
プログラミング以前

プログラミングとは

> 概要 **プログラミング**(programming)とは、**コンピュータ**に意図した動作を**行**わせるために、まとまった処理手順を作成し、与えること。作成された手順のことを**コンピュータプログラム**(**computer program**)あるいは単に**プログラム**という。プログラミングを**行**う人や職種のことを**プログラマ**(**programmer**)という。

引用

先輩やスタッフは当然のようにプログラミングをしているが、プログラミングを始める前にある程度の知識を身につける必要がある。ここでは

- プログラミング言語
- ファイルとディレクトリ
- パス
- テキストファイルの作成・編集

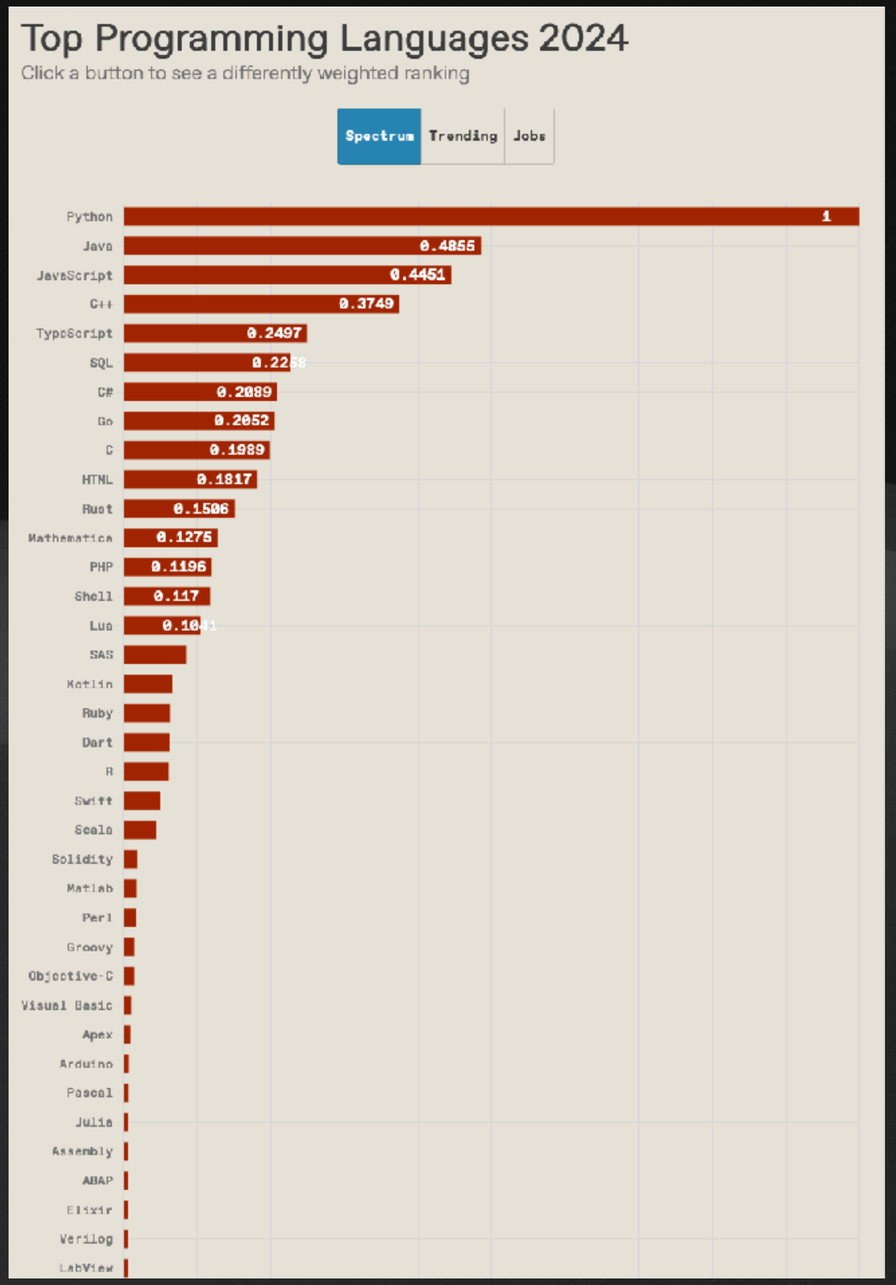
を扱う。

プログラミング言語

素粒子原子核実験分野では

- C/C++：データ解析、計算
- Shell スクリプト：Linux 操作
- SQL：データベース
- Fortran：データ解析、計算
- HTML：ホームページ
- TeX, LaTeX：論文作成
- PHP：ホームページ、計算？

といった言語がよく使われる



言語ごとの使用率

プログラミング言語

世間一般では

転職における言語別の平均年収ランキング

paiza

2023年順位	言語		2022年との平均年収増減
1位	Go	710.5万円	+51.5万円
2位	TypeScript	690.4万円	+23.3万円
3位	Scala	667.9万円	-15.0万円
4位	Kotlin	664.9万円	+20.8万円
5位	Ruby	663.5万円	+25.2万円
6位	Python	652.9万円	+8.2万円
7位	Swift	641.6万円	+25.9万円
8位	Sass	636.0万円	+54.8万円
9位	JavaScript	630.5万円	+31.1万円
10位	Perl	619.1万円	+91.5万円
11位	PHP	617.4万円	+37.0万円
12位	Java	605.5万円	+15.7万円
13位	Objective-C	601.7万円	+23.7万円
14位	C++	599.2万円	-10.3万円
15位	C#	589.1万円	+11.2万円
16位	C	579.1万円	+9.5万円
17位	Visual Basic (VB.NET)	537.6万円	+21.0万円

転職で企業からニーズが高い言語ランキング

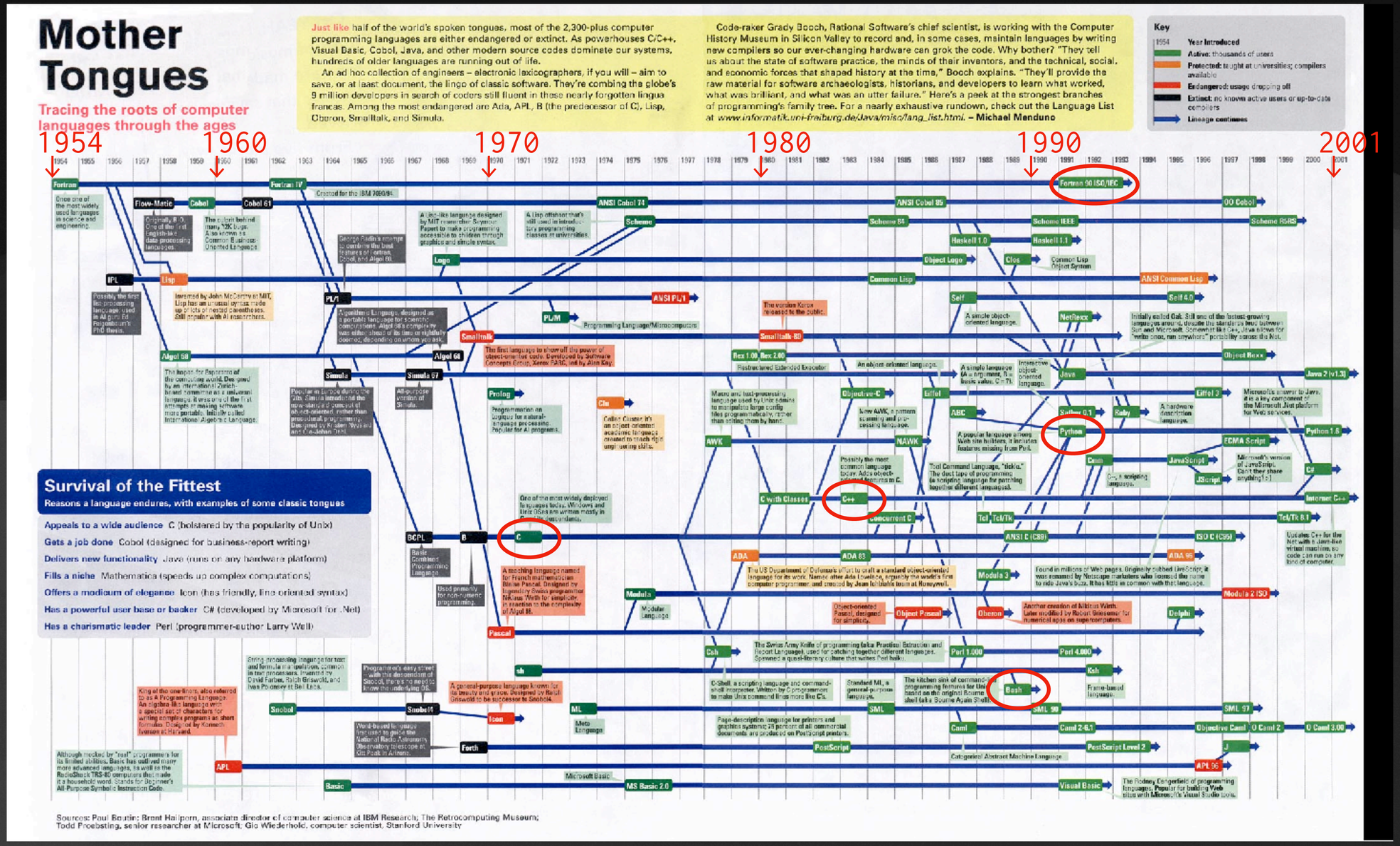
paiza

2023年順位	言語	言語別求人数比率	2022年との順位比較
1位	JavaScript	14.4%	→ 0
2位	Java	12.8%	→ 0
3位	PHP	11.8%	→ 0
4位	Python	9.6%	↑ +1
5位	TypeScript	8.8%	↑ +3
6位	C#	8.3%	↓ -2
7位	Ruby	6.0%	↓ -1
8位	C++	5.1%	↓ -1
9位	C	4.2%	↑ +1
10位	Kotlin	4.2%	↑ +3
11位	Swift	4.1%	↓ -2
12位	Go	4.0%	→ 0
13位	Visual Basic (VB.NET)	2.6%	↑ +1
14位	Objective-C	1.8%	↓ -3
15位	Sass	0.7%	→ 0
16位	Perl	0.7%	→ 0
17位	Scala	0.7%	→ 0

素粒子原子核実験とはやや趣が異なる

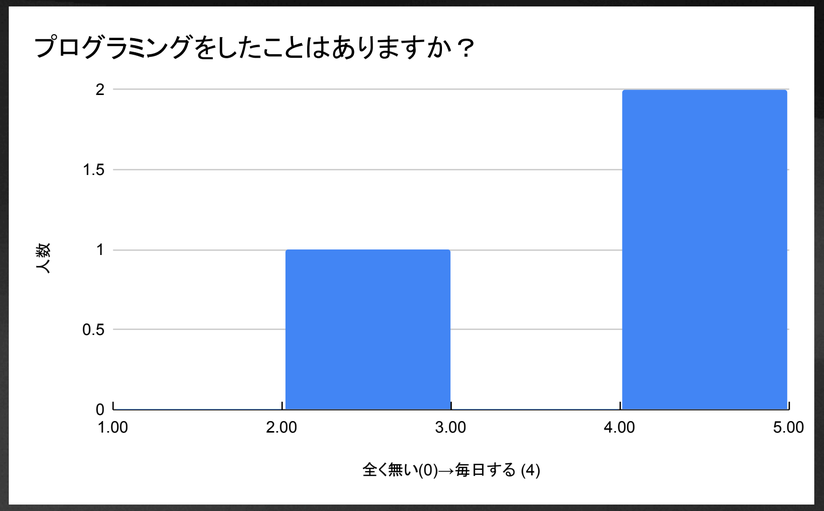
引用

プログラミング言語



引用

プログラミングやったことありますか？



ファイルとディレクトリ

引用

ファイル【file】

概要 **ファイル**（file）とは、**コンピュータ**における**データ**の管理単位の一つで、**ストレージ装置**（**外部記憶装置**）などに**データ**を記録する際に**利用者**や**オペレーティングシステム**（**OS**）から見て最小の記録単位となる**データ**のまとまり。

- ファイルにはいろいろな種類がある
  - PDF ファイル
  - Word
  - 画像（jpg, png, 等など）
  - テキスト
  - 等など
- ファイル名にある最後の．とそれ以降の文字列を拡張子といい、種類の識別に使われる
- プログラミングではテキストファイルをよく使う

ファイルとディレクトリ

引用

ファイル【file】

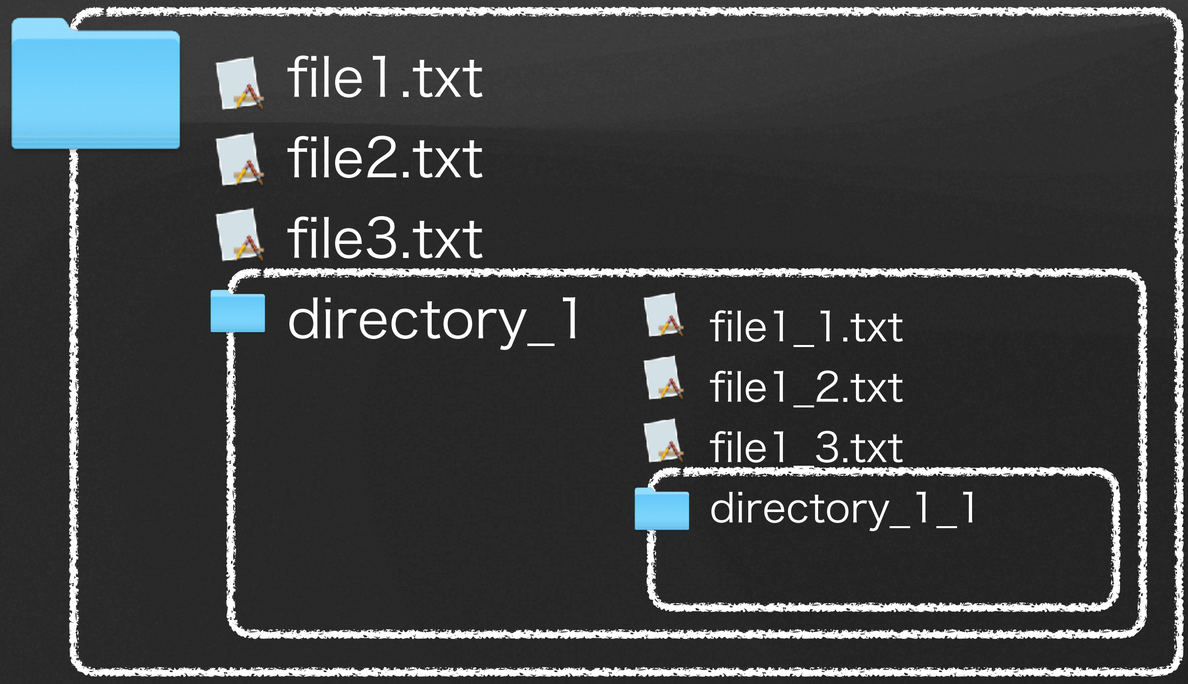
概要 **ファイル**(file)とは、**コンピュータ**における**データ**の管理単位の一つで、**ストレージ装置**(**外部記憶装置**)などに**データ**を記録する際に**利用者**や**オペレーティングシステム**(**OS**)から見て最小の記録単位となる**データ**のまとまり。

引用

ディレクトリ【directory】

概要 **ディレクトリ**(directory)とは、電話帳(phone～)、住所録、名鑑、要覧、指導書、規則集などの意味を持つ英単語。**IT**関連では、多数の対象をその所在などの**情報**と共に一覧できるよう整理したものを意味することが多い。

- ファイルにはいろいろな種類がある
  - PDF ファイル
  - Word
  - 画像（jpg, png, 等など）
  - テキスト
  - 等など
- ファイル名にある最後の．とそれ以降の文字列を拡張子といい、種類の識別に使われる
- プログラミングではテキストファイルをよく使う



ディレクトリは階層構造をとる

パス (Path)

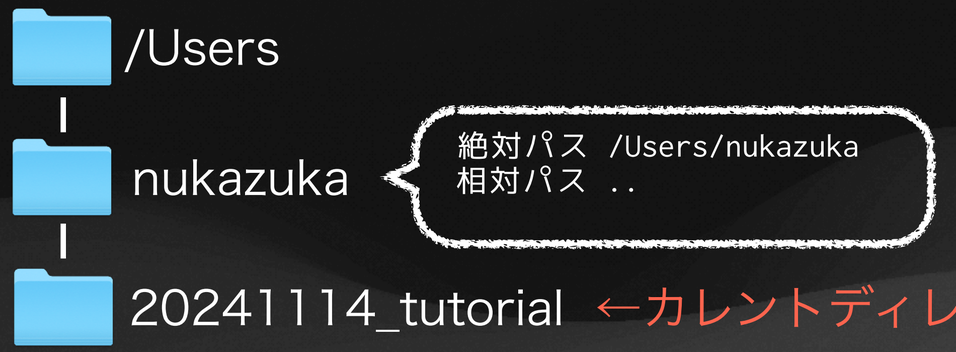
ファイルやパスの場所を示す

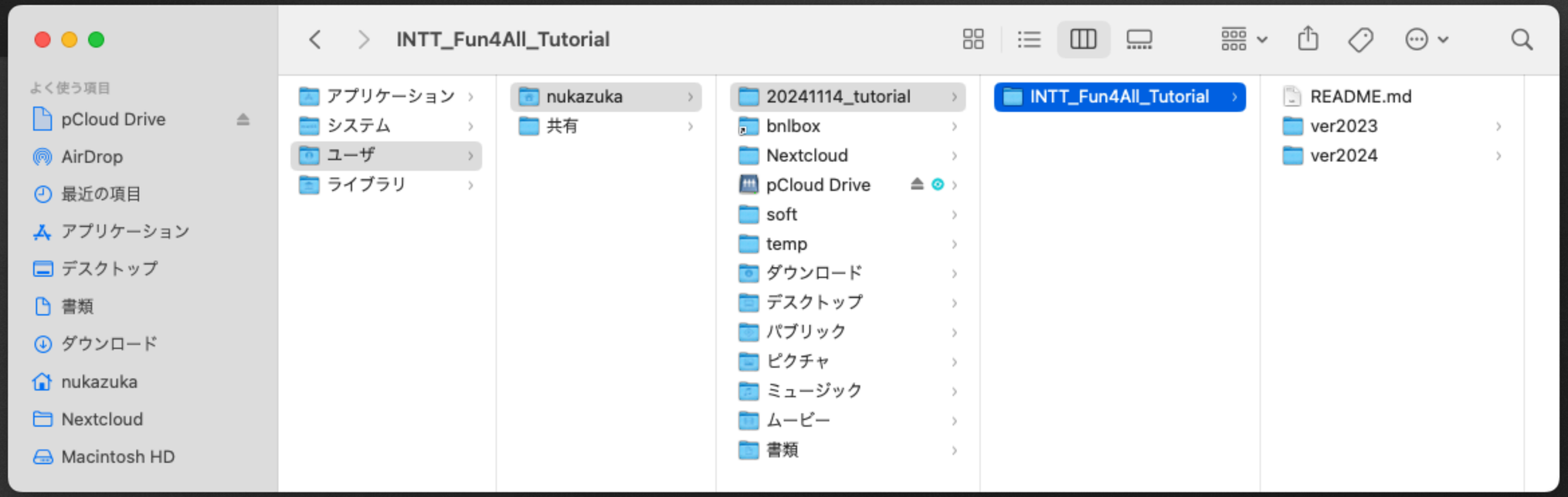
- 絶対パス: Root ディレクトリから見たパス
- 相対パス：カレントディレクトリから見たパス

※ カレントディレクトリの相対パスは “.”

※ カレントディレクトリの一つ上の階層のディレクトリの相対パスは “..”

例：糠塚 Mac の Fun4All チュートリアルのサンプル





パス (Path)

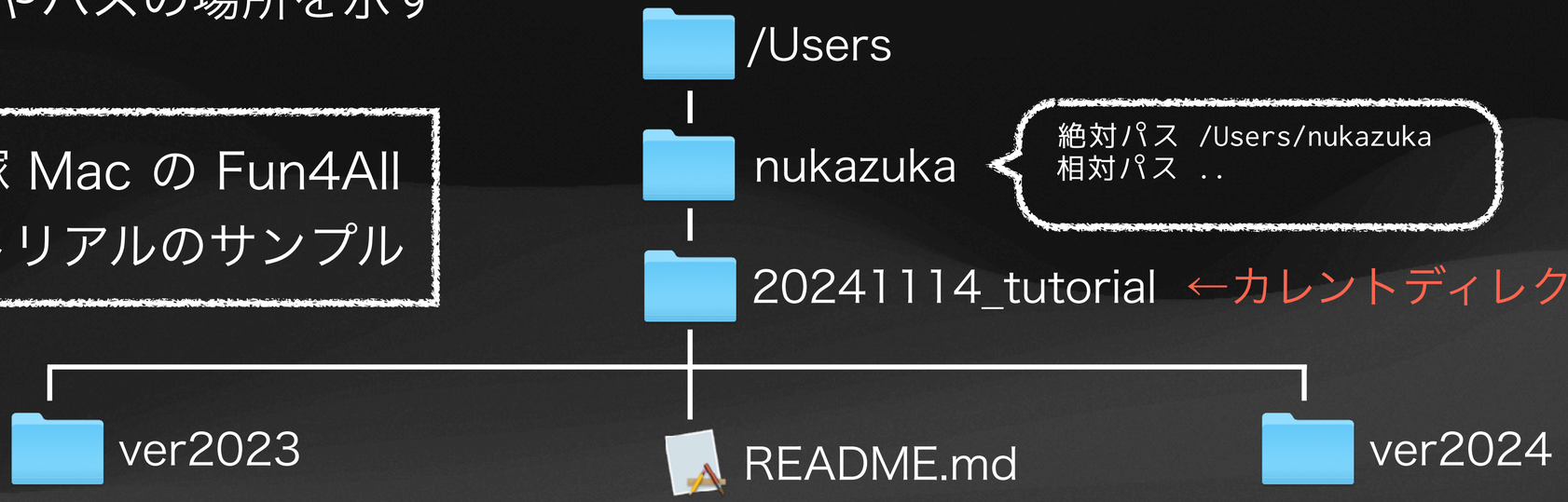
ファイルやパスの場所を示す

- 絶対パス: Root ディレクトリから見たパス
- 相対パス：カレントディレクトリから見たパス

※ カレントディレクトリの相対パスは “.”

※ カレントディレクトリの一つ上の階層のディレクトリの相対パスは “..”

例：糠塚 Mac の Fun4All チュートリアルのサンプル



パス (Path)

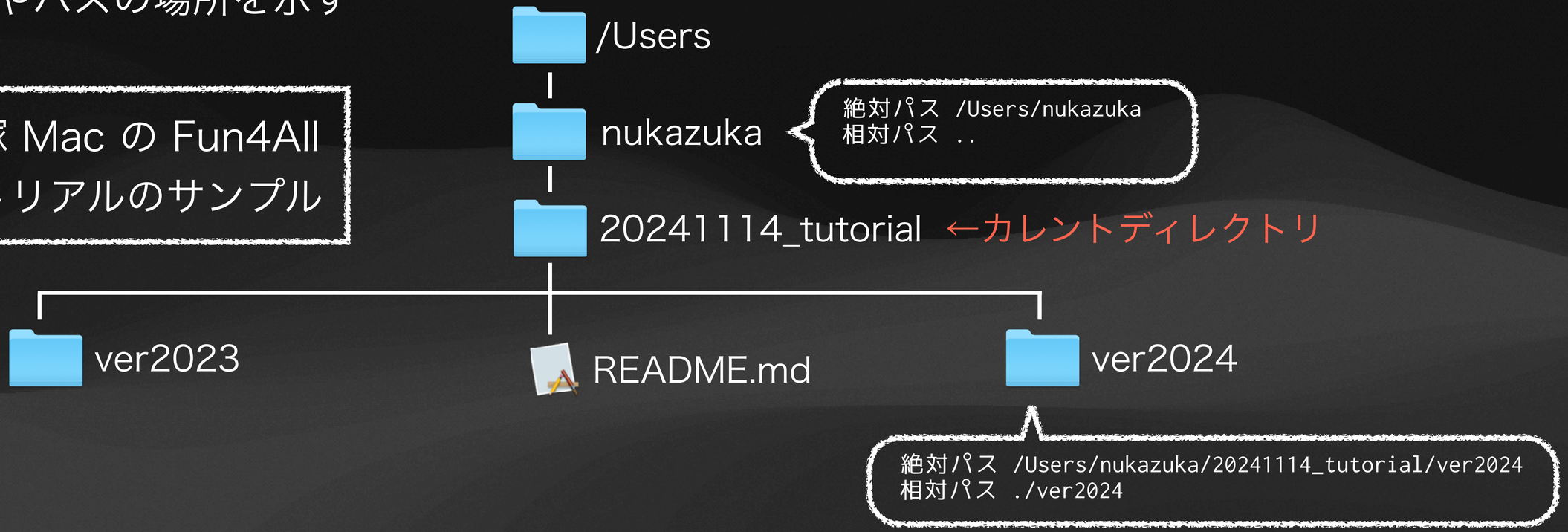
ファイルやパスの場所を示す

- 絶対パス: Root ディレクトリから見たパス
- 相対パス：カレントディレクトリから見たパス

※ カレントディレクトリの相対パスは “.”

※ カレントディレクトリの一つ上の階層のディレクトリの相対パスは “..”

例：糠塚 Mac の Fun4All チュートリアルのサンプル



パス (Path)

ファイルやパスの場所を示す

- 絶対パス: Root ディレクトリから見たパス
- 相対パス：カレントディレクトリから見たパス

※ カレントディレクトリの相対パスは “.”

※ カレントディレクトリの一つ上の階層のディレクトリの相対パスは “..”

例：糠塚 Mac の Fun4All チュートリアルのサンプル

パス (Path)

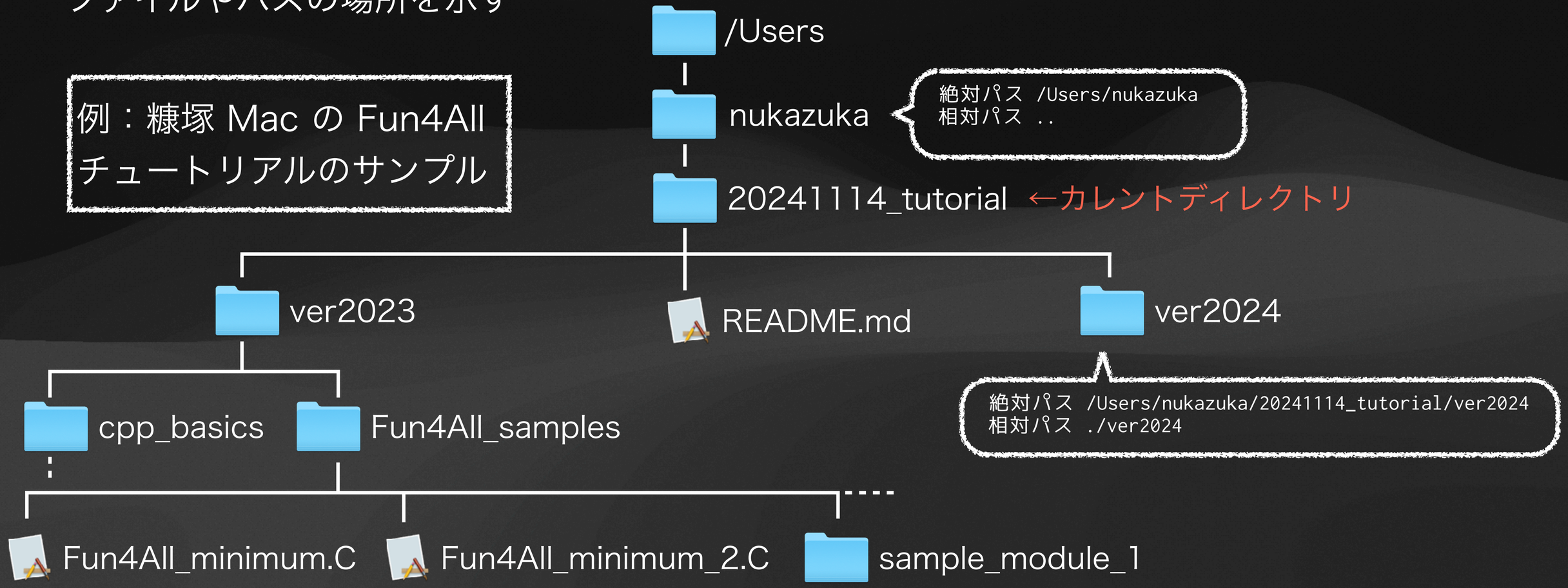
ファイルやパスの場所を示す

- 絶対パス: Root ディレクトリから見たパス
- 相対パス：カレントディレクトリから見たパス

※ カレントディレクトリの相対パスは “.”

※ カレントディレクトリの一つ上の階層のディレクトリの相対パスは “..”

例：糠塚 Mac の Fun4All チュートリアルのサンプル



10 分

パス (Path)

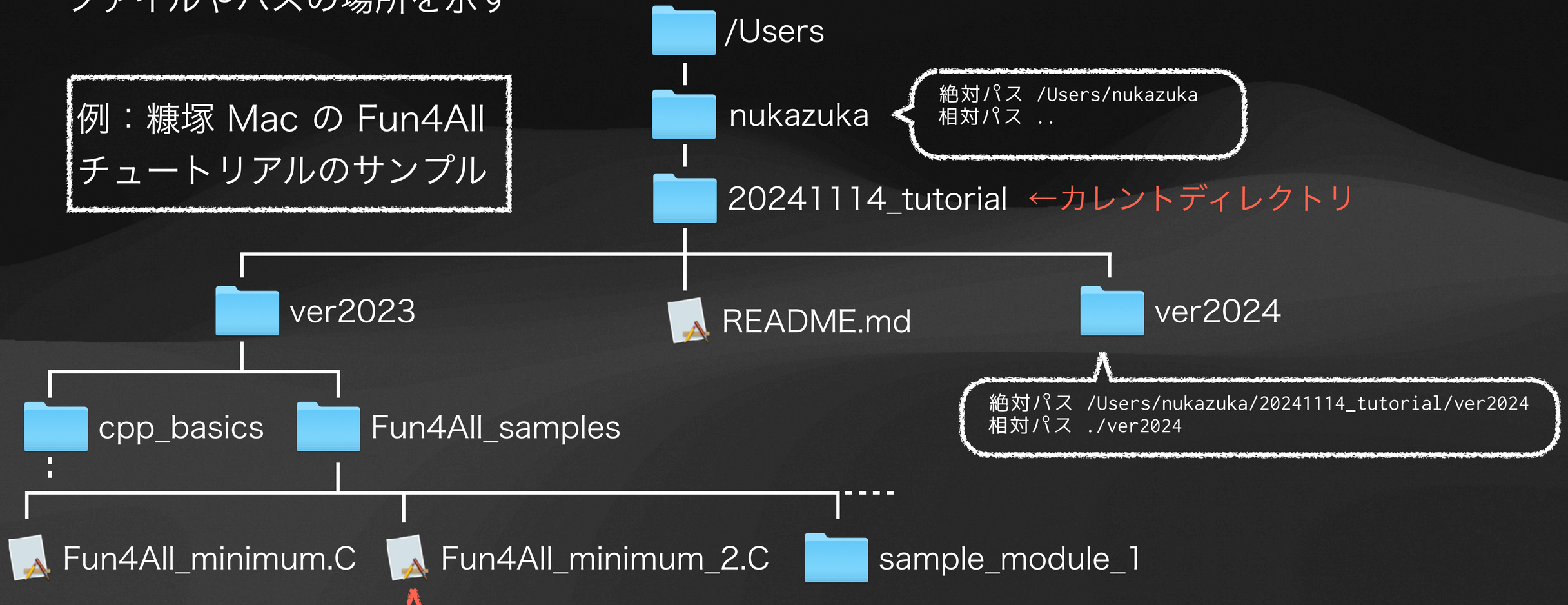
ファイルやパスの場所を示す

- 絶対パス: Root ディレクトリから見たパス
- 相対パス：カレントディレクトリから見たパス

※ カレントディレクトリの相対パスは “.”

※ カレントディレクトリの一つ上の階層のディレクトリの相対パスは “..”

例：糠塚 Mac の Fun4All チュートリアルのサンプル



どう書くでしょうか？

Linux で CUI （再び）：テキストファイルをいじってみる

- とりあえずテキストファイルを作成・編集・保存してみよう
- テキストエディタを使う

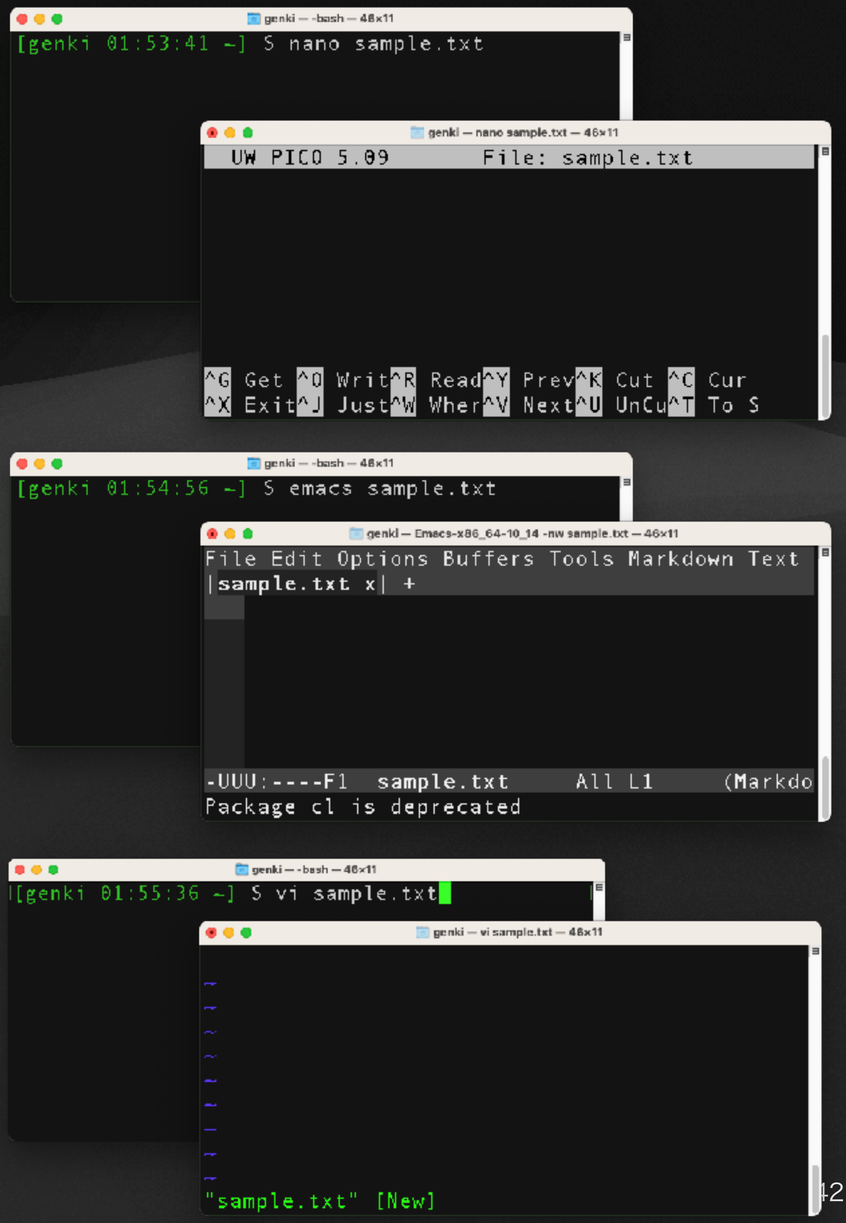
Linux で CUI （再び）：テキストファイルをいじってみる

- とりあえずテキストファイルを作成・編集・保存してみよう
- テキストエディタを使う
  - 好きなエディタ* を選び、ターミナルから起動してみましょう。
    - vi or vim (CUI): Linux/Unix (*nix と書く) 系 OS では必ず標準搭載されている有名なエディタ。蜂谷さんは vi 派。
    - Emacs (GUI): 「エディタではなく環境」という人もいるほど何でもできるが、起動がちょっと遅い。vi よりシェアが低い。中川さんは emacs 派。
    - nano (CUI): 大抵の Linux, Mac にデフォルトで標準搭載されている。初心者にはとっつきやすいが、中上級者でメインとして使っている人はあまり見ない
    - Visual Studio Code (VS code) (GUI): 現在の主流？ちゃんと設定すれば sPHENIX でも問題なく使えるが、おっさんは設定の手伝いができないかも。

* エディタ戦争

Linux で CUI （再び）：テキストファイルをいじってみる

- とりあえずテキストファイルを作成・編集・保存してみよう
- テキストエディタを使う
  - 好きなエディタ* を選び、ターミナルから起動してみましょう。
    - vi or vim (CUI): Linux/Unix (*nix と書く) 系 OS では必ず標準搭載されている有名なエディタ。蜂谷さんは vi 派。
    - Emacs (GUI): 「エディタではなく環境」という人もいるほど何でもできるが、起動がちょっと遅い。vi よりシェアが低い。中川さんは emacs 派。
    - nano (CUI): 大抵の Linux, Mac にデフォルトで標準搭載されている。初心者にはとっつきやすいが、中上級者でメインとして使っている人はあまり見ない
    - Visual Studio Code (VS code) (GUI): 現在の主流？ちゃんと設定すれば sPHENIX でも問題なく使えるが、おっさんは設定の手伝いができないかも。

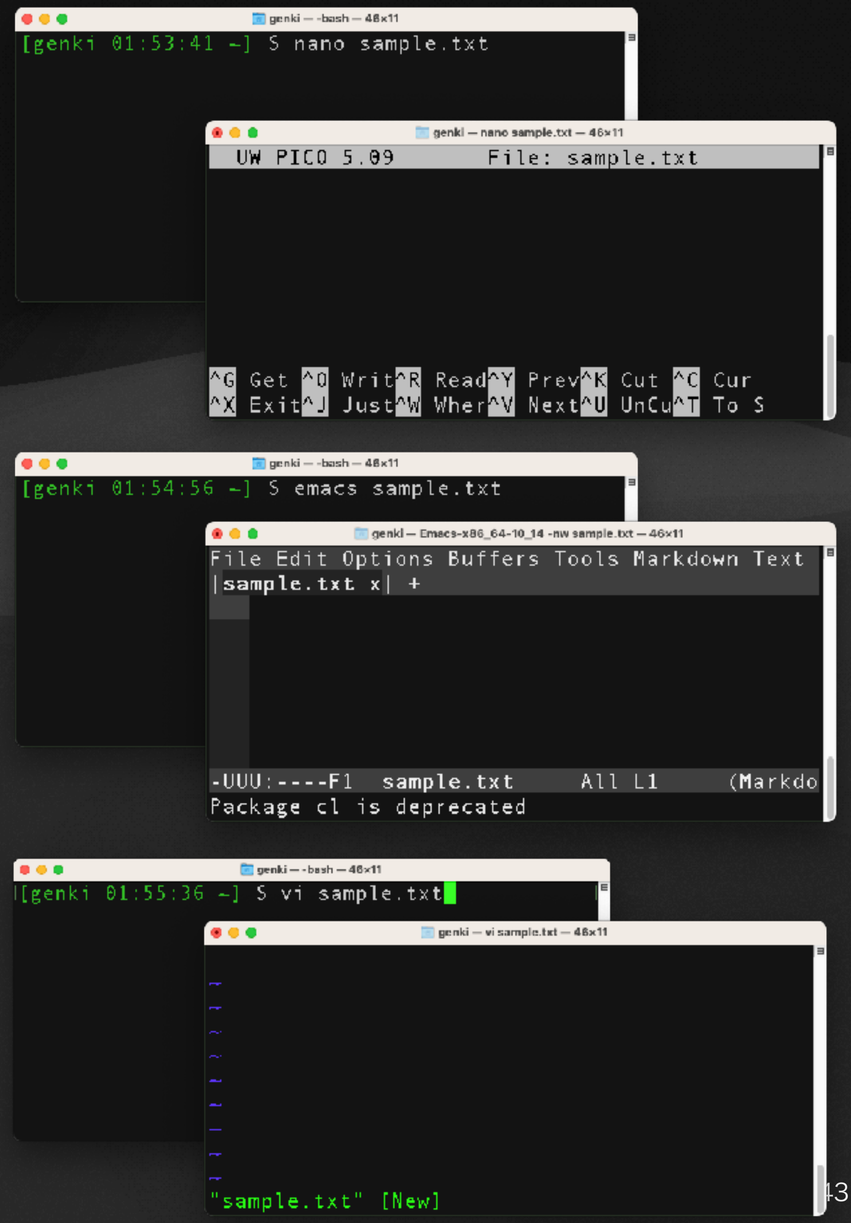


* エディタ戦争

10 分

Linux で CUI （再び）：テキストファイルをいじってみる

- とりあえずテキストファイルを作成・編集・保存してみよう
- テキストエディタを使う
  - 好きなエディタ* を選び、ターミナルから起動してみましょう。
    - vi or vim (CUI): Linux/Unix (*nix と書く) 系 OS では必ず標準搭載されている有名なエディタ。蜂谷さんは vi 派。
    - Emacs (GUI): 「エディタではなく環境」という人もいるほど何でもできるが、起動がちょっと遅い。vi よりシェアが低い。中川さんは emacs 派。
    - nano (CUI): 大抵の Linux, Mac にデフォルトで標準搭載されている。初心者にはとっつきやすいが、中上級者でメインとして使っている人はあまり見ない
    - Visual Studio Code (VS code) (GUI): 現在の主流？ちゃんと設定すれば sPHENIX でも問題なく使えるが、おっさんは設定の手伝いができないかも。



- **sample.txt という名前のファイルを作成し、ファイルに自分の名前をアルファベットで書いて保存してみよう**

* エディタ戦争