



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

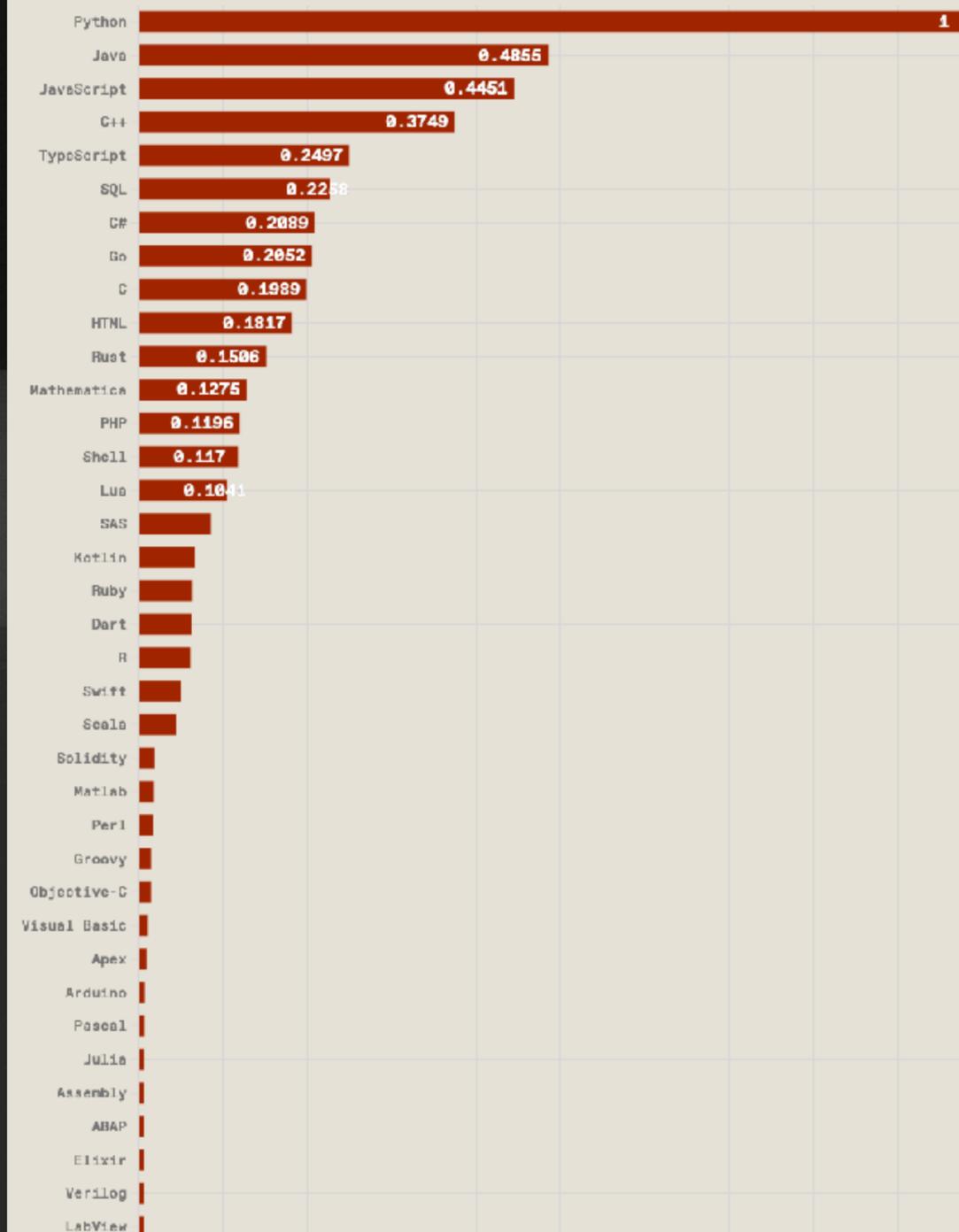
- C/C++：データ解析、計算
- Shell スクリプト：Linux 操作
- SQL：データベース
- Fortran：データ解析、計算
- HTML：ホームページ
- TeX, LaTeX：論文作成
- PHP：ホームページ、計算？

といった言語がよく使われる

## Top Programming Languages 2024

Click a button to see a differently weighted ranking

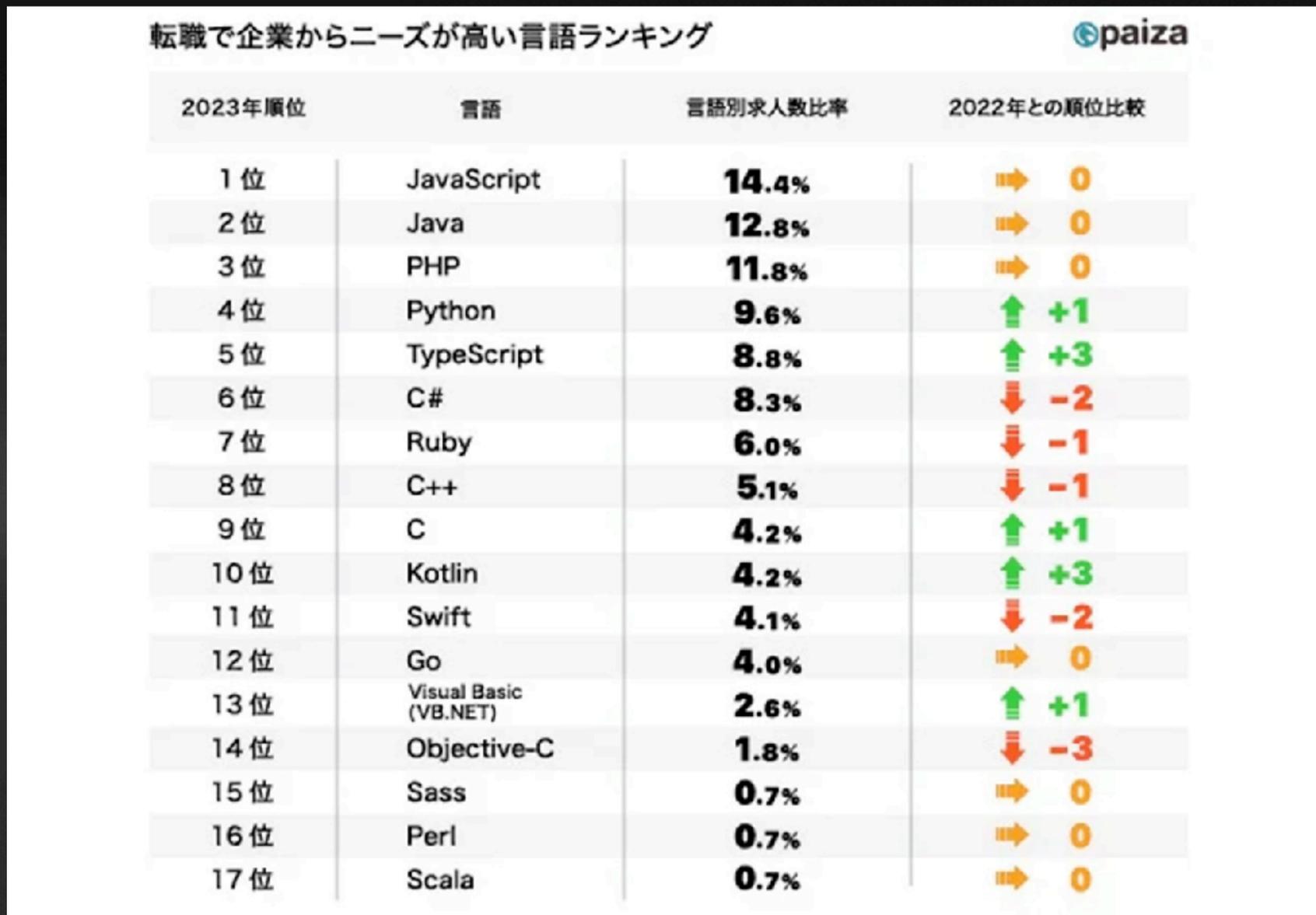
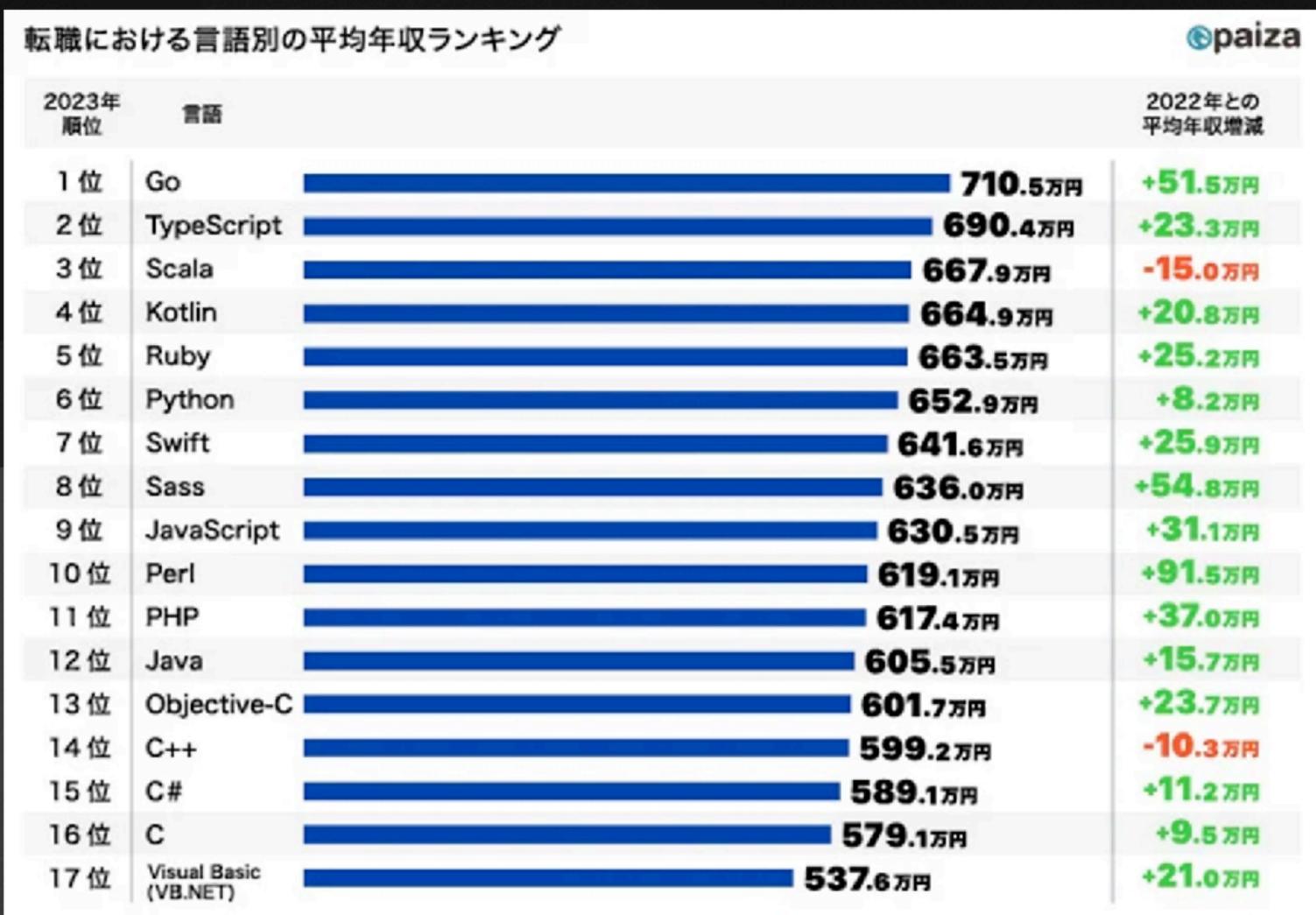
Spectrum Trending Jobs



言語ごとの使用率

# プログラミング言語

世間一般では

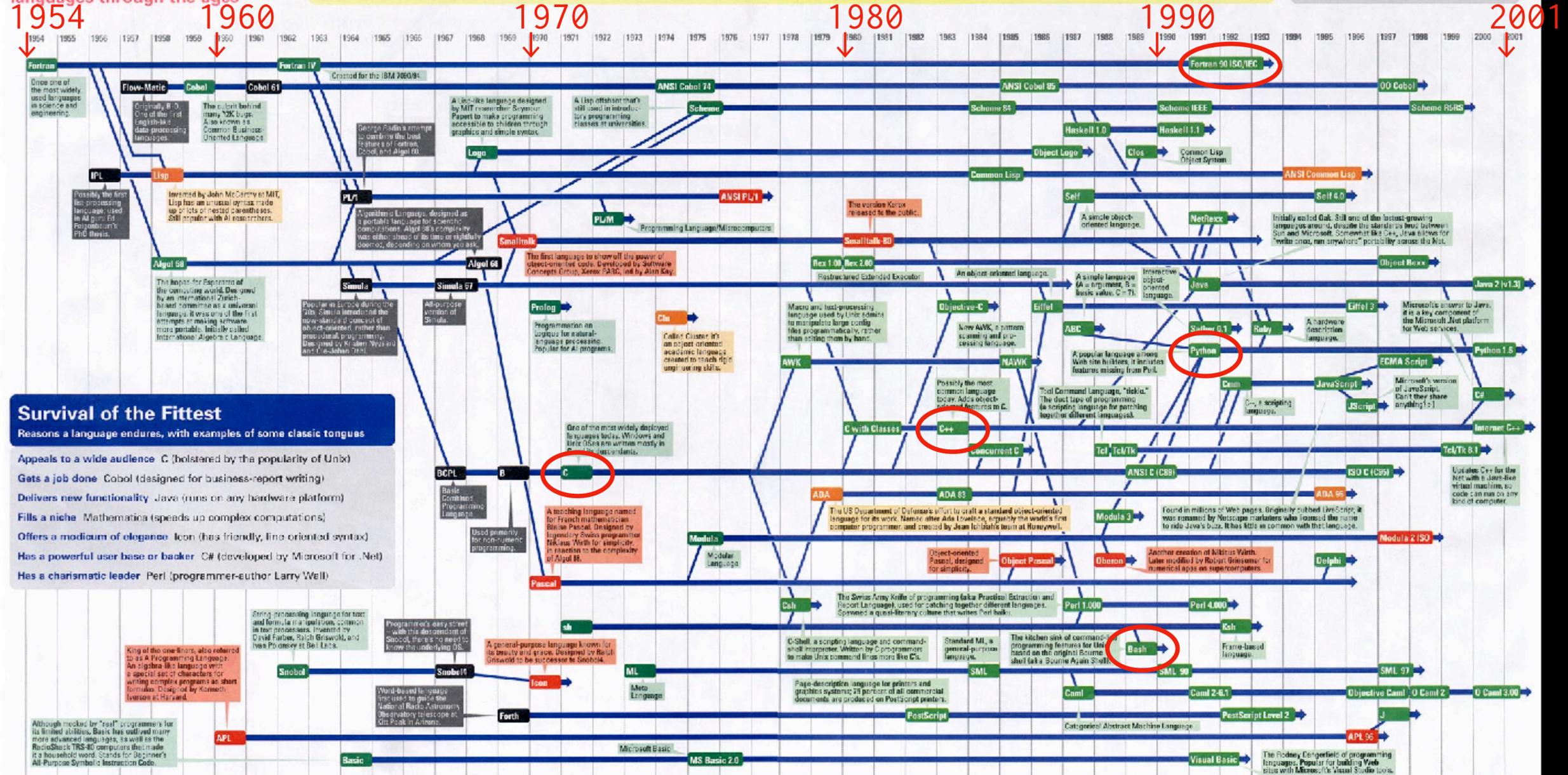


素粒子原子核実験とはやや趣が異なる

# プログラミング言語

## Mother Tongues

Tracing the roots of computer languages through the ages



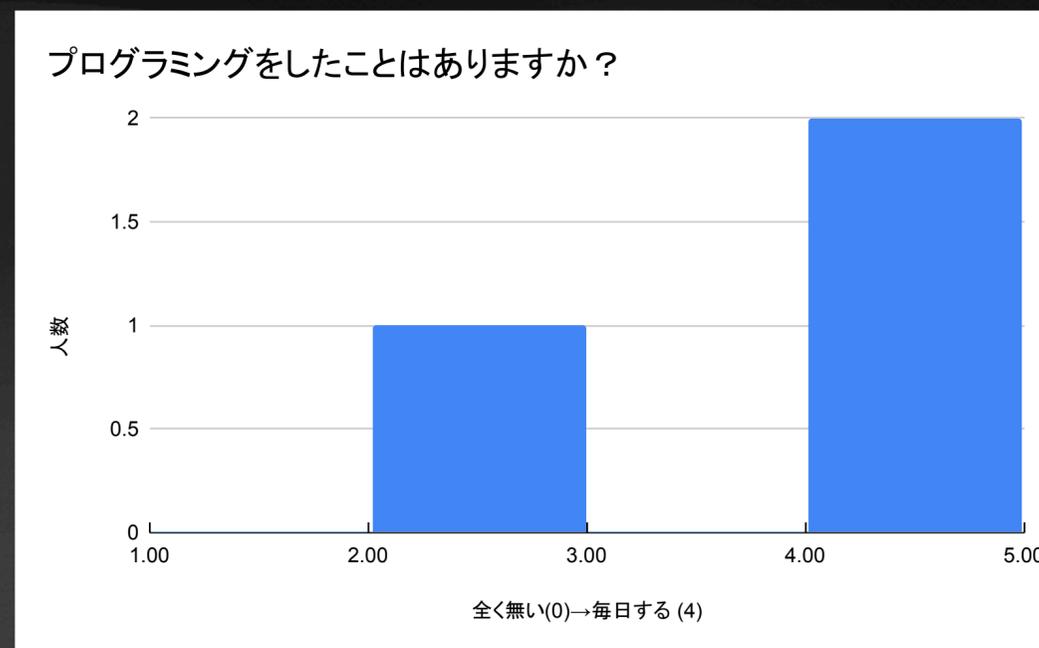
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java, and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers – electronic lexicographers, if you will – aim to save, or at least document, the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in those nearly forgotten lingua francas. Among the most endangered are Ada, APL, B (the predecessor of C), Lisp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Museum in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at [www.informatik.uni-freiburg.de/Java/misc/lang\\_list.html](http://www.informatik.uni-freiburg.de/Java/misc/lang_list.html). – Michael Menduno

Sources: Paul Bourin; Brent Halpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University

# 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, 等等)
  - テキスト
  - 等等
- 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

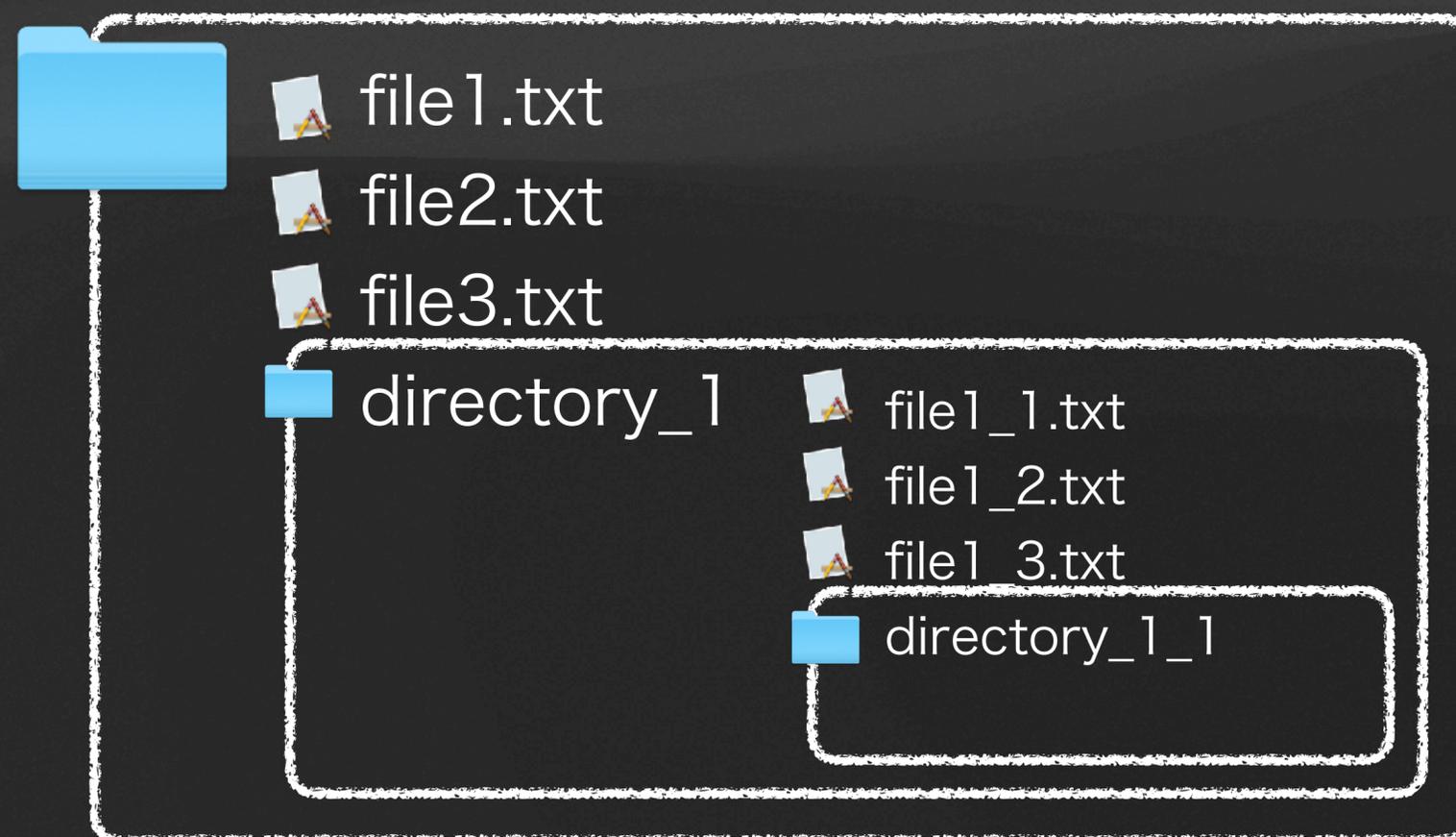
## ファイル【file】

**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。

## 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に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, 等等)
  - テキスト
  - 等等
- 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



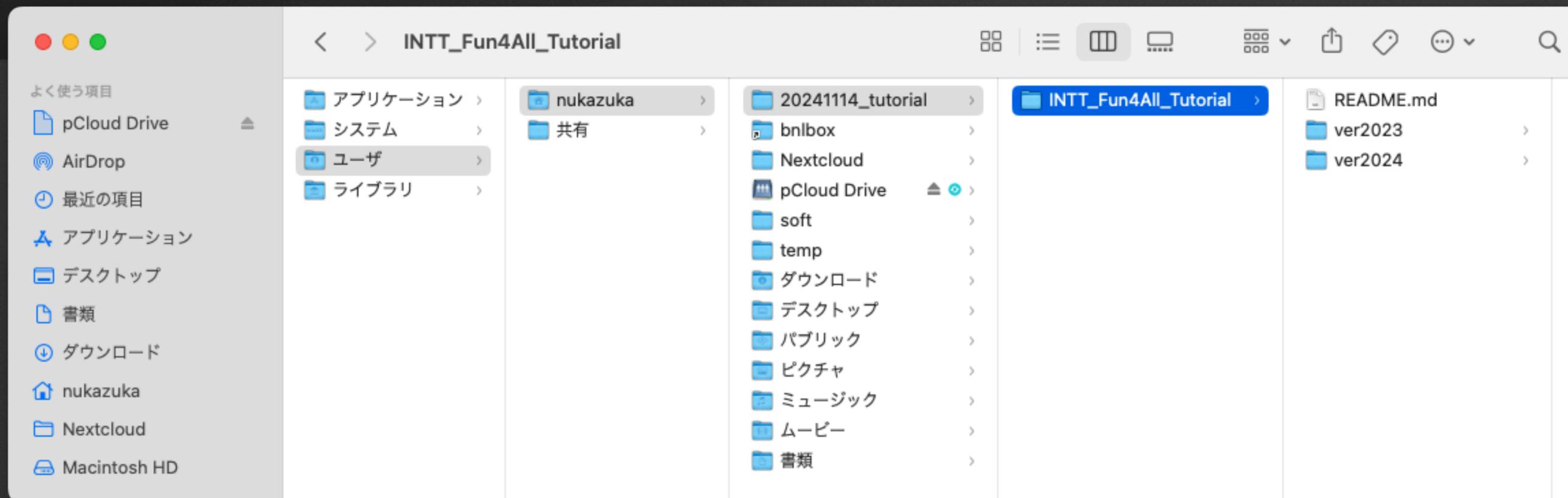
ディレクトリは階層構造をとる 33

# パス (Path)

ファイルやパスの場所を示す

例：糠塚 Mac の Fun4All  
チュートリアルサンプル

- ・ 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 “..”

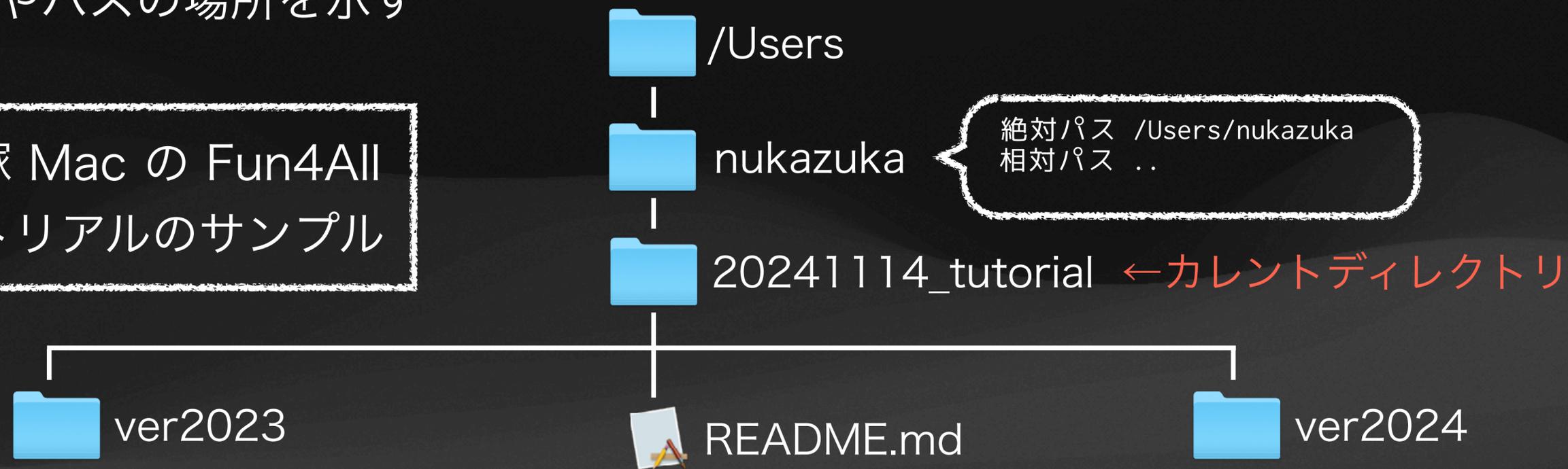


# パス (Path)

ファイルやパスの場所を示す

例：糠塚 Mac の Fun4All  
チュートリアルサンプル

- ・ 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 “..”



# パス (Path)

ファイルやパスの場所を示す

例：糠塚 Mac の Fun4All  
チュートリアルサンプル

- ・ 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 “..”



# パス (Path)

ファイルやパスの場所を示す

例：糠塚 Mac の Fun4All  
チュートリアルサンプル



- ・ 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 “..”



# パス (Path)

ファイルやパスの場所を示す

例：糠塚 Mac の Fun4All  
チュートリアルサンプル

- ・ 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 “..”

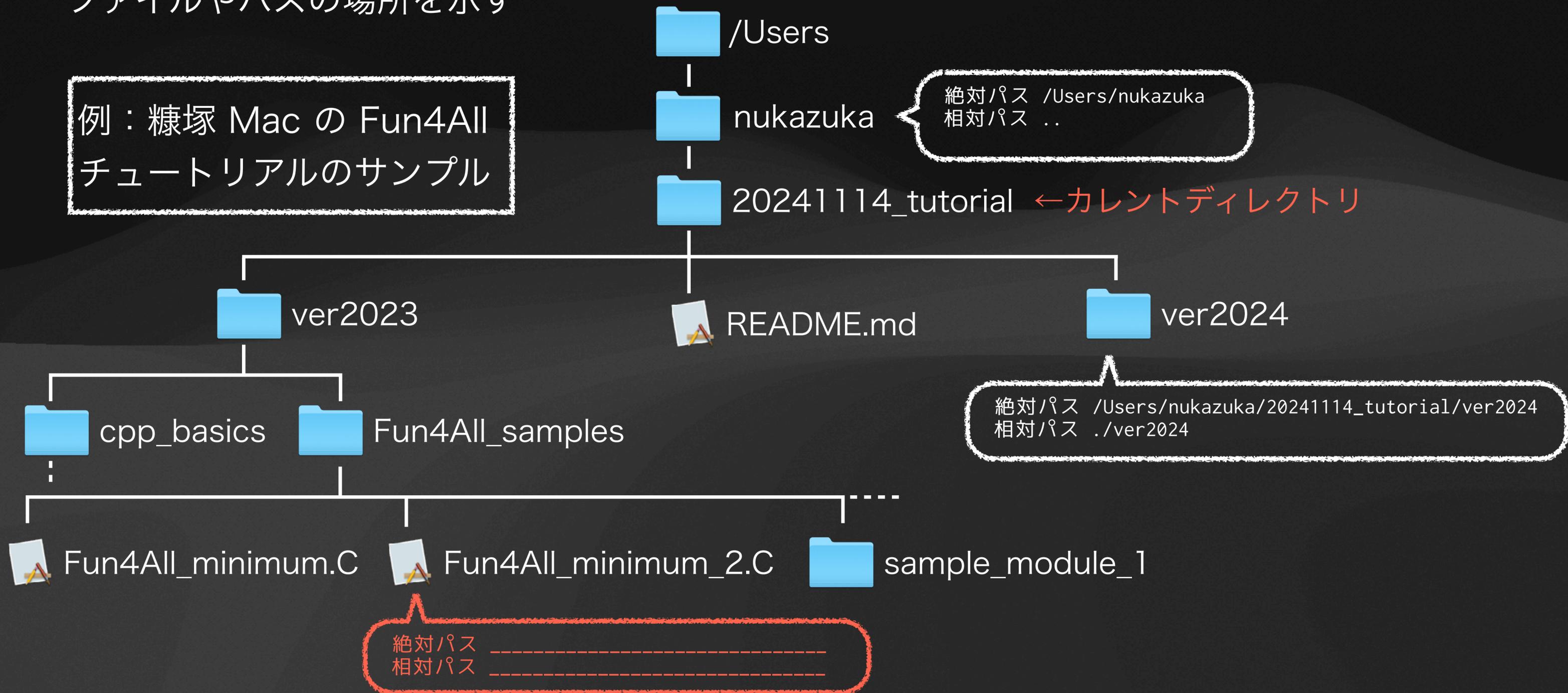


# パス (Path)

ファイルやパスの場所を示す

例：糠塚 Mac の Fun4All  
チュートリアルサンプル

- ・ 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 “..”



どう書くのでしょうか？

# 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

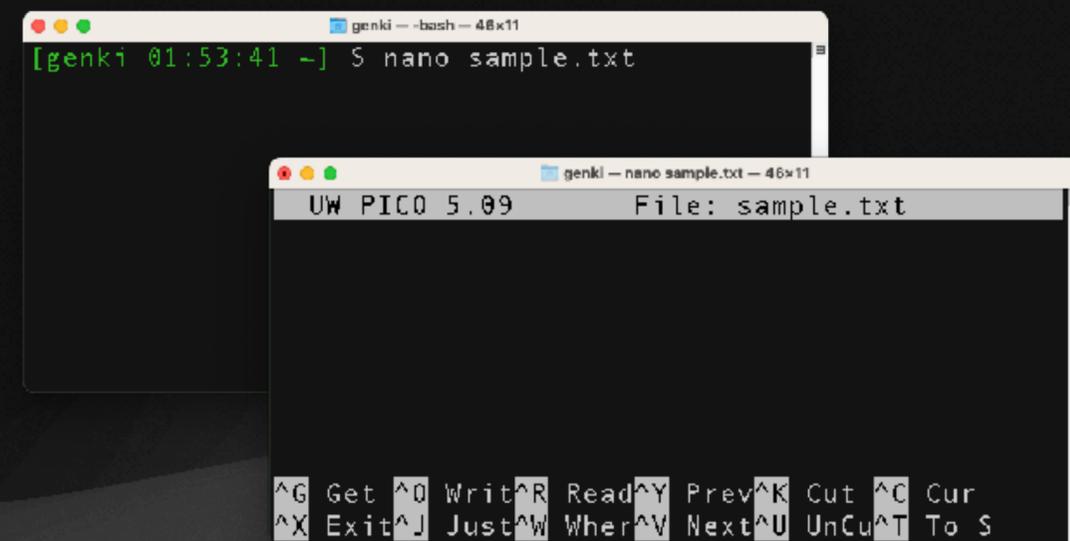
# Linux で CUI (再び) : テキストファイルをいじってみる

- 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よう。
    - ▶ [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。

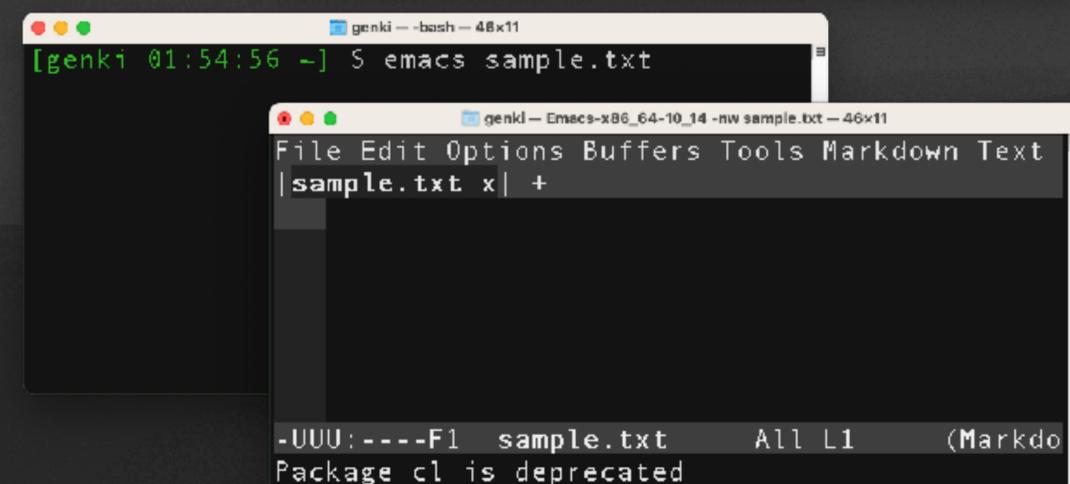
\* [エディタ戦争](#)

# Linux で CUI (再び) : テキストファイルをいじってみる

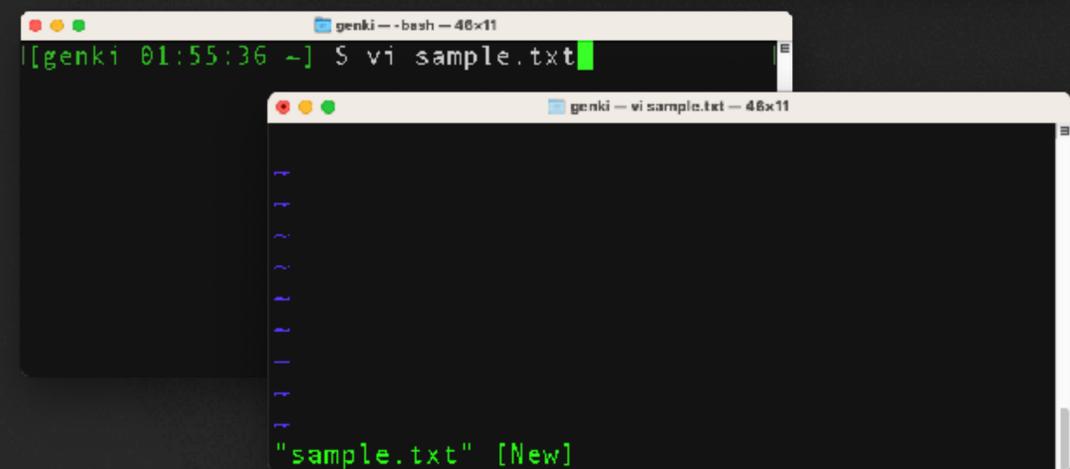
- とりあえずテキストファイルを作成・編集・保存してみよう
- テキストエディタを使う
  - 好きなエディタ\* を選び、ターミナルから起動してみましょう。
    - ▶ [vi](#) or [vim](#) (CUI): Linux/Unix (\*nix と書く) 系 OS では必ず標準搭載されている有名なエディタ。蜂谷さんは vi 派。
    - ▶ [Emacs](#) (GUI): 「エディタではなく環境」という人もいるほど何でもできるが、起動がちょっと遅い。vi よりシェアが低い。中川さんは emacs 派。
    - ▶ [nano](#) (CUI): 大抵の Linux, 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。



The image shows two overlapping windows. The top window is a terminal with the command `$ nano sample.txt` and the prompt `[genki 01:53:41 ~]`. The bottom window is the nano editor interface for `sample.txt`, showing the title bar `genki - nano sample.txt - 46x11` and the status bar with various keyboard shortcuts like `^G Get ^O Write ^R Read ^Y Prev ^K Cut ^C Cur`.



The image shows two overlapping windows. The top window is a terminal with the command `$ emacs sample.txt` and the prompt `[genki 01:54:56 ~]`. The bottom window is the Emacs editor interface for `sample.txt`, showing the title bar `genki - Emacs-x86_64-10_14 -nw sample.txt - 46x11` and the status bar with `-UUU:---F1 sample.txt All L1 (Markdo` and `Package cl is deprecated`.



The image shows two overlapping windows. The top window is a terminal with the command `$ vi sample.txt` and the prompt `[genki 01:55:36 ~]`. The bottom window is the vi editor interface for `sample.txt`, showing the title bar `genki - vi sample.txt - 46x11` and the status bar with `"sample.txt" [New]`.

\* [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。
- `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

