Linux Hands-on

KEK, High Energy Accelerator Research Organization
帯名 崇
(takashi.obina@kek.jp)

はじめに

- わからないことがあれば、その都度質問してください
- いろいろな用語が出てきます。可能な限り英語の語源を考える（調べる）ように心がけましょう
- 今日は正確さよりも「ざっくりとした」イメージをつかんでもらうことを目指します

クライアント・サーバーシステム

- たとえばWikipediaによると: クライアントサーバモデル（英: client-server model）は、機能や情報（サービス）を提供するサーバと、それを利用するクライアントとを分離し、ネットワーク通信によって接続する、コンピュータネットワークのソフトウェアモデル（ソフトウェアアーキテクチャ）
- 例：PCで使っているChromeやEdge, Safari等のソフトは、Webサーバにアクセスして情報をとってくるクライアント。サーバは複数のクライアントからの要求を処理することが多い

- 語源
  - Server：サーブする人、サービスを提供する人
  - Client: 顧客、サービスを受ける人

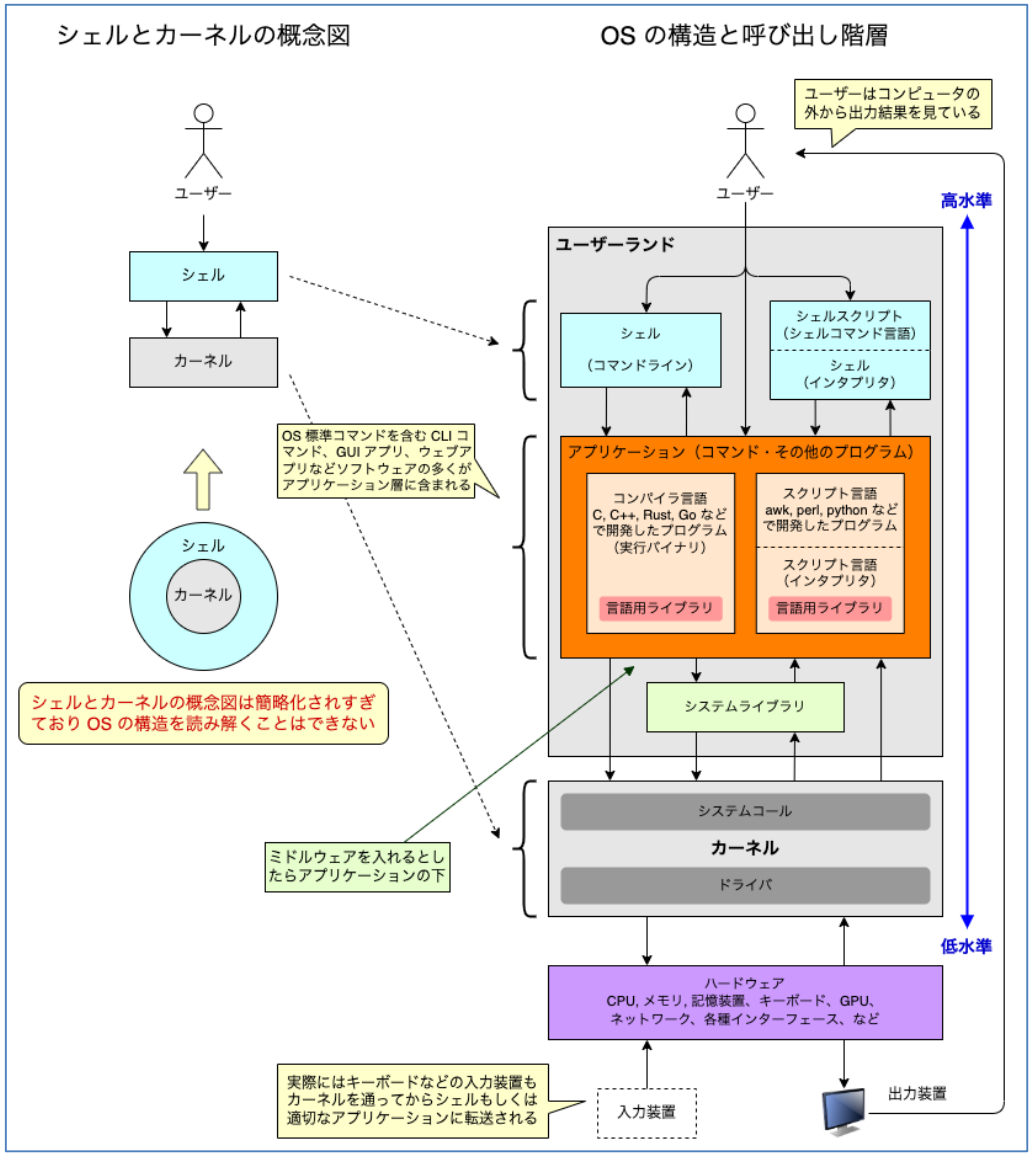
- 似た用語に「マスター・スレーブ方式」がある
  - CPUをマスターとして、周辺機器がスレーブ
  - ご主人様と奴隷モデル
  - 用語が論争になることがあるので「ペアレント・ワーカー/ヘルパー」と表現することもある

OS(Operating System)の話

- OS: Windows/MacOS/Linux/Android

- カーネルとシェル
  - カーネル（kernel, 核）： OSの中核。CPU, メモリ, Disk, networkなど、各種のハードウェアを制御
  - シェル（shell, 殻）：カーネルとユーザーとの間を橋渡しするコマンド/アプリケーション群
  - 「Linuxを使う=シェルの使い方を習熟する」、ということが第一歩となる。
  - Raspberry Pi では bash というシェルが良く使われるので実習ではこれを使用する（参考：Macのデフォルトはzsh）
  - kernelとshellを分ける理由は色々あるが、利点が多い。もちろん世の中にはいろいろな構築思想があるので適切なものを選ぶことになる→今日はその辺の話はスキップします

実習用RaspberryPiで試しながら説明します



初学者のための正しいシェルとカーネルの概念 - Qiita

実習： Rasperry Pi を起動して、さわってみる

- CUI/GUI（Character/Graphical User Interface）
  - 最近のOS（Win/Mac/Android)ではGUIがデフォルト
  - 実際に「何が動いているのか」をみるときはCUIが活躍する場面が多い

- CUIの代表： ターミナル（端末）
  - Pi : LXT Terminal, XTerm
  - Win: TeraTerm, コマンドプロンプト+ssh

- コマンド入力
  - date
  - ls
  - cd, pwd
- コマンドオプション
  - ls -l, -a, -al, -F, -R 等
  - man

コマンドラインとGUIを対比しながら、さわってみる

ファイル作成

- テキストエディタ ：Windowsではメモ帳、Macではテキストエディットなど
  - ターミナル内で使うもの
    - nano
    - vim (vi)
    - emacs
  - GUI
    - Mousepad, Leafpad など
    - VSCode (codeで起動)

- ファイル操作
  - cat, more, less
  - rm
  - mv
  - cp
- ディレクトリ操作
  - mkdir
  - rmdir

Raspberry Pi の画面で
コマンドラインとGUIの両方を表示しながら操作する

Linuxコマンドの話

- システムを少し覗いてみる
  - top　　"q"で終了だけ覚えておく
  - ps　a, u, x, f, w　:プロセスの表示、pidの確認　　Windowsのタスクマネージャーを見てみる
    - ps
    - ps a
    - ps ax
    - ps aux
    - ps auxf
    - ps auxww

a：端末を持つ全てのプロセスを表示する
u：ユーザー指向のフォーマットで表示する
x：端末(TTY)を持たない全てのプロセスを表示する
f：階層表示する

- ネットワークを覗いてみる
  - ip a　　(old) /sbin/ifconfig　　Win: ipconfig
  - ping
  - route　　(old) netstat -r　　Win: route print
  - arp
  - host　　(old) nslookup　　Win: nslookup
  - traceroute　　Win: tracert

Linuxコマンドの話

- コマンドを「組み合わせ」る
  - パイプ "|"　　ls /etc | more
  - grep
  - find
  - リダイレクト ">"　　ls -lR > tmp.txt　　画面に表示されたものをテキストファイルに保存できる

ファイルの実行

- シェルスクリプトの作成と実行
  - 複数コマンドを並べて書いて実行する
  - chmodコマンドで実行権限を与える
  - ls -l

- バイナリファイルの実行
  - C言語でプログラムを書いて実行する
  - Pythonでプログラムを書いて実行する

  - typeコマンドで属性を確認する

C言語で Hello World

- テキストエディタで以下を入力。ファイル名をhello.c として保存する

```
#include <stdio.h>

int main() {
    printf("Hello, world!¥n");
    return 0;
}
```

- ファイルを確認しコンパイルする（テキストファイルをCPUが理解・実行できるファイルに翻訳すること）

```
$ ls
hello.c
$ cat hello.c
...
$ gcc -o hello hello.c  #  gcc は "GNU C Compiler"
                        # "-o" はoutputの略で、実行ファイル名を指定する。省略した場合は a.out が出来る
$ ls -al  # ファイルの属性もチェックすること
```

- 実行し、出力を確認する

```
$ ./hello
```

Pythonで Hello World

- テキストエディタで以下を入力。ファイル名をhello.py として保存する

```
print("Hello, world!")
```

- ファイルを確認し、実行する

```
$ ls
hello  hello.c hello.py
$ cat hello.py
...
$ python hello.py
```

pythonはインタプリタ型(Interpreter, 通訳者)と呼ばれる言語で、
ソースコードを1命令ごとに解釈して逐次実行していく仕組み。

一般的にはコンパイラの方が高速だが、条件によっていろいろ変わる

確認：

- 「画面に文字を表示する」だけで色々なプログラムの書き方があることを学んだ
- コンパイラとインタプリタで異なること
- テキストファイルとバイナリファイルの差を理解すること
- ls -lR などして、ファイルの大きさなどをチェックしてみる
- fileコマンドで属性を確認してみる: file hello.py hello.c hello シェルやTerminalのなかには適当に「色」を変えてくれたりアイコンを変えてくれたりする親切なものもある

便利なシェル機能

- ヒストリ機能
- 補完機能

- タッチタイピングは重要です。必ず習得しましょう。
  - キーボードから手を離さないで。どの指でどのキーを押すか、まずは教科書で。
  - シェルの機能として ctrl-p, ctrl-n, ctrl-a, ctrl-b, ctrl-d などを使うと簡単に操作できる
  - 寿司打、EasyTypingとか、tux typing とか

自分にとって快適な（手に馴染む）プログラミング環境を模索してください。色々試すのが肝心。

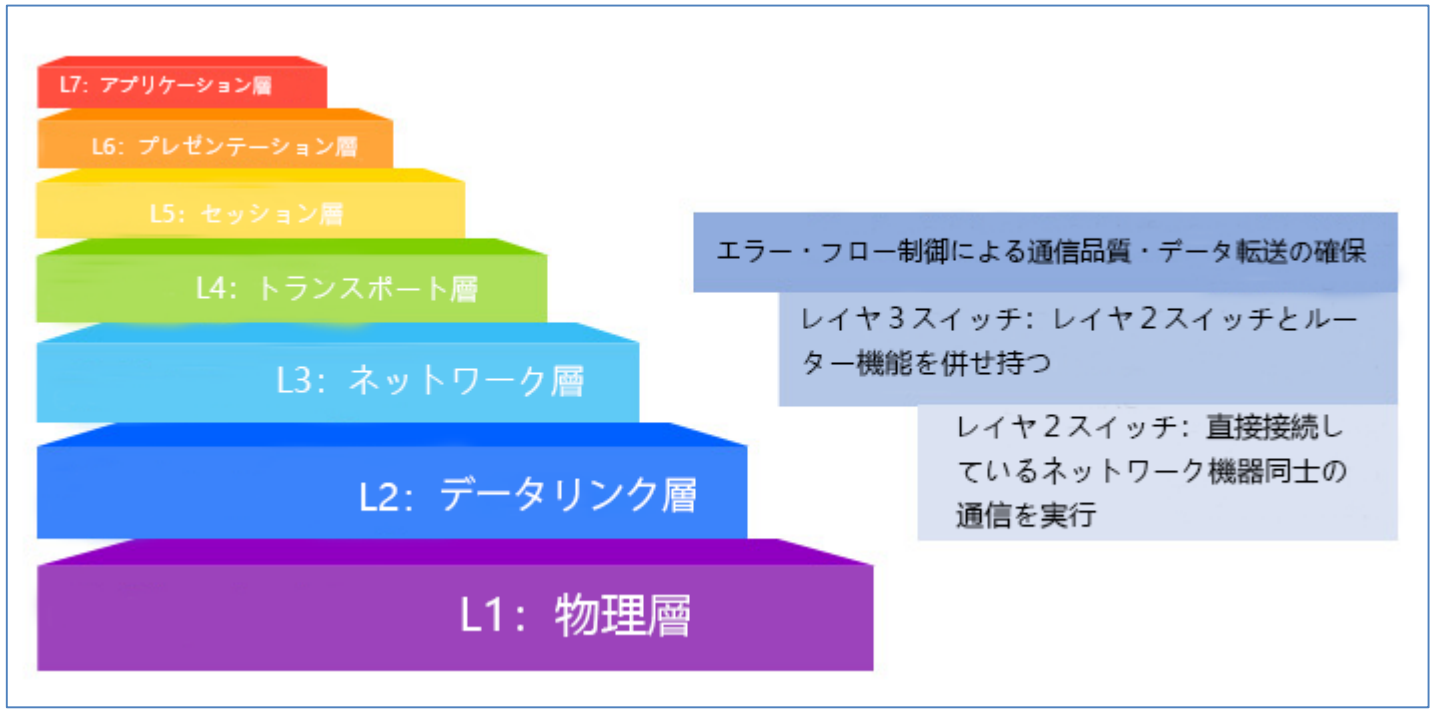
コメント： shellにも色々な種類があります。RaspberryPiではbashがデフォルトですが、高エネルギー分野では歴史的にtcshが使われている場合が多かったです。今後は fish などが流行るのかもしれませんが未来は分かりません。クラウド環境が前提での開発が増えるかもしれません。

開発環境としてVSCodeはとても便利ですが、基本となるテキストエディタを１つ習得すると強力

- 【練習シートで完全マスター】ブラインドタッチを簡単に習得する方法 (no-mark.jp)
  - どこか良いサイト知りませんか？

ネットワークの話：OSI参照モデルとTCP/IPモデル

- 色々な図があるが、たとえば



（レイヤ2）L2スイッチと（レイヤ3）L3スイッチ、その違いとは？ | FS コミュニティ
FS社はネットワーク機器を製造・販売している会社

Client/Server

クライアント・サーバーシステム

- たとえばWikipediaによると: クライアントサーバモデル（英: client-server model）は、機能や情報（サービス）を提供するサーバと、それを利用するクライアントとを分離し、ネットワーク通信によって接続する、コンピュータネットワークのソフトウェアモデル（ソフトウェアアーキテクチャ）
- 例：PCで使っているChromeやEdge, Safari等のソフトは、Webサーバにアクセスして情報をとってくるクライアント。サーバは複数のクライアントからの要求を処理することが多い

- 語源
  - Server：サーブする人、サービスを提供する人
  - Client: 顧客、サービスを受ける人

- 似た用語に「マスター・スレーブ方式」がある
  - CPUをマスターとして、周辺機器がスレーブ
  - ご主人様と奴隷モデル
  - 用語が論争になることがあるので「ペアレント・ワーカー/ヘルパー」と表現することもある

フィールドバス

- そもそも「バス」形式とは？

- real-timeが必要なのかどうかが大きな分かれ目
- いわゆる普通の「イーサーネット」はリアルタイム性は（あまり）無い。

Fieldbus

ref: https://en.wikipedia.org/wiki/Fieldbus

From Wikipedia, the free encyclopedia

Fieldbus is the name of a family of industrial computer networks[1] used for real-time distributed control. Fieldbus profiles are standardized by the International Electrotechnical Commission (IEC) as IEC 61784/61158.

A complex automated industrial system is typically structured in hierarchical levels as a distributed control system (DCS). In this hierarchy the upper levels for production managements are linked to the direct control level of programmable logic controllers (PLC) via a non-time-critical communications system (e.g. Ethernet). The fieldbus[2] links the PLCs of the direct control level to the components in the plant of the field level such as sensors, actuators, electric motors, console lights, switches, valves and contactors and replaces the direct connections via current loops or digital I/O signals. The requirement for a fieldbus are therefore time-critical and cost sensitive. Since the new millennium a number of fieldbuses based on Real-time Ethernet have been established. These have the potential to replace traditional fieldbuses in the long term.

Contents [hide]