1. システムアーキテクチャ

1.1 ハードウェアの基本知識と設定

1.1.1 基本的なシステムハードウェア

- CPU
  - メモリ上からプログラムを読み出し，処理を実行する．
  - 動作周波数が性能の指標の1つ.
- メモリ
  - データを記憶する役割を持つ．
  - メモリに記憶された内容は，システムの電源が切れると消えてしまう．
- ハードディスク
- 入力装置
- 拡張カード
  - マザーボードの拡張スロットに装着する形で様々な機能を提供する．
- USB機器

1.1.2 BIOS/UEFI

- **BIOS** (Basic Input Output System)
  - キーボードやハードディスクなどのデバイスを制御する，最も基本的なプログラム．
    - コンピュータのマザーボードや拡張カードに搭載されたフラッシュROMに書き込まれている．
    - OS やアプリケーションは，BIOS のインタフェースを利用して簡単にハードウェアにアクセスすることができる．
  - OS を起動するためのプログラムをディスクから読み込んで実行する．
  - デバイスの動作を設定する．
  - 基本的な入出力を制御する．
- **UEFI** (Unified Extensible Firmware Interface)
  - BIOS の後継となるファームウェア規格．

コンピュータの電源を入れると，システム BIOS/UEFI のメッセージが表示される．このときに特定のキーを押すことで，BIOS/UEFI セットアップ画面を呼び出すことができる．

BIOS/UEFI セットアップで設定できる項目:

- 日付や時刻 (ハードウェアクロック)
- ディスクドライブや各種デバイスのパラメータ
- キーボードの使用/不使用
- 電源管理
- 起動ドライブの順序
- デバイスへの IRQ (Interrupu Request: 割り込み要求) の割り当て
- 各種デバイスの使用/不使用

BIOS セットアップでは，一部のキーしか利用できない．

1.1.3 デバイス情報の確認

Linux は，ハードウェアのアクセスを抽象化する **デバイスファイル** を持っている．

- すべてのハードウェアはデバイスファイルとして表される．
- デバイスファイルの読み書きを通してハードウェアにアクセスできるようになっている．
- デバイスファイルは `/dev` ディレクトリ以下にある．
    - これらのデバイスファイルは，udev という仕組みによって自動的に作成される．

```
[ai@localhost ~]$ ls /dev
autofs           fuse              oldmem    tty10  tty29  tty47  tty8     vcsa1
block            hidraw0           port      tty11  tty3   tty48  tty9     vcsa2
bsg              hpet              ppp       tty12  tty30  tty49  ttyS0    vcsa3
btrfs-control    hugepages         ptmx      tty13  tty31  tty5   ttyS1    vcsa4
(以下略)
```

- `/proc` ディレクトリ
    - Linux カーネルが認識しているデバイスに関する情報の一部を確認できるファイルが格納されている．
        - ファイルとしての実態がない仮想的なファイル．

`/proc` ディレクトリ以下の主なファイル:

ファイル名	説明
`/proc/cpuinfo`	CPU 情報
`/proc/interrupts`	IRQ 情報
`/proc/ipports`	I/O アドレス情報
`/proc/meminfo`	メモリ情報

ファイル名	説明
/proc/bus/usb/*	USBデバイス情報
/proc/bus/pci/*	PCIデバイス情報

```
[ai@localhost ~]$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 94
model name      : Intel Core Processor (Skylake, IBRS)
stepping        : 3
microcode       : 0x1
cpu MHz         : 1799.995
cache size      : 16384 KB
(以下略)
```

PCI デバイスの情報を表示するには， `lspci` コマンドを使う．

```
ai@ai-Standard-PC-Q35-ICH9-2009:~$ lspci
00:00.0 Host bridge: Intel Corporation 82G33/G31/P35/P31 Express DRAM Controller
00:01.0 VGA compatible controller: Red Hat, Inc. QXL paravirtual graphic card (rev 04)
00:02.0 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.1 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.2 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.3 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.4 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.5 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
(以下略)
```

1.1.4 USB デバイス

USB の特徴

- 最大 127 台までの USB デバイスを接続可能．
- さまざまな USB デバイスを同一のコネクタで接続可能．
- 電源を入れたままの接続・取り外しに対応 (ホットプラグ)．
- プラグ & プレイをサポート．
- USB ポートから USB デバイスに電源を供給可能．

USB のバージョン:

バージョン	最大データ転送速度

バージョン	最大データ転送速度
USB 1.0	12Mビット/秒
USB 1.1	12Mビット/秒
USB 2.0	480Mビット/秒
USB 3.0	5Gビット/秒
USB 3.1	10Gビット/秒

- ハードウェアを利用するには，**デバイスドライバ** が必要．
- USB の場合は，USB デバイス固有のデバイスドライバ以外に，Linux システムに最初から搭載されている汎用のドライバ (クラスドライバ) もある．

主なデバイスクラス:

デバイスクラス	サポートするUSBデバイス
HID (Human Interface Device)	キーボード，マウスなど
マスストレージデバイス	USBメモリ，デジタルオーディオプレーヤー， ハードディスクドライブなど
オーディオ	マイク，スピーカー，サウンドカードなど
プリンタ	プリンタなど
ワイヤレスコントローラー	Wi-Fi アダプタ，Bluetooth アダプタなど

`lsbsb` コマンドで USB デバイスの情報を表示できる．

書式:

```
lsusb [option]
```

`lsusb` コマンドの主なオプション:

option	説明
`-v`	詳細に表示する

option	説明
`-t`	USB デバイスの階層構造をツリー状に表示する

```
ai@ai-Standard-PC-Q35-ICH9-2009:~$ lsusb
Bus 001 Device 002: ID 0627:0001 Adomax Technology Co., Ltd QEMU USB Tablet
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

1.1.5 udev

`/dev` ディレクトリ以下のデバイスファイルは，**udev** (Userspace DEVice management) という仕組みによって自動的に作成される．

- デバイス (ハードウェア) が接続されると，カーネルがそれを検知し， `/sys` ディレクトリ以下にデバイス情報を作成する．
- udev デーモン (`udevd`) がそのデバイス情報を参照して， `/dev` ディレクトリ以下にデバイスファイルを作成する．
- その際に， `/etc/udev/rules.d` ディレクトリ以下の設定ファイルが使われる．
    - 設定ファイルを編集することで，たとえばあるUSBメモリを `/dev/usbmemory` とするといったように，特定のハードウェアを任意の名前のデバイスファイルとして扱えるようにもできる．

デバイスの情報は，**D-Bus** (Desktop Bus) と呼ばれるアプリケーション間でやりとりを行うための機構によってアプリケーションに伝えられ，アプリケーションからデバイスを利用できるようになる．

1.1.6 デバイスドライバのロード

- **デバイスドライバ**
    - デバイスを利用するために必要な制御プログラム．
    - Linux では，デバイスドライバはカーネルの一部 (カーネルモジュール) として提供されている．
    - ロードされているカーネルモジュールを確認するには， `lsmod` コマンドを使う．

```
[ai@localhost ~]$ lsmod
Module                  Size  Used by
ip6t_rpfilter          12595  1
ip6t_REJECT            12625  2
nf_reject_ipv6         13717  1 ip6t_REJECT
ipt_REJECT             12541  2
nf_reject_ipv4         13373  1 ipt_REJECT
(以下略)
```

通常，必要なデバイスドライバは自動的にロードされる．手動でロードする場合は，`modprobe` コマンドを実行する．

1.2 Linux の起動とシャットダウン

1.2.1 システムが起動するまでの流れ

システムの電源を入れてから OS が起動するまでの流れは，コンピュータのアーキテクチャによって異なる．ここでは，一般的な PC における起動の手順を見ていく．

- 電源投入
- BIOS/UEFI
  - ハードウェアのチェック，初期化．
  - 起動デバイスに書き込まれたブートローダの読み出し．
  - ブートローダに制御を移す．
- ブートローダ
  - 起動デバイス上からカーネルをメモリ上に読み込む．
- カーネル
  - メモリの初期化やシステムクロックの設定などを行う．
  - 仮のルートファイルシステム (initramfs: 初期 RAM ディスク) をマウントする．
    - 初期 RAM ディスクには，システムの起動に必要なデバイスドライバが組み込まれている．これを使ってハードディスク等のデバイスへアクセスできるようになる．
- `init` / `systemd`
  - 必要なサービスなどを順次起動していき，最後にログインプロンプトを表示して起動処理を完了する．

1.2.2 起動時のイベント確認

`dmesg` コマンドを使うと，システム起動時にカーネルがどのような処理を行ったか確認できる．

```
ai@ai-Standard-PC-Q35-ICH9-2009:~$ dmesg | head -5
[    0.000000] Linux version 5.11.0-41-generic (buildd@lgw01-amd64-005) (gcc (Ubuntu 9.3.0-17ubuntu1~20
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-5.11.0-41-generic root=UUID=869d71ea-c63b-45ec-bd
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
```

- `dmesg` コマンド
  - カーネルが出力したメッセージを一時的に蓄えておくバッファの内容を表示する．
  - システムが起動した後もカーネルが出力するメッセージが蓄えられる．
    - バッファに収まりきらなくなった古いメッセージは消えていく．

システム起動時のメッセージは，ログファイル `/var/log/messages` や， `/var/log/dmesg` ，`/var/log/boot.log` にも保存される．

`systemd` を採用したシステムでは， `journalctl -k` コマンドでカーネルのバッファ内容を表示できる． `-b` オプションを使うと，システム起動時のメッセージを表示できる．

```
ai@ai-Standard-PC-Q35-ICH9-2009:~$ journalctl -kb
-- Logs begin at Tue 2021-11-30 12:29:00 JST, end at Tue 2021-12-21 14:30:18 JST. --
12月 21 13:01:22 ai-Standard-PC-Q35-ICH9-2009 kernel: Linux version 5.11.0-41-generic (buildd>
12月 21 13:01:22 ai-Standard-PC-Q35-ICH9-2009 kernel: Command line: BOOT_IMAGE=/boot/vmlinuz->
12月 21 13:01:22 ai-Standard-PC-Q35-ICH9-2009 kernel: KERNEL supported cpus:
12月 21 13:01:22 ai-Standard-PC-Q35-ICH9-2009 kernel:   Intel GenuineIntel
(以下略)
```

1.2.3 システムのシャットダウンと再起動

- シャットダウン
  - システム上で動作しているさまざまなプログラムを適切に終了してシステムを安全に停止させること．
  - `shutdown` コマンドを使って操作する．

書式:

```
shutdown [option] 時間 [メッセージ]
```

shutdown コマンドの主なオプション:

option	説明
`-h`	シャットダウンする
`-r`	シャットダウン時にシステムを再起動する
`-f`	次回起動時にfsckをスキップする (`-h` または `-r` と組み合わせて利用する)
`-F`	次回起動時にfsckを必ず実行する (`-h` または `-r` と組み合わせて利用する)
`-k`	実際にシャットダウンせず，警告メッセージを通知する
`-c`	現在実行中のシャットダウンをキャンセルする

- systemd を採用したシステムについて

- `shutdown -k` コマンドを使ってもメッセージが通知されないことがある．`wall` コマンドを使ってメッセージを通知する．
- `shutdown` コマンドの代わりに `systemctl reboot` (再起動)，`systemctl poweroff` (システム終了) を使う．

1.3 SysVinit

1.3.1 SysVinit による起動

SysVinit では，Linux システムで最初に実行されるプロセスである init が， `/etc/inittab` ファイルの設定に従い，システムに必要なサービスを順次起動していく．

1. init が `/etc/inittab` ファイルを読み込む
2. init が `/etc/rc.sysinit` スクリプトを読み込む
3. init が `/etc/rc` スクリプトを実行する
4. `/etc/rc` スクリプトが `/etc/rc[runlevel].d` ディレクトリ以下のスクリプトを実行する

SysVinit では，あらかじめ決められた順にサービスが起動していくので，あるサービスの起動に手間取ると，それ以後に起動するサービスが待たされてしまい，最終的な起動完了まで時間がかかってしまう．

1.3.2 ランレベル

RH 系のランレベル:

ランレベル	説明
0	停止
1	シングルユーザーモード
2	マルチユーザーモード (テキストログイン，NFS サーバは停止)
3	マルチユーザーモード (テキストログイン)
4	未使用
5	マルチユーザーモード (グラフィカルログイン)
6	再起動
S または s	シングルユーザーモード

Debian 系のランレベル:

ランレベル	説明
0	停止

ランレベル	説明
1	シングルユーザーモード
2	マルチユーザーモード
3	マルチユーザーモード
4	マルチユーザーモード
5	マルチユーザーモード
6	再起動
S または s	シングルユーザーモード

シングルユーザーモード は，root ユーザのみが利用できる特殊な状態．

1.3.3 ランレベルの確認と変更

`runlevel` コマンドで現在のランレベルとその前のランレベルを表示できる．

```
[ai@localhost ~]$ runlevel
N 3
```

ランレベルを変更するには，root で `init` コマンドまたは `telinit` コマンドを使う．

1.3.4 起動スクリプトによるサービスの管理

- SysVinit では，各種サービスの起動には， `/etc/init.d` ディレクトリ以下に用意されている起動スクリプトが使われる．
- `/etc/rc[runlevel].d` ディレクトリに，各ランレベルで起動するサービス，終了するサービスのスクリプトファイルが配置されている．

```
[ai@localhost ~]$ ls /etc/rc3.d/
K50netconsole  S10network
```

K で始まるファイルと S で始まるファイルがある．いずれも， `/etc/init.d` ディレクトリにある起動スクリプトへのシンボリックリンクである．

```
[ai@localhost ~]$ ls -l /etc/rc3.d/K50netconsole
lrwxrwxrwx. 1 root root 20 11月 30 11:25 /etc/rc3.d/K50netconsole -> ../init.d/netconsole
```

起動スクリプトは，システムサービスや各種サーバを起動したり，再起動したり，終了したりする場合に利用する．例えば，network サービスを起動するには，

```
/etc/init.d/network start
```

とする．

1.3.5 デフォルトのランレベルの設定

Linux が起動すると，最初のプロセスとして init が実行され，指定されたランレベルで起動する．デフォルトのランレベルは `/etc/inittab` に記述されている．
このファイルを書き換えることで，デフォルトのランレベルを変更できる．

1.4 systemd

1.4.1 systemd の概要

- **systemd** を採用したシステムでは，init プロセスの代わりに systemd プロセスが起動し，各種サービスを管理する．
- systemd では，以下のような複数のデーモンプロセス (常駐プロセス) が連携して動作する．
  - デーモン (daemon)
    - メモリ上に常駐してシステムサービスやサーバサービスを提供するプロセス．

systemd 関連の主なデーモンプロセス:

プロセス	説明
`systemd`	systemd のメインプロセス
`systemd-journald`	ジャーナル (ログ) 管理プロセス
`systemd-logind`	ログイン処理プロセス
`syustemd-networkd`	ネットワーク管理プロセス
`systemd-timesyncd`	システムクロック同期プロセス
`systemd-resolved`	名前解決プロセス
`systemd-udevd`	デバイス動的検知プロセス

systemd では，システムの起動処理は多数の **Unit** と呼ばれる処理単位に分かれる．

Unit の主な種類:

種類	説明
`service`	各種サービスを起動する
`device`	各種デバイスを表す
`mount`	ファイルシステムをマウントする
`swap`	スワップ領域を有効にする
`target`	複数の Unit をグループ化する
`timer`	指定した日時や間隔で処理を実行する

- systemd では，各種サービスの依存関係や順序関係を処理できる．
  - c.f. SysVinit ではサービスは順次起動する．
- サービスの起動が並列的に行われる．

1.4.2 systemd の起動手順

システムが起動すると， `default.target` という Unit が処理される．
この Unit の設定ファイルは， `/etc/systemd/system` ディレクトリ以下にある．
これは， `/lib/systemd/system/*.target` へのシンボリックリンクになっている．

ランレベルとターゲットの対応:

ランレベル	ターゲット
0	`poweroff.target`
1	`rescue.target`
2,3,4	`multi-user.target`
5	`graphical.target`
6	`reboot.target`

1.4.3 systemctl によるサービスの管理

systemd でサービスを管理するには， `systemctl` コマンドを使う．

書式:

```
systemctl サブコマンド [Unit名] [-t 種類]
```

systemctl コマンドの主なサブコマンド:

サブコマンド	説明
`start`	サービスを起動する
`stop`	サービスを終了する
`restart`	サービスを再起動する
`reload`	サービスの設定を最読み込みする

サブコマンド	説明
`status`	サービスの稼働状況を表示する
`is-active`	サービスが稼働しているかどうかを確認する
`enable`	システム起動時にサービスを自動起動する
`disable`	システム起動時にサービスを自動起動しない
`mask`	指定した Unit をマスクし，手動でも起動できないようにする
`unmask`	指定した Unit のマスクを解除する
`list-dependencies`	Unit の依存関係を表示する
`list-units`	起動しているすべての Unit と状態を表示する
`list-unit-files`	すべての Unit を表示する
`reboot`	システムを再起動する
`shutdown`	システムをシャットダウンする