊富

Debian系

- Debian GNU/Linux
- Ubuntu

まとめ

- カーネルはOSの中核となるソフトウェア
- ディストリビューションはカーネルに色々なソフトウェアを組み合わせて使いやすくしたもの
- ディストリビューションは1000種類以上あり、独自のコミュニティが開発している
- Red Hat系とDebian系が主流

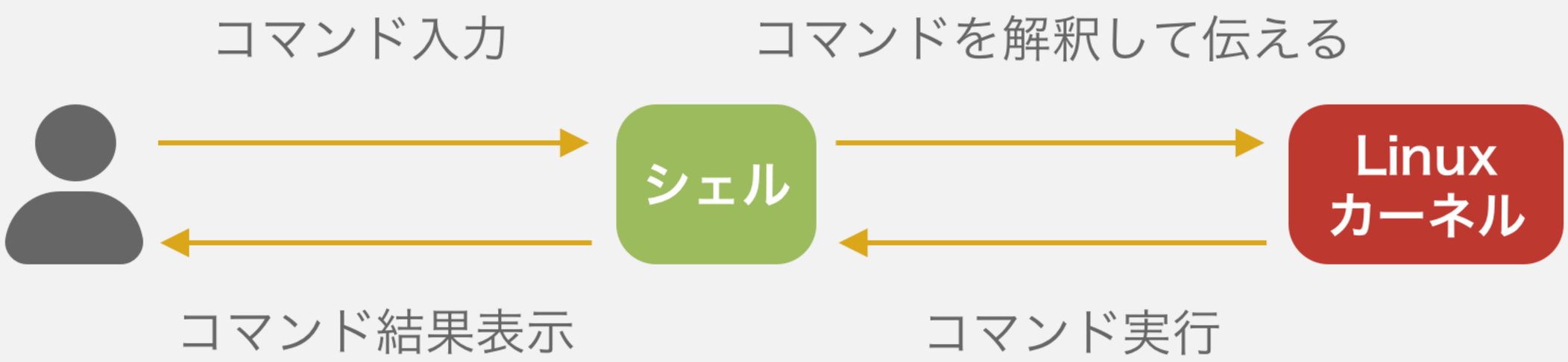
Linuxをはじめよう

- Linuxってなに？
- なぜLinuxを学ぶの？
- Linuxディストリビューションってなに？
- シェルってなに？ ここ
- VirtualBoxをインストールしよう
- コマンドを打ってみよう
- シェルの便利機能

シェルってなに？

シェルはLinuxのCLIを提供するソフトウェア。ユーザーとLinuxカーネルの仲介役（カーネルのインターフェース）をしている

CLIとは、キーボードからコマンドを入力し、
文字列として結果を出力するインターフェース



シェルの画面

ユーザー名、ホスト名、プロンプトがシェルでは表示されている。プロンプトは「コマンド入力を待ってるから、コマンド入力してね」というシェルが待っている記号

ユーザー名　ホスト名

[yamaura@localhost ~] $

プロンプト

シェルの仕組み

コマンドを入力すると実行結果が出力される

1. ユーザーがプロンプトにコマンドを打つ
2. シェルが入力された名前のコマンドを探す
3. Linuxカーネルが見つかったコマンドを実行する
4. シェルが実行結果を画面に表示する

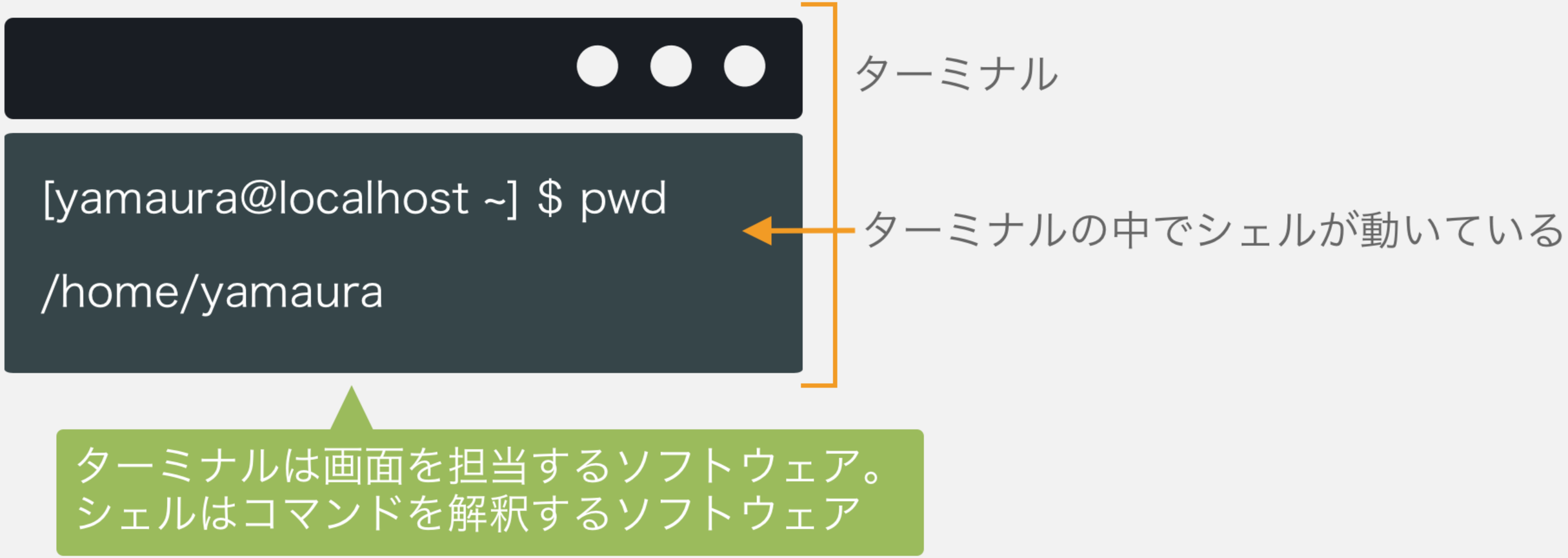
```
$ date

2019年  8月  2日 金曜日 20:39:41 UTC
```

プロンプト「$」は入力しなくていいよ。ユーザーが打ち込む場所とコンピュータが表示してくれる場所を見分けるのに、この書き方が便利なので慣習的に書かれてるよ

Linuxを操作するときはターミナルにコマンドを入力する

ターミナルは入出力の画面を提供するソフトウェア。
ターミナルとシェルは別物なので注意！



シェルの種類

sh

- もっとも古いシェル
- ほぼ全てのUnixで利用できる標準的なシェル
- 機能は少ない

bash

- shを基本として機能を拡張したシェル
- Linux環境でデフォルトのログインシェル
- 学習で使うならbashがオススメ

zsh

- bashの機能を取り込み、独自の拡張も追加
- 機能の豊富なシェル
- 講師はzshを使用

まとめ

- シェルはLinuxのCLIを提供するソフトウェア
- プロンプトは「コマンド入力を待ってるから、コマンド入力してね」というシェルが待っている記号
- ターミナルは入出力の画面を提供するソフトウェア。ターミナルとシェルは別物なので注意！
- シェルの種類には、sh、bash、zsh等がある
- カーネルとシェルが分かれていたり、シェルとターミナルが分かれているように、Linuxは「1つのプログラムには1つのことをうまくやらせる」「色々な機能をできるだけ細かく切り分けて作る」という設計思想がある

Linuxをはじめよう

- Linuxってなに？
- なぜLinuxを学ぶの？
- Linuxディストリビューションってなに？
- シェルってなに？
- VirtualBoxをインストールしよう　ここ
- コマンドを打ってみよう
- シェルの便利機能

Linux環境を用意しよう

コンピュータ上に仮想マシンを構築できるVirtualBoxを利用して、
学習用のLinux環境を用意しよう

VirtualBoxのインストール手順

インストール手順

1. Oracle VM VirtualBoxのインストール
2. CentOSのイメージファイルのダウンロード
3. Oracle VM VirtualBoxで仮想マシンを作成
4. 仮想マシンを起動し、CentOSをインストール

本動画の実行環境

項目	バージョン
Oracle VM VirtualBox	6.0.10
CentOS	7.6.1810

Linuxをはじめよう

- Linuxってなに？
- なぜLinuxを学ぶの？
- Linuxディストリビューションってなに？
- シェルってなに？
- VirtualBoxをインストールしよう
- コマンドを打ってみよう　ここ
- シェルの便利機能

今回学ぶこと

- VirtualBoxの画面を拡大して表示
- dateコマンドで現在時刻を表示

Linuxをはじめよう

- Linuxってなに？
- なぜLinuxを学ぶの？
- Linuxディストリビューションってなに？
- シェルってなに？
- VirtualBoxをインストールしよう
- コマンドを打ってみよう
- シェルの便利機能

ここ

カーソルの移動

コマンド	内容
Ctrl + f	一文字次へ移動
Ctrl + b	一文字前へ移動
Ctrl + a	行の先頭に移動
Ctrl + e	行の最後に移動
Meta + f (esc + f)	一単語次へ移動
Meta + b (esc + b)	一単語前へ移動

文字列の編集

コマンド	内容
Ctrl + h	カーソルの左を１文字削除
Ctrl + d	カーソル部分を１文字削除
Ctrl + w	カーソル位置の単語を削除（カーソルの前部分）
Ctrl + u	カーソルの位置から行頭まで削除
Ctrl + k	カーソルの位置から行末まで削除
Ctrl + y	最後に削除した内容を挿入

トラブル時の対応

コマンド	内容
Ctrl + c	実行中のコマンドを強制終了
Ctrl + l	画面の消去。文字化けしたの時のクリアにも使える
Ctrl + s	画面表示のロック（キーボードから文字の入力を受け付けない）
Ctrl + q	画面表示のロックの解除

補完・コマンド履歴

コマンド	内容
Tab	コマンドやパスの補完
Ctrl + p、または↑キー	一つ前の履歴を見る
Ctrl + n、または↓キー	一つ後の履歴を見る
Ctrl + r	コマンド履歴のインクリメンタル検索
文字の入力	コマンド履歴から検索
Ctrl + r	1つ前の検索結果へ移動
Enterキー	現在の検索結果を実行
Escキー	現在の検索結果を表示したままコマンドラインに戻る
Ctrl + g	検索結果を破棄し、コマンドラインに戻る

Day1のまとめ

- できるようになったこと
  - Linuxの概要を理解
  - Linuxディストリビューションについて理解
  - シェルの概要を理解
  - Linux環境を構築し、コマンドを実行できるようになった
  - シェルでのショートカットによる時間短縮

Day1のまとめ

- やってみよう
  - ターミナルで次のコマンドを実行してください。端末が沈黙するので、Ctrl + c でコマンドの実行を中止してください。

```
# 100秒間停止させるコマンド
$ sleep 100
```

  - ターミナルで次のコマンドを実行して、カレンダーを表示させてください

```
# 2019年のカレンダーを表示させる
$ cal 2019
# 2019年1月のカレンダーを表示させる
$ cal 1 2019
```

  - calコマンドを使って①2020年12月25日の曜日を調べてください②西暦20000年7月1日の曜日を調べてください

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう

Day2. ディレクトリを操作しよう

- Day2のゴール　ここ
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう

Day2のゴール

- Linuxでデータを格納する仕組み（ディレクトリ）を理解すること
- ディレクトリをコマンドを使って操作できるようになること
  - 【cdコマンド】ディレクトリの移動
  - 【pwdコマンド】カレントディレクトリの表示
  - 【lsコマンド】ディレクトリの中身の表示
  - 【mkdirコマンド】ディレクトリの作成
  - 【rmdirコマンド】ディレクトリの削除

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう　ここ
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう　ここ
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう

ディレクトリってなに？

ディレクトリはファイルが保存されている場所。WindowsやMacのフォルダと同じ概念

Linuxのディレクトリ構造

ルートディレクトリを頂点とした階層構造になっている。これを「ディレクトリーツリー」と呼ぶ

Windowsとの違い

Windowsはドライブから入れ子構造が始まるのに対して、
Linuxは一つのディレクリ（/）から始まる

Windows

ディスクが2つあると、2つのディレクトリツリーができる

Linux

複数ディスクがあると、ルートディレクトリ配下にディレクトリとしてくっつける。この操作を「マウント」と呼ぶ

主要なディレクトリの役割

/bin

- 一般ユーザーおよび管理ユーザーの両方が使用するコマンドの実行ファイルを格納
- 特に、Linuxの動作に最低限必要な重要度の高いコマンドの実行ファイルを格納

/dev

- デバイスファイル（ハードウェアをファイルとして扱えるようにしたファイル）を格納

/etc

- 設定ファイルを格納
- Linuxで動作するアプリケーションの設定ファイルや、Linux自体の設定に関わるファイルが置かれている

/home

- ホームディレクトリ（Linuxのユーザーごとに割り当てられる個人用ディレクトリ）が配置される
- ユーザーはホームディレクトリ内に自由にファイルやディレクトリを作成できる

主要なディレクトリの役割

/sbin

- 管理者ユーザー向けのコマンドの実行ファイルを格納

/tmp

- 一時的なファイルを格納
- アプリケーションの実行中に結果を一時的にファイルとして保存する場合に使用

/usr

- 各種アプリケーションと、それに付随するファイルを格納

/var

- 変化するデータを格納
- アプリケーションを動作する上で作成されたログなどを格納

まとめ

- ディレクトリはファイルが保存されている場所
- ルートディレクトリを頂点とした階層構造になっていて、「ディレクトリーツリー」と呼ぶ
- 一番上の「/」をルートディレクトリと呼ぶ
- Windowsはドライブから入れ子構造が始まるのに対して、Linuxは一つのディレクリ（/）から始まる
- ディレクトリごとに役割がある

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう　ここ
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう

cdコマンド

ディレクトリを移動するコマンド

何の略？　change directory

書式

```
cd [オプション] [ディレクトリ]
```

具体例

```
$ cd /home/taguchi/
```

Linuxの必須コマンドだよ。オプションはあまり使わないので覚えなくて大丈夫

特別なディレクトリの指定

カレントディレクトリ、親ディレクトリ、ホームディレクトリには特別な指定方法がある

ディレクトリ	意味	指定方法
カレントディレクトリ	現在自分がいるディレクトリ	「.」
親ディレクトリ	１つ上の階層のディレクトリ	「..」
ホームディレクトリ	「/home/ユーザー名」ディレクトリ	「~」
ルートディレクトリ	「/」ディレクトリ	「/」

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう　ここ
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう

pwdコマンド

カレントディレクトリを表示するコマンド

何の略？　print name of working directory

書式

```
pwd [オプション]
```

具体例

```
$ pwd
/home/fukushima
```

今いるディレクトリが分からなくなったらpwdコマンドで確認しよう

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう　ここ
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう

lsコマンド

ファイルやディレクトリの一覧を表示するコマンド

何の略？　list

書式

```
ls [オプション] [ディレクトリ・ファイル名]
```

具体例

```
$ ls /
bin  dev  home  lib64  mnt

# 複数のディレクトリを指定することもできる
$ ls / /dev
/:
bin  dev  home  lib64  mnt
/dev:
autofs  block  bsg  bus
```

便利なパス名展開

パス名展開を使うと複数のファイルを一度に指定できる

パス名展開

記号	意味
*	任意の文字列
?	任意の1文字

具体例

```
# 拡張子がhtmlのファイルの一覧を表示
$ ls *.html
index.html  home.html  job.html

# zから始まり4文字で終わるファイルを表示
$ ls /bin/z???
/bin/zcat  /bin/zcmp  /bin/znew
```

オプションでコマンドの動作を変えることができる

ハイフンになにかの記号を続ける形の引数をオプションと呼ぶ。
コマンドにオプションを与えることで、コマンドの動作を変えることができる

書き方

- コマンドの後ろにスペース区切りで指定する

```
$ ls -a
```

- 2つ以上指定できる

```
$ ls -a -l
$ ls -al
```

- オプションとコマンドの引数を同時に指定できる

```
$ ls -al /
```

オプションでコマンドの動作を変えることができる

書き方

- オプションが引数を取ることもある

```
$ ls -w 10
```

- 「--」で始まるロングオプションもある

```
$ ls --version
```

よく使うオプション

-l

- ファイルの詳細情報を表示する

```
$ ls -l
drwxr-xr-x   2   hori   hori   6   8月   6   11:57   dev
```

ファイルタイプ　ファイルモード　リンク数　所有者　所有グループ　サイズ　タイムスタンプ　ファイル名 ディレクトリ名

-a

- 隠しファイルも含めた全てのファイルを表示する

```
$ ls -a
.   .bash_history   .bashrc   desktop
```

. で始めるファイルは隠しファイルとみなして通常は表示されないよ

よく使うオプション

-F

- ファイル種別を表示する

```
$ ls -F
bin@  dev/  home/   lib64@
```

/　　：ディレクトリ
@　　：シンボリックリンク
*　　：実行可能ファイル
記号なし　：通常ファイル

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう　ここ
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう

mkdirコマンド

ディレクトリを作成するコマンド

何の略？　make directory

書式

```
mkdir [オプション] <作成するディレクトリ名>
```

具体例

```
# codeディレクトリを作成する
$ mkdir code
```

ディレクトリ名は日本語だと文字化けしたりするおそれがあるので、英語で作成するのがオススメ

深いディレクトリを一度に作成するオプション

-p

- 深いディレクトリを一度に作成する

```
$ mkdir -p test/2019/08
```

-pオプションを付けると、test, 2019 というディレクトリを事前に作らなくていいよ

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう　ここ
- パスについて学ぼう

rmdirコマンド

空のディレクトリを削除するコマンド

何の略？　remove directory

書式

```
rmdir <ディレクトリ名>
```

具体例

```
$ rmdir code
```

空でないディレクトリを削除しようとするとエラーになるよ。中にファイルが残っていることに気付かずにディレクトリを削除してしまう事故を防げるよ

Day2. ディレクトリを操作しよう

- Day2のゴール
- ディレクトリを操作してみよう
- ディレクトリの構造を学ぼう
- 【cdコマンド】ディレクトリを移動しよう
- 【pwdコマンド】カレントディレクトリを表示しよう
- 【lsコマンド】ディレクトリの中身を表示しよう
- 【mkdirコマンド】ディレクトリを作成しよう
- 【rmdirコマンド】ディレクトリを削除しよう
- パスについて学ぼう　ここ

パスってなに？

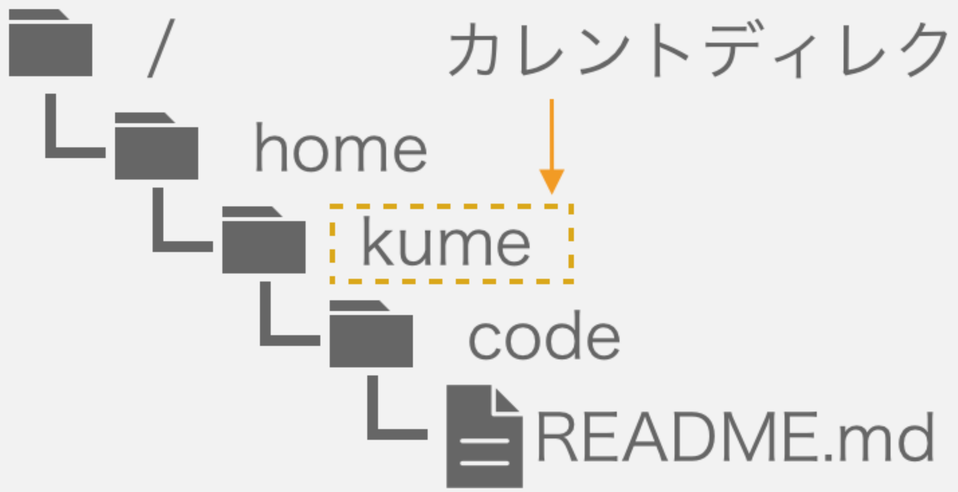
パスはディレクトリやファイルの住所情報。
ディレクトリの階層の区切りを「/」で表現する

kumeディレクトリはこう
表現できるよ

/home/kume/

絶対パスと相対パス

絶対パスはルートディレクトリ（「/」）から始まるパス。
相対パスはカレントディレクトリから始まるパス



絶対パス

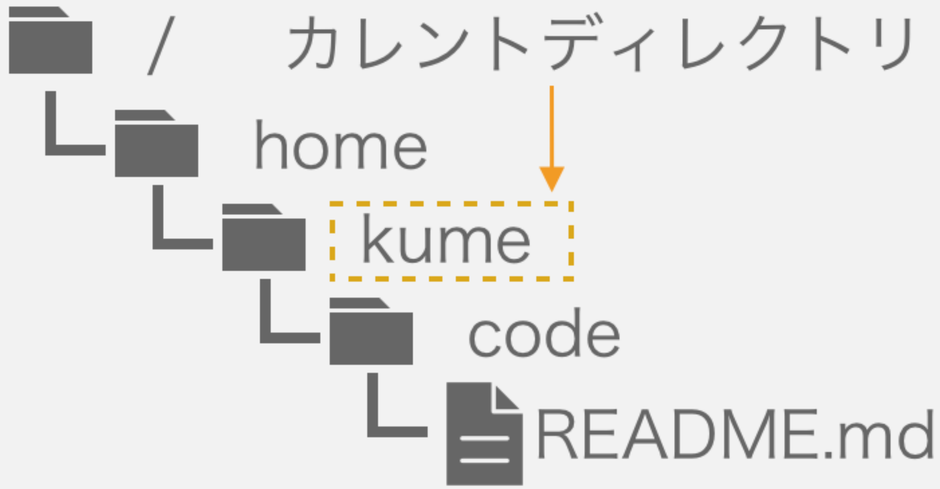
/home/kume/code/README.md

相対パス

code/README.md

特別なディレクトリ表記

「.」はカレントディレクトリを表す。
「 .. 」は親ディレクトリを表す



相対パス	絶対パス
../..	/
..	/home
.	/home/kume
code または ./code	/home/kume/code
code/work または ./code/work	/home/kume/code/README.md

まとめ

- パスはディレクトリやファイルの住所情報
- ディレクトリの階層の区切りを「/」で表現する
- 絶対パスはルートディレクトリ（「/」）から始まるパス
- 相対パスはカレントディレクトリから始まるパス
- 「.」はカレントディレクトリを表す
- 「 .. 」は親ディレクトリを表す

Day2のまとめ

- できるようになったこと
  - ディレクトリとパスの構造について理解した
  - cdコマンドでディレクトリを移動できるようになった
  - pwdコマンドでカレントディレクトリを確認できるようになった
  - lsコマンドでディレクトリの内容を表示できるようになった
  - mkdirコマンド・rmdirでディレクトリの作成・削除ができるようになった

Day2のまとめ

- やってみよう
  - ルートディレクトリに移動してみよう
  - ルートディレクトリの下にある任意のディレクトリの中身を
    隠しファイルも含めて表示しよう
  - 下記コマンドを実行後、test1、test2のディレクトリを削除してみよう

```
$ cd ~/work/
$ mkdir test1
$ ls
$ mkdir test1/test2
```

コラム：コマンドの使い方を調べよう

--helpオプションで、コマンドのヘルプメッセージを表示する

書式

```
<コマンド> --help
```

具体例

```
$ ls --help
使用法: ls [オプション]... [ファイル]...
```

使用方法、コマンドの概要、利用可能なオプション一覧を教えてくれるので、コマンドの概要を理解したい時に使うよ

コラム：コマンドの使い方を調べよう

manコマンドで、コマンドのマニュアルを表示する

何の略？　manual

書式

```
man <調べたいコマンド名>
```

具体例

```
$ man ls
NAME
        ls - list directory contents
$ man -k move # キーワードから検索もできる
```

--helpオプションより詳しいので、コマンドをきちんと理解したい時に使うよ

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

Day3. ファイルを操作しよう

- Day3のゴール　ここ
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

Day3のゴール

- Linuxのファイルの仕組みを理解すること
- コマンドを使ってファイルを操作できるようになること
  - 【catコマンド】ファイルの中身を表示しよう
  - 【lessコマンド】ファイルの中身をスクロールしよう
  - 【touchコマンド】ファイルを作成しよう
  - 【rmコマンド】ファイルを削除しよう
  - 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
  - 【cpコマンド】ファイル・ディレクトリをコピーしよう
  - 【lnコマンド】リンクを貼ろう
  - 【findコマンド】ファイルを検索しよう

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう　ここ
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

catコマンド

ファイルの中身を表示するコマンド

何の略？　concatenate

書式

```
cat [オプション] <ファイル名>
```

具体例

```
$ cat /etc/hosts
127.0.0.1    localhost
::1          localhost6
```

ファイルの内容を表示する時によく使われるのがcatコマンドだよ

オプションや使い方の注意

-n

- 行番号を表示するオプション

```
# ファイルの詳細情報を表示
$ cat -n /etc/hosts
1  127.0.0.1    localhost
2  ::1          localhost6
```

使い方

- 複数のファイルをまとめて表示することもできる

```
# 一度に複数のファイルを指定するとファイルの内容が順に表示される
$ cat /etc/hostname /etc/hosts
local.localdomain        ← /etc/hostname の内容
127.0.0.1    localhost
::1          localhost6  ← /etc/hosts の内容
```

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう　ここ
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

lessコマンド

ファイルの中身をスクロール表示するコマンド

何の略？

less - opposite of more

書式

```
less [オプション] <ファイル名>
```

具体例

```
$ less /etc/vimrc
if v:lang =~ ...
```

長いファイルの中身を見たい時に使うコマンドだよ

スクロールの操作

コマンド	内容
スペースキー, f	一画面下にスクロール
b	一画面上にスクロール
j	一行下にスクロール
k	一行上にスクロール
q	lessコマンドを終了

検索の操作

コマンド	内容
/<文字列>	下方向に向かって検索
?<文字列>	上方向に向かって検索
n	次の検索結果に移動
N	前の検索結果に移動

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう　ここ
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

テキストファイルってなに？

テキストファイルとは文字列が書かれたファイルのこと

詳細

- テキストファイルは文字コードで定められた文字だけが並んで格納されているファイル
- 文字コードとは、文字をコンピュータで処理したり通信したりするために，文字の種類に数値を割り振ったもの

コンピュータの中で文字は特定の数値の並びで表現されているよ

- 例：Aは1000001、Bは1000010、など
- Linuxではテキストファイルはとても大切。OSに関わるほとんどの設定情報をテキストファイルとして管理している

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう　ここ
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

バイナリファイルってなに？

バイナリファイルとは、画像ファイルや音声ファイル、実行ファイルなど

詳細

- 文字コード以外の数値も含むようなファイルをバイナリファイルと呼ぶ
- テキスト以外のファイルがバイナリファイルと考えてOK
- catコマンドなどでバイナリファイルが文字化けするのは、無理矢理テキストファイルとして解釈しようとして、文字コードに該当した数値をその文字に置き換えて表示したので意味不明の文字列になったため

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう　ここ
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

touchコマンド

空のファイルを作成するコマンド

何の略？　touch

書式

```
touch <新規ファイル名1> <新規ファイル名2> ...
```

具体例

```
$ touch README.md
```

ファイル名が存在していないときは空のファイルを作成して、
存在しているときはタイムスタンプを更新するよ

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう　ここ
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

rmコマンド

ファイル・ディレクトリを削除するコマンド

何の略？

remove

書式

```
rm [オプション] <削除するファイル名1>
<削除するファイル名2> ...
```

具体例

```
$ rm README.md
```

rmコマンドを実行すると、ごみ箱に行くのではなく、
本当に削除されるので気を付けよう

よく使うオプション

-r

- ディレクトリも合わせて削除する

```
$ rm -r dir
```

ディレクトリの中のファイルやディレクトリもまとめて
削除されるので気を付けよう

-f

- ファイルを削除する際に警告文を表示しない

```
$ rm -f file
```

-i

- ファイルの削除前に確認する

```
$ rm -i file
```

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう ここ
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

mvコマンド

ファイルの移動、ファイル名を変更するコマンド

何の略？

move

書式

```
mv [オプション] <移動元>... <移動先>
```

具体例

```
$ mkdir dir
$ touch file

# ファイル名を変更
$ mv file file1

# ファイルを移動
$ mv file1 dir/
```

オプション

-i

- 上書きする前に確認する

```
$ mv -i file file1
```

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう　ここ
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう

cpコマンド

ファイル・ディレクトリをコピーするコマンド

何の略？

copy

書式

```
cp [オプション] <コピー元>... <コピー先>
```

具体例

```
# ファイルをコピー
$ cp file new_file
# ファイルをディレクトリ内にコピー
$ cp file dir
```

コピー先のファイルがすでにあると上書きするので
気を付けよう

オプション

-i

- 上書きする前に確認する

```
$ cp -i file new_file
```

-r

- ディレクトリをコピーする

```
$ cp -r dir new_dir
```

Day3. ファイルを操作しよう

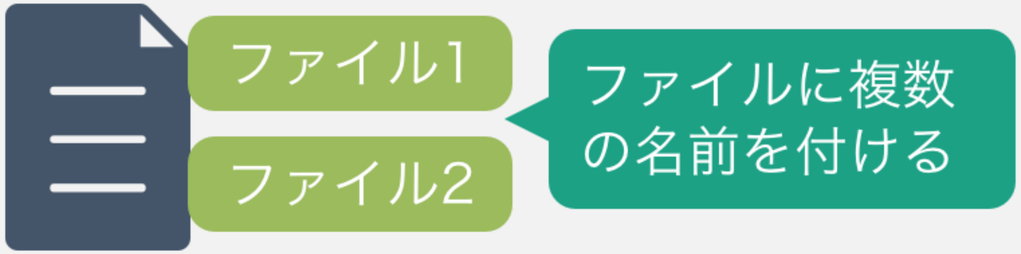
- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう　ここ
- 【findコマンド】ファイルを検索しよう

リンクを張るってどういうこと？

Linuxではファイルに別名を付けることができ、別名を付けることを「リンクを張る」という。Windowsで言うショートカットのようなもの

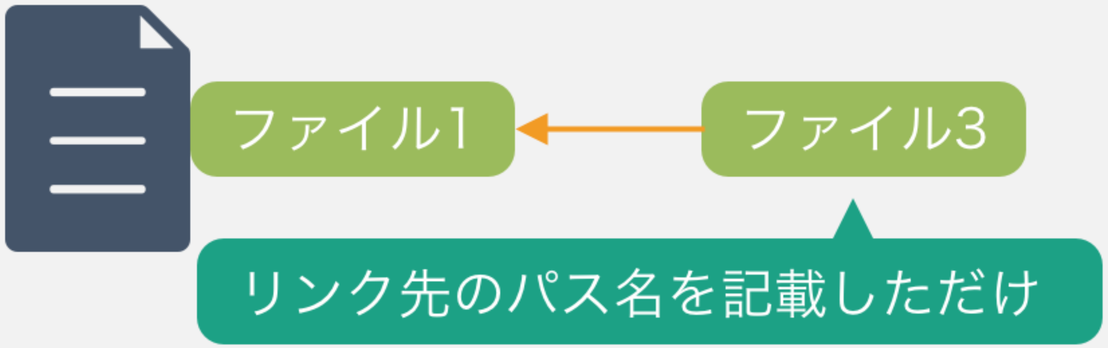
ハードリンク

- １つのファイルの実体に複数の名前を付ける機能
- 元のファイルを削除しても消えない
- すべてのハードリンクがなくなった時に削除される



シンボリックリンク

- リンク先のパス名が書かれた特殊ファイル。リンク先がファイルの実体
- シンボリックリンクを残したままファイルの実体を削除したり、ファイルを移動すると、ファイルを参照できなくなる



lnコマンド

リンクを張るコマンド

何の略？

link

書式

```
ln [オプション] <リンク元ファイル名> <リンク名>
```

具体例

```
# ハードリンクを張る
$ ln file1 file2
```

オプションを何も付けないとハードリンクになるよ

オプション

-s

- シンボリックリンクを張る

```
$ ln -s file1 file3
```

リンクをどういうときに使うの？

- 長いパス名を省略したいとき
- 複数バージョンを共存させ最新を区別したいとき

※ハードリンクとシンボリックリンクだと、シンボリックリンクの方が使うことが多い。ハードリンクはいくつか制限があって、異なるディスクをまたいで作成できなかったり、ディレクトリに対して使えない。シンボリックリンクはそのような制限を受けない。

Day3. ファイルを操作しよう

- Day3のゴール
- 【catコマンド】ファイルの中身を表示しよう
- 【lessコマンド】ファイルの中身をスクロールしよう
- テキストファイルについて学ぼう
- バイナリファイルについて学ぼう
- 【touchコマンド】ファイルを作成しよう
- 【rmコマンド】ファイルを削除しよう
- 【mvコマンド】ファイルの移動、ファイル名の変更をしよう
- 【cpコマンド】ファイル・ディレクトリをコピーしよう
- 【lnコマンド】リンクを貼ろう
- 【findコマンド】ファイルを検索しよう　ここ

findコマンド

ファイルを検索するコマンド

何の略？　find

書式

```
find <検索開始ディレクトリ> <検索条件> <アクション>
```

具体例

```
$ find . -name README.md -print
```

検索開始ディレクトリ　検索条件　検索アクション

./wordpress/.../README.md

ファイルがどこにあるか探す時findコマンドを使うと便利だよ

よく使われる検索条件

-name

- ファイル名を指定してファイルを検索。ファイル名の大文字小文字は区別する

```
$ find . -name README.md

# ワイルドカードが使える。* を使って指定する時は " で囲うこと
$ find . -name '*.html' -print
```

* は任意の文字列に一致するよ。
? も使えて、任意の一文字に一致するよ

-iname

- ファイル名を指定してファイルを検索。ファイル名の大文字小文字は区別しない

```
$ find . -iname readme.md
```

よく使われる検索条件

-type

- ファイルの種類で検索

```
# -type f は通常ファイル
$ find . -type f -print

# -type l はシンボリックリンク
$ find . -type l -print

# -type d はディレクトリ
$ find . -type d -print
```

-a

- 複数の検索条件を指定。なお、-aは省略可能

```
$ find . -type d -a -name images -print
```

Day3のまとめ

- できるようになったこと
  - テキストファイルとバイナリファイルについて理解した
  - ファイルの中身を表示できるようになった
  - ファイルを作成できるようになった
  - ファイルの削除ができるようになった
  - ファイルの移動ができるようになった
  - ファイルをコピーできるようになった
  - リンクを貼れるようになった
  - ファイルを検索できるようになった

Day3のまとめ

- やってみよう
  - index.htmlという空ファイルを作成してみよう
  - index.htmlの名前をhome.htmlに変更してみよう
  - publicというディレクトリを作成し、その中にpractice.txtというファイルを作成しよう
  - publicというディレクトリを作成し、その中にpractice.txtというファイルを作成しよう
  - findコマンドを使って上記で作成したpractice.txtファイルを見つけよう
  - Day3で作成した全ファイル・ディレクトリを削除しよう

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール　ここ
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

標準入出力のゴール

- 入出力の仕組みを理解すること
- 複数のコマンドを連携できるようになること
  - リダイレクトを使って標準入出力先を変更しよう
  - パイプラインを使って複数のコマンドを連携しよう

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう　ここ
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

標準入出力ってなに？

標準入出力というのは、プログラムが、特に何も指定されていない場合に
利用するデータの標準的な入出力元

標準入力、標準出力、標準エラー出力

標準入力、標準出力、標準エラー出力の3つを合わせて標準入出力と呼ぶ

標準入力
- プログラムの標準の入力
- キーボードが通常は使われる

標準出力
- プログラムの標準の出力
- ディスプレイが通常は使われる

標準エラー出力
- プログラムのエラーメッセージを出力する
- ディスプレイが通常は使われる

Linuxではコマンドの入出力先を抽象化することで、
入出力先を柔軟に変更できる

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう ここ
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

【>】リダイレクト

リダイレクトというのは入出力先を変更する機能

入力のリダイレクト

- キーボードの代わりにファイルから入力する機能

```
# /etc/hostsファイルを入力元にしてcatコマンドを実行

$ cat < /etc/hosts
1  127.0.0.1    localhost
2  ::1          localhost6
```

出力のリダイレクト

- コマンドの実行結果を画面に表示するのではなくファイルに保存する機能

```
# lsコマンドの出力先をoutput.txtにして保存

$ ls > output.txt
```

エラー出力のリダイレクト

出力とエラー出力は別で、エラー出力は「2>」を使う

標準入出力の数値

- 標準入出力などをOSが識別するために用いる識別子

入出力	数値
標準入力	0
標準出力	1
標準エラー出力	2

エラー出力のリダイレクト

- エラー結果を画面に表示するのではなくファイルに保存する機能

```
$ ls /hoge 2> error.txt
$ cat error.txt
ls:  /hoge にアクセスできません:
```

リダイレクトのその他の使用方法

出力とエラー出力をまとめる

- まとめる場合は、出力をリダイレクトした後に「2>&1」と書く

```
$ ls / /hoge > output.txt 2>&1
$ cat output.txt
ls:  /hoge にアクセスできません:
bin  boot  dev  etc  home  lib
```

リダイレクトで追記する

- 「>」でリダイレクト指定するとファイルを上書きする
- 「>>」でリダイレクト指定するとファイルに追記する

```
$ echo Hello! >> output.txt
```

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう　ここ
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

/dev/nullってなに？

入力先として指定しても何も返さず、出力先として指定してもデータは消え何も表示されない、スペシャルファイル

入力先として指定

- 入力が何もない状態になる
- コマンドのテストなどで入力を空にしたい時に使用する

```
$ cat < /dev/null
$
```

何も表示されない

出力先として指定

- 結果が何も表示されない
- 結果が大量にあり、結果を非表示にしたい時に使用する

```
$ ls / > /dev/null
$
```

何も表示されない

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう ここ
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

パイプラインってなに？

複数のコマンドを連携させる機能。
コマンドの標準出力を別のコマンドの標準入力につなげる

書式

```
コマンド | コマンド
```

具体例

```
$ ls /bin | less

$ ls /bin | grep systemd | less
```

コマンドを複数つなげることができるよ。
ちなみにgrepというのは指定した文字列で検索して該当するものを表示するコマンドで、結果を絞りたい時によく使うよ

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ　ここ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

標準入出力のまとめ

- できるようになったこと
  - 標準入力、標準出力、標準エラー出力について理解した
  - 標準入出力をファイルに変更できるようになった
  - 入出力先に /dev/null を指定できるようになった
  - パイプラインを使って複数のコマンドをつなげられるようになった
- やってみよう
  - wcコマンドは入力元の行数、語数、文字数を計算して左から順に出力します。wcコマンドを使って /bin ディレクトリの行数をカウントしてください
  - /etc ディレクトリに host という文字列を含む語の数をカウントしてください

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール　ここ
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

パーミッションのゴール

- ファイルの権限管理ができるようになり、システムを安全に管理・使用できるようになること
  - ファイルのオーナーとグループの概念を理解しよう
  - ファイルの権限を確認・設定できるようになろう
  - 管理者権限でコマンドを実行できるようになろう

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

ここ

ファイルのオーナーとグループ

ファイルには全てオーナー（所有者）とグループ（所有グループ）が設定れていて、権限管理されている

詳細

- 前提：Linuxではユーザーごとにアカウントを発行する
- オーナー：ファイルの所有者
- グループ：ファイルの所有グループ。ユーザーをグループに所属させ、グループに権限を与えることで、複数ユーザーの権限を一括で管理できるようにする

確認方法

- ls -l コマンドでファイルのオーナーとグループが確認できる

```
$ ls -l /etc/less
-rwxr-xr-x. 1 root root 158240 7月 31 2015 /bin/less
```

オーナー　グループ

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

ここ

パーミッション

ファイルやディレクトリを「誰がどのように操作できるか」を定めたのがパーミッション

確認方法

- ls -l コマンドでファイルやディレクトリのパーミッションが確認できる

```
$ ls -l /bin/less
-rwxr-xr-x. 1 root root 158240 7月 31 2015 /bin/less
```

ファイルタイプ

オーナーのパーミッション

グループのパーミッション

その他のユーザーのパーミッション

詳細

パーミッションの記号

記号	内容
r	read（読み取り）
w	write（書き込み）
x	execute（実行）

rwxr-xr-xだと

対象	読み取り	書き込み	実行
オーナー	○	○	○
グループに属するユーザー	○	×	○
その他のユーザー	○	×	○

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう ここ
- スーパーユーザーについて学ぼう
- パーミッションのまとめ

chmodコマンド

ファイルやディレクトリのパーミッションを設定する

何の略？

change mode

書式

```
chmod [ugoa] [+-=] [rwx] <ファイル名>
```

詳細

記号	内容
u	所有者の権限
g	グループの権限
o	その他のユーザーの権限
a	全ユーザーの権限

記号	内容
+	権限を追加
-	権限を削除
=	記述した権限にする

具体例

```
$ chmod u+r file
```

chmodコマンド（数値で指定）

書式

```
chmod <8進数の数値> <ファイル名>
```

詳細

パーミッション	数値
r（読み取り）	4
w（書き込み）	2
x（実行）	1

オーナー	グループ	その他
rwx	r-x	r-x
↓	↓	↓ 数値で置き換える
421	4 1	4 1
↓	↓	↓ 足し合わせる
7	5	5

具体例

```
$ chmod 755 file
```

通常の指定は相対指定、数値指定は絶対指定だよ

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう ここ
- パーミッションのまとめ

スーパーユーザーってなに？

管理者権限を持つ特別なユーザーがスーパーユーザー

背景

- システムの根幹に関わる作業が誰にでもできると危険
- そのため、システムの根幹に関わる作業はスーパーユーザー（root）だけができるようにしている

コマンド

- コマンドの前にsudoを付けるとスーパーユーザーとしてコマンドを実行

```
sudo <コマンド>
```

何の略？

switch user and do

具体例

```
$ sudo mkdir /storage
```

Day4. 標準入出力とパーミッションについて学ぼう

標準入出力

- 標準入出力のゴール
- 標準入出力について学ぼう
- 【>】リダイレクトについて学ぼう
- /dev/nullについて学ぼう
- 【|】パイプラインについて学ぼう
- 標準入出力のまとめ

パーミッション

- パーミッションのゴール
- ファイルのオーナーとグループについて学ぼう
- ファイルのパーミッションについて学ぼう
- 【chmodコマンド】ファイルモードを変更しよう
- スーパーユーザーについて学ぼう
- パーミッションのまとめ ここ

パーミッションのまとめ

- できるようになったこと
  - ファイルやディレクトリのオーナーとグループの概念を理解した
  - ファイルやディレクトリのパーミッションを確認できるようになった
  - ファイルやディレクトリのパーミッションを設定できるようになった
  - スーパーユーザー権限でコマンドを実行できるようになった
- やってみよう
  - ~/work/output.txt ファイルのパーミッションを rw-r--r--（オーナーは読み書きできてその他のユーザーは読み取りのみ）に変更してください
  - インストールしたgitをアンインストールしてください。
yumコマンドの使い方はmanで確認して、難しい場合は
ネットで検索してみてください

Day5. プロセス・ジョブを管理しよう

- Day5のゴール
- プロセスについて学ぼう
- 【psコマンド】プロセスを表示しよう
- ジョブについて学ぼう
- 【fg・bgコマンド】ジョブの状態を変更しよう
- 【killコマンド】プロセス・ジョブを終了しよう

Day5. プロセス・ジョブを管理しよう

- Day5のゴール　ここ
- プロセスについて学ぼう
- 【psコマンド】プロセスを表示しよう
- ジョブについて学ぼう
- 【fg・bgコマンド】ジョブの状態を変更しよう
- 【killコマンド】プロセス・ジョブを終了しよう

Day5のゴール

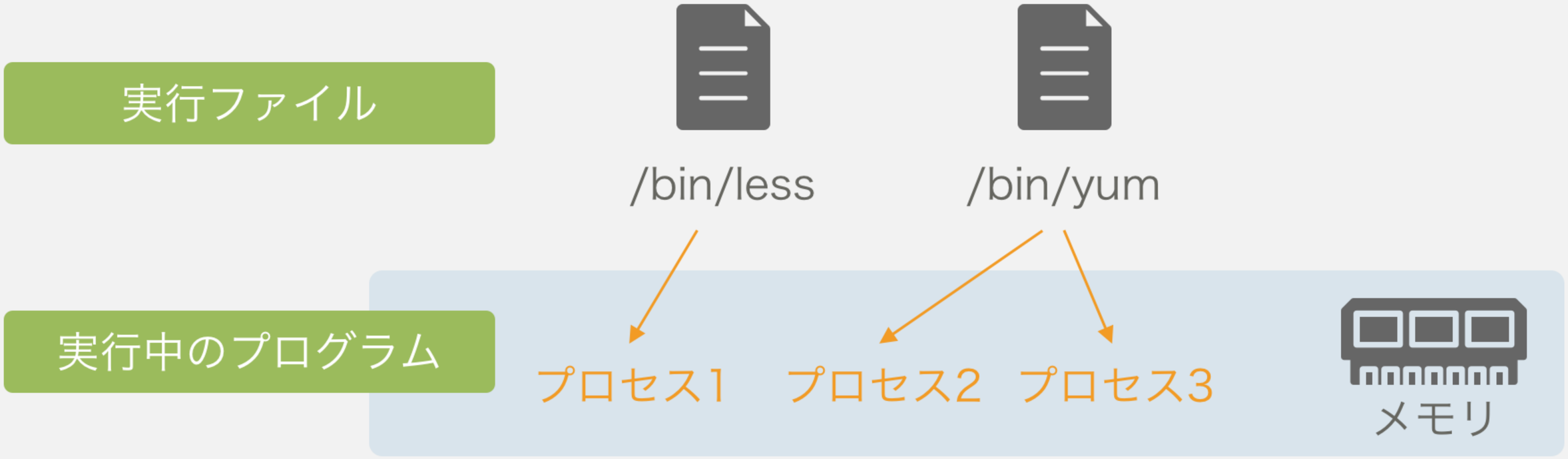
- プロセスの仕組みを理解すること
- 現在実行しているプロセスを確認できるようになること
- ジョブの仕組みを理解すること
- ジョブをバックグラウンドで実行できるようになること
- プロセスとジョブを終了できるようになること

Day5. プロセス・ジョブを管理しよう

- Day5のゴール
- プロセスについて学ぼう　ここ
- 【psコマンド】プロセスを表示しよう
- ジョブについて学ぼう
- 【fg・bgコマンド】ジョブの状態を変更しよう
- 【killコマンド】プロセス・ジョブを終了しよう

プロセスってなに？

実行中のプログラムのことをプロセスという



コマンドを実行すると、実行ファイルをメモリに格納して、メモリ上でCPUがプログラムを実行する。一つ実行すると新しいプロセスが一つ作られるよ

Day5. プロセス・ジョブを管理しよう

- Day5のゴール
- プロセスについて学ぼう
- 【psコマンド】プロセスを表示しよう　ここ
- ジョブについて学ぼう
- 【fg・bgコマンド】ジョブの状態を変更しよう
- 【killコマンド】プロセス・ジョブを終了しよう

psコマンド

実行中のプロセスを表示するコマンド

何の略？

process status

書式

```
ps [オプション]
```

具体例

```
$ ps
  PID   TTY          TIME CMD
 3935  pts/0    00:00:00 ps
20961  pts/0    00:00:00 bash
```

一行ずつが一つのプロセスだよ。PIDにプロセスIDが、CMDに実行中のコマンドが表示されているよ

オプション

aux

- 全ユーザーのプロセスを、詳細情報を合わせて表示

```
$ ps aux
USER       PID  %CPU %MEM     VSZ    RSS   TT  STAT
root       441  84.4 11.8  9259228 1972816   ??  S
root      1462  63.6  7.6  5816956 1267760   ??  R
```

オプションだけど - を付けなくていいよ。
xがpsコマンドを実行したユーザーの全てのプロセス、aが全てのユーザーのプロセス、uが詳細情報を合わせて表示するという意味

Day5. プロセス・ジョブを管理しよう

- Day5のゴール
- プロセスについて学ぼう
- 【psコマンド】プロセスを表示しよう
- ジョブについて学ぼう　ここ
- 【fg・bgコマンド】ジョブの状態を変更しよう
- 【killコマンド】プロセス・ジョブを終了しよう

ジョブってなに？

まとめて実行されるプロセスの集まりがジョブ。シェルごとに管理される

プロセスは3つ生成される

```
$ ps aux | grep bash | less
```

ジョブは一つ生成される

ジョブとプロセスの違い

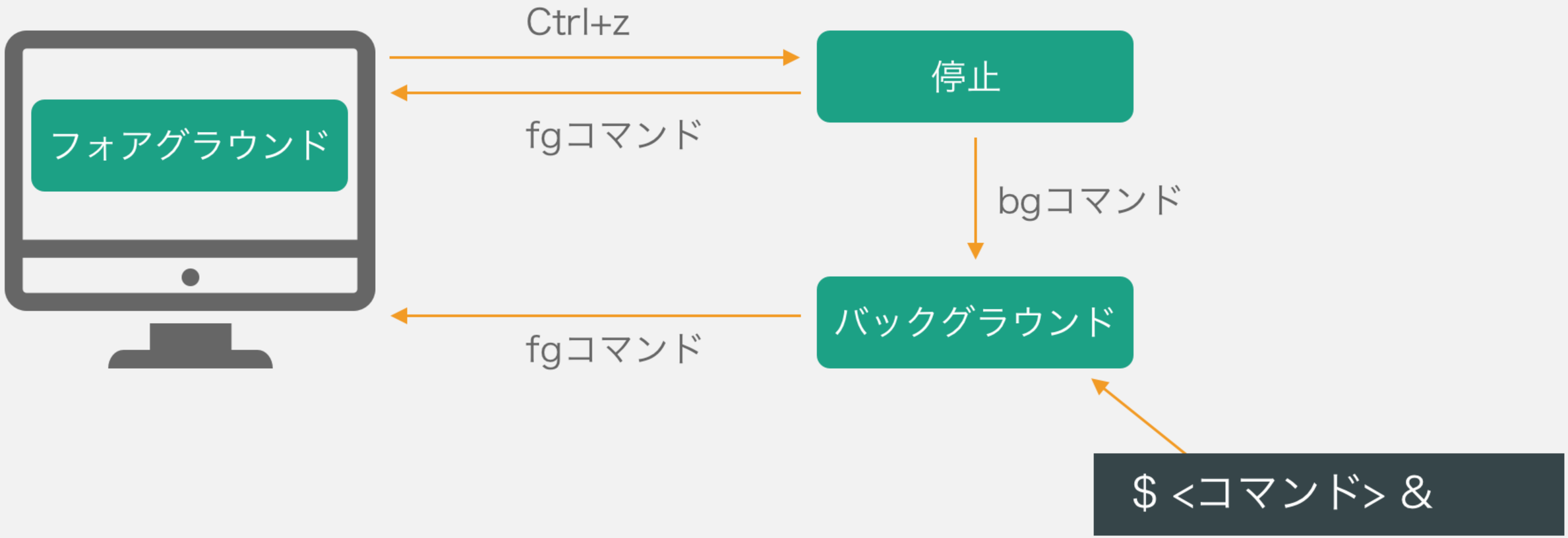
- プロセスはOS全体で統一して管理される。システムが自動で実行しているプログラムや他のユーザーが実行しているコマンドを一括して管理
- ジョブは、そのジョブを実行するシェルで管理される

Day5. プロセス・ジョブを管理しよう

- Day5のゴール
- プロセスについて学ぼう
- 【psコマンド】プロセスを表示しよう
- ジョブについて学ぼう
- 【fg・bgコマンド】ジョブの状態を変更しよう　ここ
- 【killコマンド】プロセス・ジョブを終了しよう

バックグラウンドとフォアグラウンド

シェルの画面外でコマンドを実行することをバックグラウンド、
シェルの画面内で実行することをフォアグラウンドと呼ぶ



jobsコマンド

現在のジョブの一覧を表示するコマンド

何の略？　jobs

書式

```
jobs
```

具体例

```
$ jobs
[1]+    実行中    sleep 10 &
```

ジョブID、実行状況、実行コマンドを表示するよ

fgコマンド

ジョブをフォアグラウンドにするコマンド

何の略？　foreground

書式

```
fg %<ジョブID>
```

具体例

```
$ fg %1
```

ジョブをフォアグラウンドに戻して対話的に操作したい時に使うよ

bgコマンド

ジョブをバックグラウンドにするコマンド

何の略？　background

書式

```
bg %<ジョブID>
```

具体例

```
$ bg %1
```

長い時間がかかる処理をバックグラウンドで実行することで、
コマンドが終了するのを待たなくてよくなるよ

Day5. プロセス・ジョブを管理しよう

- Day5のゴール
- プロセスについて学ぼう
- 【psコマンド】プロセスを表示しよう
- ジョブについて学ぼう
- 【fg・bgコマンド】ジョブの状態を変更しよう
- 【killコマンド】プロセス・ジョブを終了しよう ここ

killコマンド

ジョブやプロセスを終了させるコマンド

何の略？

kill process

書式

```
kill [オプション] %<ジョブID>
kill [オプション] <プロセスID>
```

具体例

```
$ kill %1
$ kill 4965
```

フォアグラウンドのジョブは Ctrl+c で終了できるよ。
停止中やバックグラウンドのジョブはkillコマンドで終了できるよ

補足

補足

- 正確には、killコマンドはシグナルを送信するコマンド

```
$ kill -<シグナル名> <プロセスID>
```

- デフォルトでは、TERMという、終了を指示するシグナルを送信している

```
# この2つのコマンドは同じ
$ kill 4965
$ kill -TERM 4965
```

時々使う
シグナル

- 時々使うのは SIGKILL という強制終了のシグナル

```
# この2つのコマンドは同じ
$ kill -SIGKILL 4965
$ kill -9 4965
```

Day5のまとめ

- できるようになったこと
  - プロセスとジョブについて理解した
  - プロセスを確認できるようになった
  - ジョブをバックグラウンド・フォアグラウンドで実行できるようになった
  - ジョブやプロセスを終了できるようになった

- やってみよう
  - yes > /dev/null & コマンドを実行してCPU負荷をかけてみよう
  - 上記のyesコマンドのプロセスIDを調べよう
  - 上記のyesコマンドを終了しよう

本講座のまとめ

- できるようになったこと
  - Linuxの必須コマンドを使えるようになった
  - Linuxの仕組みについて理解した
  - Linuxの考え方が身に付いた
- 今後に向けて
  - 普段からターミナルを使おう
  - LinuCレベル1もしくはLPIC-1を受講しよう
  - 「UNIXという考え方」を読もう
  - Linuxをパソコンにインストールしてみよう