

# モバイル プログラミング2

本日の予定

# 午前

- Linuxコンソール
  - パイプとリダイレクト
- Linux基礎
  - デーモンとサーバ

# 午後

- ネットワーク基礎
  - 「ドメイン」は誰の物？
- HTTP演習
  - JavaScript

前回休んだ人

(°▽°)ｼ

PC借りた人

(°▽°)ｼ

# まずは追いつこう

1. GitHubの資料見てね
2. 環境構築
3. 環境構築で困ったらすぐに聞いてください

前回のFB



# 前回のアンケートに 答えるコーナー

1. チョークの色を変えて
2. 文字が小さい
3. 大切な所は繰り返し喋って
4. もこみち好きなんですか？
5. 波平さん

前回の復習

# ユーザー権限

- Linuxは1台を複数人が使う前提
- 管理者と一般ユーザー



# ユーザー権限

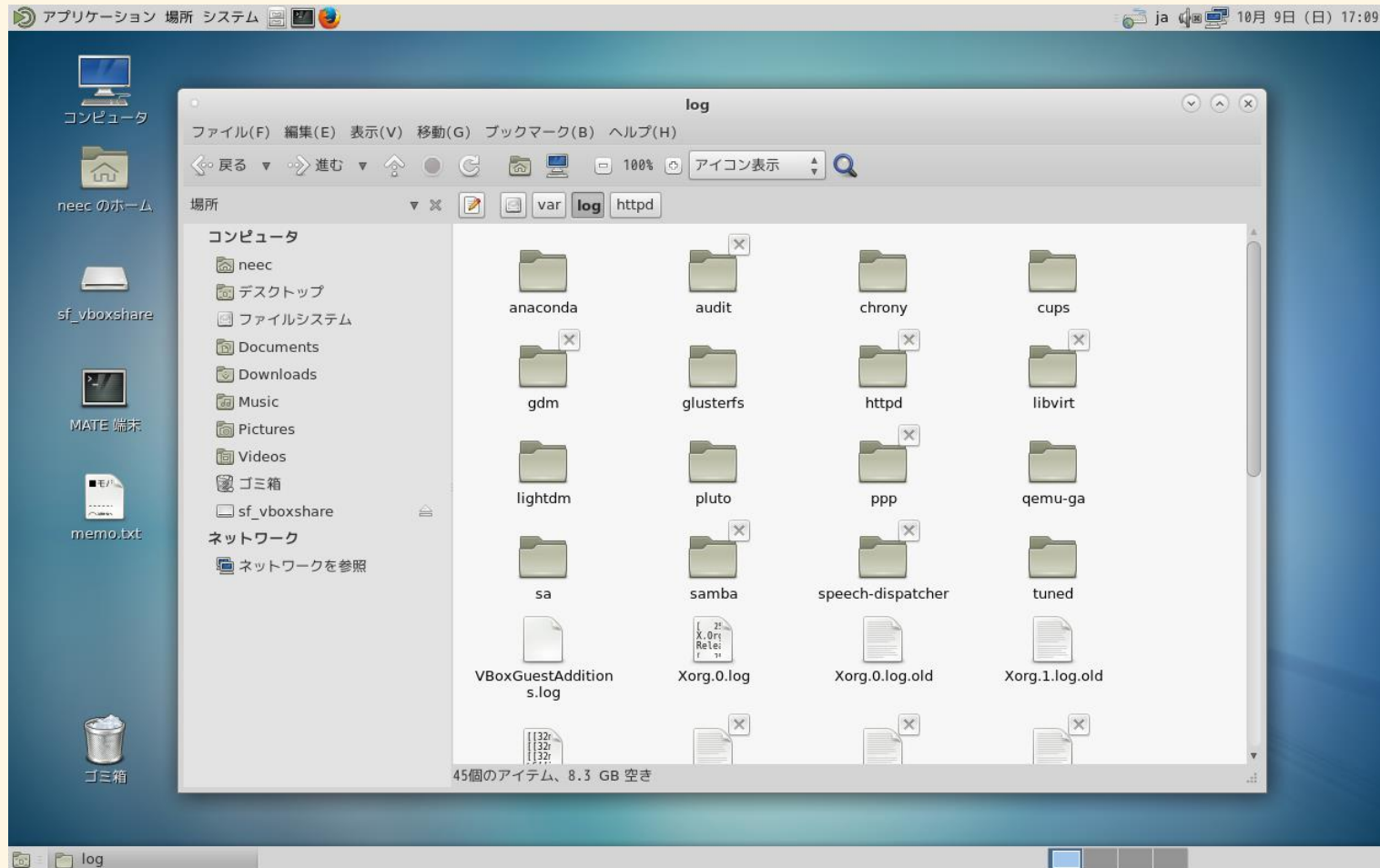


スーパーユーザー  
(root)

一般ユーザー

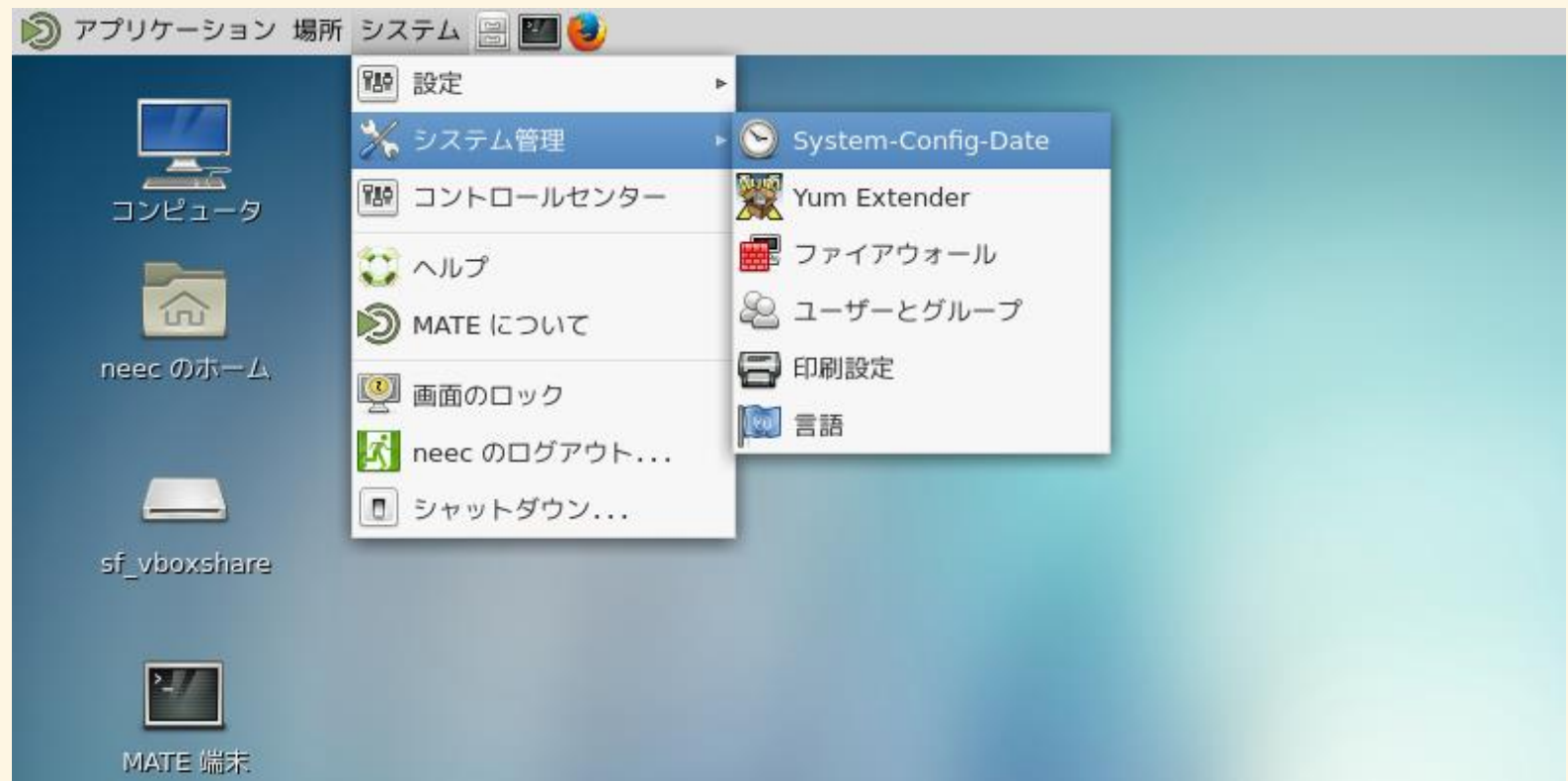
# やってみよう

## **/var/log/** の下のディレクトリ(フォルダ)やファイル



# やってみよう

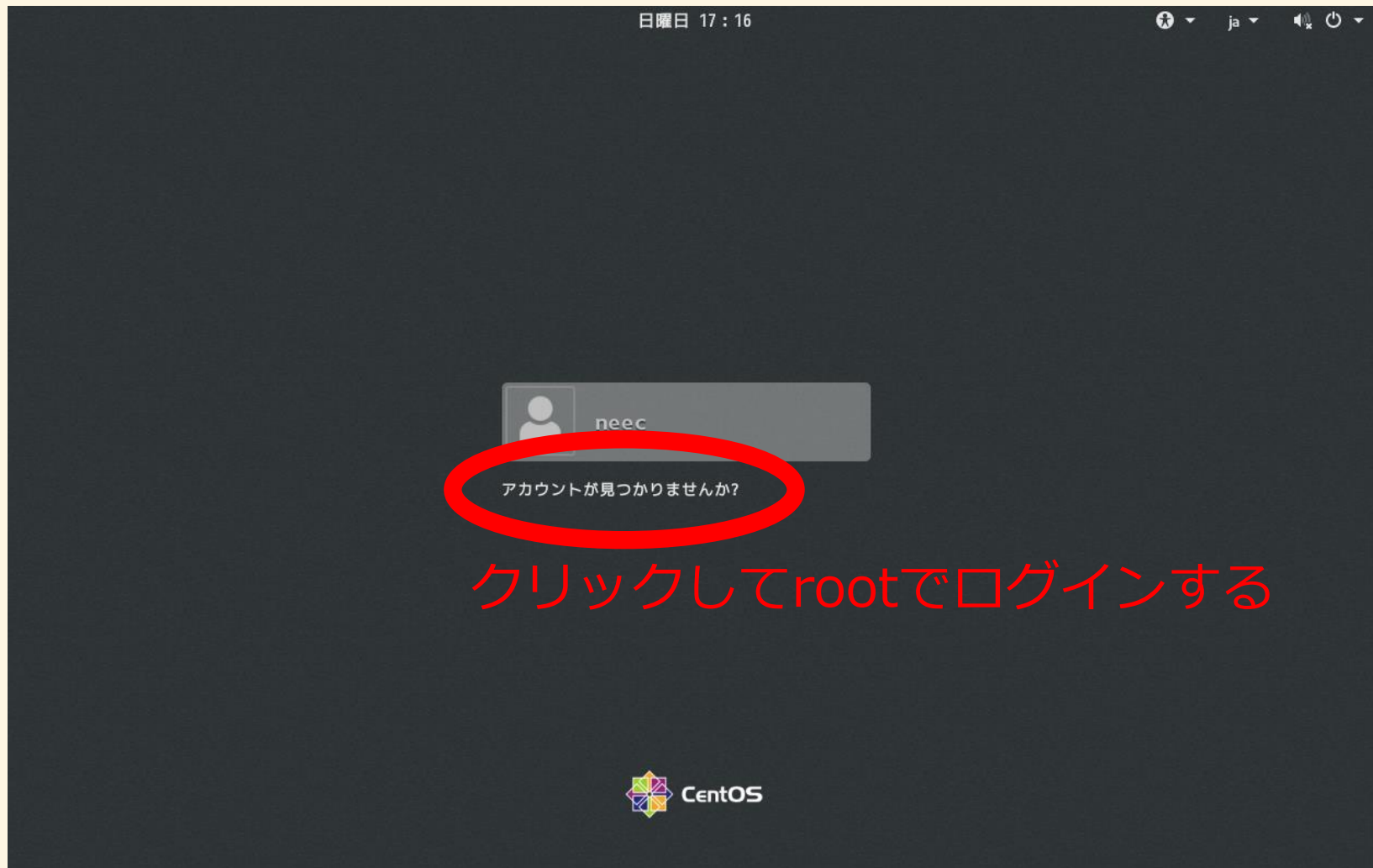
## OSの設定変更



# やってみよう



# やってみよう





# やってみよう

ユーザー名 : root

パスワード : root2016



ユーザー名:

キャンセル 次へ

# rootは特別な時だけ



スーパーユーザー  
は何でも出来てし  
まうので、**非常に  
危険**。

普段は必ず一般  
ユーザーで。

# ユーザーの権限

- 「スーパーユーザー」と「一般ユーザー」
- スーパーユーザー = root
- 普段の作業は一般ユーザー

# Linuxコンソール

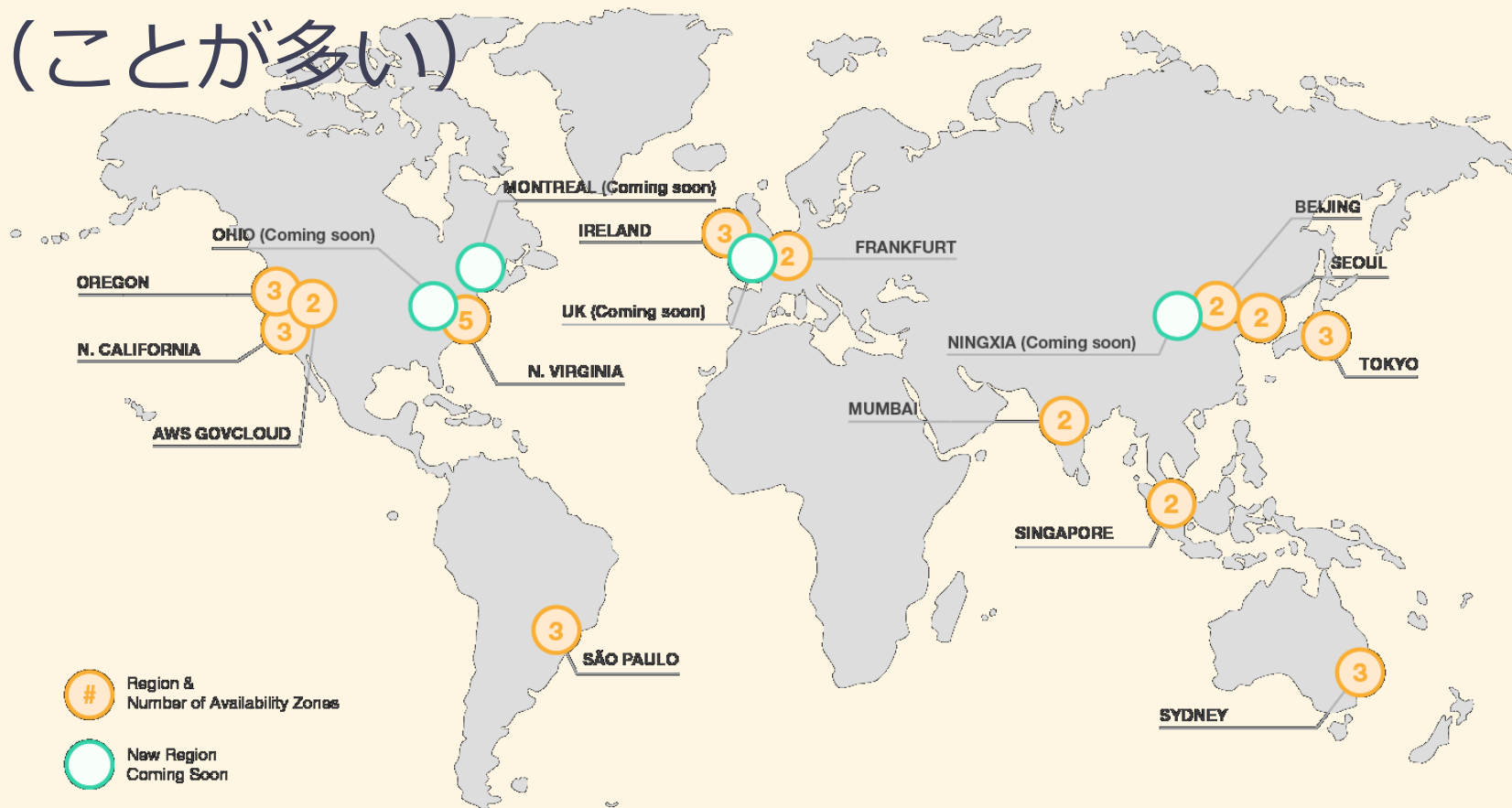


なぜCUI ?

# なぜCUIを使うの？

前提として、

**サーバは遠く離れたところ**にある  
(ことが多い)



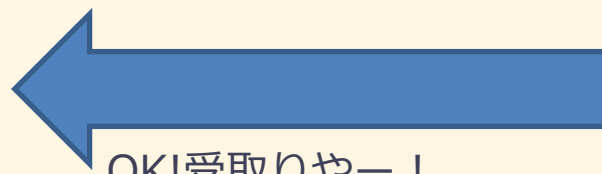
<https://aws.amazon.com/jp/about-aws/global-infrastructure/>

# なぜCUIを使うの？

CUIなら文字情報だけ。  
1つのコマンドで完結する。



ファイルの一覧ください



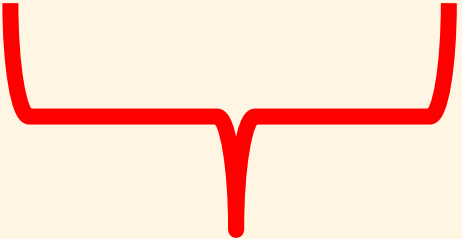
OK!受取りやー！

# コマンド練習



# 書式

```
$ host twitter.com
```



コマンド



オプション

ファイル

# ls

```
$ ls
```

- ファイルの一覧を表示

# ls

```
$ ls -l
```

- -l をつけるとファイルの詳細情報も表示

# ls

```
$ ls -l -a
```

```
$ ls -la
```

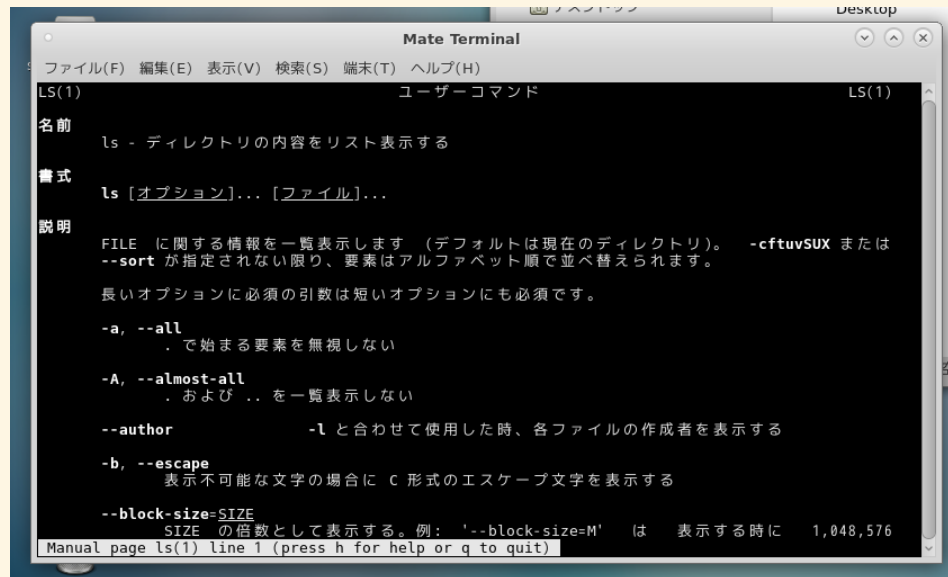
```
$ ls -la Desktop
```

- 同時に複数指定することも可能

# man ls

# \$ man ls

- 使い方を忘れたら man コマンド



```

Mate Terminal
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
LS(1) ユーザーコマンド LS(1)

名前
ls - ディレクトリの内容をリスト表示する

書式
ls [オプション]... [ファイル]...

説明
FILE に関する情報を一覧表示します (デフォルトは現在のディレクトリ)。 -cftuvSUX または
--sort が指定されない限り、要素はアルファベット順で並べ替えられます。

長いオプションに必須の引数は短いオプションにも必須です。

-a, --all
    . で始まる要素を無視しない

-A, --almost-all
    . および .. を一覧表示しない

--author
    -l と合わせて使用した時、各ファイルの作成者を表示する

-b, --escape
    表示不可能な文字の場合に C 形式のエスケープ文字を表示する

--block-size=SIZE
    SIZE の倍数として表示する。例: '--block-size=M' は 表示する時に 1,048,576

Manual page ls(1) line 1 (press h for help or q to quit)
```

- スペースでページ送り
- ↑, ↓で行送り
- “q”で終了

# cat

```
$ cat Desktop/memo.txt
```

- ファイルの内容を表示

ディレクトリ



cd

```
$ cd Desktop
```

- カレントディレクトリを移動

# pwd

```
$ pwd
```

- カレントディレクトリのフルパスを表示

※カレントディレクトリは、正式にはカレントワーキングディレクトリ

※print working directory

# ホームディレクトリ

```
$ cd ~
```

- ‘~’はホームディレクトリを表す特別な文字

# ルートディレクトリ

```
$ cd /
```

- 最も上位に位置する階層をルートディレクトリ、または単にルートと呼びます

# ディレクトリ

- 「カレント」
  - 今、自分がいる場所
- 「ホーム」
  - 最初場所 (/home/neec)
- 「ルート」
  - 一番上

var → www → html

```
$ cd /var/www/html
```

- Linuxではディレクトリの階層構造を **'/'** で区切って記述

# ひとつ上の階層へ行く

```
$ cd ..
```

- 一つ上の階層を '..' で表現する
- 自分自身は '.'

# 絶対パスと相対パス

絶対パス

```
$ cd /var/www/html
```

相対パス

```
$ cd ../../log
```



その他  
コマンド

# mkdir

```
$ mkdir ディレクトリ名
```

- ディレクトリを作成します

cp

```
$ cp コピー元 コピー先
```

- ファイルをコピーします。
- -r をつけるとディレクトリも可能

rm

\$ rm ファイル

- ファイルを削除します。
- -r をつけるとディレクトリも可能

※remove direvtory

mv

\$ mv 移動元 移動先

- ファイル、ディレクトリを移動します。
- 応用するとリネーム

su と sudo

# su

```
$ su
```

- 一時的にrootになります
- 特別な事情がなければ“su -” と “-”オプションをつける

# sudo

```
$ sudo 実行コマンド
```

- rootとしてコマンドを実行



Where is  
Linux  
Command?

/usr/bin

```
$ cd /usr/bin
```


```
$ ls
```

```
$ cd /usr/sbin
```

```
$ ls
```

# UNIXという哲学

# 書籍「UNIXという考え方」



すべて

誕生


Amazonポイント: 85

マイストア ギフト券 タイムセール

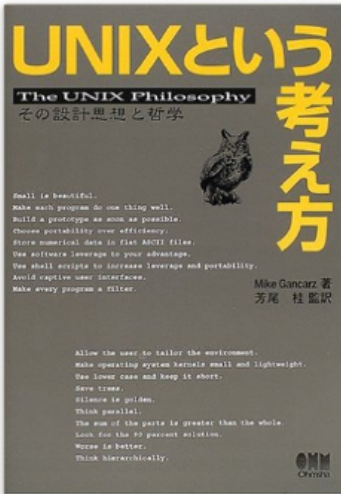
JP 勝部麻季人さん アカウントサービス

本 詳細検索 ジャンル一覧 新刊・予約 Amazonランキング コミック・ラノベ 雑誌 文庫・新書 Amazon Student

本 > コンピュータ・IT > コンピュータサイエンス

 お客様は、2002/5/19にこの商品を注文しました。  
[この注文を表示](#)

なか見!検索↓



UNIXという考え方

The UNIX Philosophy

その設計思想と哲学

Mike Gancarz 著  
芳尾 桂 監訳

OMM  
O'Reilly

UNIXという考え方—その設計思想と哲学 単行本

— 2001/2

Mike Gancarz (著), 芳尾 桂 (翻訳)

★★★★★ 23件のカスタマーレビュー

▶ その他 ( ) の形式およびエディションを表示する

単行本

¥ 1,728

¥ 1,299 より 14 中古品の出品

¥ 1,728 より 3 新品

10/10 月曜日 にお届けするには、今から**13 時間 41 分**以内に「**お急ぎ便**」または「**当日お急ぎ便**」を選択して注文を確定してください（有料オプション。Amazonプライム会員は無料）

# 定理1: スモール・イズ・ビューティフル

## 2.1 **定理1**: スモール・イズ・ビューティフル

プログラムを書くときは小さなものから始めて、それを小さなままに保っておく。簡単なフィルタプログラムでも、グラフィックスパッケージでも、巨大なデータベースを構築するときでも、同じく小さな実用的なプログラムにする。一つの巨大なプログラムにしようとする誘惑に負けないで、シンプルさを追及する。

伝統的なプログラマは、巨大なアメリカンプログラムを書きたいという欲求を内に秘めていることが多い。このようなプログラマがプロジェクトに携わると、数週間、数カ月、あるいは数年かけてでも、世界中のすべての問題を一つのプログラムで解決しようとする。コスト的にビジネスに見合わないだけでなく、何よりも彼らは現実を無視している。現実には、その場かぎりの小さな解決策でも解決できない問題はそれほど多くはない。ほとんどの場合、問題を完全に理解していないから巨大な解決策を実装しようとしてしまうのだ。

# Linuxコンソール



パイプ

# よくあるタスク

- “/category/games” へ何件のアクセスがあったか集計したい
- アクセスログは10万行ほどある



# grep

```
$ grep 'パターン' ファイル
```

- ファイルから "パターン" に該当する行を表示します。

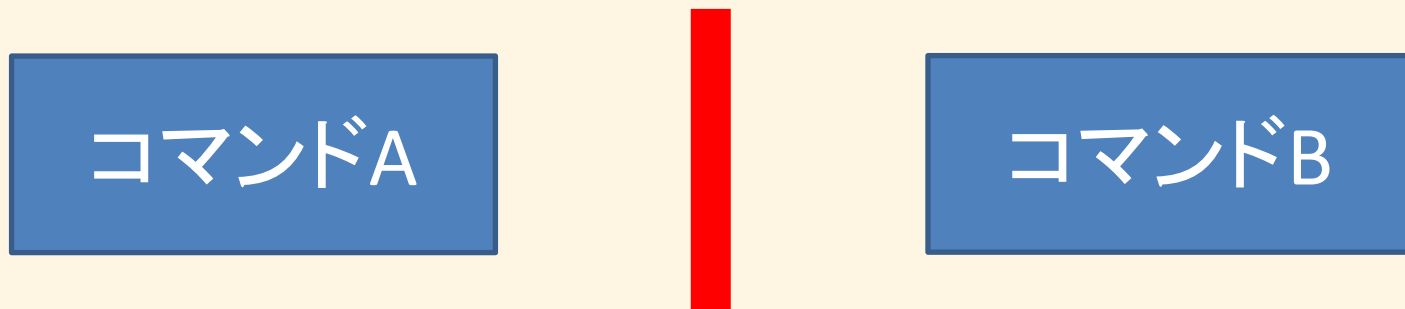
WC

```
$ wc -l ファイル
```

- ファイルの行数をカウントする

# パイプを利用する

- コマンドの実行結果を、次のコマンドに引き継ぐことができます。



# パイプを利用

```
$ grep 'GET /category/games' access.log | wc -l
```

- grepでaccess.log から"/category/games"を検索する
- 検索結果の行数をwcでカウント

# やってみよう

- “/category/games” へ何件のアクセスがあったか集計したい
- アクセスログは10万行あるが、先頭の1万件だけ対象にしたい

# head

```
$ head ファイル
```

- ファイルから先頭10行を表示
- オプションで「-100」を指定すると先頭の100行を表示する

# パイプを利用

- パイプは何個でもコマンドを連結することができます。



# パイプ

- |
  - コマンドの実行結果(標準出力)を、次のコマンドに渡す
  - パイプは一度に何個でも指定できる
- Linuxの1コマンドが1機能なのは、パイプの利用を前提としている



リダイレクト

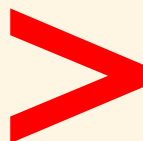
# よくあるタスク

- アクセスログの中から、iPhoneのアクセスだけ抽出して欲しい
- iPhoneかどうかの判断はUserAgentに “iPhone” が含まれていればOK

# リダイレクトを利用

- コマンドの実行結果を、ファイル等に保存することができます。

コマンドA



ファイルX

# リダイレクトを利用

```
$ grep 'iPhone' access.log > iphone.log
```

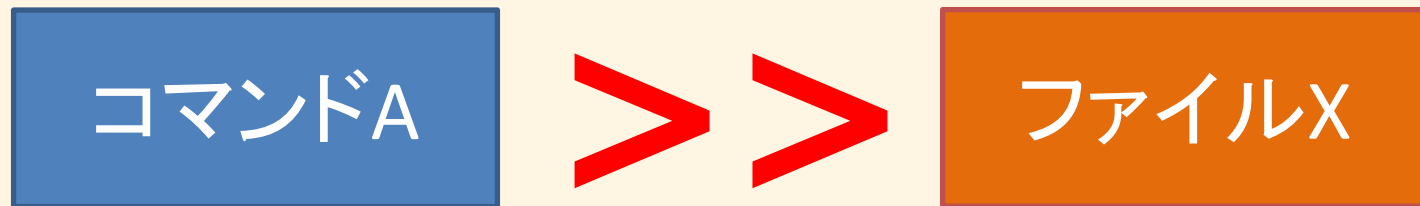
- grepでaccess.log から文字列*"iPhone"*を検索する
- iphone.logへ結果を保存する

# よくあるタスク

- 先ほどのログにMacOSからのアクセスも追加して欲しい
- MacOSかどうかの判断はUserAgentに “Macintosh” が含まれていればOK

# リダイレクトを利用する

- “>>”を用いることで、コマンドの実行結果を、ファイル等に追記することができます。



# リダイレクトを利用

```
$ grep 'Macintosh' access.log >> iphone.log
```

- grepでaccess.log から文字列“Macintosh”を検索する
- iphone.logへ結果を追記する

# リダイレクト

- > (名前をつけて保存)
  - 標準出力に出力された情報を、指定されたファイル等に**上書き**する
- >> (追記)
  - 標準出力に出力された情報を、指定されたファイルの**末尾に追記**する



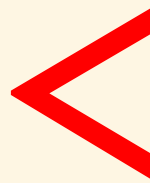
# よくあるタスク

- 先ほどのiphone.logの中から、  
"/category/games"へのアクセス  
件数を集計して欲しい

# リダイレクトを利用

- “<”を用いることで、ファイルの内容をコマンドに送ることができます。

コマンドA



ファイルX

# リダイレクトを利用

```
$ grep 'GET /category/games' < iphone.log | wc -l
```


- grepにiphone.logの内容を送り、文字列"/category/games"を検索する
- 検索結果の行数をカウントする

# リダイレクト

- < (入力)
  - コマンドにファイル等の情報を標準入力として渡す

# UNIXという哲学

# 書籍「UNIXという考え方」



すべて

誕生


Amazonポイント: 85

マイストア ギフト券 タイムセール

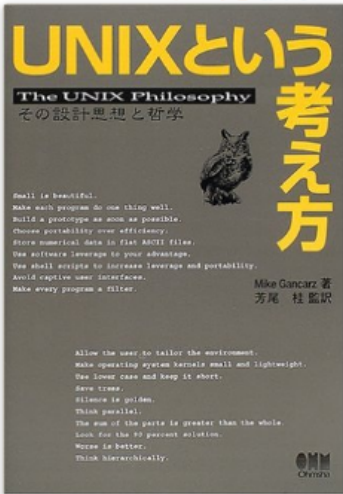
JP 勝部麻季人さん アカウントサービス

本 詳細検索 ジャンル一覧 新刊・予約 Amazonランキング コミック・ラノベ 雑誌 文庫・新書 Amazon Student

本 > コンピュータ・IT > コンピュータサイエンス

 お客様は、2002/5/19にこの商品を注文しました。  
[この注文を表示](#)

なか見!検索↓



## UNIXという考え方—その設計思想と哲学 単行本

— 2001/2

Mike Gancarz (著), 芳尾 桂 (翻訳)

★★★★☆ 23件のカスタマーレビュー

▶ その他 ( ) の形式およびエディションを表示する

単行本

¥ 1,728

¥ 1,299 より 14 中古品の出品

¥ 1,728 より 3 新品

10/10 月曜日 にお届けするには、今から**13 時間 41 分**以内に「**お急ぎ便**」または「**当日お急ぎ便**」を選択して注文を確定してください（有料オプション。Amazonプライム会員は無料）

# 定理9：すべてのプログラムをフィルタにする

## 6.2 **定理9**：すべてのプログラムをフィルタにする

コンピュータの出現以来、書かれてきたすべてのプログラムはフィルタだ。すべてのプログラムは、簡単なものでも複雑なものでも、何らかの形式のデータを入力として受け入れ、何らかの形式のデータを出力として生成する。プログラムがどのようにデータをフィルタするかは、そこに含まれているアルゴリズムによる。

ほとんどの人は、テキストフォーマッタや変換プログラムをフィルタとみなすことにはそう抵抗を感じない。しかし、フィルタとはみなされない他のプログラムも同じだということを受け入れるのは難しいようだ。例えば、リアルタイムデータ収集システムを考えてみよう。典型的なシステムは、アナログデジタルコンバータを定期的にサンプリングして、データを収集する。これが入力ストリームとなる。次に、このデータから適切な一部を抽出し、出力ストリームに送り出す。それがさらにユーザーインターフェースに送られ、他のアプリケーションに送られ、あるいはファイルに保存されることになる。

# 定理8：過度の対話的インターフェイスを避ける

## 6.1 定理8：過度の対話的インタフェースを避ける

拘束的ユーザーインタフェースを避ける理由について議論を始める前に、まずそれを定義しておこう。拘束的ユーザーインタフェースとは、システムに存在するあるアプリケーションが、最上位のコマンドインタプリタの範囲外にあって、ユーザーと対話するスタイルをいう。いったん、そのアプリケーションをコマンドインタプリタから起動すると、そのアプリケーションが終了するまでコマンドインタプリタとの対話ができなくなる。ユーザーはアプリケーションのユーザーインタフェースの内部に拘束され、拘束を解くための行動を起こさない限り、その拘束から逃れられない。

分かりやすくするために、例を見てみよう。二つのプログラムがある。一つ（mail）は電子メールボックスの内容一覧を書き出し、もう一つ（search）はファイル中からテキスト文字列を探し出す。mailプログラムとsearchプログラムは、共に拘束的ユーザーインタフェースを採用しているとすると、システムとの対話は次のようになる。

\$mail	コマンドラインからmailプログラムを起動する
MAIL> dir	メールボックスの内容を表示する
	:
	:
MAIL> exit	mailプログラムを終了し、コマンドインタプリタレベルに戻る
\$search	searchプログラムを起動する
SEARCH> find jack *.txt	検索する
SEARCH> exit	searchプログラムを終了する
	:
	:
\$	もう一度コマンドインタプリタレベルに戻る



# Linux基礎



デーモン

Deamon  
is  
Not Demon



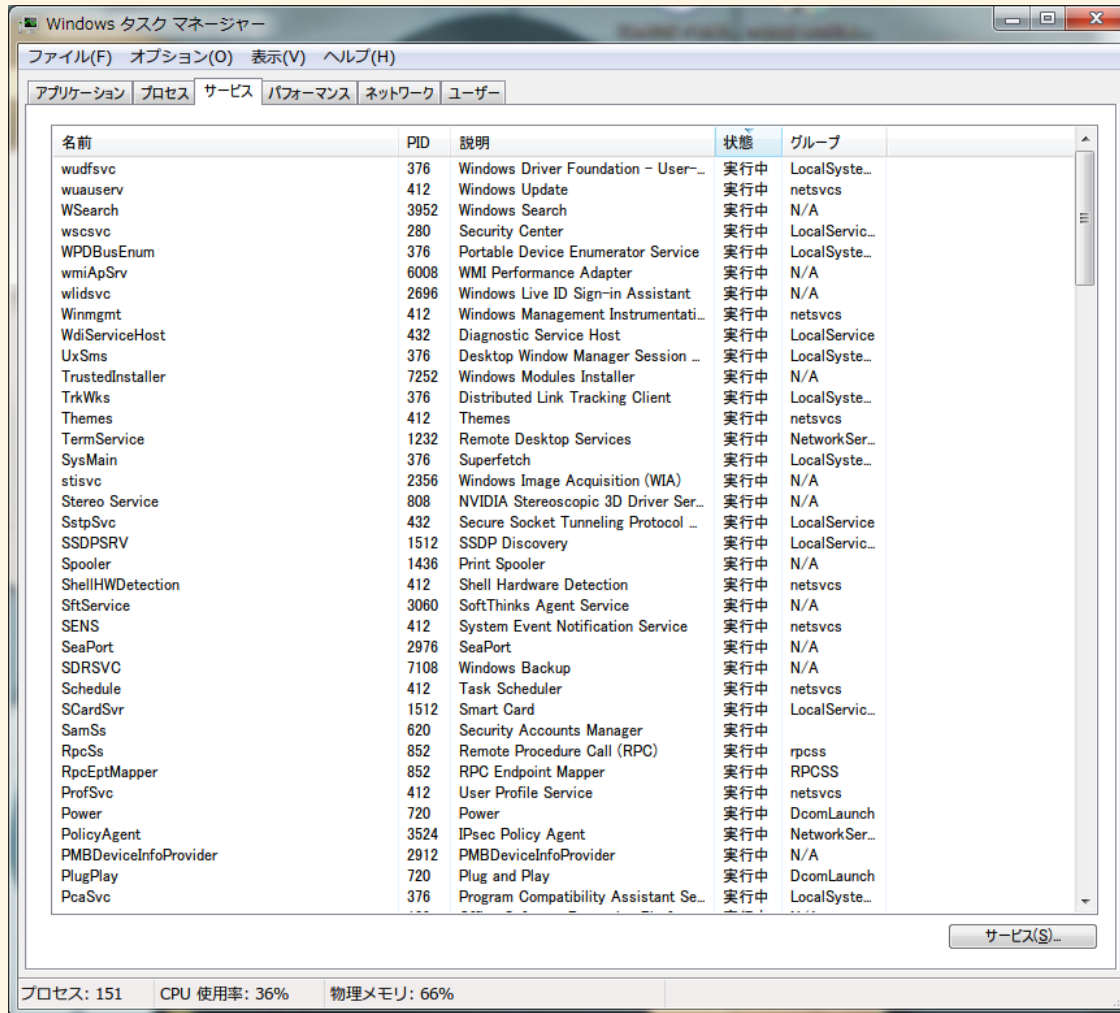
# Deamon

=

# 守護神

(ギリシャ神話、ダイモーン)

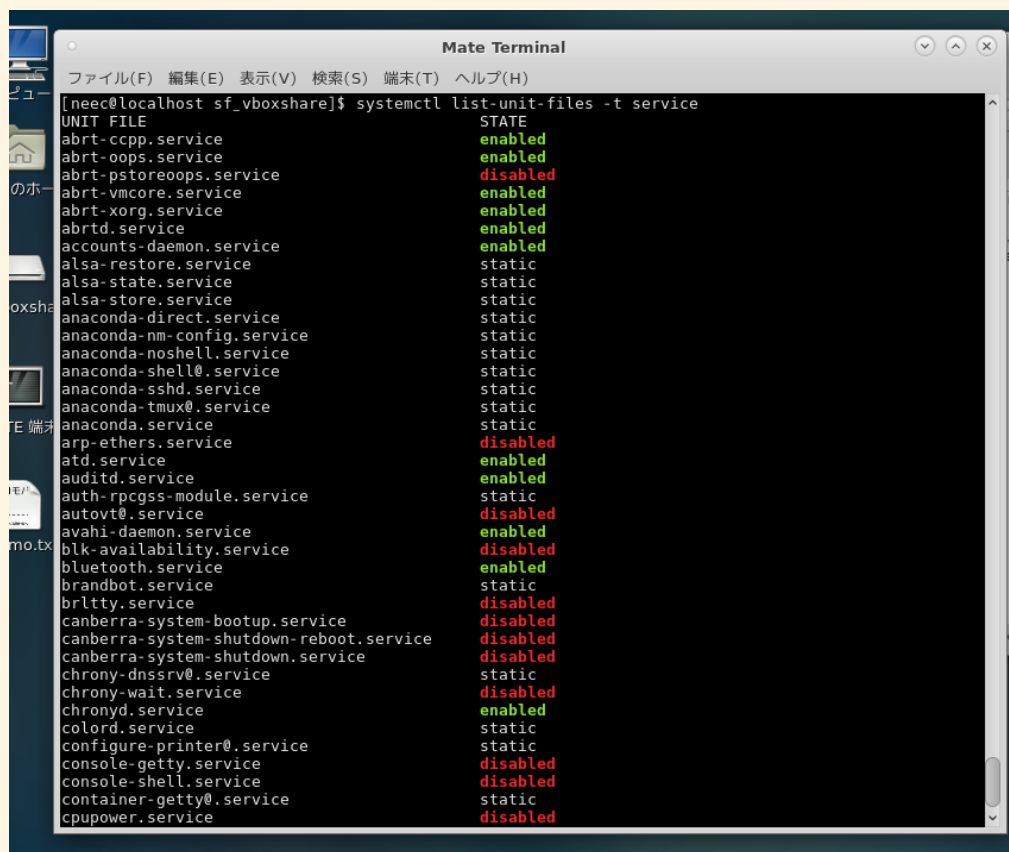
# Windowsの場合



「Ctrl」  
「Shift」  
「Esc」  
同時押し

# Linuxの場合

```
$ systemctl list-unit-files -t service
```



```
Mate Terminal
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[nec@localhost sf_vboxshare]$ systemctl list-unit-files -t service
UNIT FILE                                STATE
abrt-ccpp.service                        enabled
abrt-oops.service                        enabled
abrt-pstoreoops.service                  disabled
abrt-vmcore.service                      enabled
abrt-xorg.service                        enabled
abrt.service                             enabled
accounts-daemon.service                  enabled
alsa-restore.service                    static
alsa-state.service                       static
alsa-store.service                       static
anaconda-direct.service                  static
anaconda-nm-config.service               static
anaconda-noshell.service                 static
anaconda-shell.service                   static
anaconda-sshd.service                    static
anaconda-tmux.service                    static
anaconda.service                         static
arp-ethers.service                       disabled
atd.service                             enabled
auditd.service                          enabled
auth-rpcgss-module.service                static
autovt.service                           disabled
avahi-daemon.service                     enabled
blk-availability.service                  disabled
bluetooth.service                        enabled
brandbot.service                         static
brltty.service                           disabled
canberra-system-bootup.service            disabled
canberra-system-shutdown-reboot.service   disabled
canberra-system-shutdown.service          disabled
chrony-dnssrv.service                     static
chrony-wait.service                       disabled
chronyd.service                           enabled
colord.service                            static
configure-printer.service                 static
console-getty.service                     disabled
console-shell.service                     disabled
container-getty.service                   static
cpupower.service                          disabled
```

# デーモン

- バックグラウンドにて動作するプログラム(プロセス)
- サーバはデーモンの一種

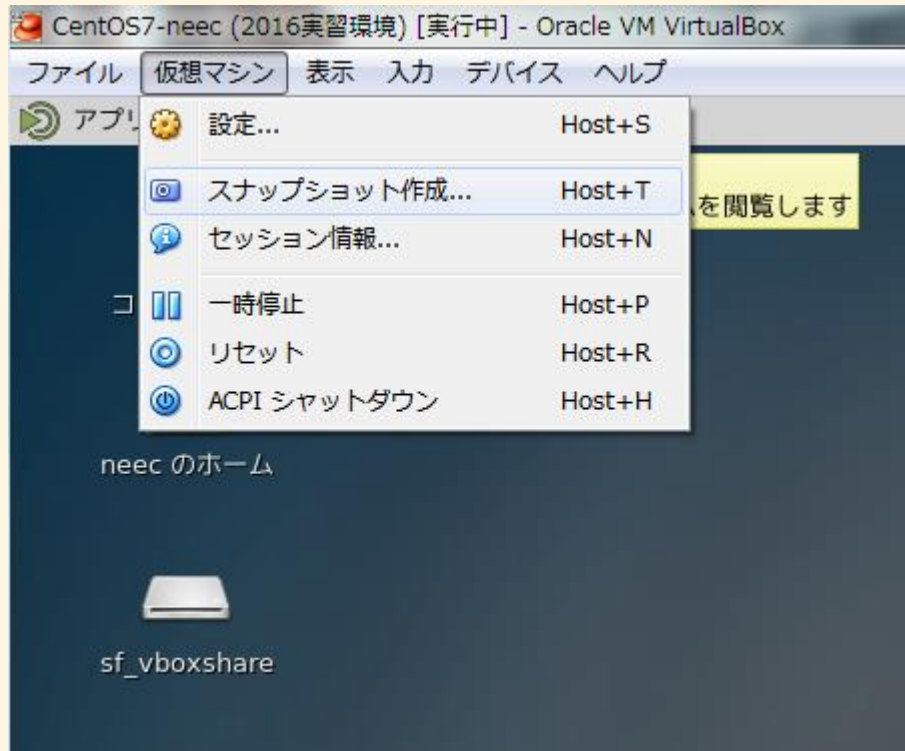
「サーバ」を  
開発する



# スナップショット

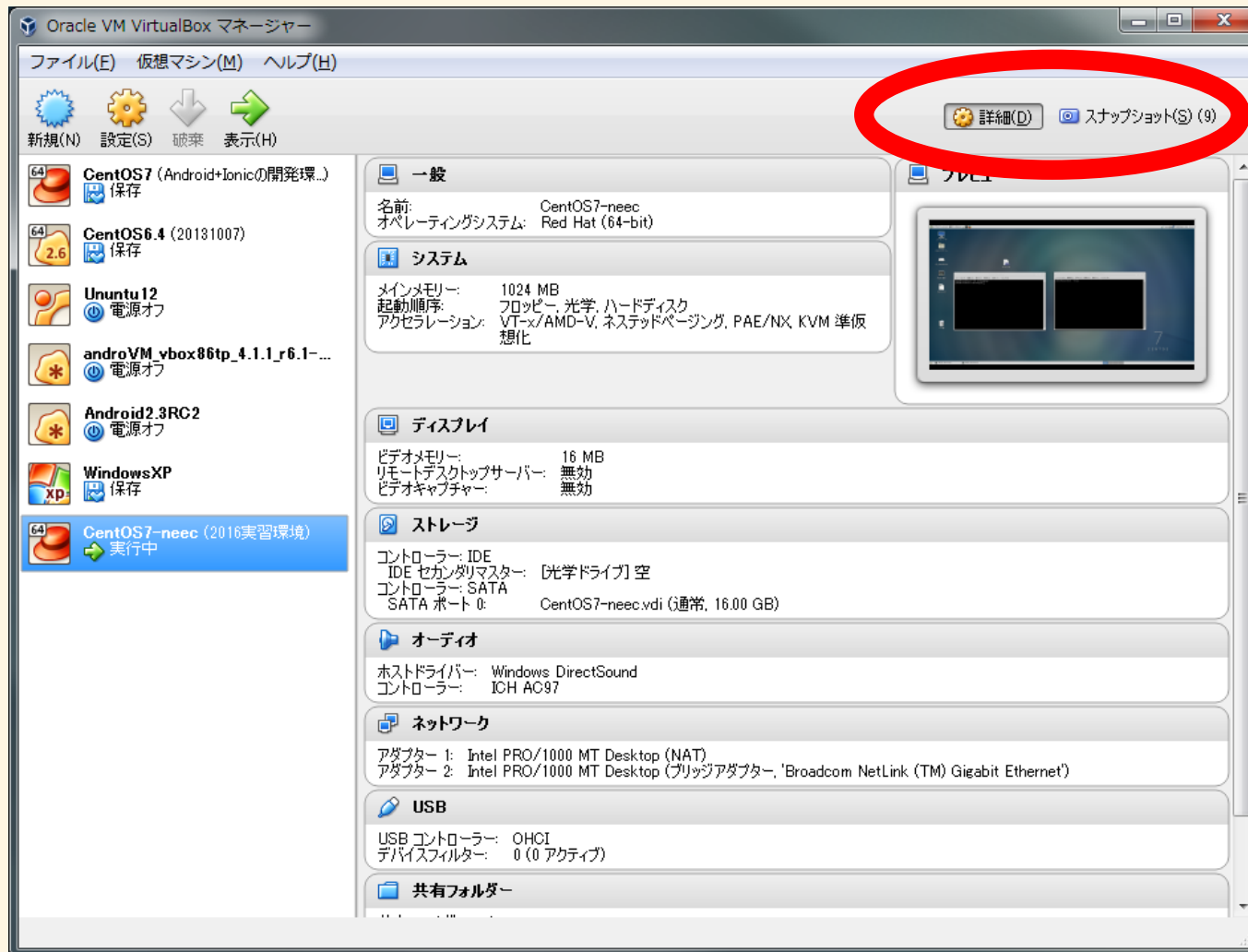


# VirtualBoxの 「スナップショット」機能



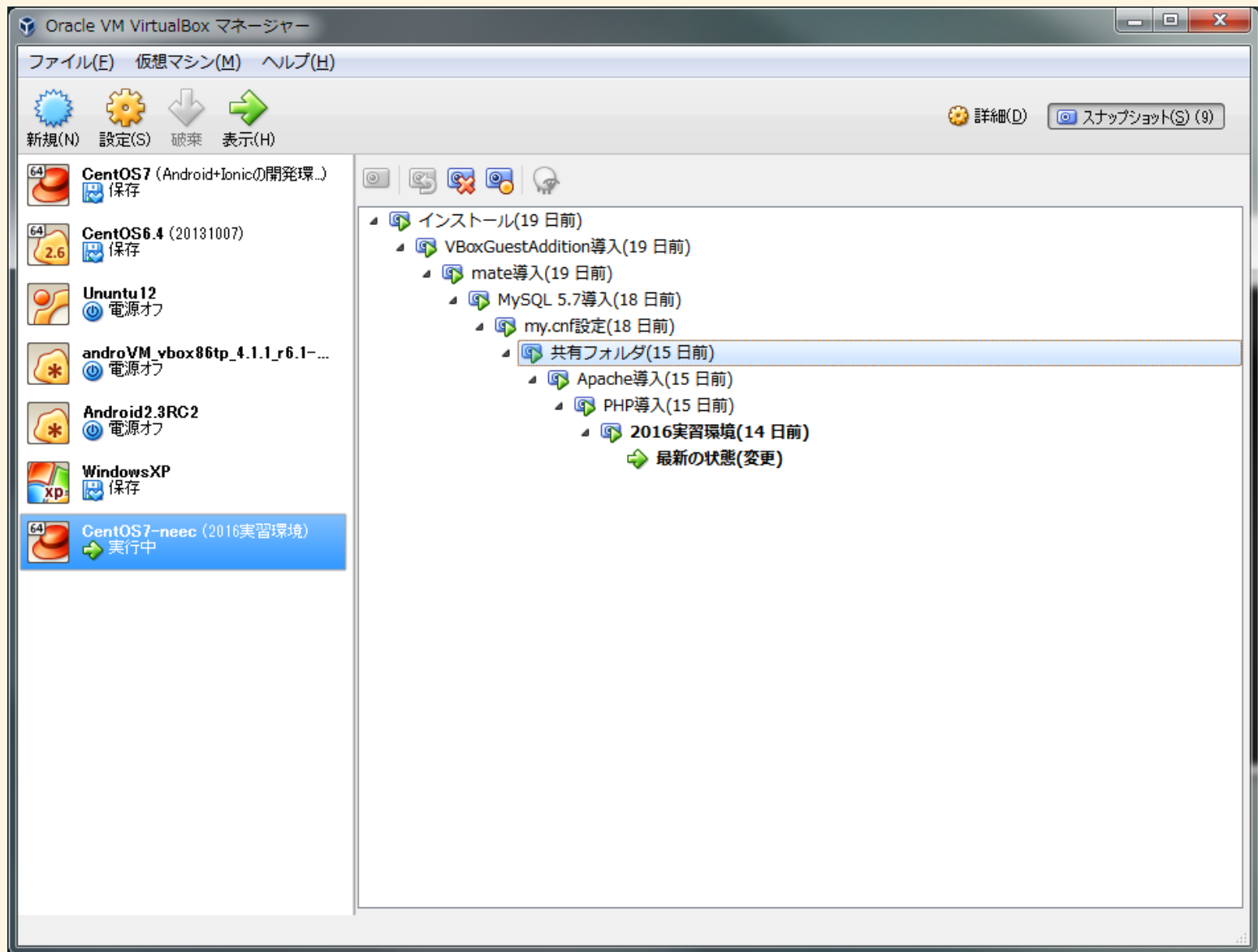
スナップショット  
を作成しておくと、  
いつでもその時点  
の状態に戻せます。

# 復元するには



「スナップ  
ショット」を  
クリック

# 復元するには



# 「telnet」 コマンド

# telnetコマンドの インストール

```
$ su
```

```
パスワード:
```

```
# cd /media/sf_vboxshare/
```

```
# ls
```

```
# yum localinstall [ファイル名]
```

# telnetコマンドの インストール

```
# exit
```



用事が終わったら必ず  
root から抜けてください！

# telnetコマンドの インストール

```
$ telnet  
telnet>
```

- telnetコマンドを打ち、このような画面になったら成功。
- “quit”と打つと、対話型のtelnetコマンドが終了します



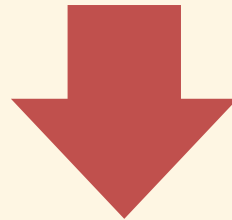
「echoサーバ」  
を動かす

# サンプルコードを保存

<https://github.com/katsube/neec/>



mobileprogramming2



20161017/echoserver.php

# サンプルコードを保存

This repository

Search

Pull requests

Issues

Gist

🔔

+

katsube / neec

👁 Unwatch

1

★ Star

0

🍴 Fork

1

<> Code

🔔 Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki

📶 Pulse

📊 Graphs

⚙ Settings

Branch: master

neec / mobileprogramming2 / 20161017 / echoserver.php

Find file

Copy path

katsube Update echoserver.php

a1be0f5 an hour ago

1 contributor

129 lines (103 sloc) | 3.68 KB

Raw

Blame

History

```
1 <?php
2 /**
3  * Echo Server
4  *
5  * クライアントから入力された文字列をそのまま返却する
6  *
7  * @author M.katsube < katsube@winning-section.net >
8  * @copyright M.katsube All Rights reserved
9  * @license The MIT License < https://opensource.org/licenses/mit-license.php >
10 */
11
12 /*-----
13  * 定数定義
14  *-----*/
15 define('IPADDRESS', '127.0.0.1');
16 define('PORT', 10000);
17
18 function echo_server($data) {
19     return $data;
20 }
```

# サンプルコードを保存

Wiki Pulse Graphs Settings

choser.php

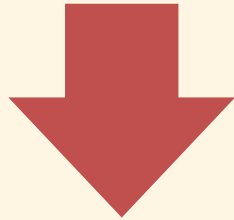
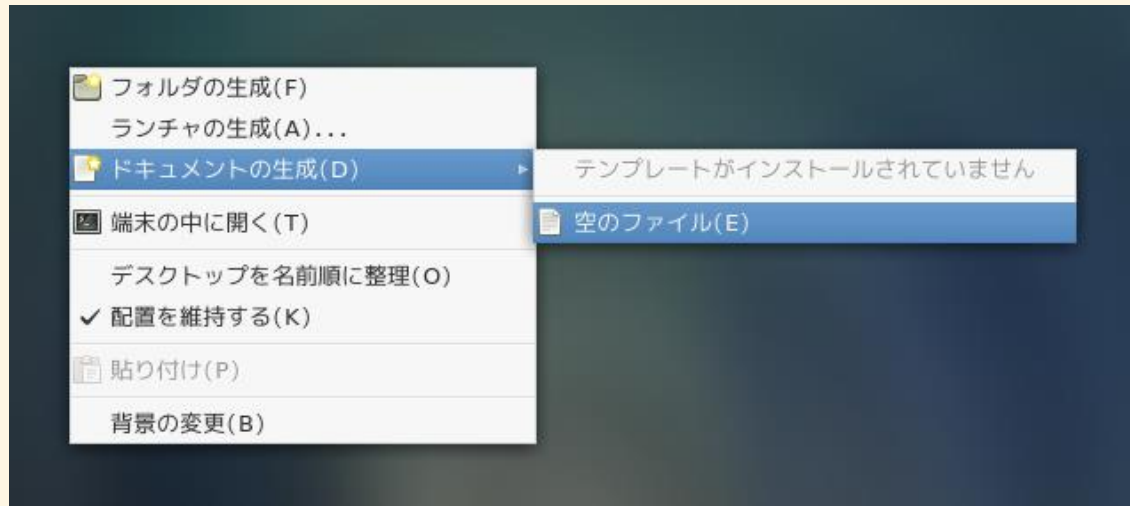
Find file Copy path

a1be0f5 an hour ago

Raw Blame History

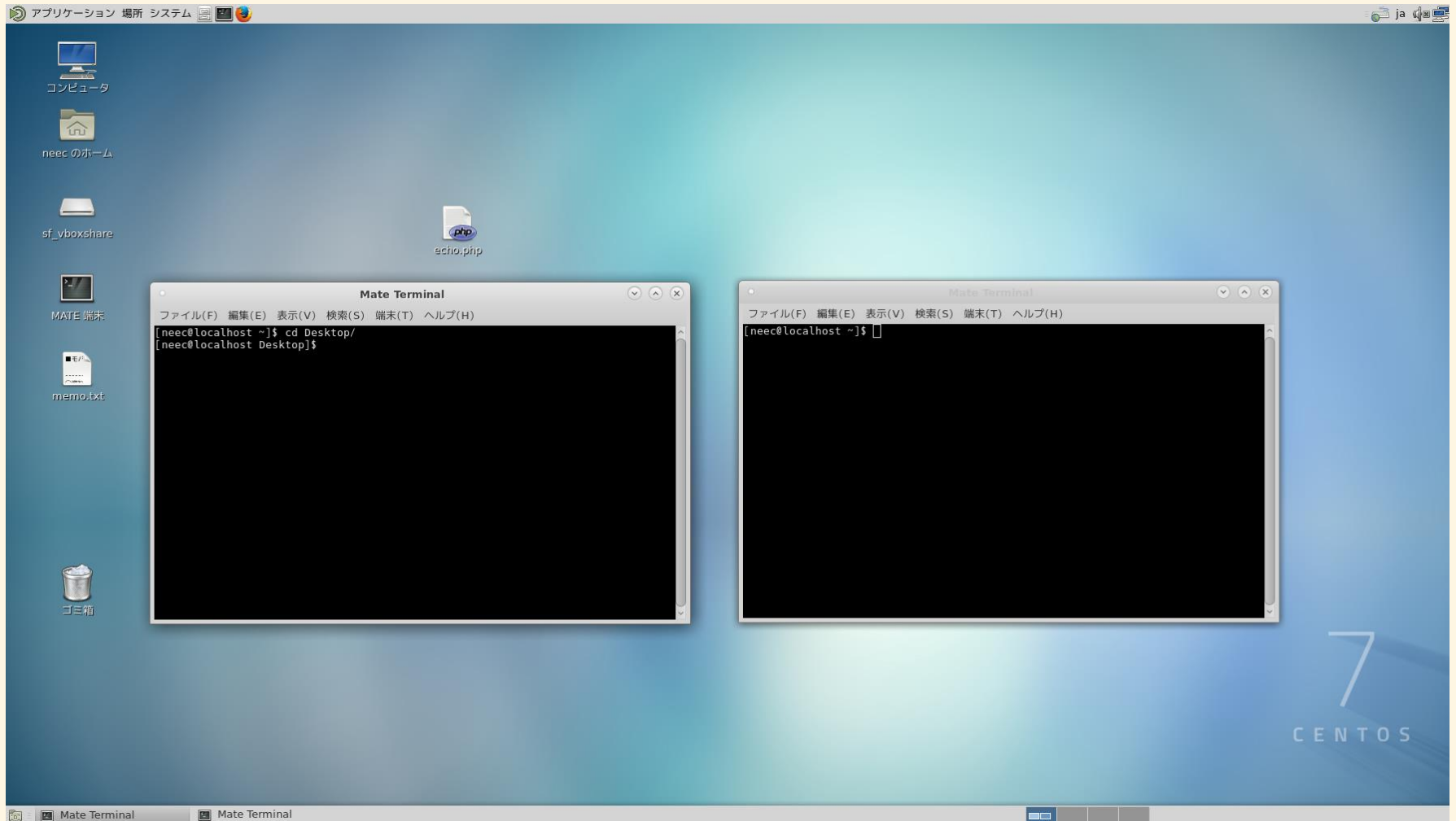
「Raw」 ボタンをクリックし  
表示されたコードをコピー

# サンプルコードを保存



「echo.php」と命名。  
先程コピーしたソース  
をここに保存します。

# 実行してみよう



# Socket

# Socket





# 「echoサーバ」 を改造する

# 演習1 「shutdown」

```
$ telnet 127.0.0.1 10000  
[Client] shutdown  
$
```

- クライアントで「shutdown」と打つと、サーバプログラムを終了

## 演習2 「quit」

```
$ telnet 127.0.0.1 10000  
[Client] quit  
$
```

- クライアントで「quit」と打つと、クライアントとの接続を切断。
- echoサーバは稼働したまま

## 演習3 「特定の文字列に反応する」

```
$ telnet 127.0.0.1 10000  
[Client] mountain  
[Server] river
```

- クライアントで「**mountain**」と打つと、サーバから「**river**」と返すように修正しましょう。

## 演習4「ファイルを返却」

```
$ telnet 127.0.0.1 10000  
[Client] /var/www/html/hello.html  
[Server]  
(hello.htmlの中身を返却)
```

- クライアントでファイルのパスを指定されると、サーバはそのファイルの内容を返却する。

## 演習5 「プログラムの実行結果」

```
$ telnet 127.0.0.1 10000
```

```
[Client] now
```

```
[Server] 2016年 10月 16日 日曜日 22:06:20 JST
```

- **now** 現在時刻を返却
- **ls (ノパス)** ls -lの結果を返却

「Webサーバ」  
にする

## 演習6 「Webサーバ」

```
$ telnet 127.0.0.1 80
```

```
GET / HTTP/1.0¥n
```

```
¥n
```

```
HTTP/1.0 200 OK
```

```
Content-Type: text/html¥n
```

```
¥n
```

```
(ファイルの内容)
```