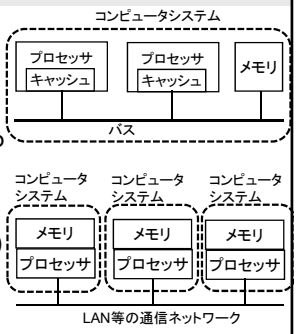
オペレーティングシステム

資料　第3分冊(H30)

村田正幸　(murata@ist.osaka-u.ac.jp)
○松田秀雄(matsuda@ist.osaka-u.ac.jp)

並列コンピュータ(または分散コンピュータ)の構成

- **マルチプロセッサ**
  - 密結合(Tightly coupled)
  - SMP(symmetric multi-processor)とも呼ばれる
  - 共有メモリ
- **マルチコンピュータ**
  - 疎結合(loosely coupled)
  - 専用メモリ
  - 自立的

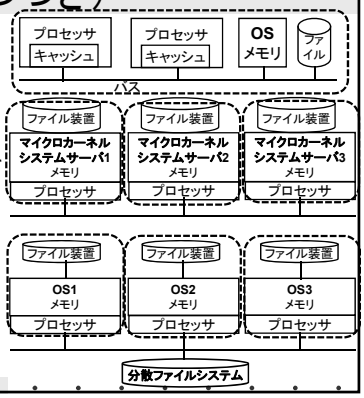


並列／分散コンピュータのOS

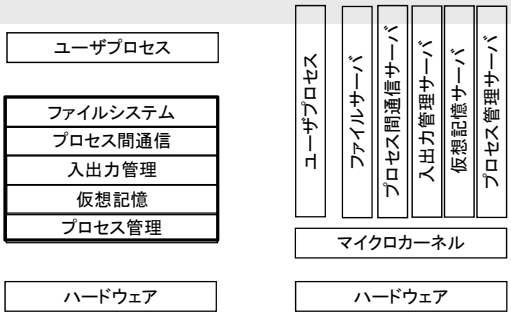
- マルチプロセッサOS
  - **物理的に**単一計算機のOS(プロセッサだけが複数個ある)
  - 1つのOS(カーネル)が複数のプロセッサで実行される
- 分散OS
  - **仮想的に**単一計算機のように見ることができる(プロセッサだけでなく、メモリもファイル装置も複数個ある)
  - OS(カーネル)はプロセッサごとに複数個あり、相互に協調して動作する
  - 通信はメッセージを介して行われる
- ネットワークOS
  - 独立した複数の計算機で、OSもそれぞれに独立している
  - 各計算機のファイル装置が、**分散ファイルシステム**により共有されている

並列／分散コンピュータのOS（つづき）

- **マルチプロセッサOS**
  - 共有メモリ上に存在
- **分散OS**
  - 専用メモリ上に存在
  - マイクロカーネル方式を取ることが多い(**システムサーバプロセスを分散**)
- **ネットワークOS**
  - 専用メモリ上に存在
  - ファイル装置の仮想化・共通化(**分散ファイルシステム**)



参考　マイクロカーネル方式



単層カーネル方式
(単一のプログラムなので、モジュールごとの分離は不可能で、**各マシンでそれぞれ実行**するしかない)

マイクロカーネル方式
(システムサーバプロセスはマイクロカーネルとは分離でき、**別のマシンで実行したシステムサーバプロセスを利用できる**)

1.3.2 OSの実際

[1] OSの実際的な動作　－概要－
OSの起動時(**ブート**(boot)と呼ばれる)
① ファイル装置からメインメモリに読み出す
② メインメモリに割り付ける
③ メインメモリに保持して、OS自身を起動する
OSによる仮想化
- ハードウェアの仮想化により、OSの機能が同じであれば、ハードウェアに依存せずに、ユーザプログラムがオブジェクトプログラムのレベルで動作する

→ **オブジェクト互換性**(プログラムをコンパイルし直さなくても、別のコンピュータで実行できる)
⇔ **ソース互換性**(ソースプログラムをコンパイルし直せば、別のコンピュータで実行できる)

7

OSの起動

OSの起動(boot)過程

1. ブートROMにある**ブートストラップローダ**(bootstrap loader)による、ファイル装置から**IPL** (initial program loader)をメインメモリに読み出す
2. IPLに制御権が移り、IPLがOSプログラムをメインメモリへ読み出す(ロードする)
3. OSを初期化する
   ① ハードウェア装置の認識
   ② ファイル装置上のファイルの位置を識別
   ③ メインメモリ上にOSを常駐する領域や空間を確保する

8

OSの起動(つづき)

OSの再起動(reboot)

- **ハードリセット**(コールドブートともいう)
  - 電源オンからやり直す
- **ソフトリセット**(ウォームブートともいう)
  - 動手順の2. からやり直す

OSの停止(shutdown)

1. すべてのユーザプロセス(ユーザプログラム)について停止処理を行う
   - オープンしていたファイルはクローズする
2. 電源をオフする

9

1.3.3 OSと仮想マシン ーOS機能の隠ぺいー

- ハードウェアだけでなく、**OSそのものも隠ぺい**する
- なぜ、OSを隠ぺいする必要があるのか?

(参考)*ハードウェア機構の隠ぺいの目的*

- *個々のハードウェアが持つ物理的な特性や相違点を隠す*

→*隠ぺいにより、**共通の(論理的な)装置に統一化***

OSの隠ぺいの目的

- 個々のOSが持つ個別の特徴や相違点を隠す
  (例:Windows XP/ Vista/ 7/ 8/ 10, macOS, Linux, FreeBSD)
  → 隠ぺいにより、OSを統一化することで**多様なOSに対応できる**(詳細は後述)

10

仮想マシン

- **Virtual Machine, VM**ともいう
  - ソフトウェアによって、物理的な(実際の)ハードウェア機構や機能をシミュレーション
  - 仮想的あるいは論理的機構や機能に見せかける
- ソフトウェア機能で実現する
  - ハードウェア機構の長所である高速処理能力が、ある程度失われる
  - 仮想マシンの管理や切り替えにオーバヘッドが生じる
    - 最近の実装(VMWare, Xen, KVMなど)ではそれほど性能が落ちるわけではない

11

エミュレーションとファームウェア

- **エミュレーション**
  - 物理的なハードウェア機構や機能をシミュレーションを、ハードウェアにより行うこと
  - ファームウェアにより実現することが多い
  - 1台の実コンピュータ上で複数の仮想マシンをエミュレーションすることができる(図1.13)
- **ファームウェア**
  - ハードウェアを制御するソフトウェア(マイクロプログラムともいう)
  - FPGA (Field Programmable Gate Array)もファームウェアの一種

12

OSと仮想マシンOS

- 実コンピュータ上で複数の仮想マシンを設定する場合は、次の2種類のOSが機能することになる(図1.14)
  - **ホストOS**(実コンピュータのOS)
    実コンピュータ(ホストコンピュータ)上で稼働し、その実コンピュータを管理・制御するOS
  - **ゲストOS**(仮想マシンのOS)
    実コンピュータ上で稼働する各仮想マシンを管理・制御するOS
- 仮想マシンのそれぞれに異種のゲストOSを搭載し、同時に稼働させるコンピュータシステムを、「**マルチOSコンピュータ**」という

13

OS機能の隠ぺい

- マルチOSコンピュータでは、次の2段階の隠ぺいが行われている
  1. **ホストOS**による実コンピュータのハードウェア機構の隠ぺい
  2. **ゲストOS**による仮想マシンの隠ぺい
- この2段階の隠ぺいにより、「**仮想マシンによるホストOSの隠ぺい**」が行われる
- より隠ぺいのレベルが高くなっている（図1.15）
  - **ゲストOS： 仮想マシンを隠ぺい**
  - **仮想マシン： ホストOSを隠ぺい**
  - ホストOS: 実ハードウェアを隠ぺい

14

OS機能の隠ぺいの目的

目的

- **多様なゲストOSに対処**する
  - 例：複数のOS（例えば、Windows, Linux, macOSなど）を、1台の実コンピュータ上で動作させることで、多様なプログラム実行環境を提供できる
- **ゲストOSそのものの可搬性**を高める
  - ハードウェアの違いを仮想マシンが隠ぺいするので新たに別のゲストOSを移植するのが容易となる（例：MacでもWindowsを動作させることができる）

15

OS機能の隠ぺいの効果

- 1台の実コンピュータ上で、多種多数のユーザプログラムの実行環境、開発環境を提供
- 実コンピュータの台数を絞れる
  - **管理コストを削減**（例 情報科学科の演習室）
  - 地球温暖化や災害時の**電力削減にも有効**（**グリーンIT**）
- 仮想マシンを複数台動作させて別々のプログラムを実行し、冗長性を持たせて**耐故障性を向上**
- 昔のOSを稼働させ、ソフトウェアの寿命を延ばす
  - 保守やセキュリティ対策が切れたOSで動作していた、**レガシーソフトウェア**（古いソフトウェア）の寿命の延長

16

OS機能の隠ぺいの実現

仮想マシンの構成方式としては、次の2種類がある（図1.16）

(A)ホストOSあり・複数分散仮想マシン

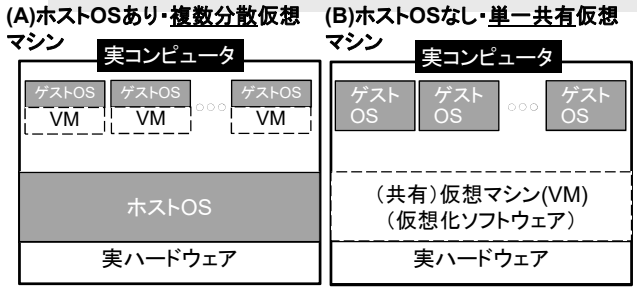
- 異種複数の仮想マシンをホストOSとは独立した機能として構成する
- 既存のコンピュータシステム上で、仮想化ソフトウェアを追加インストールするだけで実現可能
- ホストOSと異種複数の仮想マシンが共存する形となり、(B)に比べるとコストが大きい

(B)ホストOSなし・単一共通仮想マシン

- 単一の仮想化ソフトウェアをホストOSの代わりにあらかじめ組み込んで構成する
- そのコンピュータシステムに本来あるOSを置き換える必要あり
- 仮想化ソフトウェアを実ハードウェア上で稼働させて共有すればよく、ホストOSがないので、(A)に比べるとコストが小さい

17

仮想マシンの構成方式（図1.16）



なぜ(A)ではVMが複数個あるのに、(B)ではVMが一つなのか?

18

なぜ(A)ではVMが複数個あるのに、(B)ではVMが一つなのか?

- (A)の方式
  - ホストOSが実ハードウェアを隠ぺいし、プロセス管理を行う
  - VMは**ホストOS上でプロセスとして実行される**ため、**複数個を実行でき**、ホストOSを隠ぺいする
  - ゲストOSは、VMにより提供される仮想化されたハードウェアの基で、**VMの数だけ動作**する
- (B)の方式
  - **ホストOSがない**ので、VMはプロセスとして実行できず、**1個しか実行できない**
  - ゲストOSは、VMにより提供される仮想化されたハードウェアの基で動作する
  - VMがマルチタスキングのプロセス管理機能を持たない限り、ゲストOSは**1個しか動作できない**（ただし、ユーザ等の指示により、別のゲストOSに切り替えて動作させることは可能）

19

OS機能のファームウェア化

- OS機能の一部をソフトウェアではなく、**ファームウェアによって実現**し、OS機能を改善あるいは強化すること
- 一部の機能で特別な条件が要求される場合（リアルタイムシステムなど）や、利用できるハードウェア装置に制約が強いとき（組み込みシステムなど）に有効
- ファームウェア化の例
  - 仮想マシンや言語処理プログラムのオーバヘッドの発生防止
  - ハードウェア機能のテストや保守での柔軟性確保
  - プロセス管理でのスケジューリングアルゴリズムや、仮想メモリでのページ置換アルゴリズム、通信プロトコルなどの実現
  - 多種多様な割り込み要因に対する柔軟かつ高速な対処

20

1.4 割り込み　－OSとの通信－

1.4.1 割り込みとは？

- **割り込みは「OSとの通信」**
- ユーザプログラムおよびハードウェア装置のそれぞれからOSへの通信を統一的に行う仕組み

[1] ユーザプログラムとOSとハードウェア装置との通信

- ユーザプログラム→OSの通信
- ハードウェア装置→OSの通信

21

マシン命令の実行と割り込み

- 割り込み機構および機能
  - マシン命令の実行フローを、不測の事態の発生時に強制的かつ動的に変更する手段
  - 割り込みの原因となる不測の事態を「**割り込み要因**」という
  - 割り込みの発生：割り込み要因が生じたことで、実行中のプログラムを一時中断して、割り込み機構の動作を開始（図1.17）
  - 割り込み処理：割り込み要因ごとに決められた処理

22

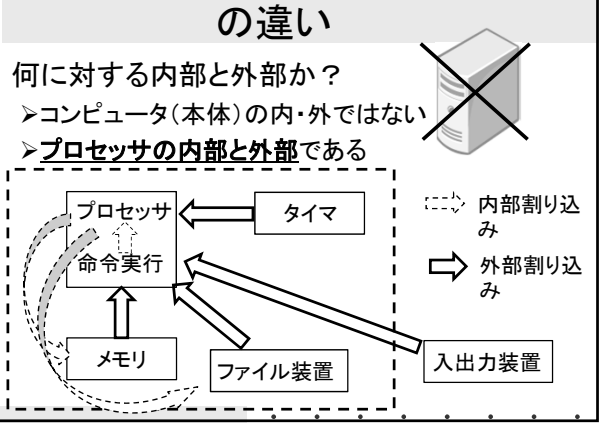
割り込み要因の分類

- **内部割り込み**（ソフトウェア割り込み）
  - 暗黙的（例：命令実行例外）
    - 不測の事態に伴って発生する割り込みを、**例外(exception)**と呼ぶ（0除算、不正なアドレス参照）
  - 明示的（例：**システムコール(SVC)**, ブレークポイント）
- **外部割り込み**（ハードウェア割り込み）
  - ハードウェア障害
  - リセット
  - タイマ割り込み
  - 入出力割り込み

23

「内部割り込み」と「外部割込み」の違い

- 何に対する内部と外部か？
  - コンピュータ（本体）の内・外ではない
  - **プロセッサの内部と外部**である



24

マシン命令の実行との関係

- **内部割り込み**
  - マシン命令の実行に**合わせて**発生（実行に**「同期」して**発生）
- **外部割り込み**
  - マシン命令の実行とは**独立に**発生（実行に対して**「非同期」に**発生）

外部割り込み

任意の時点で発生

内部割り込み

```
LD   GR0, DATA1  アドレスが不正なら割り込み
DIV  GR0, DATA2  除数が0なら割り込み
ST   GR0, DATA3
SVC ADR1     システムコール（割り込みの一種）
```