Q1
Linuxとはなにか？

A1
WindowsやmacOSなどと同じ、OS（オペレーティング・システム）のひとつ

これらのOSとは異なり、オープンソースのOSであることが最大の特徴
利用によるライセンス料がかからないのもあって、サーバー用OSではトップシェア

狭義では「Linuxカーネル」、広義では「Linuxディストリビューション」を指す

Q2
OSとはなにか？

A2
コンピュータを動作させるための基盤となるソフトウェアのこと

一般的なPCではWindowsやmacOS、サーバー用のコンピュータではLinuxが主流

OSの入っていないコンピュータだけでは何もできず、
OSをインストールすることで初めて、人間やアプリケーションがコンピュータを扱えるようになる

OSの中核には「カーネル」という、ハードウェアの制御を行うソフトウェアがある

Q3
なぜLinuxを学ぶ必要があるのか？

A3
サーバーのOSはほとんどがLinuxなので、
サーバーを操作する上で知っておく必要がある

オープンソースで無料で利用できることもあり、
サーバー用OSとしてはトップシェアになっている

Q4
Linuxカーネルとはなにか？

A4
LinuxにおいてOSの中核となる、コンピュータのハードウェア制御を行うソフトウェアのこと
Linuxカーネルには、実際にユーザーが使うツールやアプリケーションは含まれない
なので実際には、ディストリビューションとして
その他必要なツールとまとめて提供されることがほとんど

Q5
Linuxディストリビューションとはなにか？

A5
Linuxとは狭義の意味では、「Linuxカーネル」というOSの中核となる部分のことを指す
これは、コンピュータのハードウェア制御を行うソフトウェアで、
実際にユーザーが使うツールやアプリケーションは含まれない

これでは不便なので、その他必要なツールをLinuxカーネルとまとめて提供することで、
すぐにユーザーが使えるようにしたものをディストリビューションという

単に「Linux」と言った場合には「Linuxディストリビューション」を指すことも多い
ディストリビューションにはいくつか種類があるが、
主流なのはCentOSなどRed Hat系と、UbuntuなどのDebian系

Q6
物理マシン・仮想マシンとはそれぞれなにか？

A6
・物理マシン

普通のコンピューター

・仮想マシン
物理マシンと同じ機能をソフトウェアで実現したコンピューター
仮想マシンの実体は物理マシン上のファイルであり、
このファイルが物理コンピューター上で実行されることで、
物理コンピューターと同じ機能をソフトウェアで実現している

仮想マシンを用意するには「Oracle VM VirtualBox」などの仮想化ソフトウェアが必要

Q7
ホストOS・ゲストOSとはそれぞれなにか？

A7
・ホストOS
物理マシンのOSのこと
仮想マシンの基盤となる

・ゲストOS
仮想マシン上で稼働するOSのこと

Q8
「GUI」と「CLI」とはそれぞれなにか？

A8
「GUI」：グラフィカルユーザーインターフェース（Graphical User Interface）の略
「CLI」：コマンドラインインターフェース（Command Line Interface）の略

WindowsやmacOSで使われているような、
画面上のアイコンをマウスなどを用いてクリックすることで操作するインターフェースがGUI
Linuxで使われているような、キーボードなど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がCLI

Q9
シェルとはなにか？

A9
Linuxカーネルを操作するためのインターフェースなっているソフトウェアのこと

コマンドは以下のような流れで実行される
①シェルがキーボードなどから「date」といった入力を受け取る
②シェルが「date」という名前のコマンドを探し出し、Linuxカーネルに実行を依頼する
③Linuxカーネルがコマンドを実行する
④実行結果をシェルが受け取り、画面などに表示する

このように、Linuxカーネルを「殻」のごとく包み込むように
動作することから「shell（殻）」と呼ばれる

Q10
Linuxカーネルとシェルを一体化せず、分けているのはなぜか？

A10
Linuxカーネルとシェルを分離しておくと以下のようなメリットがある
・シェルを自分好みに取り換えることができる
・OSの違いをシェルが隠ぺいしてくれるので、異なるOSも同じシェルで操作できる
・シェルにエラーが起きても、Linuxカーネルへの影響が少ない

UNIXの「１つのプログラムには、１つのことをうまくやらせる」という思想がよく表れた設計になっている

Q11
UNIXとはなにか？

A11
WindowsやmacOSなどと同じ、OS（オペレーティング・システム）のひとつ

これらのOSとは異なり、オープンソースのOSである
LinuxはUNIXを参考にして作られているので共通点が多く、UNIXの思想が色濃く反映されている

Q12
bashとはなにか？

A12
シェルの種類のひとつ、最も主流でLinuxのデフォルトのシェル
「sh」というシェルを基本として機能を拡張しているので、
「sh」と後方互換性を持ち、shのコマンドはbashでも動く
シェルの種類は他にも、「zsh（ズィーシェル）」や「tcsh（ティーシーシェル）」などがあるが、
とりあえずは「bash」を使っておけば間違いない

Q13
ターミナルとはなにか？

A13
ユーザーがコンピュータに入出力する際に利用するハードウェアのこと、「端末」とも言う
ターミナルの入力装置としてはキーボード、出力装置としてはディスプレイが主に使われる

「ターミナルエミュレータ」のことを指して「ターミナル」というケースも多い
ターミナルをソフトウェアとして扱えるように実装したものを、ターミナルエミュレータという
代表的なターミナルエミュレータは以下
・Windows：PuTTY, Tera Term
・macOS：ターミナル, iTrem2
・Linux：GNOME端末, Konsole

ターミナルエミュレータとシェルは、まったく異なるソフトウェアであることに注意
ターミナルエミュレータは、入出力画面を提供するだけで、内部で処理を動かしているのはシェル

Q14
プロンプトとはなにか？

A14
`[hiramatsu@localhost ~] $`
などというように表示されている部分のこと

プロンプトの右側にはコマンドを入力することができる
「prompt」は「促す」という意味なので、ユーザーに入力を促している部分といえる

Q15
コマンドラインとはなにか？

A15
プロンプトの後ろのコマンド入力部分のこと

Q16
bashでカーソルを
・後方(左)に1文字移動する
・前方(右)に1文字移動する
・行頭(一番左)に移動する
・行末(一番右)に移動する

A16
・後方(左)に1文字移動する
Ctrl + b、もしくは矢印キー(←)
「backward（後方）」の「b」

・前方(右)に1文字移動する
Ctrl + f、もしくは矢印キー(→)
「forward（前方）」の「f」

・行頭(一番左)に移動する
Ctrl + a
「頭（atama）」の「a」と覚えよう

・行末(一番右)に移動する

Ctrl + e
「end（末端）」の「e」

Q17
bashで以下の操作をするには？
・カーソル位置の後ろの1文字を削除する
・カーソル位置の1文字を削除する
・カーソル位置の後ろの1単語を削除する

A17
・カーソル位置の後ろの1文字を削除する
BackSpaceキー

・カーソル位置の1文字を削除する
Deleteキー

・カーソル位置の後ろの1単語を削除する
Ctrl + w
「word（単語）」の「w」

Q18
bashにおいて、「カット」と「ヤンク」とはそれぞれなにか？

A18
「カット」：「切り取り」のこと
「ヤンク」：「ペースト」のこと

Q19
bashで以下の操作をするには？
・カーソル位置から行末までをカットする
・カーソル位置から行頭までをカットする
・最後にカットした内容をヤンクする

A19
・カーソル位置から行末までをカットする
Ctrl + k

・カーソル位置から行頭までをカットする

Ctrl + u

・最後にカットした内容をヤンクする
Ctrl + y
「yank」の「y」

Q20
入力途中のコマンドやパスの補完を行う

A20
Tabキー

「cd /ho」までコマンドラインに入力してTabキーを押すと、
「cd /home/」というように補完してくれる

Q21
「/bin/bash」というファイルはどのようなファイルか？

A21
bashというシェルを利用するためのファイル

Linuxで扱うデータや機能はすべて、ファイルとして表現されている
bashのようなシェルだけでなく、各種コマンドやLinuxカーネルも1つのファイル

Q22
パスとはなにか？

A22
「/usr/bin」というような、
あるファイルやディレクトリにたどり着くまでの経路のこと

Q23
カレントディレクトリを表示する

A23
pwd

「print working directory」の略

Q24
「絶対パス」と「相対パス」とはそれぞれなにか？

A24
・「絶対パス」
ルートディレクトリを起点として表記するパス
曖昧さはないが、ファイルやディレクトリが深い位置にあるとパスが長くなってしまったり、
開発したマシンのディレクトリ構成に依存してしまうという欠点がある

・「相対パス」
カレントディレクトリを起点として表記するパス
「.」はカレントディレクトリを表し、「..」は親ディレクトリを表す
パスを短くなる上に、開発したマシンのディレクトリ構成にも依存しない

Q25
カレントディレクトリを一つ上のディレクトリに変更する

A25
```
cd ..
```

「change directory」の略

「..」は相対パスによる指定で、親ディレクトリの意味

Q26
カレントディレクトリを「/usr/lib」に変更する

A26
```
cd /usr/lib
```

「change directory」の略

絶対パスによる指定を行っている

Q27
カレントディレクトリをホームディレクトリに変更する

A27
以下のいずれか
①cd
②cd ~
③cd /home/ユーザー名

「change directory」の略

引数をつけずにcdコマンドを実行すると、ホームディレクトリへ移動する（①）
シェルでチルダ「~」を入力すると、ホームディレクトリのパスに自動的に置き換えられる
これを「チルダ展開」という（②）
ホームディレクトリのパスは「/home/ユーザー名」なので、絶対パスによる指定も可能（③）

Q28
「/bin」のファイルとディレクトリを一覧表示する

A28
ls /bin

「list」の略

Q29
パス名展開とはなにか？

A29
ファイル名の一部を「*」や「?」などを用いてパターンで指定することにより、複数ファイルを指定する機能のこと
パス名展開では、任意の文字列を「*」、任意の1文字を「?」で表す

例えば、カレントディレクトリにある拡張子が「.txt」のファイルを

すべて表示したいときは以下のように書ける
```
cat *.txt
```

もしこのとき、カレントディレクトリに「A.txt」と「B.txt」があったら、
このコマンドは以下のように変換されて実行される
```
cat A.txt B.txt
```

パス名展開ではこのように、「*」や「?」で書かれたパターンを、具体的なファイルのリストに変換して実行する
こうすることによって、カレントディレクトリ内のすべての「.txt」ファイルを表示することができる

Q30
カレントディレクトリのファイル中で、拡張子が「.html」のファイルを表示する

A30
```
ls *.html
```

「ls」は「list」の略

「*」は任意の文字列を表現しているので、「.html」で終わるすべてのファイルを指定していることになる

このようにファイル名の一部を「*」などを用いてパターンで指定することにより、
複数ファイルを指定する機能を「パス名展開」という

パス名展開では、任意の文字列を表す「*」のほかに、任意の1文字を表す「?」も使われる

Q31
カレントディレクトリのファイル中で、「ba」から始まる4文字のファイルやディレクトリを表示する

A31
```
ls ba??
```

「ls」は「list」の略

「?」は任意の1文字を表現しているので、「ba」で始まる4文字のファイルを指定していることになる

このようにファイル名の一部を「?」などを用いてパターンで指定することにより、
複数ファイルを指定する機能を「パス名展開」という

パス名展開では、任意の1文字を表す「?」のほかに、任意の文字列を表す「*」も使われる

Q32
コマンドの「オプション」とはなにか？

A32
コマンドラインに、ハイフン「-」で始まる引数を追加することで、デフォルトとは違う動作をするようにするための機能のこと

例えば、「lsコマンド」に「-aオプション」をつけて「ls -a」というようにプロンプトの後ろに入力すると、隠しファイルも含めたすべてのファイルが表示されるようになる

オプションには、「-a」のような短いものだけでなく、「--quate-name」のような長いオプションである「ロングオプション」という種類もある
引数をとるオプションもある

Q33
カレントディレクトリのファイルとディレクトリを、詳細情報を含めて一覧表示する

A33
```
ls -l
```

「ls」は「list」の略

「-l」は「long」の略、長く表示→詳細に表示

「-l」オプションをつけると、パーミッションやファイルのオーナーなどの情報を見ることができる

Q34
カレントディレクトリのファイルとディレクトリを、隠しファイルを含めて一覧表示する

A34
```
ls -a
```

「ls」は「list」の略

「-a」は「all」の略、隠しファイルも含めてすべて（all）ということ

「-a」オプションをつけると、「.gitignore」のようなピリオド「.」から始まる隠しファイルも表示される

Q35
カレントディレクトリのファイルとディレクトリを、ファイル種別を含めて一覧表示する

A35
```
ls -F
```

「ls」は「list」の略

「-F」は「classify（種別、分類）」から

「-F」オプションをつけると、「bin@」や「home/」のようにファイル種別を末尾に含めて表示される

ファイル種別と記号の対応は以下
・通常ファイル：なし
・ディレクトリ：「/」
・実行可能ファイル：「*」

・シンボリックリンク：「@」

Q36
カレントディレクトリに「exapmle」という名前のディレクトリを作成する

A36
```
mkdir example
```

「mkdir」は「make directory」の略

「sales/2025/01」のような深いディレクトリを一気に作成するときは、
以下のように「-p」オプションをつける必要がある
```
mkdir -p sales/2025/01
```

「-p」は「parents」の略、「親ディレクトリ（parent）もまとめて」ということ

Q37
カレントディレクトリに「sales/2025/01」というディレクトリの階層を作成する

A37
```
mkdir -p sales/2025/01
```

「-p」は「parents」の略、「親ディレクトリ（parent）もまとめて」ということ

「sales/2025/01」のような深いディレクトリを一気に作成するときは、
「-p」オプションをつける必要がある

Q38
「newfile」という名前のファイルを新規作成する

A38
```
touch newfile
```

「touch」は本来、ファイルのタイムスタンプを変更するための機能

指定したファイルが存在しない場合には新規ファイルが作成されるので、
空ファイルの新規作成に使われる

Q39
「newfile」という名前のファイルを削除する

A39
```
rm newfile
```

「rm」は「remove（削除する）」の略

Q40
「newdir」という名前のディレクトリを削除する
中に入っているファイルも、まとめて削除するものとする

A40
```
rm -r newdir
```

「rm」は「remove（削除する）」の略

「-r」は「recursive（再帰的な）」の略、再帰的にディレクトリツリーを削除する

Q41
カレントディレクトリのファイル中で、拡張子が「.html」のファイルを削除する

A41
```
rm *.html
```

「rm」は「remove（削除する）」の略

「*」は任意の文字列を表現しているので、「.html」で終わるすべてのファイルを指定していることになる

このようにファイル名の一部を「*」などを用いてパターンで指定することにより、
複数ファイルを指定する機能を「パス名展開」という
パス名展開では、任意の文字列を表す「*」のほかに、任意の1文字を表す「?」も使われる

Q42
カレントディレクトリにある、「kara」という空のディレクトリを削除する

A42
```
rmdir kara
```

「rmdir」は「remove directory」の略

「rm」と違って、ディレクトリの中に何らかのファイルがあるとエラーになる
なので、隠しファイルなどに気づかず、間違えて必要なファイルを消してしまう心配がない

Q43
「/etc/crontab」というファイルを表示する

A43
```
cat /etc/crontab
```

「cat」は「concatenate（連結する）」の略

以下のようにファイルを複数指定すると、
それらのファイルを「連結」して順番に表示するので「concatenate」
```
cat /etc/crontab /etc/hostname
```

Q44
「/etc/crontab」というファイルを行番号をつけて表示する

A44

```
cat -n /etc/crontab
```

「cat」は「concatenate（連結する）」の略

「-n」は「number」の略、行"番号"のこと

Q45
「/etc/crontab」をスクロール表示する

A45
```
less /etc/crontab
```

「less」は「more」の対義語

「more」というスクロール表示するためのコマンドもあり、そのコマンドをより使いやすくしたのが「less」
「more」でできることは「less」ですべてできるので「less」が使えればOK

スクロールやlessコマンドの終了には、専用のコマンドを入力する必要がある
最低限覚えておくものとして、以下のようなコマンドがある
・1画面下にスクロールする：スペースキー
・1画面上にスクロールする：「b」キー
・lessコマンドを終了する：「q」キー

Q46
lessコマンドの実行中に、以下の操作を行うためのコマンドはそれぞれなにか？
・1画面下にスクロールする
・1画面上にスクロールする
・1行下にスクロールする
・1行上にスクロールする
・lessコマンドを終了する

A46
lessコマンド実行中（スクロール表示中）に、
スクロールやlessコマンドを終了するには、専用のコマンドを入力す

る必要がある

以下のようなコマンドがある
・1画面下にスクロールする：スペースキー
・1画面上にスクロールする：「b」キー
・1行下にスクロールする：「j」キー
・1行上にスクロールする：「k」キー
・lessコマンドを終了する：「q」キー、「quit（終了する）」

「less」は「more」の対義語

「more」というスクロール表示するためのコマンドもあり、そのコマンドをより使いやすくしたのが「less」
「more」でできることは「less」ですべてできるので「less」が使えればOK

Q47
lessコマンドの実行中に、以下の操作を行うためのコマンドはそれぞれなにか？
・文字列を検索する
・次の検索結果に移動する
・前の検索結果に移動する

A47
・文字列を検索する
/文字列（「/etc」と入力すると「etc」という文字列を検索する）

・次の検索結果に移動する
「n」キー、「next」の意味

・前の検索結果に移動する
Shift + 「n」キー

「less」は「more」の対義語

「more」というスクロール表示するためのコマンドもあり、そのコマンドをより使いやすくしたのが「less」
「more」でできることは「less」ですべてできるので「less」が使

えればOK

Q48
カレントディレクトリの「file」というファイルを、「cpfile」という名前でコピーする

A48
```
cp file cpfile
```

「cp」は「copy」の略

「cp コピー元 コピー先」というように書く

Q49
カレントディレクトリの「file1」「file2」という2つのファイルを、「dir1」にコピーする

A49
```
cp file1 file2 dir1
```

「cp」は「copy」の略

「cp コピー元 コピー先」というように書く

コピー元は複数書くこともできる
コピー先をディレクトリにすると、コピー元のファイルをコピー先のディレクトリに同じファイル名でコピーする

Q50
カレントディレクトリの「dir1」というディレクトリを、「dir2」という名前でコピーする

A50
```
cp -r dir1 dir2
```

「cp」は「copy」の略

「cp オプション コピー元 コピー先」というように書く

「-r」は「recursive（再帰的な）」の略、再帰的にディレクトリツリーをコピーする

Q51
カレントディレクトリの「file1」というファイルを、「file2」という名前に変更する

A51
```
mv file1 file2
```

「mv」は「move」の略

「mv 移動元 移動先」というように書く

移動先にディレクトリの名前以外を指定すると、名前の変更になる
移動先にディレクトリを指定すると、ファイルやディレクトリの移動になる

Q52
カレントディレクトリの「file1」「file2」という2つのファイルを、「dir1」という名前のディレクトリに移動する

A52
```
mv file1 file2 dir1
```

「mv」は「move」の略

「mv 移動元 移動先」というように書く

移動元には、複数のファイルやディレクトリを指定することができる
移動先にディレクトリを指定すると、ファイルやディレクトリの移動になる

Q53
リンクとはなにか？

A53

ファイルに別名をつける機能のこと

1つのファイルに複数の名前を付ける「ハードリンク」と、
リンク先のパスが書かれた小さな特殊ファイルを作る「シンボリックリンク」の２つがある
シンボリックリンクの方が後にできたのもあり使いやすいので、シンボリックリンクが使われることが多い

用途としては以下のようなものがある
・長いパス名の省略（ショートカット）
「dir1/dir2/dir3/file」というパスを、別名をつけることで「file」という名前で呼び出せるようにすれば、パス指定が楽になる

・別のファイルを同じ名前で呼べるようになるので、呼び出し方を変更しなくてよくなる
例えば、バージョン番号を含んだファイルを、「latest」といった別名をつけて利用するなど
こうしておくと、バージョンを変えるときに「latest」のリンク先を新しいバージョンのファイルに変えれば、呼び出し方は「latest」のまま変えずに、参照するファイルを変えることができるようになる

Q54
「ハードリンク」と「シンボリックリンク」の違いは？

A54
リンクという「ファイルに別名をつける機能」の2つの種類として、
「ハードリンク」と「シンボリックリンク」がある

・ハードリンク
1つのファイルに複数の名前を付ける機能
ディレクトリに対して設定できないなど不便な点もあることから、シンボリックリンクの方がよく使われる

・シンボリックリンク
リンク先のパス名が書かれた小さな特殊ファイル
ハードリンクのように1つのファイルに別名をつけるのではなく、ファイルを参照するファイルを新しく作る

あるファイルを参照したいときに、そのファイルを直接指定するか、そのファイルを参照するファイルを指定するかという2通りの方法があるので、これは別名をつけているとも解釈できる
ディレクトリに対しても設定することができる

Q55
カレントディレクトリの「file1」に、「file2」という名前のハードリンクを作成する

A55
```
ln file1 file2
```

「ln」は「link」の略

「ln リンクをつけるファイル リンク名」というように書く

こうした後に「catコマンド」などで「file2」を表示すると、「file1」と同じ内容が表示される
「file1」を削除したうえで「catコマンド」などで「file2」を表示しても、別名の一つが消えたに過ぎないので「file1」と同じ内容が表示される

Q56
カレントディレクトリの「file1」に、「file2」という名前のシンボリックリンクを作成する

A56
```
ln -s file1 file2
```

「ln」は「link」の略

「-s」は「symbolic（シンボリック）」の略
「ln -s リンクをつけるファイル リンク名」というように書く

こうした後に「catコマンド」などで「file2」を表示すると、「file1」と同じ内容が表示される
「file1」を削除したうえで「catコマンド」などで「file2」を表示すると、参照ファイルがなくなるのでエラーになる

Q57
「lsコマンド」のヘルプメッセージを表示して、オプションの一覧を見る

A57
```
ls --help
```

多くのLinuxコマンドでは、「--help」オプションを指定すれば、ヘルプメッセージを見ることができる

ヘルプメッセージでは以下のような内容を確認できる
・コマンドの概要
・使用方法
・オプションなどの意味

より詳細な情報を表示したいなら、マニュアルを表示する「manコマンド」を以下のように使う
```
man 調べたいコマンド名
```

Q58
「catコマンド」のマニュアルを表示する

A58
```
man cat
```

「man」は「manual」の略

マニュアルでは「--help」オプションよりも詳細な情報を見ることができる

manコマンドを使うと、lessコマンドを使ってマニュアルがスクロール表示される
なので、スペースキーで下スクロール、「q」キーで終了など、操作方法はlessコマンドと共通になっている

Q59
コマンドの履歴を表示する

A59
`history`

`history`コマンドを実行すると、過去に実行したコマンドが「`5 ls`」というように、番号とともに表示される

「`!`番号」と入力すると`history`コマンドを実行したときに書いている番号のコマンドを再び実行できる
なので、「`5 ls`」と書いてあったなら、「`!5`」とすると「`ls`」が実行される

Q60
カレントディレクトリ以下のすべてのファイルから、拡張子が「`.txt`」のファイルを検索して表示する

A60
`find . -name '*.txt' -print`

「`find` 検索開始ディレクトリ 検索条件 アクション」というように書く
検索開始ディレクトリ：「`.`」、カレントディレクトリから開始して
検索条件：「`-name '*.txt'`」、ファイル名（`-name`）が「任意の文字列`.txt`」（`*.txt`）で終わるファイルを
アクション：「`-print`」、表示する

ファイル名で検索するときには「`-name`」を検索条件に指定し、続けて検索ファイル名を書く
アクションが「`-print`」の場合は省略可能

「`'*.txt'`」は「パス名展開」ではなく、単に「ワイルドカード」を用いた指定であることに注意
シングルコーテーションで囲まないと、シェルに「パス名展開」として解釈されてしまうので注意
パス名展開では「`*`」や「`?`」で書かれたパターンを、具体的なファイルのリストに変換して実行する

例えば、カレントディレクトリに「`A.txt`」と「`B.txt`」があるとす

ると、
シングルコーテーションで囲まず「パス名展開」として解釈された場合、以下のように変換されて実行される
`find . -name A.txt B.txt -print`

こうすると、このコマンドの意味が
「カレントディレクトリ以下のすべてのファイルから`A.txt`と`B.txt`という2つのファイルを検索して表示する」という
本来の目的とは違う意味になってしまう
パス名展開ではこのように、「`*`」や「`?`」で書かれたパターンを、具体的なファイルのリストに変換して実行する

なので、シングルコーテーションで囲むことで、ワイルドカードとして実行してあげる必要がある
ワイルドカードとして利用する場合でもパス名展開と同じく、「`*`」は「任意の文字列」、「`?`」は「任意の1文字」を表現している

Q61
カレントディレクトリ以下のすべてのディレクトリを表示する

A61
`find . -type d -print`

「`find` 検索開始ディレクトリ 検索条件 アクション」というように書く

検索開始ディレクトリ：「`.`」、カレントディレクトリから開始して
検索条件：「`-type d`」、ファイルタイプ（`-type`）が「`d`」（ディレクトリ）を
アクション：「`-print`」、表示する

ファイルタイプには「`f`（ファイル）」「`d`（ディレクトリ）」「`l`（シンボリックリンク）」などがある

アクションが「`-print`」の場合は省略可能

Q62
カレントディレクトリ以下のすべてのファイルから、「`latest`」とい

う名前のシンボリックシンクを検索して表示する

A62

```
find . -type l -a -name latest -print
```

検索条件が2つ以上あるときには、以下のように「-a（AND）」で区切って、
「find 検索開始ディレクトリ 検索条件1 -a 検索条件2 アクション」というように書く
「-a」を省略して、「find 検索開始ディレクトリ 検索条件1 検索条件2 アクション」とも書ける

検索開始ディレクトリ：「.」、カレントディレクトリから開始して
検索条件1：「-type l」、ファイルタイプ（-type）が「l」（シンボリックリンク）で
検索条件2：「-name latest」、ファイル名（-name）が「latest」のファイルを
アクション：「-print」、表示する

ファイルタイプには「f（ファイル）」「d（ディレクトリ）」「l（シンボリックリンク）」などがある
ファイル名で検索するときには「-name」を検索条件に指定し、続けて検索ファイル名を書く
アクションが「-print」の場合は省略可能

Q63

「bash」という文字列を含むパス名を高速に検索する

A63

```
locate bash
```

「locate」は「見つける」という意味

「findコマンド」はディレクトリツリーを下ってファイルを探すなので、ファイルが深い場所にあったり、大量のファイルがあったりすると非常に時間がかかる
一方「locateコマンド」は事前にファイルパスのデータベースが作られていて、そこから検索するので高速な検索が可能

デフォルトの設定では、ファイルをデータベースへの登録を1日1回行う
なので、作ったばかりのファイルはデータベースに登録されておらず、locateでは検索できない
手動でデータベースへの登録を行うこともできるので、すぐに検索したい場合はそうする
locateは初期状態のLinuxにはインストールされていない場合も多いので、その場合はインストールする必要がある

Q64
「bash」と「doc」という2つの文字列をどちらも含むパスを高速に検索する

A64
```
locate -A bash doc
```

AND検索を行うには「-A」オプションを指定する
OR検索を行うなら「-A」オプションは不要で、以下のように書く
```
locate bash doc
```

「locate」は「見つける」という意味

「findコマンド」はディレクトリツリーを下ってファイルを探す
なので、ファイルが深い場所にあったり、大量のファイルがあったりすると非常に時間がかかる
一方「locateコマンド」は事前にファイルパスのデータベースが作られていて、そこから検索するので高速な検索が可能

デフォルトの設定では、ファイルをデータベースへの登録を1日1回行う
なので、作ったばかりのファイルはデータベースに登録されておらず、locateでは検索できない
手動でデータベースへの登録を行うこともできるので、すぐに検索したい場合はそうする
locateは初期状態のLinuxにはインストールされていない場合も多いので、その場合はインストールする必要がある

Q65
「document」と「doc」という2つの文字列のどちらかを含むパスを高速に検索する

A65
```
locate document doc
```

OR検索を行うなら「-A」オプションは不要
一方、AND検索を行うには「-A」オプションを指定して、以下のように書く
```
locate -A document doc
```

「locate」は「見つける」という意味

「findコマンド」はディレクトリツリーを下ってファイルを探す
なので、ファイルが深い場所にあったり、大量のファイルがあったりすると非常に時間がかかる
一方「locateコマンド」は事前にファイルパスのデータベースが作られていて、そこから検索するので高速な検索が可能

デフォルトの設定では、ファイルをデータベースへの登録を1日1回行う
なので、作ったばかりのファイルはデータベースに登録されておらず、locateでは検索できない
手動でデータベースへの登録を行うこともできるので、すぐに検索したい場合はそうする
locateは初期状態のLinuxにはインストールされていない場合も多いので、その場合はインストールする必要がある

Q66
カレントディレクトリの「file」というファイルから、「bin」という文字列を含む行だけを出力する

A66
```
grep bin file
```

「grep」は「global regular expression print」の略

「ファイル全体（global）のうち、正規表現（regular expression）に一致する行を出力（print）」という意味

正規表現を使わずに文字列を検索することもでき、
その場合は「grep 検索文字列 検索するファイル」というように書く

Q67
「/etc」ディレクトリにある、「cron」という文字列を含むファイルやディレクトリを表示する

A67
```
ls /etc | grep cron
```

「grep」は「global regular expression print」の略

「ファイル全体（global）のうち、正規表現（regular expression）に一致する行を出力（print）」という意味

正規表現を使わずに文字列を検索することもでき、その場合は「grep 検索文字列 検索するファイル」というように書く
パイプラインを使うことで「lsコマンド」などの結果を入力にして、文字列を検索することもできる

Q68
Linuxのファイルにおいて、「オーナー」と「グループ」とはそれぞれなにか？

A68
・オーナー
ファイルの所有者のこと
Linuxで扱われるすべてのファイルにはオーナーが設定されており、ファイルのオーナーはファイルへのアクセス権限を自由に設定できる

・グループ
ユーザーをまとめた集まりのこと
グループ単位でファイルへのアクセス権限を設定することができる
例えば、システム管理を行うユーザーをwheelグループに所属させたうえで、

wheelグループに権限を付与することで、システム管理者にまとめて権限を与えられるようになる

Q69
パーミッションとはなにか？

A69
ファイルのひとつひとつに設定されている、「誰に、どのような操作を許可するか？」という権限のこと
「ls -l」を実行すると、「-rwxr-xr-x」というように表示される部分がパーミッションを表す

「-rwxr-xr-x」というパーミッションなら、以下の意味になる
・通常ファイル（-）
・オーナーは読み取り、書き込み、実行ができる（rwx）
・グループは読み取り、実行ができる（r-x）
・その他のユーザーは読み取り、実行ができる（r-x）

Q70
「ls -l」コマンドを用いて、あるディレクトリのパーミッションを調べたところ、「drwxrw-r-- tanaka teamA ..(省略)..」と書いてあった
このパーミッションはどのような意味か？

A70
以下の意味になる
・ディレクトリ
・tanakaユーザーのパーミッションは読み取り、書き込み、実行
・teamAグループに所属するユーザーのパーミッションは読み取り、書き込み
・その他のユーザーのパーミッションは読み取り

「tanaka teamA」はファイルのオーナーと、ファイルが所属するグループを指している
「tanaka」がファイルのオーナーで、「teamA」がファイルの所属グループ
teamAグループに属するユーザーは、そのグループのパーミッション

が与えられる

「drwxrw-r--」は「d」「rwx」「rw-」「r--」という4つからなり、左から以下のように解釈できる
「d」：ファイルの種類を表す、「-（通常ファイル）」「d（ディレクトリ）」「l（シンボリックリンク）」
「rwx」：オーナー（tanaka）のパーミッション、「r（読み取り）」「w（書き込み）」「x（実行）」が許されている
「rw-」：グループ（teamA）に所属するユーザーのパーミッション、「r（読み取り）」「w（書き込み）」が許されている
「r--」：その他のユーザーのパーミッション、「r（読み取り）」が許されている

ディレクトリにおいて「r（読み取り）」「w（書き込み）」「x（実行）」とは、それぞれ以下のような意味
「r」：ディレクトリに含まれるファイル一覧の取得
「w」：ディレクトリの下にあるファイル・ディレクトリの作成・削除
「x」：ディレクトリをカレントディレクトリにする

Q71
「ls -l」コマンドを用いて、あるファイルのパーミッションを調べたところ、「-rwxr-xr-x root wheel ..(省略)..」と書いてあった
このパーミッションはどのような意味か？

A71
以下の意味になる
・ファイルは通常ファイル
・rootユーザーのパーミッションは読み取り、書き込み、実行
・wheelグループに所属するユーザーのパーミッションは読み取り、実行
・その他のユーザーのパーミッションは読み取り、実行

「root wheel」はファイルのオーナーと、ファイルが所属するグループを指している
「root」がファイルのオーナーで、「wheel」がファイルの所属グループ

wheelグループに属するユーザーは、そのグループのパーミッションが与えられる

「-rwxr-xr-x」は「-」「rwx」「r-x」「r-x」という4つからなり、左から以下のように解釈できる
「-」：ファイルの種類を表す、「-（通常ファイル）」「d（ディレクトリ）」「l（シンボリックリンク）」
「rwx」：オーナー（root）のパーミッション、「r（読み取り）」「w（書き込み）」「x（実行）」が許されている
「r-x」：グループ（wheel）に所属するユーザーのパーミッション、「r（読み取り）」「x（実行）」が許されている
「r-x」：その他のユーザーのパーミッション、「r（読み取り）」「x（実行）」が許されている

Q72
「file.txt」のパーミッションを「- rwx r-x r-x」に設定する

A72
```
chmod 755 file.txt
```

パーミッションを変更するには、ファイルモードを変更する「chmodコマンド」を用いる

「chmod」は「change mode」の略

「chmodコマンド」には「数値モード」と「シンボルモード」があり、ここでは数値モードで記述している
元のパーミッションにかかわらず、新しいパーミッションに変更したい場合には、数値モードが便利

数値モードでは、
「r」を「4」
「w」を「2」
「x」を「1」
というように数値を割り振ることで、
「rwx」→「4+2+1=7」
「r-x」→「4+1=5」
というように、パーミッションを数字で表現できる

Q73
「file.txt」のパーミッションを「- rwx r-- r--」に設定する

A73
```
chmod 744 file.txt
```

パーミッションを変更するには、ファイルモードを変更する「chmodコマンド」を用いる

「chmod」は「change mode」の略

「chmodコマンド」には「数値モード」と「シンボルモード」があり、ここでは数値モードで記述している
元のパーミッションにかかわらず、新しいパーミッションに変更したい場合には、数値モードが便利

数値モードでは、
「r」を「4」
「w」を「2」
「x」を「1」
というように数値を割り振ることで、
「rwx」→「4+2+1=7」
「r-x」→「4+1=5」
というように、パーミッションを数字で表現できる

Q74
「file.txt」のパーミッションを「- -wx -w- --x」に設定する

A74
```
chmod 321 file.txt
```

パーミッションを変更するには、ファイルモードを変更する「chmodコマンド」を用いる

「chmod」は「change mode」の略

「chmodコマンド」には「数値モード」と「シンボルモード」があ

り、ここでは数値モードで記述している
元のパーミッションにかかわらず、新しいパーミッションに変更したい場合には、数値モードが便利

数値モードでは、
「r」を「4」
「w」を「2」
「x」を「1」
というように数値を割り振ることで、
「rwx」→「4+2+1=7」
「r-x」→「4+1=5」
というように、パーミッションを数字で表現できる

Q75
「file.txt」のその他のユーザーのパーミッションを「r--」から「r-x」に変更する

A75
```
chmod o+w file.txt
```

パーミッションを変更するには、ファイルモードを変更する「chmodコマンド」を用いる

「chmod」は「change mode」の略

「chmodコマンド」には「数値モード」と「シンボルモード」があり、ここではシンボルモードで記述している
パーミッションの一部だけを変更したい場合には、シンボルモードが便利

シンボルモードでは「誰に」「何を」「どうする」かを指定する
「誰に」：「u（オーナー）」「g（グループ）」「o（その他のユーザー）」「a（ugoすべて）」
「何を」：「r（読み取り）」「w（書き込み）」「x（実行）」
「どうする」：「+（権限の追加）」「-（権限の削除）」「=（指定した権限と等しくする）」

「o+w」→その他のユーザー（o）に書き込み権限（w）を追加する（+

）
「go-x」→グループに所属するユーザー（g）とその他のユーザー（o）から実行権限（x）を削除する（-）
「o=rw」→その他のユーザー（o）の権限を「rw-」にする

解答は「chmod o=rx file.txt」でもOK

Q76
スーパーユーザーとはなにか？

A76
管理者権限をもつ特別なユーザーのこと
ユーザー名が「root」であることから、「rootユーザー」とも呼ばれる

スーパーユーザーはあらゆる操作が許可された強い権限を持つユーザーで、
システムの設定ファイルの変更や、アプリケーションのインストールなどができる

スーパーユーザーはファイルのパーミッションの影響を受けずに、
すべてのファイルに対して読み込み、書き込み、実行が可能

権限が強いあまり、Linuxの動作に必要なファイルも削除できてしまうので、扱いには細心の注意が必要

Q77
スーパーユーザーになる

A77
su

「su」は「substitute user（代わりのユーザー）」の略

「su」は本来ユーザーを切り替えるためのコマンドだが、主にスーパーユーザーになるために使われる

「suコマンド」を実行すると、スーパーユーザーのパスワードの入力

が求められる
パスワードを入力してスーパーユーザーになると、プロンプトが「$」から「#」に変わる

スーパーユーザーから一般ユーザーに戻るには「exit」と入力する

Q78
「/etc/shadow」というパスワードファイルの内容を、スーパーユーザーとして「catコマンド」を実行することで表示する

A78
```
sudo cat /etc/shadow
```

「sudo」は「substitute user do（代わりのユーザーが実行）」の略

「sudo 実行したいコマンド」と書くことで、コマンドをスーパーユーザーとして実行できる
「sudo」も「su」と同様に、本来はユーザーを切り替えてコマンドを実行するためのコマンドだが、主にスーパーユーザーとして実行するために使われる

「sudo 実行したいコマンド」と書くと、現在ログインしているユーザーのパスワードが求められる
パスワードを入力すると、コマンドがスーパーユーザーとして実行される

Q79
初心者のうちは、「suコマンド」と「sudoコマンド」はどのように使い分けるべきか？

A79
基本的には「sudoコマンド」で済まし、「sudoコマンド」が使えない場面では「suコマンド」を使う

スーパーユーザーは良くも悪くも何でもできてしまい、
Linuxを動作させるのに必須のファイルも変更・削除できてしまう

なので、誤操作で致命的な変更を加えないようにするために、
スーパーユーザーを限定的に利用できる「sudoコマンド」を中心に使うのがよい

Q80
標準入出力とはなにか？

A80
標準入力（stdin）、標準出力（stdout）、標準エラー出力（stderr）の３つをまとめて標準入出力という

・標準入力（stdin）
プログラムの標準的な入力、キーボードやファイルなどが使われる

・標準出力（stdout）
プログラムの標準的な出力、ディスプレイやファイルなどが使われる

・標準エラー出力（stderr）
プログラムのエラーメッセージを出力するためのもの
標準出力と同じく、ディスプレイやファイルなどが使われる

各コマンドは「キーボードから入力されている」とか「ディスプレイに出力する」とは考えておらず、
シンプルに「標準入力から入力し、標準出力に出力する」だけ

「標準入力はキーボード、標準出力はディスプレイ、標準エラー出力はファイル」というように、
標準入出力をどこに繋ぐかはユーザーが自由に設定できる

もし「キーボードから入力」とか「ディスプレイに出力」というように入出力先が固定されていると、出力をディスプレイからファイルに変えたときに、コマンドの内容を変更しなければいけなくなってしまう
特定の入出力先に依存させるのではなく、「標準入力から入力し、標準出力に出力する」という標準入出力を土台にして、入出力をユーザーが自由に切り替えられるようなインターフェースにすることで、コマンドの内容を変更しなくてもいいようにしている

Q81
リダイレクトとはなにか？

A81
標準入出力を変更するための機能のこと

Q82
カレントディレクトリのファイルとディレクトリを一覧表示した結果を、「ls.txt」という名前のテキストファイルとして保存する

A82
```
ls > ls.txt
```

「>」はリダイレクト（標準入出力の切り替え）のための記号
標準出力をターミナルのディスプレイから、ファイルに変更している

リダイレクトの書き方は以下
`< file`：標準入力を`file`に変更
`> file`：標準出力を`file`に変更
`2> file`：標準エラー出力を`file`に変更
`>> file`：標準出力の出力を`file`の末尾に追記
`2>> file`：標準エラー出力の出力を`file`の末尾に追記
`>file 2>&1`：標準出力と標準エラー出力を`file`に変更

Q83
「/notexist」という存在しないディレクトリ内のファイルとディレクトリを、一覧表示しようとしたときに発生するエラーメッセージを「error.txt」というテキストファイルに保存する

A83
```
ls /notexist 2> error.txt
```

「2>」はリダイレクト（標準入出力の切り替え）のための記号
標準エラー出力をターミナルのディスプレイから、ファイルに変更している

リダイレクトの書き方は以下
`< file`：標準入力を`file`に変更

> file：標準出力をfileに変更
2> file：標準エラー出力をfileに変更
>> file：標準出力の出力をfileの末尾に追記
2>> file：標準エラー出力の出力をfileの末尾に追記
>file 2>&1：標準出力と標準エラー出力をfileに変更

Q84
「cat file」というコマンドのエラーメッセージだけを、ターミナルのディスプレイに表示する

A84
```
cat file > /dev/null
```

「/dev/null」はスペシャルファイルと呼ばれる特殊なファイルで、以下のような性質を持つ

・入力先に指定すると、何もデータを返さない
・出力先に指定すると、書き込まれたデータをすべて破棄する

このような性質から「/dev/null」は以下の用途で用いられる
・エラーメッセージだけを表示したい（> /dev/null）
・エラーメッセージを非表示にしたい（2> /dev/null）

Q85
「/dev/null」とはなにか？

A85
「/dev/null」はスペシャルファイルと呼ばれる特殊なファイルで、以下のような性質を持つ

・入力先に指定すると、何もデータを返さない

・出力先に指定すると、書き込まれたデータをすべて破棄する

このような性質から「/dev/null」は以下の用途で用いられる

・エラーメッセージだけを表示したい（> /dev/null）

・エラーメッセージを非表示にしたい（2> /dev/null）

Q86
パイプラインとはなにか？

A86
コマンドの標準出力を別のコマンドの標準入力につなぐための機能
パイプラインによって、シンプルなLinuxコマンドを組み合わせて、
複雑な処理を実現することができるようになる

Q87
「/」のファイルとディレクトリを一覧表示した結果を、行番号付きで表示する

A87
```
ls / | cat -n
```

「|」によって、コマンドの標準出力を別のコマンドの標準入力につなぐ「パイプライン」を利用できる
「ls /」の標準出力を、「cat -n」の標準入力に繋いでいる

「ls /」の標準出力は、「/」のファイルとディレクトリを一覧表示
これを「cat -n」（行番号付きで表示）の標準入力にすれば実現できる

Q88
フィルタとはなにか？

A88
標準入力を入力として受け取り、標準出力に出力するコマンドのこと

入力をそのまま出力する「catコマンド」や、入力の先頭部分だけを出力する「headコマンド」など、
たくさんのフィルタがLinuxには存在している

フィルタのひとつひとつはごく簡単な機能しか持たないが、

複数のフィルタをパイプラインでつなぐことによって、複雑な処理を行うことができる

Q89
カレントディレクトリにある「file」の行数を表示する

A89
```
wc -l file
```

「wc」は「word count」の略

「-l」オプションは「lines」の略で、行数を数える

その他のオプションとしては以下
「-w」オプションは「words」の略で、単語数を数える

Q90
ルートディレクトリの直下のファイルとディレクトリの数をカウントして表示

A90
```
ls / | wc -l
```

「wc」は「word count」の略

「-l」オプションは「lines」の略で、行数を数える

「ls」とパイプラインで組み合わせることで、ファイルやディレクトリの数をカウントできる

Q91
カレントディレクトリにある「file」というファイルの内容を、アルファベット順に並べ替える

A91
```
sort file
```

「sort」はその名の通り、並べ替えのためのフィルタ

オプションとしては、以下のようなものがある
「-n」オプションは「number」の略で、数値順に並べ替える
「-r」オプションは「reverse」の略で、デフォルトの逆順（降順）に並べ替える（デフォルトは昇順）

Q92
カレントディレクトリにある「file」というファイルの内容を、数値が小さい順に並べ替える

A92
```
sort -n file
```

「sort」はその名の通り、並べ替えのためのフィルタ

「-n」オプションは「number」の略で、数値順に並べ替える

その他のオプションとしては、以下のようなものがある
「-r」オプションは「reverse」の略で、デフォルトの逆順（降順）に並べ替える（デフォルトは昇順）

Q93
カレントディレクトリにある「file」というファイルの内容を、数値が大きい順に並べ替える

A93
```
sort -rn file
```

「sort」はその名の通り、並べ替えのためのフィルタ

「-n」オプションは「number」の略で、数値順に並べ替える
「-r」オプションは「reverse」の略で、デフォルトの逆順（降順）に並べ替える（デフォルトは昇順）

「-rn」とすることで、複数のオプションを同時に指定している

Q94
カレントディレクトリにある「file」というファイルの内容を、重複

行を取り除いて表示する

A94

```
sort file | uniq
```

「uniq」は「unique（唯一）」の略

「uniq」では同じ内容の行が連続している場合のみ重複を取り除く仕様になっているので、
単に「uniq file」とするだけでは、離れた行にある重複を取り除けない
なので、「sort」で並べ替えることで重複行を集めてから「uniq」を実行する必要がある

「uniq」は常に「sort」とセットで使うのがよい

Q95

カレントディレクトリにある「file」というファイルの内容を、重複行を取り除いて表示する
ただし、重複数が多い順に並べ替えて表示するものとする

A95

```
sort file | uniq -c | sort -rn
```

「uniq」は「unique（唯一）」の略

「uniq」では同じ内容の行が連続している場合のみ重複を取り除く仕様になっているので、
単に「uniq file」とするだけでは、離れた行にある重複を取り除けない
なので、「sort」で並べ替えることで重複行を集めてから「uniq」を実行する必要がある

「-c」オプションは「count」の略で、行数をカウントするためのオプション
「-c」オプションを指定すると、重複数を表示したうえで重複が取り除かれる

その重複数が多い順に並べ替えるので、さらに「sort -rn（数値が大きい順に並べ替える）」を実行している

Q96
カレントディレクトリの「file」というファイルの、末尾5行を表示する

A96
```
tail -n 5 file
```

「tail」はファイルの末尾だけを表示するためのコマンド
「-n」オプションは「number」の意味で、表示する行数を指定する

Q97
カレントディレクトリの「file」というファイルの、先頭5行を表示する

A97
```
head -n 5 file
```

「head」はファイルの先頭だけを表示するためのコマンド
「-n」オプションは「number」の意味で、表示する行数を指定する

Q98
カレントディレクトリの「output.log」というログファイルの内容を監視して、変更をリアルタイムで確認できるようにする

A98
```
tail -f output.log
```

「-f」オプションは「follow（ついていく）」の略で、
ファイルの内容の更新を監視するためのオプション

上記コマンドを入力すると、「output.log」を監視し、
変更内容がリアルタイムに表示されるようになる

監視を終了するにはCtrl + 「c」キーを入力する

Q99
プロセスとはなにか？

A99
メモリ上で実行状態にあるプログラム（ファイル）のこと

コマンドの実態はディスク上に保存されたファイルであり、
シェルからコマンドを実行すると、Linuxカーネルはディスクからコマンドのファイルを読みだしてメモリに格納する

そのメモリに格納された内容に従って、CPUがプログラムを実行する
ここでメモリ上で実行状態にあるプログラムのことをプロセスという

Linuxカーネルから見た時の処理の単位、とも言える

Q100
現在のターミナルで動作しているプロセスを表示する

A100
ps

「ps」は「process status」の略

「psコマンド」を実行すると、以下のような項目が表示される
「PID」：プロセスID。プロセスIDはシステム全体で一意の値を持つので、プロセスIDを指定すればどのプロセスかわかる
「TTY」：ターミナル。デーモンは「?」と表示される
「CMD」：実行されているコマンドのパス。

Q101
「psコマンド」で、以下を実現するためのオプションは？
①別のターミナルやデーモンを含めたすべてのプロセスを表示
②すべてのユーザーのプロセスを表示
③詳細情報も含めてプロセスを表示

A101
①ps x
②ps a

③ps u

「ps」は「process status」の略

「psコマンド」では、BSDオプションと呼ばれるハイフンのつかないオプションも使われる

「psコマンド」は、以下のようなBSDオプションを持つ
「x」：別のターミナルやデーモンを含めたすべてのプロセスを表示
「a」：すべてのユーザーのプロセスを表示
「u」：詳細情報も含めてプロセスを表示

Q102
デーモンとはなにか？

A102
コンピュータ上で常時起動されている、特定の処理を行うプロセスのこと

例えば、Webサーバーはいつでも HTTPリクエストを処理できるように、
httpd（HTTPデーモン）というプロセスを常時動かしている

Q103
ジョブとはなにか？

A103
シェルのコマンドラインに入力している1つの行のこと
シェルから見た処理の単位、とも言える

コマンドが一つだけの場合、プロセスとジョブは同じ数（1つ）になる
一方で、パイプラインにより2つ以上のコマンドを組み合わせると、
ジョブは1つだが、プロセスはコマンドの数だけ生成される

Q104
実行中のコマンドを一時停止する

A104

Ctrl + 「z」キー

一時停止したコマンドは、「fgコマンド」や「bgコマンド」によって再開することが可能

Q105
現在のジョブの一覧を表示する

A105
jobs

「jobsコマンド」を実行すると、[1]や[2]というようにジョブ番号が表示される

ジョブ番号はシェルごとにシェル番号を持つので、
2つ以上のシェルを同時に使っているならジョブ番号は重複する

Q106
現在のジョブの一覧をプロセスIDを含めて表示する

A106
jobs -l

「jobsコマンド」を実行すると、[1]や[2]というようにジョブ番号が表示される

プロセスIDも含めて表示したい場合は「-l」オプションを用いる
「-l」は「long」の意味、長く表示→詳細に表示

Q107
停止状態にあるジョブ番号1のジョブを、フォアグラウンドにする

A107
fg %1

「fg %ジョブ番号」というように書く

「fg」は「foreground」の略

人間に見える部分がフォアグラウンド、見えない部分がバックグラウンド

ジョブの実行状態には「フォアグラウンド」「バックグラウンド」の2種類がある
それぞれは以下のような意味
フォアグラウンド：「ユーザーが対話的に操作できる」
バックグラウンド：「ユーザーが対話的に操作できない」

Q108
停止状態にあるジョブ番号1のジョブを、バックグラウンドにする

A108
`bg %1`

「`bg %`ジョブ番号」というように書く

「`bg`」は「`background`」の略
人間に見える部分がフォアグラウンド、見えない部分がバックグラウンド

ジョブの実行状態には「フォアグラウンド」「バックグラウンド」の2種類がある
それぞれは以下のような意味
フォアグラウンド：「ユーザーが対話的に操作できる」
バックグラウンド：「ユーザーが対話的に操作できない」

Q109
フォアグラウンドジョブを終了させる

A109
`Ctrl +` 「`c`」キー

バックグラウンドジョブを終了するには「`kill`コマンド」を使う

Q110
ジョブ番号1のバックグラウンドジョブを終了させる

A110
`kill %1`

「`kill %`ジョブ番号」というように書く

「`kill`」は「プロセス・ジョブを死なせる（終了する）」の意味

フォアグラウンドジョブを終了するには「`Ctrl + c`キー」を使う

Q111
プロセス`ID`「`4133`」のプロセスを終了させる

A111
`kill 4133`

「`kill` プロセス`ID`」というように書く

「`kill`」は「プロセス・ジョブを死なせる（終了する）」の意味

プロセスを終了させることができるのは、「実行したユーザー」か「スーパーユーザー」のどちらか

Q112
プロセス`ID`「`4133`」のプロセスを強制終了させる

A112
以下のどちらか
`kill -SIGKILL 4133`
`kill -9 4133`

「`kill`コマンド」は、正確に言うと「シグナルを送信するコマンド」

シグナルは、プロセスに送信される信号のことで、
プロセスは受け取ったシグナルの種類によって「終了」や「停止」などの動作をする

「`kill -l`」を実行することで、シグナル名とその番号（シグナル番号）の一覧ができる

代表的なシグナルとしては、以下のようなものがある
「TERM」：終了
「TSTP」：一時停止
「SIGKILL」：強制終了

Ctrl ＋ 「z」キーでジョブを一時停止したが、内部的には「TSTP」というシグナルが送られている
「kill -シグナル名（シグナル番号） プロセスID」というように書くことでシグナルを送れる
プロセスの強制終了のシグナルは、シグナル番号9の「SIGKILL」なので解答のようになる
