DMM WEBCAMP

Linuxを学ぼう

目次

- カーネルとシェルの関係
- プロセスとは
- コマンドを実行するということ
- 代表的なコマンド
- Linuxユーザとは
- Linuxグループとは
- パーミッションについて
- クライアントとサーバについて

DMM WEBCAMP

カーネルとシェルの関係

カーネルやシェルは何でしょうか？

DMM WEBCAMPで例えていきます

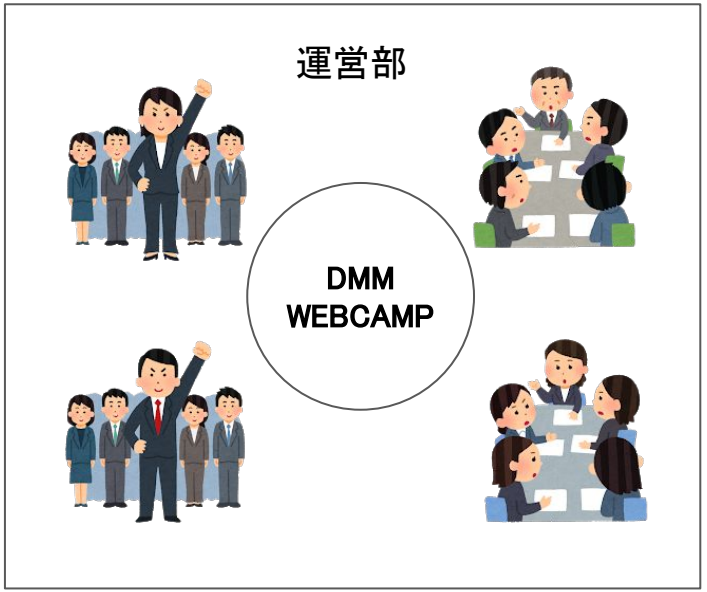
受講生の皆さんは運営部に課題のレビュー依頼や
わからないことがあればSlackから質問をします

受講生

依頼

回答

イメージ



DMM WEBCAMP

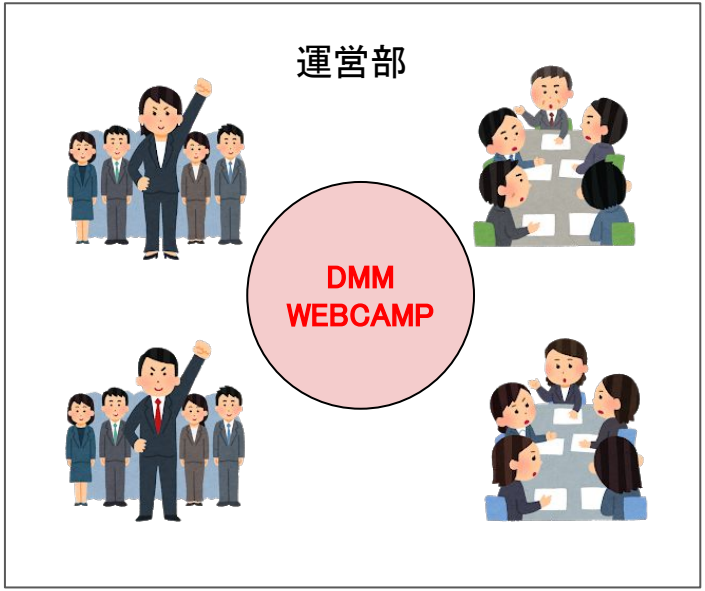
カーネルとシェルの関係

DMM WEBCAMPのサービスそのものが**カーネル**です

カーネルはサービス(OS)の核であり、
とても大事な部分を担っています。

DMM WEBCAMPがあるからこそ
運営部や受講生が存在し、
カーネルがあるからこそ
コマンドを実行したりアプリケーションを
動かすことができます。

イメージ



DMM WEBCAMP

カーネルとシェルの関係

DMM WEBCAMPのサービスに関して、

依頼の仲介役の運営部を**シェル**と言います。

運営部の役割は受講生からの依頼を受け、

DMM WEBCAMPのサービス内容を確認し

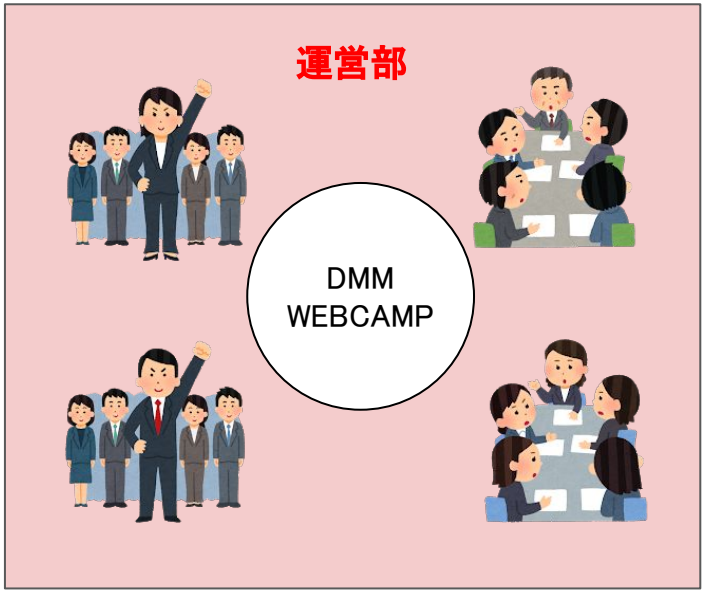
回答します。

それと同じで、

シェルはユーザからの依頼(コマンド)を

カーネルに伝え、結果をユーザに返します。

イメージ



DMM WEBCAMP

カーネルとシェルの関係

シェルに依頼をするためのツール(Slack)は
ターミナルやTera Termに置き換えられます。

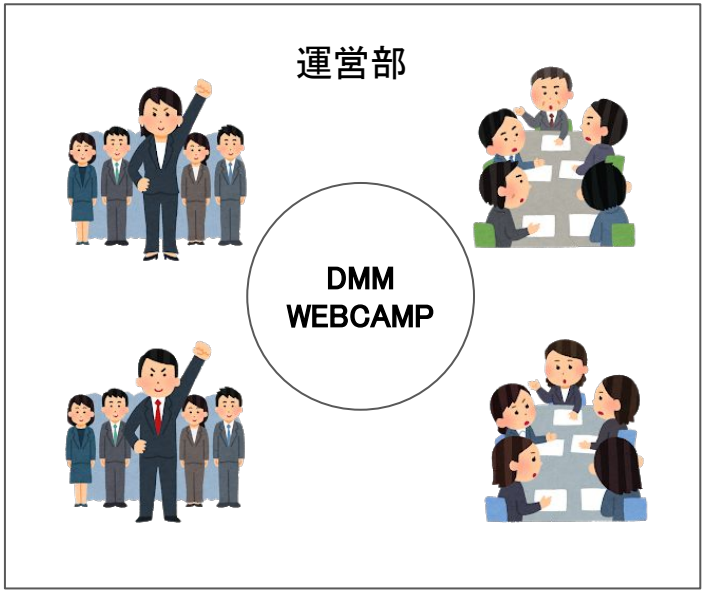
ターミナルやTera Termはシェルを表示させる
ツールです。

受講生

依頼

回答

イメージ

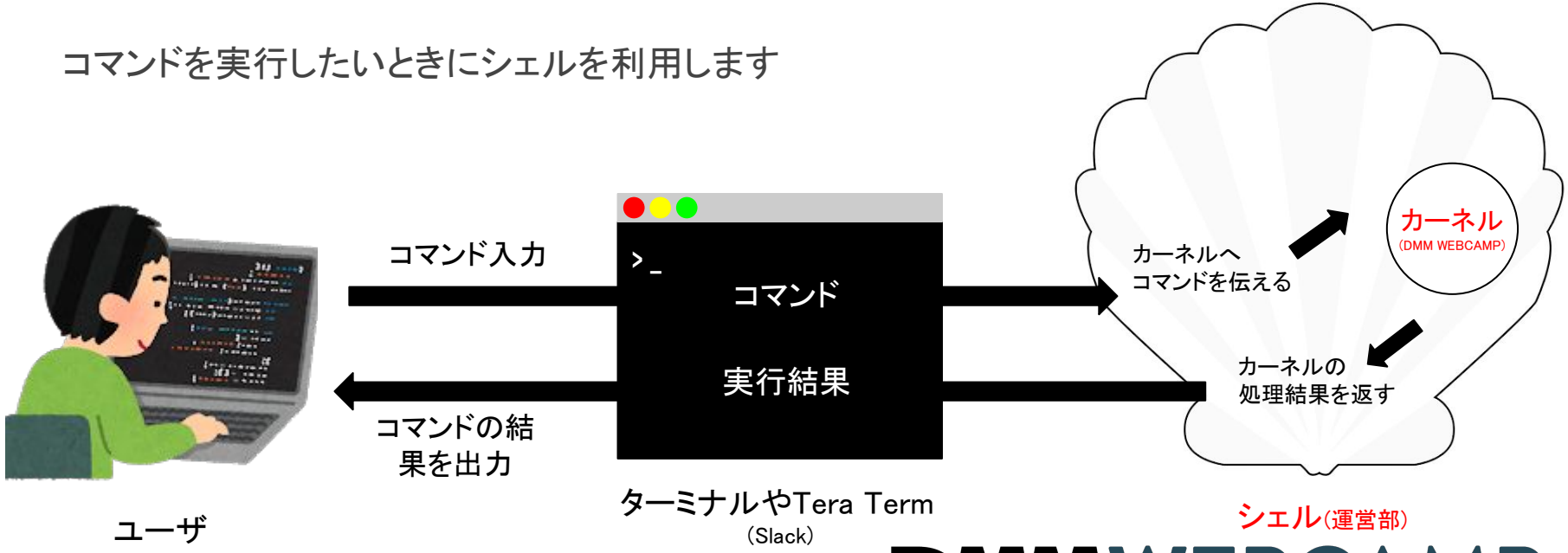


DMM WEBCAMP

カーネルとシェルの関係

カーネルやシェルは何でしょうか？

コマンドを実行したいときにシェルを利用します



DMM WEBCAMP

カーネルとシェルの関係

まとめ&補足

- シェルはユーザからの依頼をカーネルに渡す中継の役割がある

- ターミナルやTera Termはシェルではなく、シェルを表示させるツールです

- カーネルはシステムの核であり、
  ユーザからの依頼はシェルを中継してカーネルで実行される

- sh, csh, bash, tcsh, zshといった種類がある

プロセスとは

イメージ

プロセスは「実行中」である処理のことを指します。

例えば、受講生はDMM WEBCAMPのサービスに契約しています。

1プロセスとして管理されています。

契約が終わるとプロセスが終了し、プロセスから破棄されます。

DMM WEBCAMP

プロセスとは

プロセスは「実行中」である処理のことを指します。

例えば、$ rails s -b 0.0.0.0 を実行します。

1プロセスとして管理されています。

Ctrl-Cを実行するとプロセスから破棄されます。

プロセスとは

動作しているプロセスは、「ps」コマンドで調べることができます。

基本的に「ps」コマンドはオプションを利用して用いられます。

$ ps aux

→プロセスを表示する

$ ps aux | grep 3000

→プロセスの中で3000と表示されているプロセスのみ抽出

他にも様々なオプションがあります。

プロセスとは

動作しているプロセスは、「kill」コマンドで終了させることができます。
コンピュータに負荷をかけている動作や異常な動作をしているプロセスを
終了させます。

$ kill プロセスID
(プロセスIDは$ ps aux で表示される2列目の数字)

※通常フローでプロセスを終了させることができない時に利用します。

プロセスとは

実際に確認します

```
[vagrant@localhost sample_app]$ rails s -b 0.0.0.0
=> Booting Puma
=> Rails 5.2.4.3 application starting in development
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.12.6 (ruby 2.5.7-p206), codename: Llamas in Pajamas
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://0.0.0.0:3000
Use Ctrl-C to stop
```

```
[vagrant@localhost work]$ ps aux
:
:
root      9290  0.0  1.1 154672  5580 ?        Ss   13:56   0:00 sshd: vagrant [priv]
vagrant   9293  0.0  0.5 154988  2604 ?        S    13:56   0:00 sshd: vagrant@pts/0
vagrant   9294  0.0  0.6 116508  3244 pts/0    Ss   13:56   0:00 -bash
vagrant   9499  1.4 12.6 856912 63088 pts/0    Sl+  13:57   0:01 puma 3.12.6 (tcp://0.0.0.0:3000) [sample_app]
root      9546  0.1  1.1 154672  5580 ?        Ss   13:57   0:00 sshd: vagrant [priv]
vagrant   9549  0.0  0.5 154988  2596 ?        S    13:57   0:00 sshd: vagrant@pts/1
vagrant   9550  0.0  0.6 116508  3232 pts/1    Ss   13:57   0:00 -bash
vagrant   9633  0.0  0.3 155372  1864 pts/1    R+   13:59   0:00 ps aux
```

実際に$ rails s -b 0.0.0.0, $ ps aux コマンドを実行しました

プロセスに実行中のRailsアプリケーションが追加されていることが確認できます

DMM WEBCAMP

プロセスとは

実際に確認します

```
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://0.0.0.0:3000
Use Ctrl-C to stop


- Gracefully stopping, waiting for requests to finish
=== puma shutdown: 2020-05-26 14:07:15 +0000 ===
- Goodbye!
Exiting
Terminated
```

```
root      9290  0.0  1.1 154672  5580 ?        Ss   13:56   0:00 sshd: vagrant [priv]
vagrant   9293  0.0  0.5 154988  2604 ?        S    13:56   0:00 sshd: vagrant@pts/0
vagrant   9294  0.0  0.6 116508  3244 pts/0    Ss   13:56   0:00 -bash
vagrant   9499  0.3 12.6 856912 63088 pts/0    Sl+  13:57   0:01 puma 3.12.6 (tcp://0.0.0.0:3000) [sample_app]
root      9546  0.0  1.1 154672  5580 ?        Ss   13:57   0:00 sshd: vagrant [priv]
vagrant   9549  0.0  0.5 154988  2596 ?        S    13:57   0:00 sshd: vagrant@pts/1
vagrant   9550  0.0  0.6 116508  3232 pts/1    Ss   13:57   0:00 -bash
root      9635  0.0  0.0      0     0 ?        S    14:00   0:00 [kworker/0:1]
root      9651  0.0  0.0      0     0 ?        R    14:05   0:00 [kworker/0:0]
vagrant   9653  0.0  0.3 155372  1864 pts/1    R+   14:06   0:00 ps aux
[[vagrant@localhost work]$ kill 9499
```

プロセスを終了させることができました

※ 注意

通常フローで終了させることができるプロセスは
「kill」コマンドを使用しないこと

DMM WEBCAMP

プロセスとは

まとめ&補足

- プロセスは「ps」コマンドで確認する

- プロセスは「kill」コマンドで終了できる

- 基本的に通常フローでプロセスを終了させる
  通常フローで終了できない場合にのみ「kill」コマンドを使用する

コマンドを実行するということ

イメージ

受講生(ユーザ)は運営部(シェル)に対して依頼することで、
やりたいことを実現することができます。
依頼された運営部(シェル)はそれをどう解釈するのでしょうか。

依頼された運営部(シェル)はDMM WEBCAMP(カーネル)に伝え、結果を待ちます。
この時どこを探すかは前もって決められています。

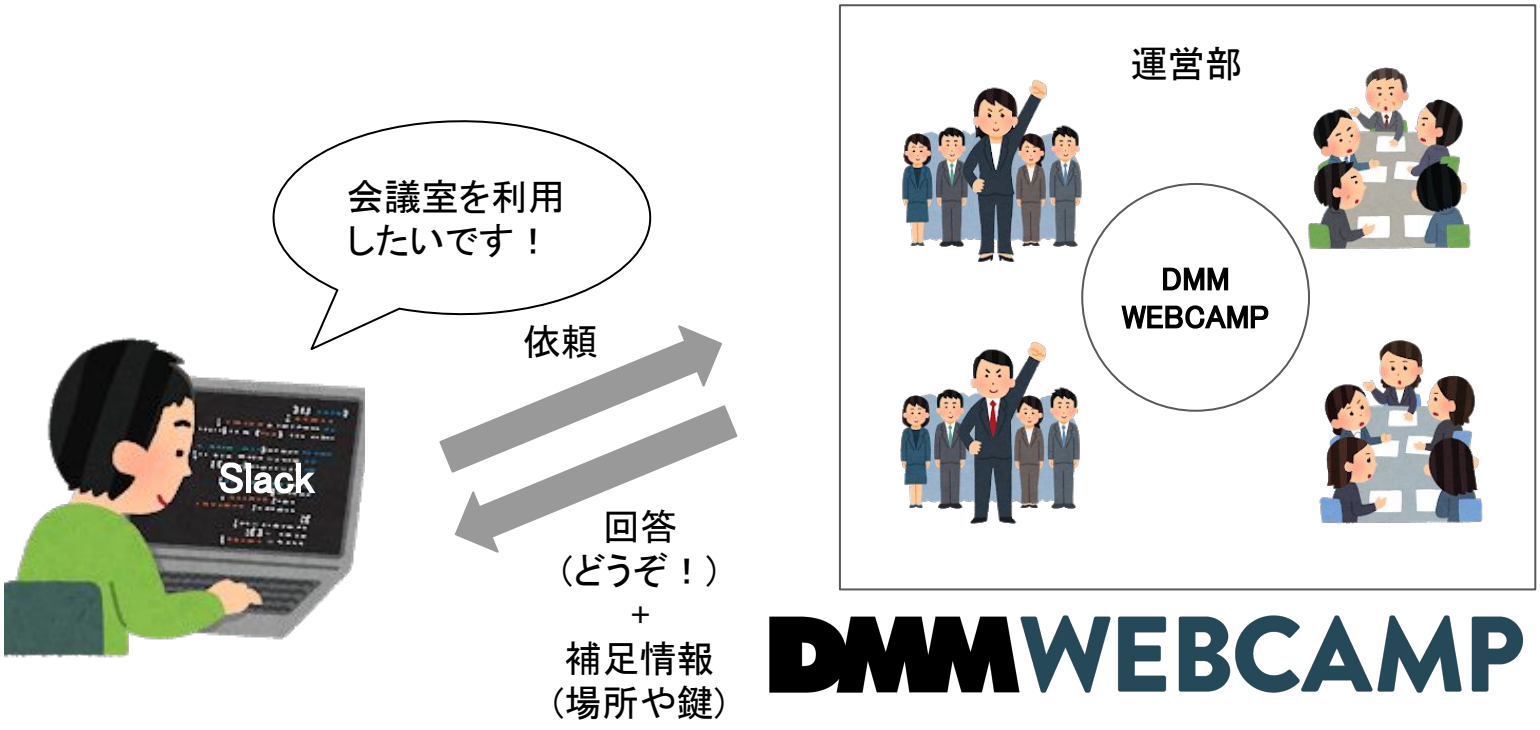
最後に、DMM WEBCAMP(カーネル)から運営部(シェル)に結果が伝えられ、
受講生(ユーザ)にその結果を伝えます。

DMM WEBCAMP

コマンドを実行するということ

受講生(ユーザ)は会議室を利用したいときに運営部(シェル)に依頼をします

イメージ

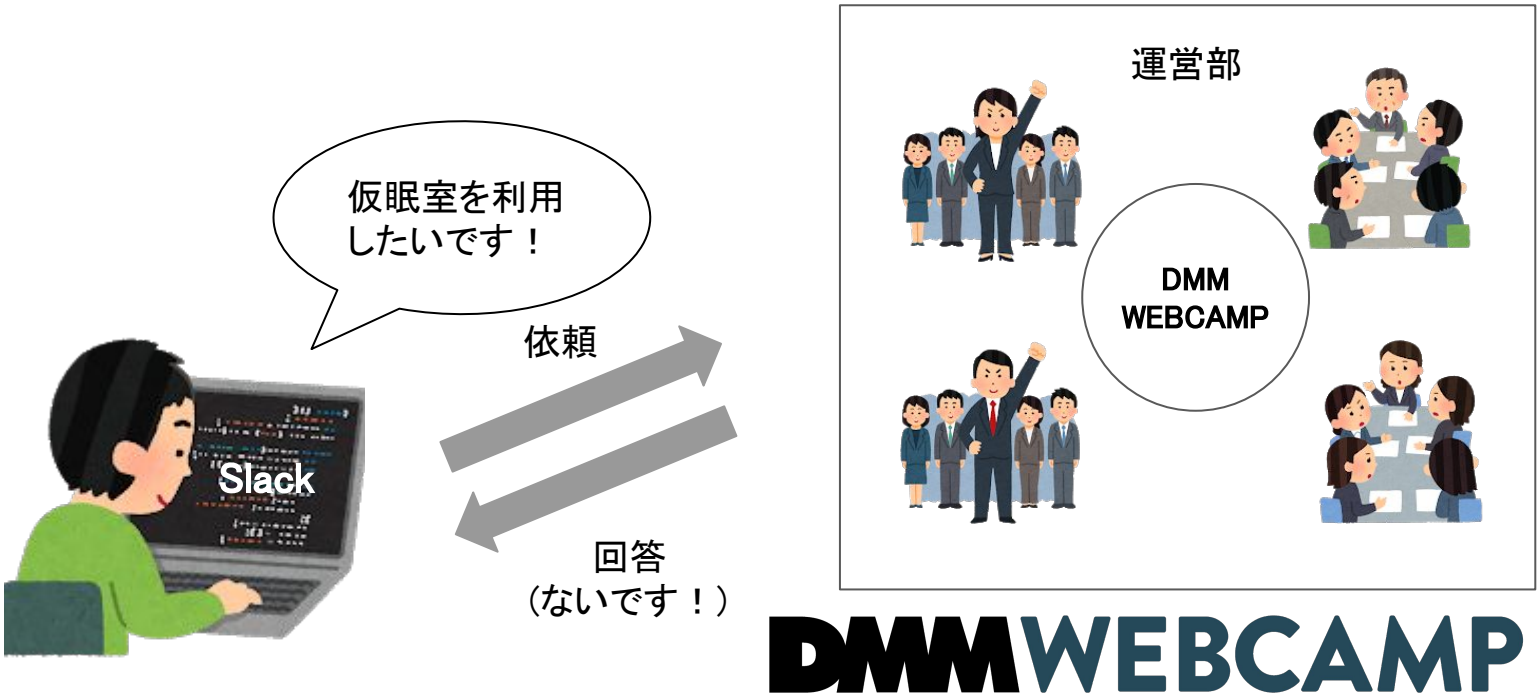


DMM WEBCAMP

コマンドを実行するということ

受講生(ユーザ)は仮眠室を利用したいときに運営部(シェル)に依頼をします

イメージ

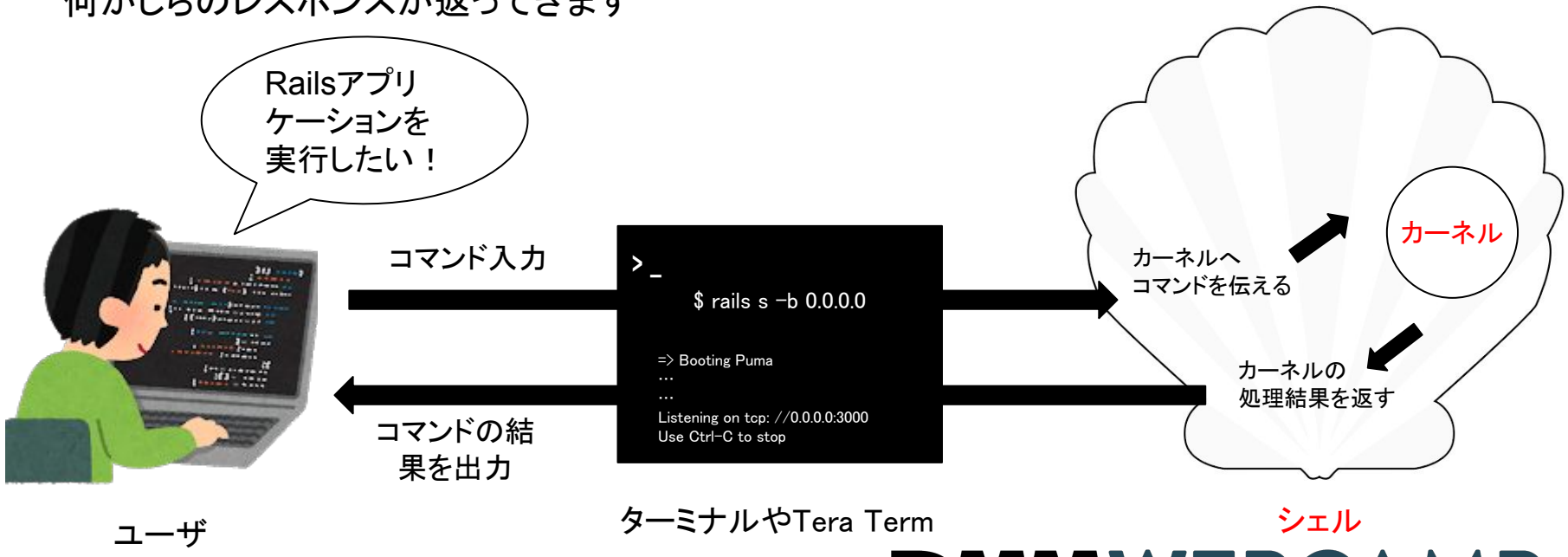


DMM WEBCAMP

コマンドを実行するということ

Railsアプリケーションを実行したい時は$ rails s -b 0.0.0.0を実行します

何かしらのレスポンスが返ってきます

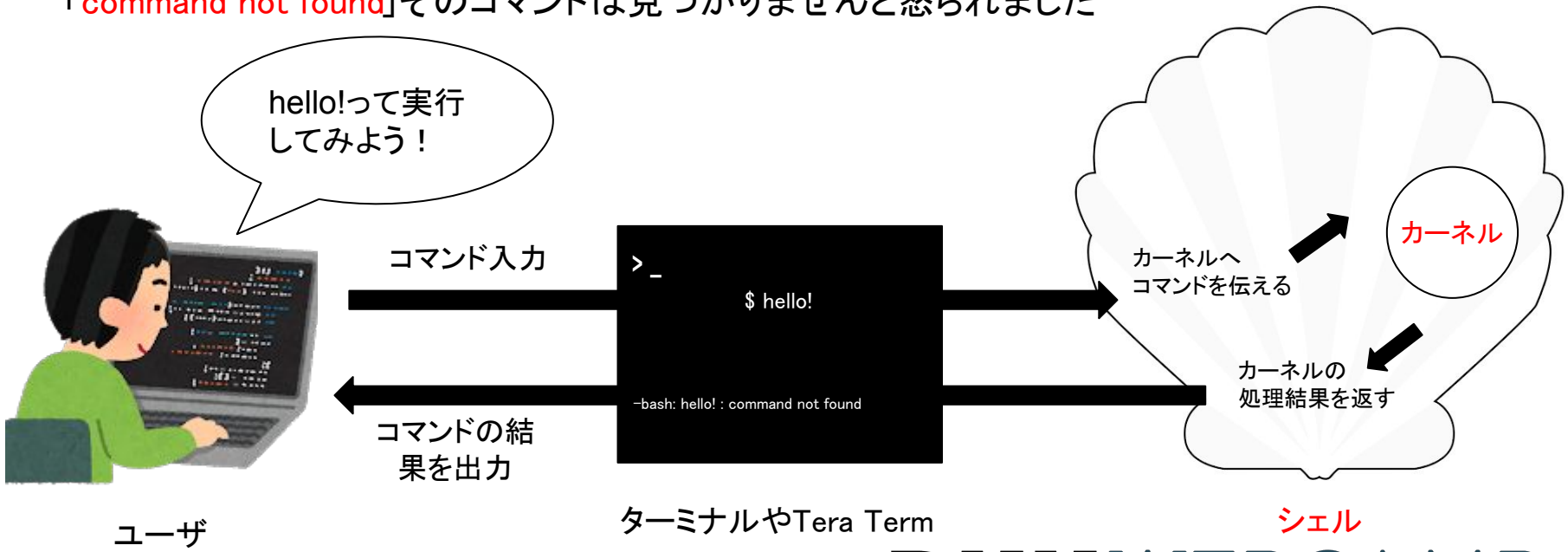


DMM WEBCAMP

コマンドを実行するということ

コマンドでスペルミスや存在しないコマンドを実行します

「command not found」そのコマンドは見つかりませんと怒られました



DMM WEBCAMP

代表的なコマンド

ls：ファイルやディレクトリを表示する

cd：ディレクトリ間を移動する

mkdir：ディレクトリを作成する

touch：中身がないファイルを作成する

mv：ファイルやディレクトリを移動する/名前を変更する

cp：ファイルやディレクトリをコピーする

cat：ファイルの中身を表示する

rm：ファイルやディレクトリを削除する

pwd：カレントディレクトリのパスを表示する

DMM WEBCAMP

代表的なコマンド

lsコマンドを例にコマンドを分解します。

コマンドは仕事の役割が決められています。

ls：カレントディレクトリ(①)の内容をリストアップして表示する

命令を受けたlsは役割を実行し、また待機に入ります

ちなみにカレントディレクトリとは、

DMM WEBCAMPの中で、受講生が現在いる部屋のことです。

DMM WEBCAMP

代表的なコマンド

それでは、決められている役割を変更したり、追加する場合はどうするかというと、

オプションや引数を指定することで実現させます。

ls -la /tmp/

先ほどのlsコマンドに文字列が追加されました。lsコマンドにオプションが加わりました

-laをオプションといい、/tmpのことを引数と言います。

引数を付与したことでコマンドの解釈が異なります。

ls：カレントディレクトリの内容をリストアップして表示する

↓

ls -la /tmp/：/tmp/の内容を[詳細に][全て]リストアップして表示する

代表的なコマンド

-lオプションは、[詳細に]という言葉を追加し、

-aオプションは、[全て]という言葉を追加します。

指定の仕方は、-laでも-alでも機能します。

詳細に全て、でも

全て詳細に、でも意味が通じます。

/tmp/という引数はオプションとは異なり、

引数を指定する順番は決められています。

代表的なコマンド

まとめ＆補足

- 利用できるコマンドは環境ごとに異なる
- コマンドが見つからない場合「command not found」と怒られる
- コマンドはたくさん実行して慣れることが大事

DMM WEBCAMP

Linuxユーザとは

ユーザは以下に区分されます

- スーパーユーザ(LS社員)
- 一般ユーザ(受講生)
- 特殊ユーザ(清掃員…)

特殊ユーザはあることをするだけのユーザで出入りすることはできません。

住み込みです

DMM WEBCAMP

Linuxグループとは

ユーザは基本的にはDMM WEBCAMPに関わる全ての人のことを指します。

DMM WEBCAMPでは、このユーザをわかりやすくするために

グループにまとめて管理しています。

「入学月」や「コース」でグルーピングすることで、

管理しやすくしています。

DMM WEBCAMP

Linuxグループとは

グループに関しては以下のルールがあります。

1. グループには、メイン(プライマリ)とサブ(セカンダリ)の2種類ある
2. ユーザは必ずどこかのグループに属さなければならない

メイングループは1つしか指定できませんが、

サブグループは複数指定できます。

また、ユーザを作成した時点でユーザ名のグループが作成されることが多いです。

例:「メンター太郎」というユーザを作成すると「メンター太郎」というグループも作成されます。

パーミッションについて

ファイルやディレクトリにはパーミッションという権限管理が行われています。

drwxr-xr-x 2 root root 4096 12月 26 15:35 hogedir

-rw-r--r-- 1 root root 0 12月 26 15:36 hogefile

パーミッションに必要な情報としては、

root root

まずはこの部分です。

最初のrootはそのファイル、ディレクトリの所有ユーザを表します。

次のrootはそのファイル、ディレクトリの所有グループを表します。

パーミッションについて

次に

rwxr−xr−x

この部分です。

3文字ずつ区切って見ていきます。

先頭の3文字、rwxは所有者に対する権限を表します。

次の3文字、r−xは所有グループに対する権限を表します。

最後の3文字、r−xは

第三者に対する権限を表します。

パーミッションについて

r w x は、属性を表しています。

r属性：読み込み属性

w属性：書き込み属性

x属性：実行属性

所有者、所有グループ、第三者それぞれに、

読み込み、書き込み、実行の属性を組み合わせることで、

権限を設定していきます。

Linuxユーザ・Linuxグループ・パーミッションについて

まとめ＆補足

- ファイルやディレクトリはユーザ・グループごとに権限管理されている
- 権限がない場合はsudoを利用して閲覧・実行することができる
- 「chmod」コマンドで読み取り・書き込み・実行権を変更できる
- 「chown」コマンドで所有ユーザ・所有グループを変更できる
- 権限は最低限に狭めておくと安全

クライアントとサーバについて

クライアントとサーバの関係は以下のようにイメージすることが多いです。

クライアント＝お客さん

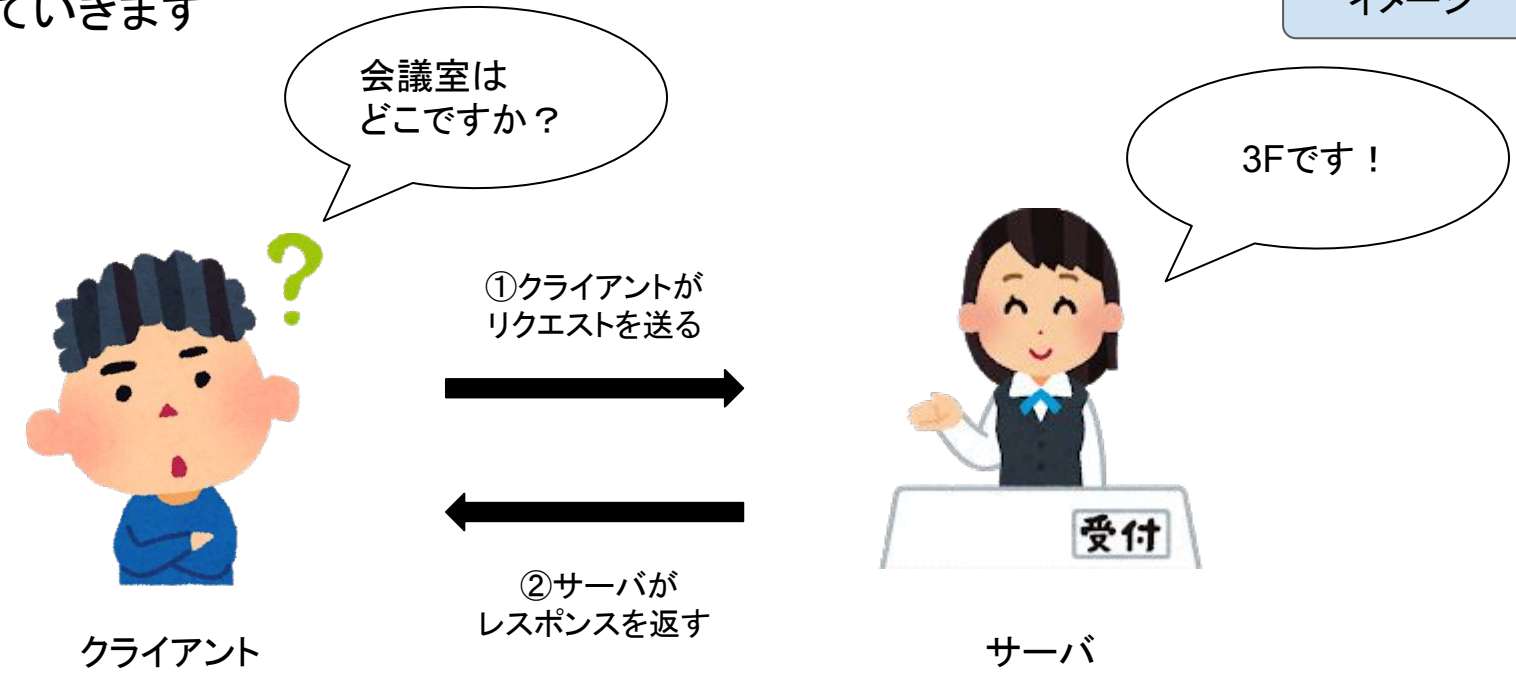
サーバ＝窓口担当

大事なことは、「やりとりを始めるのはいつもクライアント」ということです。

クライアントとサーバについて

受付で例えていきます

イメージ



DMM WEBCAMP

クライアントとサーバについて

まずはお客さんから店に出向くことからやりとりは始まります。

お客さんが依頼をする＝URL or IPアドレスに通信をする(ブラウザで検索)

このことをリクエストと言います。

窓口はこの要件に応え、何らかの応答をします。

このことをレスポンスと言います。

クライアントとサーバについて

まとめ&補足

- クライアント = 依頼をする側
  サーバ = 依頼を受けて応答をする側

- やりとりはいつもクライアントから始まる