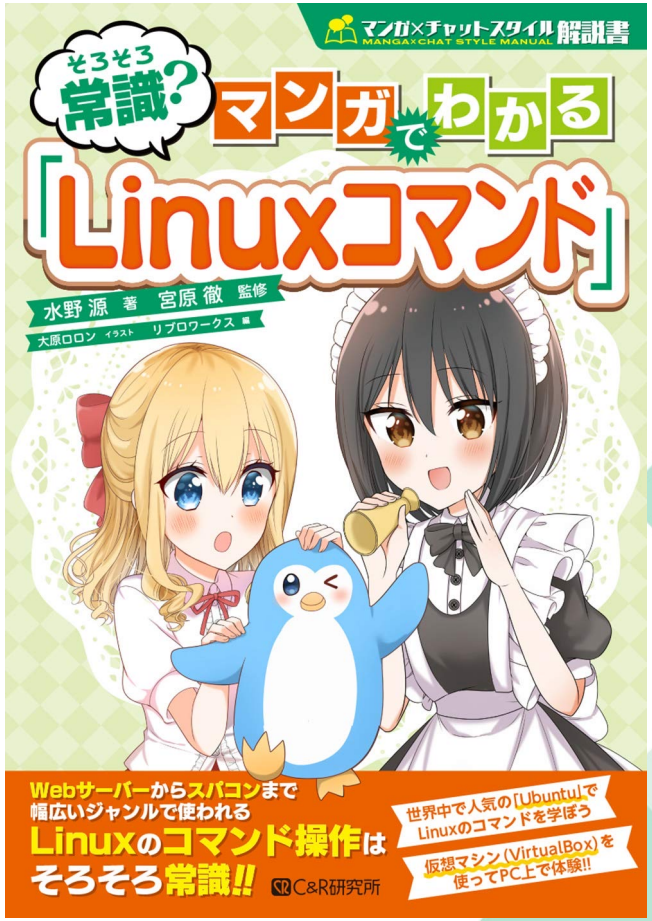
今からはじめる！ Linuxコマンド入門

日本仮想化技術株式会社
水野 源 mizuno@VirtualTech.jp

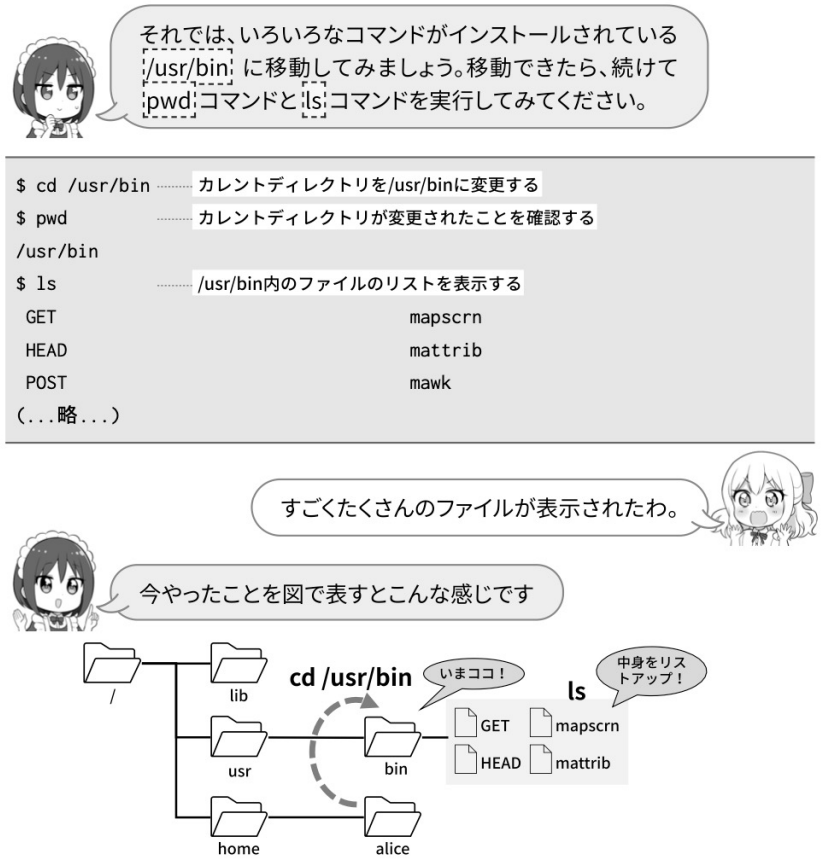
OSC-do 2021/06/26

そろそろ常識？ マンガでわかる「Linuxコマンド」

- はじめてCLIに触れる人向けのガイドブックを書きました。
- コマンドの概要、ファイル操作、SSH接続、Webサーバー構築までを解説しています。
- 2021年4月20日発売。
- Kindle版もあるよ。



マンガ&チャットスタイルでCLIを解説しています



OSSのCSS組版システムで執筆しています

```
# <span class="ch_category">【基礎編】</span><br>Linuxコマンドで<br>

@div:chap_lead
この章のお話は……

* <span class="tobira_num">第1話</span>そもそもLinuxとは？
* <span class="tobira_num">第2話</span>ディストリビューションとは？
* <span class="tobira_num">第3話</span>コマンドラインのメリットとは？

@divend

@divend

<p id="pagetitle">【基礎編】Linuxコマンドで何ができるの？</a>

## そもそもLinuxとは？

====

@アリス「starting in fwide
雪ちゃ～ん、聞きたい<br>
ことがあるんだけど～？

#なんでしょう、<br>
アリス様
@」


@アリス「hatena fslim
雪ちゃんはLinuxのコマンドって使ったことあるかしら？
@」

@雪「iyada fslim
もちろんございますが、突然どうされたのですか？
@」

@アリス「fuan centering fwide
最近、あちこちでLinuxって聞くじゃない？  <br>
```

VirtualTech Japan

本セミナーの想定受講者

- Linuxのコマンドラインの経験がない人。
  - コマンドとか使ったことないんだけど、業務で必要になってしまったので、よくわからないまま検索してQiitaをコピペしてます……みたいな人を想定。

本セミナーのゴール

- 黒い画面アレルギーがなくなる。
- Linuxのコマンドラインの基礎がわかる。

本日のアジェンダ

- CLIを使うメリットとは
- CLI環境の準備
  - ターミナルとシェルとは
  - Linux/Windows/MacでCLI環境を準備する
- コマンド操作に慣れよう
  - シェルの扱い方
  - オプションと引数
  - コマンド履歴、編集
- 頻出コマンド紹介
- コマンド実行例

CLIを使うメリットとは

CLIを使うメリット(1)

- 大量のデータの処理や自動化が容易。
  - マウスでは手作業でやらなければならないことを、一括で処理したり、自動化しやすい。
  - 複数のコマンドを組み合わせることで、複雑な処理も可能。
- 手順を再現しやすい。
  - 操作手順=コマンドはテキストなので、コピペすれば再現可能。
  - シェルスクリプト化すれば、手順の自動化&再利用で効率アップ。
  - メニューやアイコンが行方不明になりがちなGUIより操作が簡単な場合も。

CLIを使うメリット(2)

- 作業の記録を厳密に残しやすい。
  - 実行したコマンドと出力結果を記録したり、それを再生するコマンドもある。
- ミスに気づきやすい。
  - 行った操作がすべて文字として残るので、やらかしてもリカバリーしやすい。
- シンプルで、GUIほど多くのリソースを必要としない。
  - 少ないリソースで動かせるのでサーバーとの相性がよい。
  - 必要なコンポーネントが少ない→セキュリティ的にも有利。

つまりCLI環境とは

- CLIにはGUIにはないメリットがある。
- CLIは仕方なく使うものではなく、メリットがあるから敢えて使うもの。
- GUIとCLIは排他ではなく、用途に合わせて併用するもの。
  - CLI vs GUI論争は無意味。
- ただしCLIはGUIほど直感的ではないため、なんとなく触ってるだけで使えるようになるわけではない。
  - 最初はきちんとした学習が必要。

CLI環境の準備

参考: 端末(ターミナル)とは

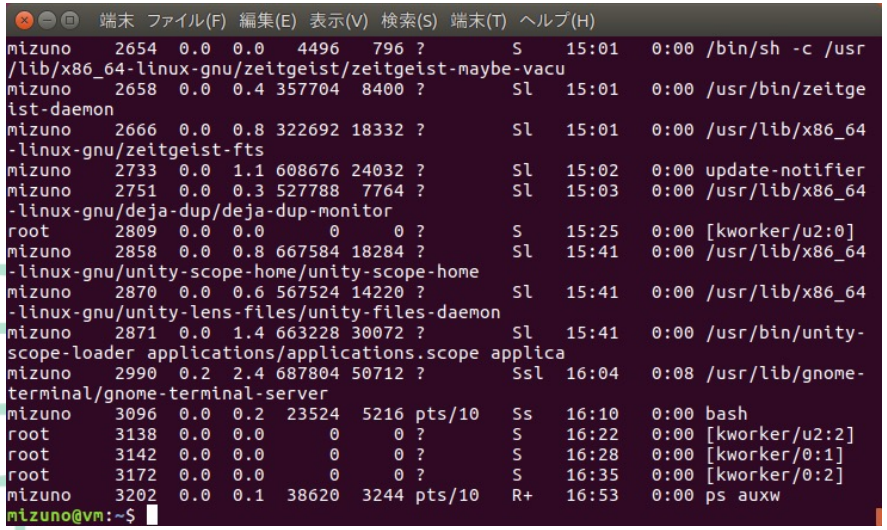
- 端末とは、大昔に存在した「キーボードとディスプレイを備えた、コンピューターを操作するための専用ハードウェア」のこと。
- 当時コンピューター本体は非常に大型だったため、専用の部屋に設置し、ネットワークを介して別の場所から端末で操作を行っていた。

←VT100端末 (Wikipediaより (CC BY-SA 3.0))

VirtualTech Japan

参考: 端末エミュレーターとは

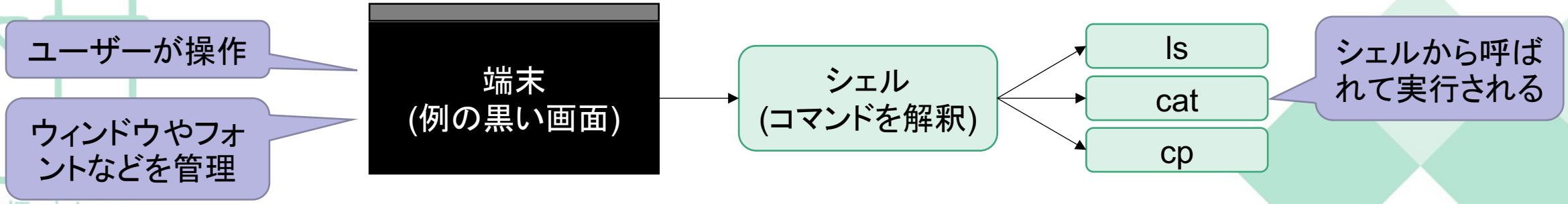
- 現在のLinuxでは、ソフトウェア的に再現された端末を使ってCLIを利用している。
- このソフトが端末エミュレーター(略して端末と呼ぶ)。



←Ubuntuに標準で用意されている端末の例

前提知識: シェルとは

- ユーザーにインターフェイスを提供するプログラムの総称。
  - グラフィカルなシェル(デスクトップ)とコマンドラインを提供するシェルがある。
  - Windowsのエクスプローラーはグラフィカルシェルの一種。
- Linuxでシェルと言えば、コマンドラインシェルを指すのが一般的。
  - Linuxで一般的に使われているシェルは「bash」。
- 端末はウィンドウの見た目や大きさなどを決めているだけで、コマンドの解釈と実行は、コマンドインタプリタであるシェルが行っている。



ここまでのまとめ

- CLIでコマンドを実行するには、コマンドを解釈するインタプリタであるシェルが必要。
- LinuxではBashというシェルが一般的。
- 端末(をエミュレートするソフトウェア)を介してシェルを操作する。

→ つまりCLIを使うには、端末とシェルを用意する必要がある。

CLI環境の準備(Linux編)

- Linuxコマンドと言った場合、一般的には標準シェルのBashと、基礎的なコマンドを集めたcoreutilsを前提としている。
- 大抵のディストリではBashが標準シェルとなっている。
  - ただし特殊なディストリやコンテナだとBashがないこともある。
- coreutilsや(何かしらの)端末も標準でインストールされているため、端末を起動するだけでよい。
- Linuxコマンドの学習には、Linuxを使うのが一番簡単。
  - 環境構築が簡単で、参考書と同じものが動く。
  - メジャーなディストリなら何を使っても構わない。

CLI環境の準備(Windows編)

- 端末はWindows Terminal、コマンド実行環境としてはWSL2のUbuntuがおすすめ。
  - 他にメジャーな端末としてはTeraTermやPuTTYなどがある。
- WSL2を使うと、Ubuntuをそのまま Windows上で動かせる。
  - CLIからWindowsのファイルシステムにもアクセスできるので、Windows上のデータをLinuxのツールを使って編集、がシームレスに行えるのがメリット。

←Windows TerminalやWSLは、Microsoft Storeからインストールできる

CLI環境の準備(Mac編)

- 端末は標準でインストールされている「ターミナル」を起動するだけ。
  - これ以外の端末としてはiTerm2が有名。
- macOS 10.15 Catalina以降は、シェルがbashからzshに変更された。
- 標準でインストールされているコマンドの差異にも注意。
  - MacはBSDベースのため、一部のコマンドの挙動がGNU coreutilsベースのLinuxと異なる。
- Macのデフォルトの環境で、Linuxの参考書と同じことをやろうとするとハマることがあるので注意。
  - シェルをbashに変更して、HomebrewあたりでGNU coreutilsを入れるとだいたい同じになる(はず)。

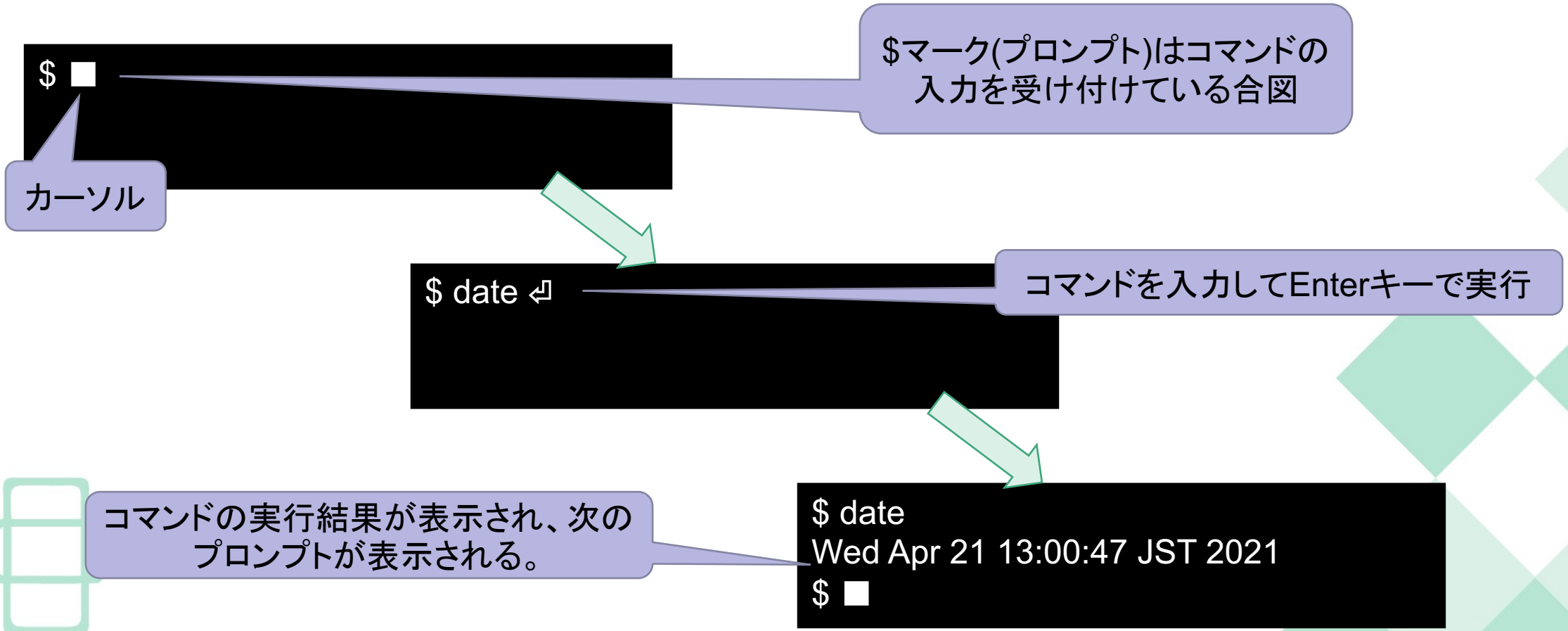
コマンド操作に慣れよう

コマンド操作の基本

1. プロンプトにコマンドを入力してEnterで実行する。
2. 実行が終了するとプロンプトに復帰する。
3. 次のコマンドを入力する。

この繰り返しが基本。

コマンド操作の基本

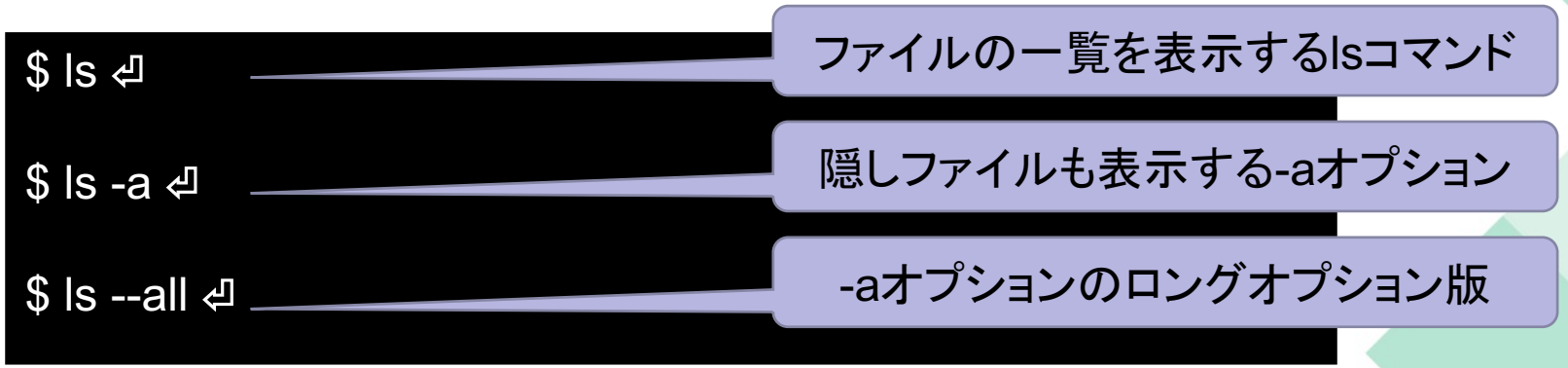

$ ■
$マーク(プロンプト)はコマンドの
入力を受け付けている合図
カーソル
$ date ⏎
コマンドを入力してEnterキーで実行
$ date
Wed Apr 21 13:00:47 JST 2021
$ ■
コマンドの実行結果が表示され、次の
プロンプトが表示される。

カレントディレクトリを意識しよう

- カレントディレクトリとは、シェルが現在開いているワーキングディレクトリのこと。
- コマンド実行時、対象のディレクトリを指定しなかった場合は、暗黙的にカレントディレクトリが利用される。
  - たとえば、指定したディレクトリ内のファイル一覧を表示するlsコマンドは、対象を省略するとカレントディレクトリの中身を表示する。
  - カレントディレクトリ内のファイルは、ファイル名のみで指定できる、など。
- カレントディレクトリを常に意識するクセをつけるのが上達のコツ。
  - だいたいプロンプトに表示されるようになっているよ！

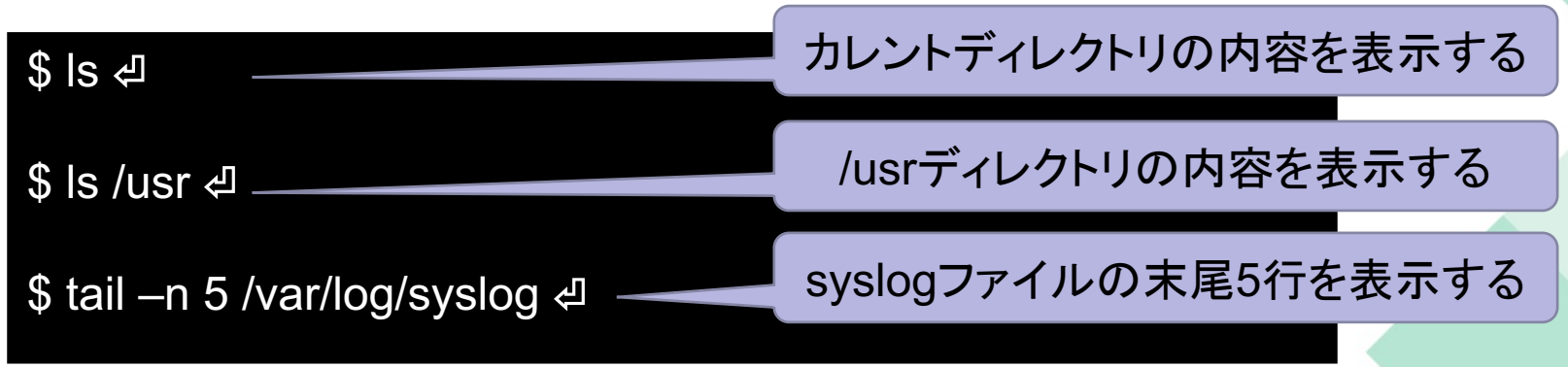
オプションと引数(1)

- オプションとはコマンドの挙動を変化させるスイッチのこと。
  - たとえば結果の表示の仕方を変えるオプションなどがある。
- オプションはハイフン+英数字1文字が基本。
  - コマンド名の後に、半角スペースで区切って指定する。
  - ハイフンふたつ+英単語で指定されるロングオプションもある。



オプションと引数(2)

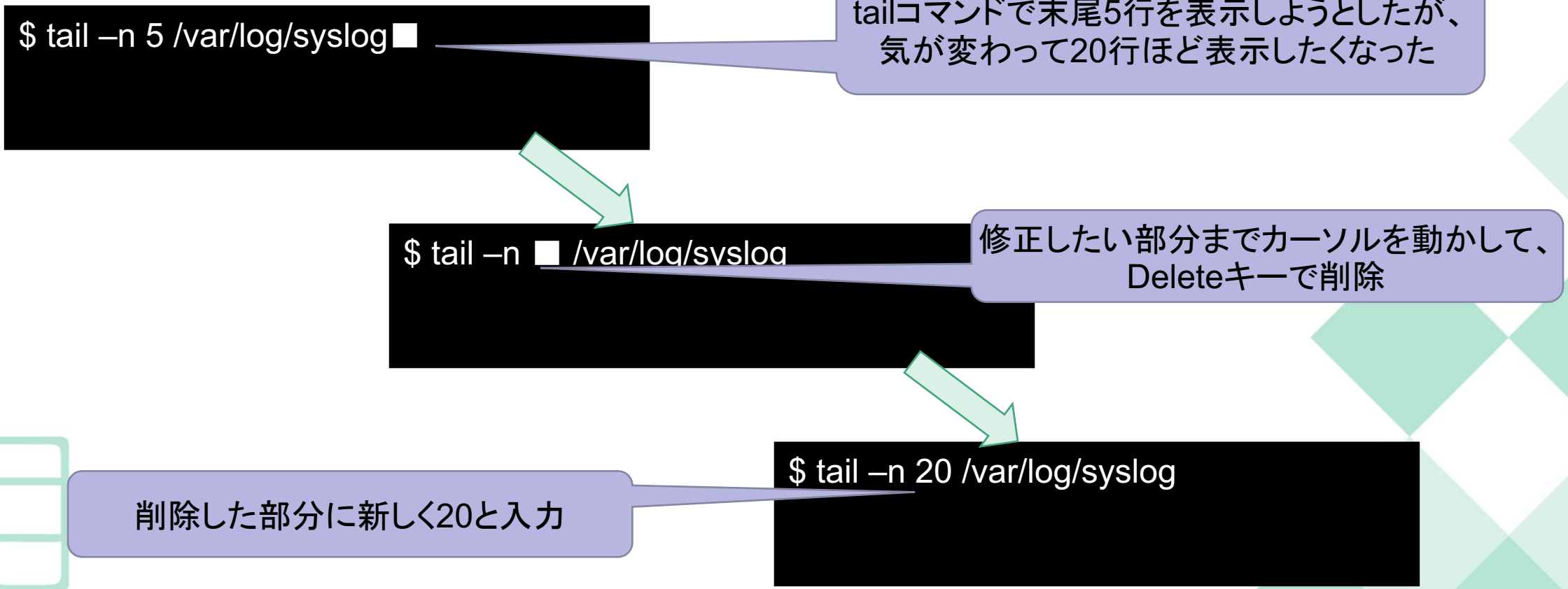
- 引数(パラメータとも)とはコマンドに渡す値のこと。
  - 開くファイルを指定する、など。
- 引数も半角スペースで区切って指定する。
- 引数を指定できるオプションもある。
  - 引数、オプションは複数組み合わせて使うこともできる。



コマンドラインを編集する

- 入力中のコマンドはテキストエディタのように編集が可能。
  - 正しいコマンドを一発で書き上げる必要はなく、タイプミスしても大丈夫。
- カット&ペースト機能も用意されている。
- コマンドは実行する前にきちんと確認して、間違っていた場合は落ち着いて修正しよう。
  - 危険なコマンドを手くせで実行してあわわ……という事故はよくある。

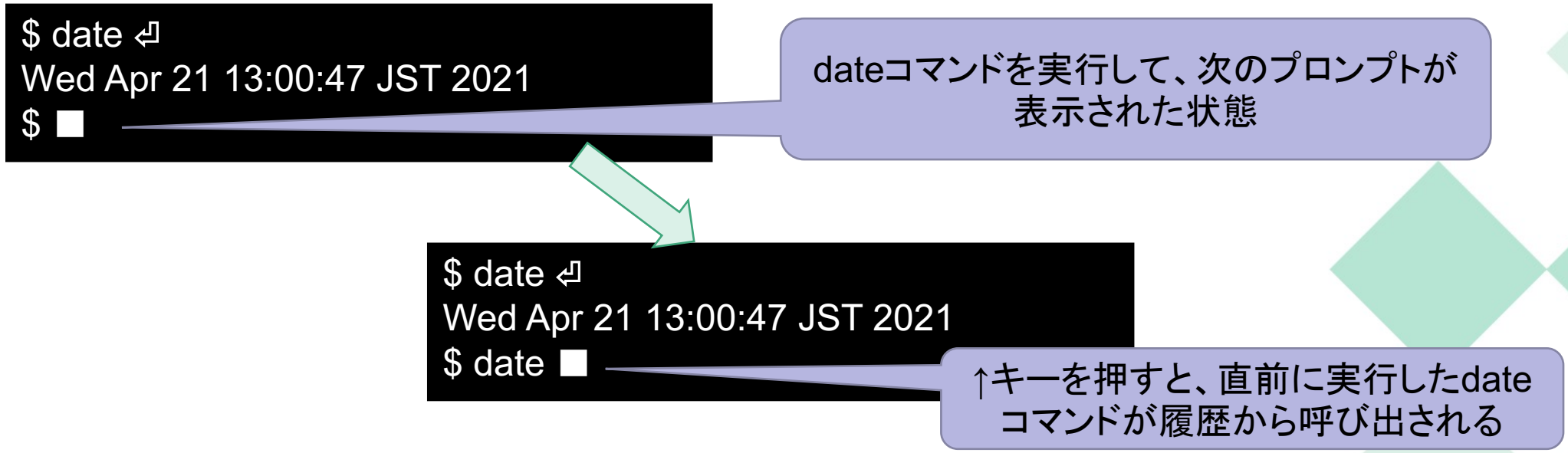
コマンドラインを編集する



コマンド履歴の活用

- ↑ / ↓ キーか、Ctrl+n/pで、過去に実行したコマンドを呼び出せる。
  - プロンプトが表示されている状態で↑キーを1回押すと、直前に実行したコマンドが、コマンドラインに入力済みの状態になる。
  - historyコマンドで履歴の表示や管理が可能。
- Ctrl+Rで履歴を検索できる。
  - 大昔に実行したコマンドであっても、コマンド名や引数の断片さえ覚えていれば、検索で呼び出せる。
- 呼び出した履歴は実行前に編集できる。
  - 引数だけを書き換えて再利用する、など。
- 同じコマンドや似たコマンドは何度もタイプせず、履歴を再利用すると便利。

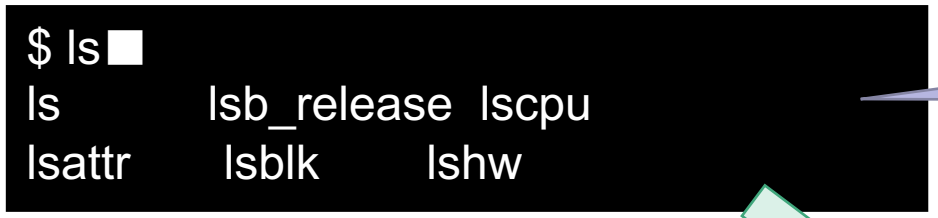
コマンド履歴の活用


$ date ⏎
Wed Apr 21 13:00:47 JST 2021
$ ■
dateコマンドを実行して、次のプロンプトが表示された状態
$ date ⏎
Wed Apr 21 13:00:47 JST 2021
$ date ■
↑キーを押すと、直前に実行したdateコマンドが履歴から呼び出される

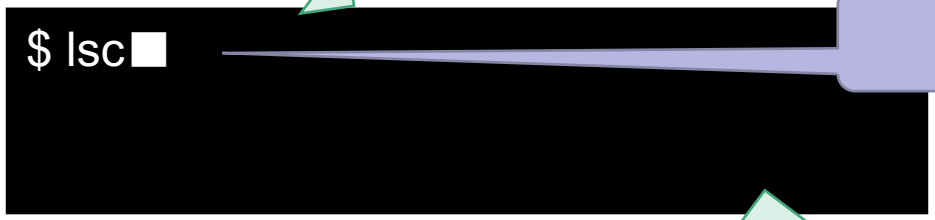
コマンド補完の活用

- 入力中のコマンドはTabで補完できる。
  - 入力した文字列から、実行可能なコマンド候補を前方一致で検索する。
  - 候補が複数あり、一意に特定できない場合は、候補が一覧表示される。
  - 候補を一意に絞れた場合は、残りの部分を自動補完する。
  - 候補がない場合は何も表示されない。
- 補完対象や候補は自分で定義、拡張できる。
  - bash-completionという仕組み。
  - これによりオプションや引数もTab補完できる(場合もある)。
- タイプミスをなくすためにも、手入力は最小限に抑えて補完を使おう。

コマンド補完の活用



lsと入力してTabキーを押すと、lsではじまる名前を持つ複数の候補が表示される



lscと入力してTabキーを押す

先頭がlscではじまるコマンドはlscpuのみなので、残りの部分が自動的に補完される

コマンドのマニュアルの読み方

- 各コマンドにはオンラインマニュアルが用意されている。
- マニュアルを読むためのコマンドがmanコマンド。
  - manコマンドの引数に読みたいマニュアル名を指定して使う。
- マニュアルはジャンルごとに複数のセクションに分かれている。
  - セクションは番号で指定しよう。
- -kオプションで検索もできる。
- 詳しくはマニュアルのマニュアルを読んでおこう。
  - つまりman manコマンドを実行。

頻出コマンド紹介

ディレクトリの内容を表示する(ls)

- 引数に指定したディレクトリの中身を一覧表示する。
  - 省略するとカレントディレクトリが対象になる。
- よく使うオプションはltarあたり。
  - lオプション → 詳細なリストを表示する。
  - aオプション → 隠しファイルも表示する。
  - tオプション → タイムスタンプ順に表示する。
  - rオプション → 昇順降順を逆にして表示する。

カレントディレクトリを変更する(cd)

- カレントディレクトリを、引数で指定したディレクトリに変更する。
  - 省略した場合はホームディレクトリに変更する。
- CLIではカレントディレクトリがとても重要なので、まずはcdからはじまると言っても過言ではないくらい基本のコマンド。
  - シェルによっては、カレントディレクトリの位置を記録しておき、あとから取り出す「ディレクトリスタック」や、ディレクトリ名のみでcdできるauto_cdなど、cdをもっと便利にする機能が提供されているほど重要。

ファイルをコピー/移動する(cp/mv)

- ファイルやディレクトリをコピー/移動する。
  - つまりcopy/move
- コピー/移動時は名前を変えることもできる。
  - 同一ディレクトリ内で名前を変えて移動することでリネーム。
- どちらもコピー元とコピー先のふたつの引数が必要。
- 色々と奥の深いコマンドなので詳しくはマニュアルを参照。

ファイルの内容を表示する(cat)

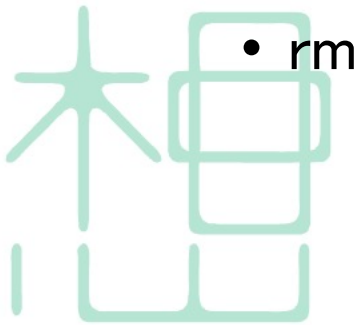
- 引数に指定したファイルの内容を表示する。
  - テキストの内容の確認によく使う。
- 複数のファイルを指定すると、それらを連結できる。
  - そもそもコマンド名の由来はconcatenate
- 実はバイナリファイルも処理できる。
  - ファイルサイズ制限のあるファイルを分割/結合したり。
- Linuxのファイル抽象化と組み合わせると、もっと色々できる。
  - cat /dev/videoでビデオ録画している人がいる(実話)。

ファイルを削除する(rm)

- ファイルやディレクトリを削除する。
  - つまりremove
- ディレクトリを削除するには-rオプションが必要。
- rm -rf /はよくあるジョーク。
  - ちなみに↑のコマンドはデフォルトでは動かないようになっている。

ディレクトリを作成/削除する(mkdir/rmdir)

- ディレクトリを作成/削除する。
  - つまりmake/remove directory
- 引数に指定したディレクトリを作成/削除する。
- 削除する場合、対象のディレクトリは空である必要がある。
  - rmdirする前に中身を消しておかないといけないため、ある意味面倒。
  - とはいえ、うっかり必要なファイルが入っているディレクトリを消してしまうという事故を防げるという面もある。
  - rm -rfと使い分けよう。



シェルを終了する(exit)

- シェルを終了する。
- GUIで端末を使っている場合はウィンドウを閉じてもいいけれど、スクリプトの中で終了したい場合や、サーバーのコンソールからログアウトしたいような場合はexitで終了しよう。
- exitと似たコマンドにlogoutがある。
  - logoutはログインシェルを終了するコマンド。
  - サブシェルは終了できないので、とりあえずどちらも終了できるexitを覚えておけばOK。

なお筆者のコマンド履歴より

- 普段作業しているUbuntuマシンのシェルの履歴から、直近13000件ほどをカウントした結果の上位ランキング。

やはり圧倒的

順位	回数	コマンド	順位	回数	コマンド
1	1602	ls	8	449	exit
2	1434	cd	9	323	cat
3	748	pass	10	269	kubectl
4	589	apt	11	237	rm
5	535	docker	12	236	less
6	512	git	13	224	aws
7	459	ssh	14	202	vi

コマンドの実行例

ディレクトリを作る/移動する

```
$ mkdir work ⏎
$ ls ⏎
work
```

workディレクトリを作成する

作成されたディレクトリ

```
$ mv work work_new ⏎
$ ls ⏎
work_new
```

workディレクトリをwork_newという名前で移動(リネーム)する

```
$ mkdir backup⏎
$ mv work_new backup/ ⏎
$ ls ⏎
backup
$ ls backup/ ⏎
work_new
```

backupディレクトリを作成し、その中にwork_newディレクトリを移動する

backupディレクトリの中身を表示する

カレントディレクトリを変更する

```
$ pwd ⏎
/home/ubuntu
$ mkdir work ⏎
$ cd work ⏎
$ pwd ⏎
/home/ubuntu/work
```

カレントディレクトリを確認する

workディレクトリを作成して、カンレントディレクトリを変更する

カレントディレクトリが変更された

```
$ pwd ⏎
/home/ubuntu/work
$ cd ..⏎
$ pwd ⏎
/home/ubuntu
```

カレントディレクトリは/home/ubuntu/work

..(ドットドット)は1階層上のディレクトリを表す

カレントディレクトリを親ディレクトリに変更できた

ファイルをコピーする

```
$ cp /var/log/syslog ./⏎
$ ls
syslog
```

/var/log/syslogファイルを、カレントディレクトリへコピーする

```
$ cp syslog syslog.bak⏎
$ ls ⏎
syslog    syslog.bak
```

コピーしたsyslogを、syslog.bakという名前でさらにコピーする(バックアップを作る)

ファイルがコピー(バックアップ)された

```
$ mkdir work⏎
$ cp syslog syslog.bak work/⏎
$ ls ⏎
work    syslog    syslog.bak
$ ls work/ ⏎
syslog    syslog.bak
```

workディレクトリを作成し、複数のファイル(syslogとsyslog.bak)をまとめてコピーする

カレントディレクトリ(コピー元)とworkディレクトリ(コピー先)の両方にファイルが存在する

ファイルを削除する

```
$ ls ⏎
syslog    syslog.bak
$ rm syslog.bak ⏎
$ ls ⏎
syslog
```

syslog.bakを削除する

```
$ ls ⏎
work
$ rm work ⏎
rm: cannot remove ‘work': Is a directory
```

workディレクトリを削除する

rmコマンドではディレクトリを削除できない

ディレクトリを削除する

```
$ ls work/⏎
syslog
$ rmdir work⏎
rmdir: failed to remove 'work/': Directory not empty
$ rm work/syslog
$ rmdir work
```

workディレクトリの中にはsyslogがあるため、rmdirで削除できない

ディレクトリ内のsyslogファイルを削除することで、rmdirが可能となる

```
$ ls work/ ⏎
syslog
$ rm –r work ⏎
```

rmコマンドは-rオプションをつけることで、ディレクトリを中身ごと削除できる

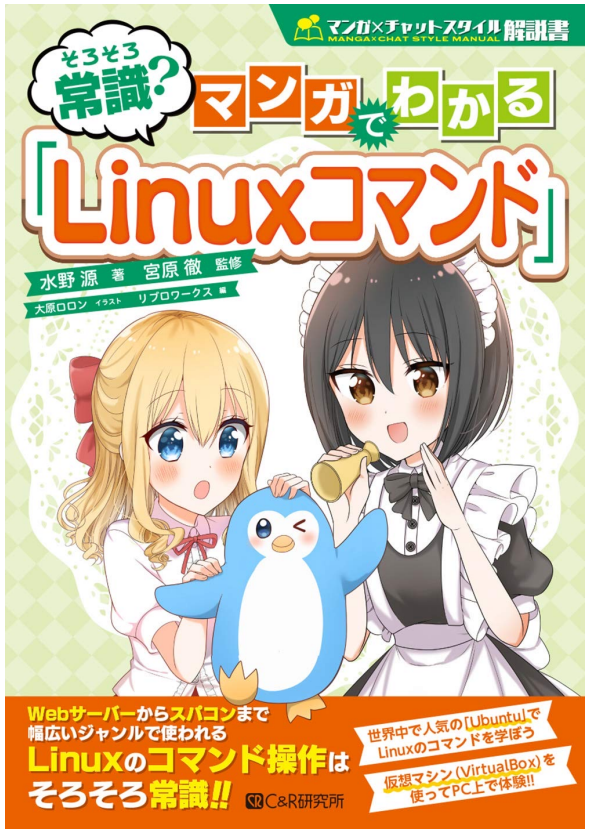
まとめ

コマンドを学ぶ意味

- コマンドに習熟すると、その時その時に合わせた処理を、その場で組み上げることもできる。
  - 決まった機能しか持たないGUIアプリに比べ、自由で柔軟な処理が可能。
  - 単純に作業効率アップが期待できる。
- 一度組み立てたコマンド群は再利用しやすい。
  - 使えば使うほど効率が上がっていく。
  - コマンド履歴は財産。
- コマンドが使えるということは、GUIしか使えない場合に比べて強力な武器を持っているということに他ならない。
  - とにかく便利になるので、怖がらずに使ってみよう。

OSCのアンケートにご協力ください

- アンケートに回答すると、抽選でこちらの書籍が当たります！
- アンケートページはQRコード、もしくはURLから。
  https://bit.ly/OSC2021Hokkaido_Form



VirtualTech Japan