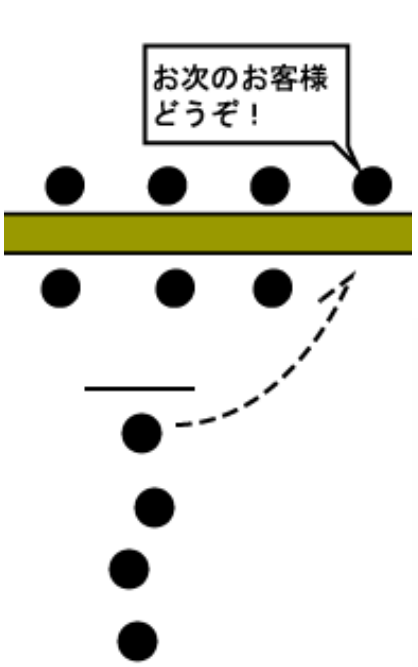
オペレーティングシステム

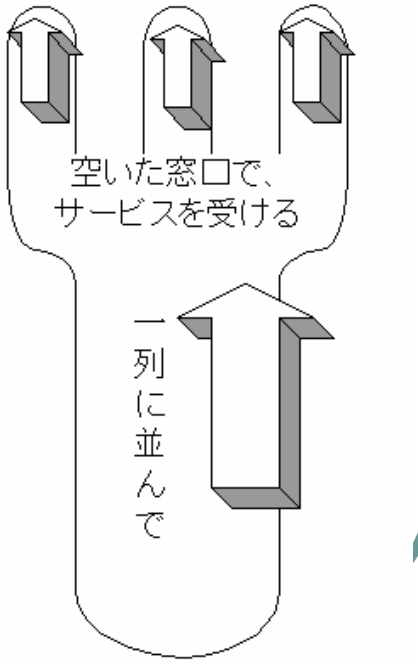
第9回(2009.06.11)

並列処理とプロセス間通信

「フォーク並び」

- ５年位前に、誰かが提唱したフォーク並び
  - 複数のサービス窓口
  - 個人商店とスーパーマーケットの違い！？
    - レジが一つしかないのが「個人商店」
    - 複数のレジがあるのが、スーパーマーケット？
- 「公平」な待ち時間
  - を目指したモデル

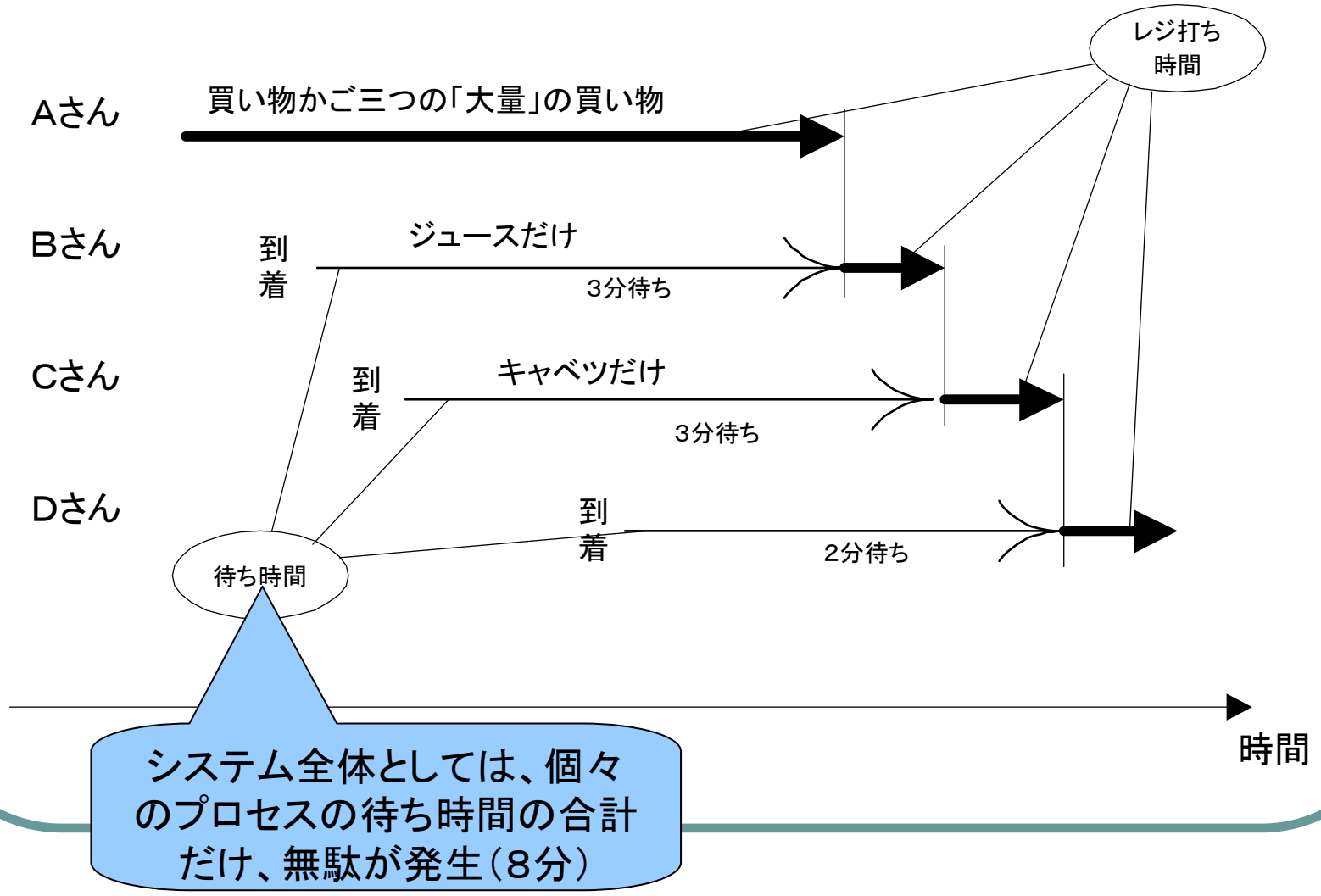




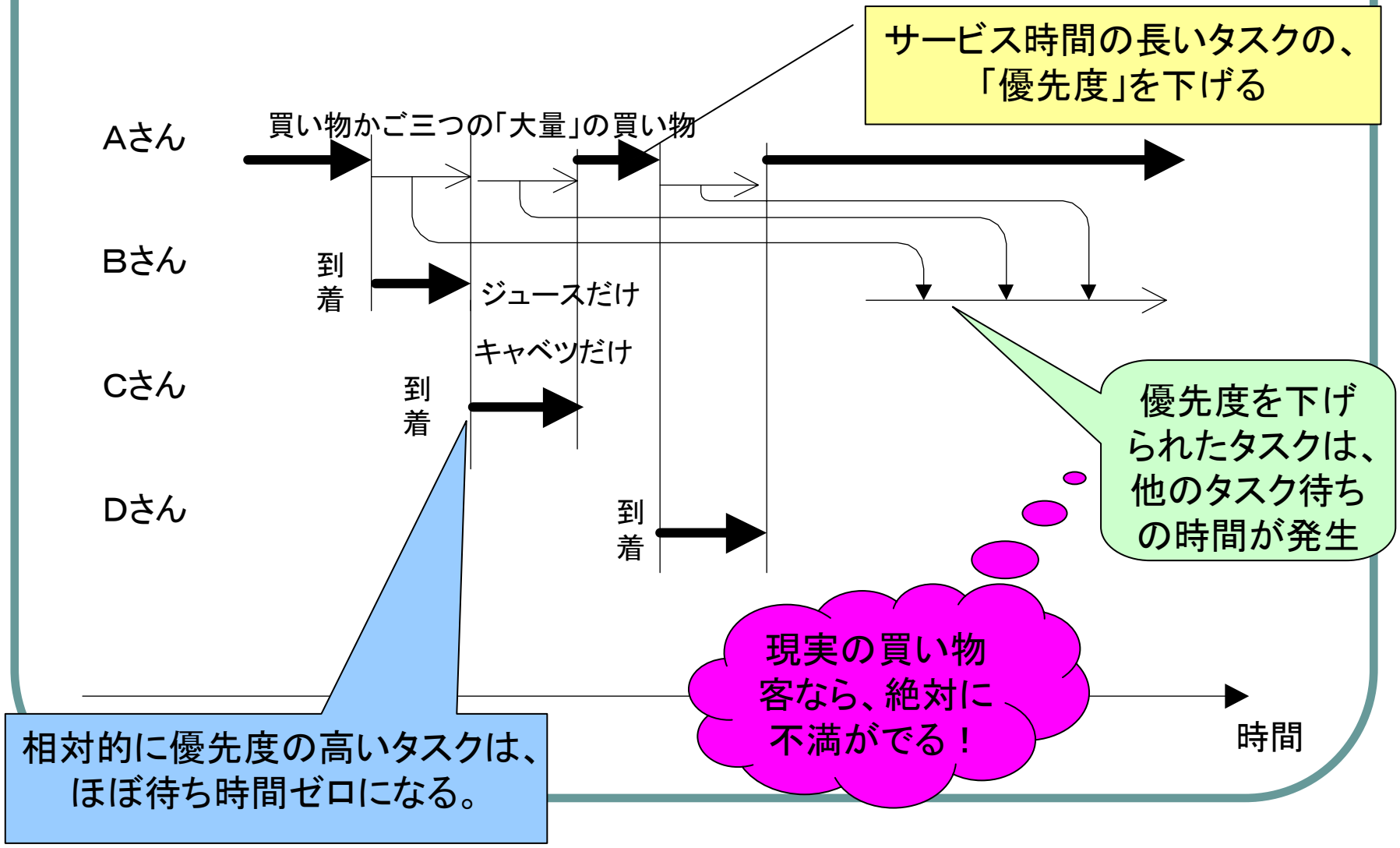
最初は「割込み処理」

- 「資源」を効率的に使う
  - ここでは「時間」を効率的に使う
- 割込み処理
  - ハードウェアを待たない
  - 割込み処理
    - ⇒　プロセスの切り替えが実現
  - 「プロセス」という概念が出来た。
  - 複数のプロセスに「優先度」という概念が発生

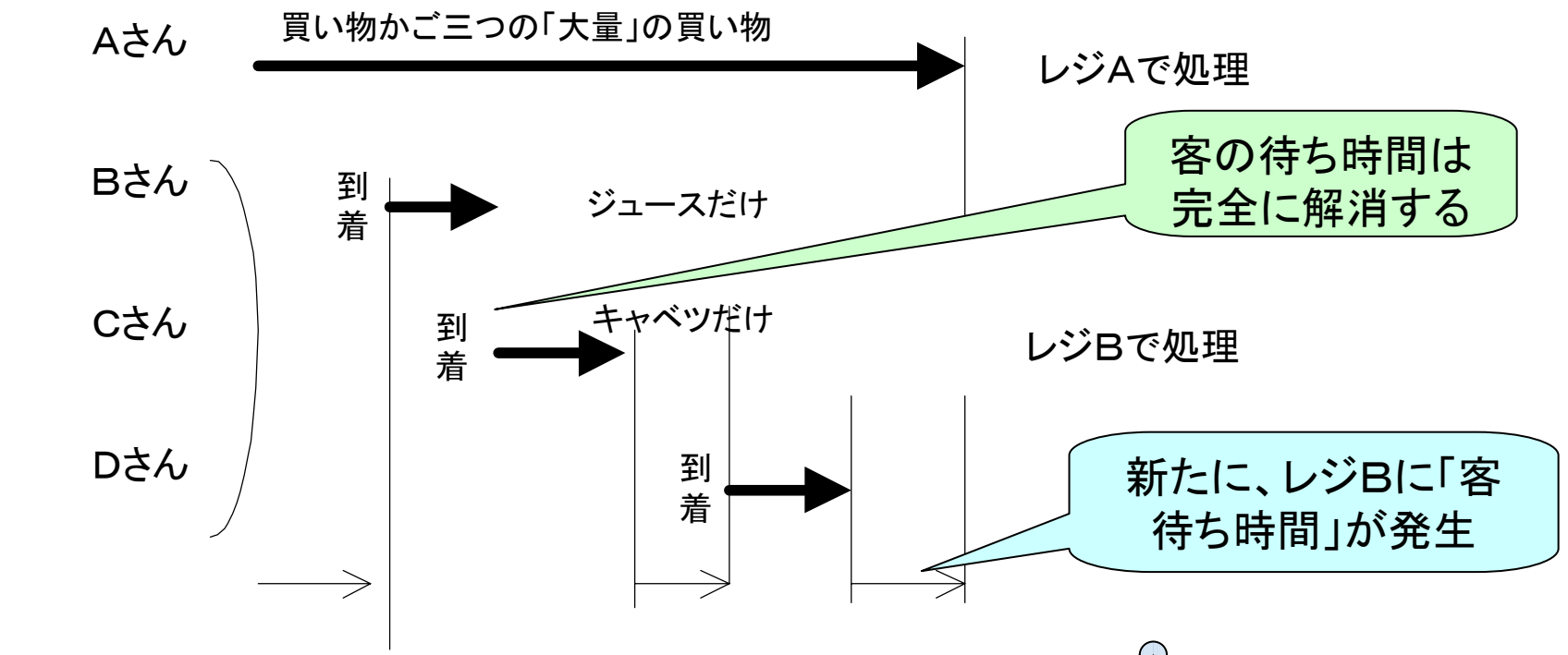
たった一つのレジで、割込みなし



たった一つのレジで、「割込み」あり



複数のレジがあったら・・・



レジの数を増やして、誘導がうまくいけば、
客の待ち時間という資源は浪費されなくなるが、
新たに、店の「客待ち」という資源の浪費が発生する

多重プロセス

- 「客」はプロセスで、「レジ」はCPU時間
- 教科書P99
- 一つのCPUで複数のプロセスを実行できる
  - シングルタスクではなく、できるだけ多くの客の「不満」を同時に処理するモデル
- 複数のCPUで複数のプロセスを実行
  - 次回以降に「分散システム」のモデルで扱う

多重プロセスでの必須要件

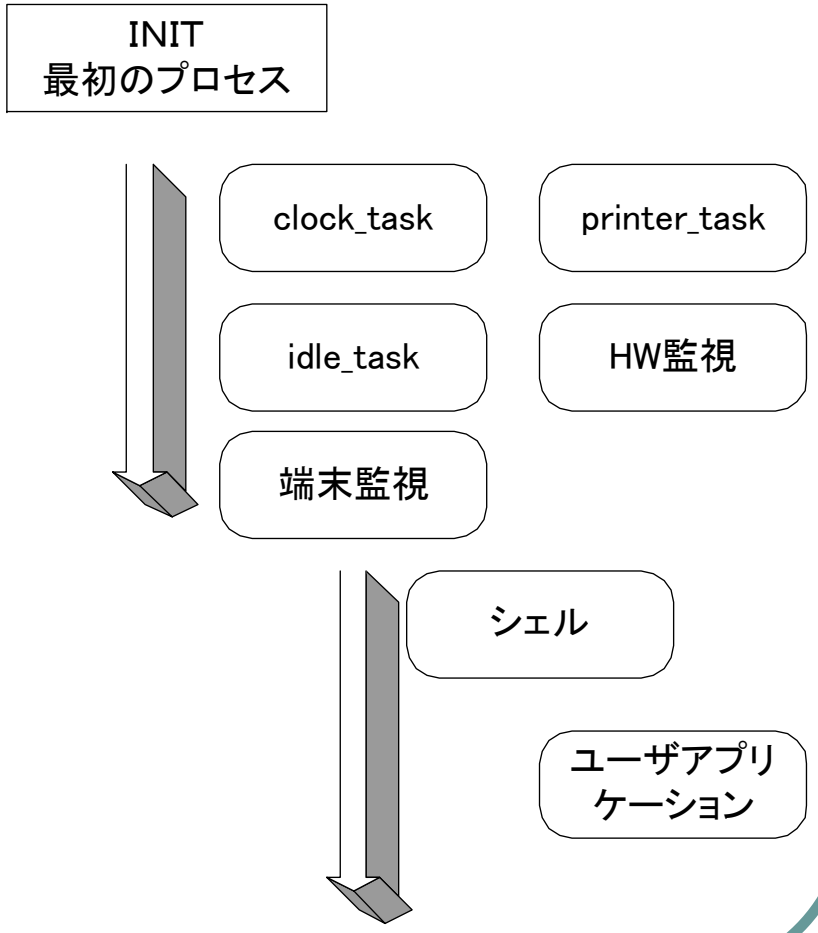
- 排他制御機能
  - （第2回の授業で扱った）
  - 資源へのアクセスの交通整理
- 事象の連絡機能
  - Eventの発生を互いに通知しあう
    - 割り込みによる通知
    - ポーリングや、セマフォの利用
- プロセス間の通信機能
  - それぞれが、お互いの状態を知る必要がある。

プロセスの生成と消滅

- Fork
- 実は、「プロセスの生成」もforkと言う
  - 教科書P102
  - fork() 関数
    - 自分自身のコピーを作成する。
    - fork()関数の戻り値以外は、完全にコピーされた内容を持つ。
      - 記憶領域などは、新たに割り付けられる。
- 分岐- 親プロセスと子プロセス
  - 親と子の対話 = プロセス間通信の原点

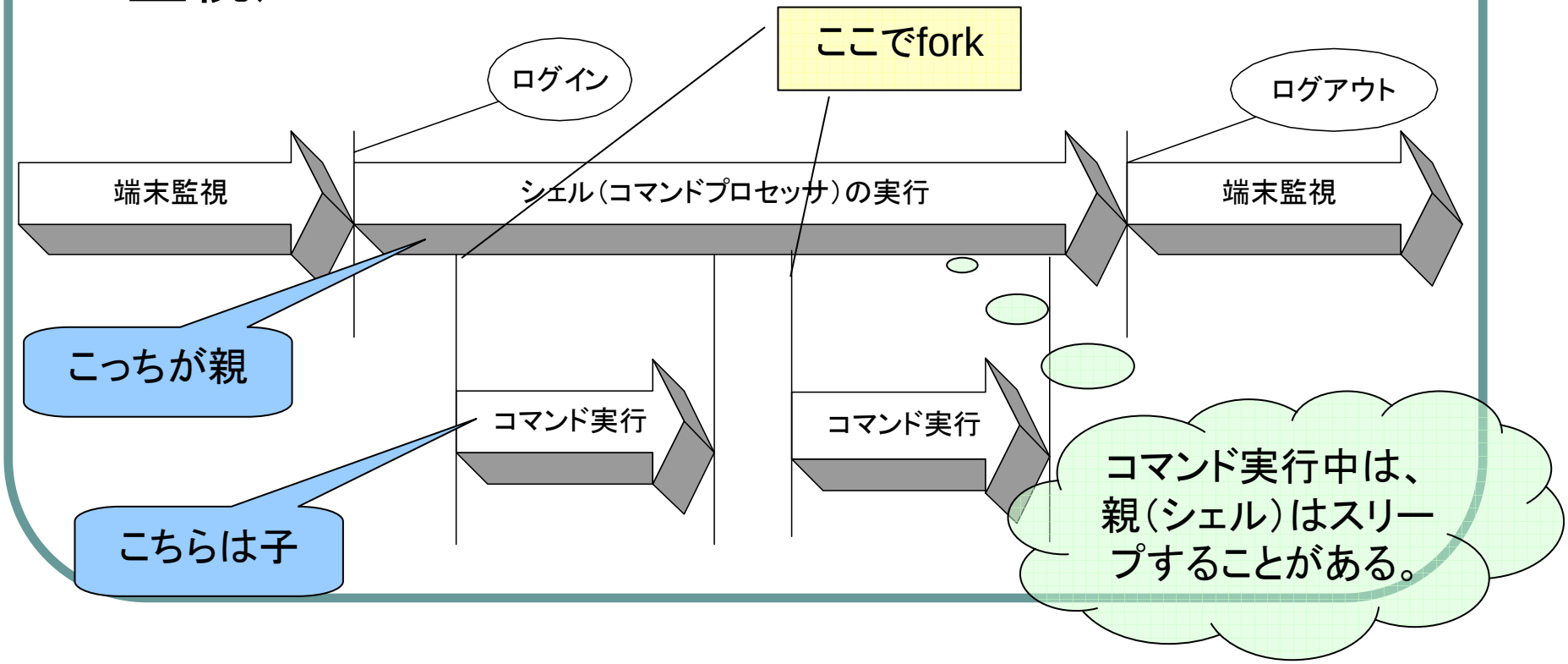
UNIXのプロセス生成と消滅

- OS（親）が起動すると、システムタスク群を起動する。（子）
- この中の端末監視モニタが、シェルを起動する。（生まれ代わり）
  - 起動されたシェルが、ユーザアプリケーション（孫）を起動する。
- Exitすると、そのプロセス（孫たち）は消滅する。



OSのコマンド処理とプロセス

- 端末監視プログラムがそのままシェルへと移行し、ログアウトと同時に「端末監視」に戻る。（ログイン監視）



なぜ、多重プロセス？

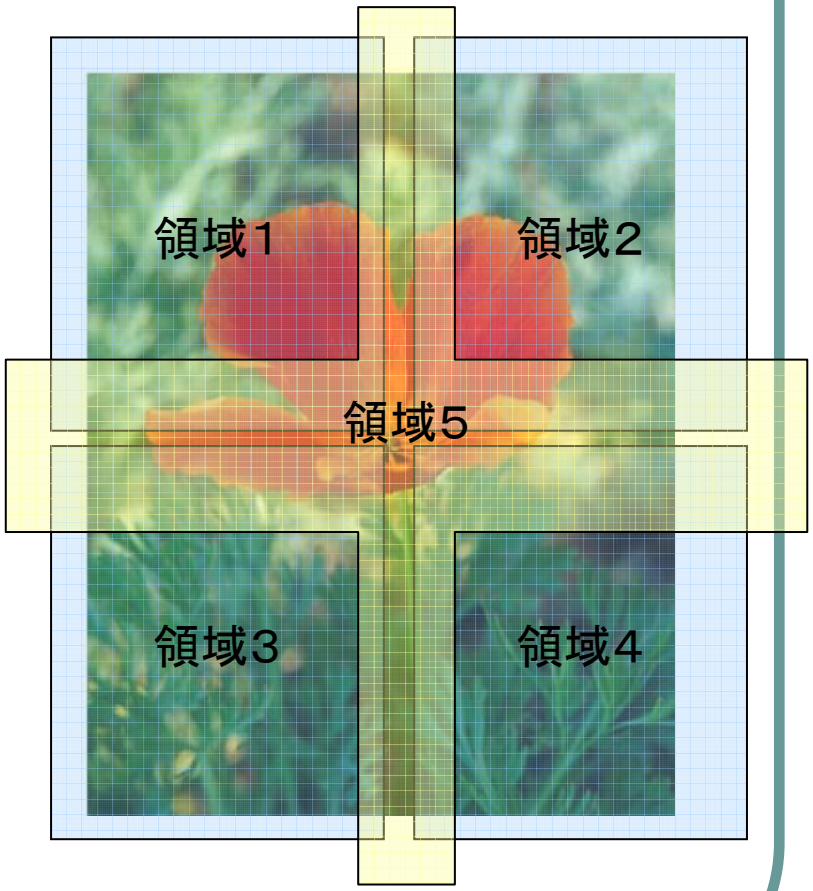
- CPUの空き時間が、無駄に多いため！
  - 一つ一つの「プロセス」のライフサイクルで、「入力」と「出力」に占めるウェイトが、大きい
    - アプリケーションの種類によって異なる
      - 数値計算型
      - データ処理型
      - 制御型
    - 入出力の時間を、他のプロセスの計算に使用させる。

マトリックスの演算

- 画像処理のプログラム

```
for( i=0; i<N; i++ )
    for( j=0; i<N; j++ )
```

  - 全体をループで回し、画素ごとに演算する。
- もし、CPUコアが複数あったら・・・
  - 画像（行列）を分割して、各ゾーンごとに、同時並行的に処理を行うことができる。



並列処理⇒分散処理

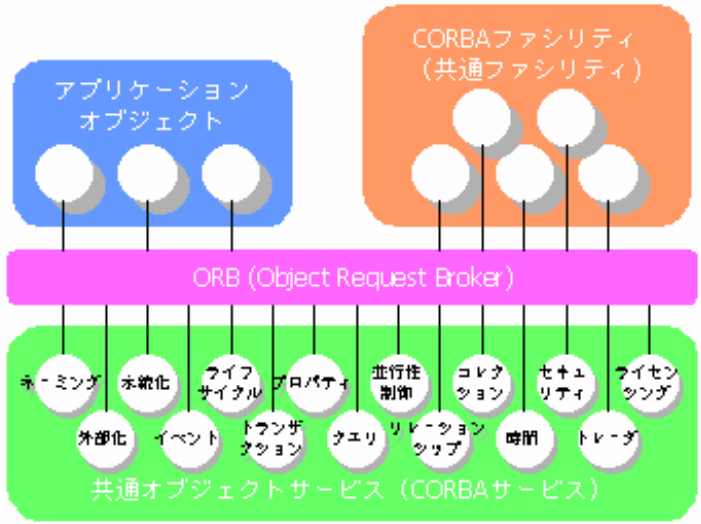
- プロセスの分割
  - 一つのCPUに、複数のコアがある場合
    - Dual CORE
    - CPU内部の配線で、通信
  - 一つの筐体内部に複数のCPUがある場合
    - マルチCPU
    - バスを経由して通信
  - ネットワークを経由したコンピュータ連携
    - ネットワーク経由で通信、分散並列処理

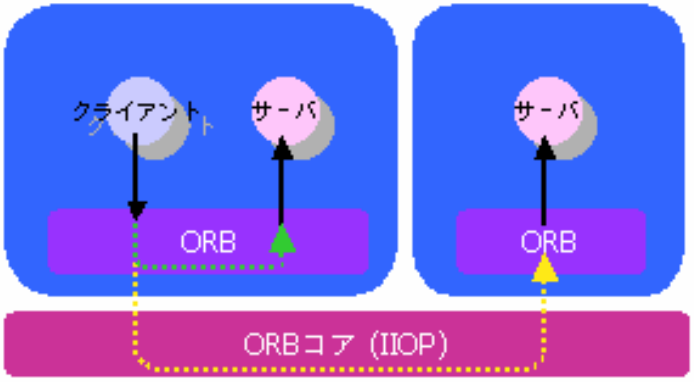
分散オブジェクト処理

- 「異なる種類」の演算を、機能を特化したそれぞれの「サーバ」に分割して実行する。
- 「同種で大量」の演算を、同じ機能を持った複数のサーバに分割して実行する。
- 処理を分散することで、効率を上げる。

CORBA

- http://e-words.jp/w/CORBA.html より
- 読み方：コルバ
- Common Object Request Broker Architecture
- 分野：プログラミング > オブジェクト指向 > CORBA
- 　OMGが定めた分散オブジェクト技術の仕様。異機種分散環境上のオブジェクト(プログラム部品)間でメッセージを交換するためのソフトウェア(ORBと呼ばれる)の仕様を定めている。具体的には、ORBの基本構造や、プログラミング言語からORBを利用する際の手順、異なるORB間で相互にメッセージを交換する際の規定などを定めている。
- 教科書：P159





再び、UNIX系OSに戻って・・・

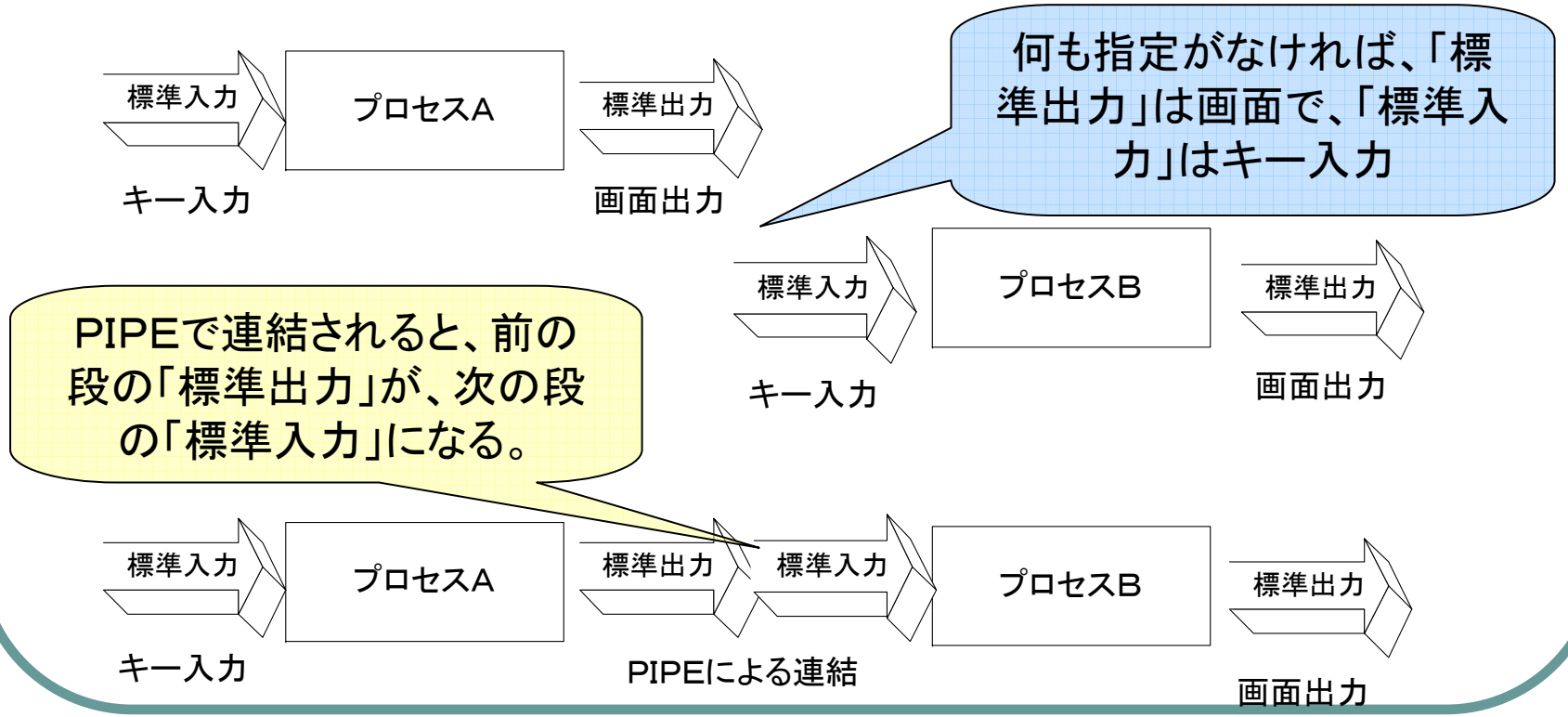
- プロセス間通信の、技法
  - 分岐したプロセス間では、PIPEと呼ばれる通信手段を持つ　⇒　APIとしてのPIPE
    - IPC：Inter Process Communication
    - 教科書：P113
  - 自分のパソコン上で、マルチな分散オブジェクト環境をシミュレーションする方法
    - マルチタスクでプロセス間通信の環境を作って、分散型オペレーティングシステムに見なす。
    - 待機時間、不接続時間などのモデルを作成し、シミュレーションする！

PIPE（パイプ）

- APIとしてのPIPEと、OSの提供する機能のPIPE
  - 分岐したプロセス間では、PIPEと呼ばれる通信手段を持つ　⇒　APIとしてのPIPE
    - IPC：Inter Process Communication
    - 教科書：P113
  - シェルスクリプトの記述で、標準入出力を連結させるPIPE機能　⇒　OSの提供する機能
    - “|”（縦棒）で、プロセスを連続的に記述すると、「直列」に実行されるプロセスの間で、データを受け渡しできる。

標準入出力を切り替えるパイプ

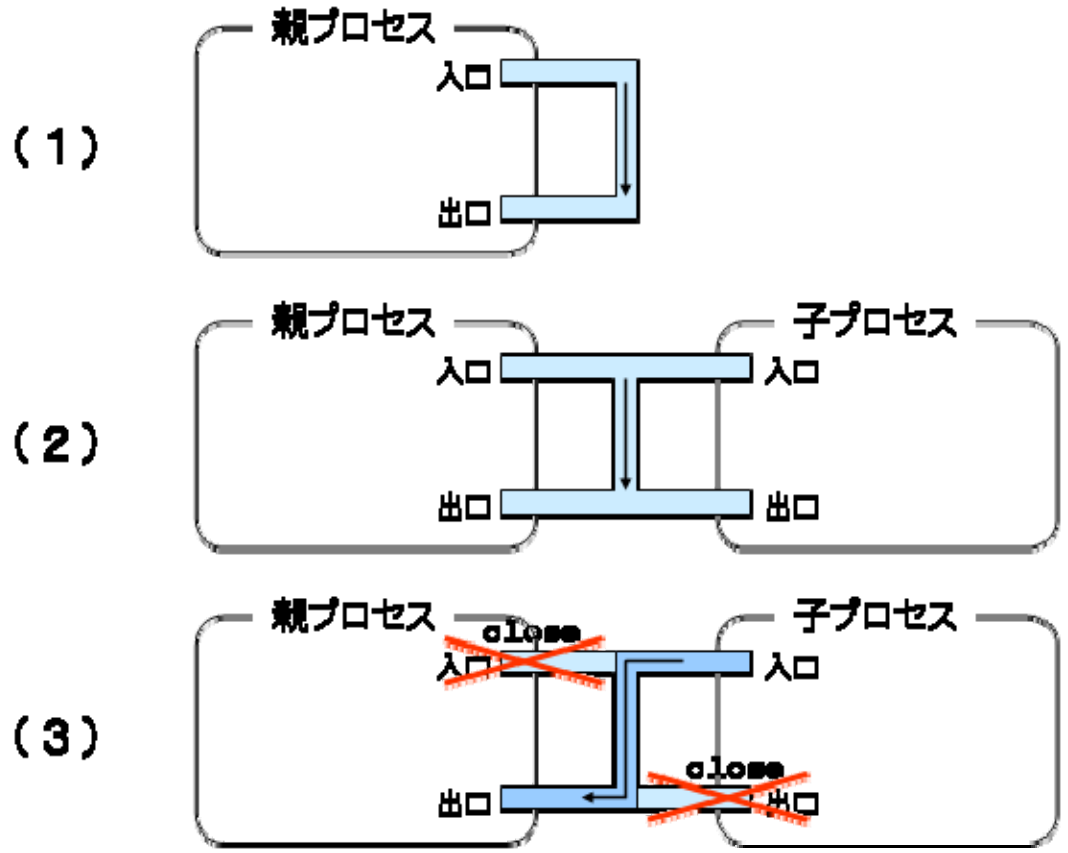
- プログラムA　|　プログラムB
- 「縦棒」で二つのプログラムを「連結」する



プロセス間データ通信のパイプ

- 教科書P114
- シェルが「標準入出力」を連結しているのと同様に、「親」「子」の間で、プロセス間の通信ができる。
- pipe()関数コールを行う。

パイプによる「親子」の連結



（引用元：http://www.coins.tsukuba.ac.jp/~syspro/2005/No3.html） 2005年

まとめ

- 時間などのシステム資源を有効に活用するため、オペレーティングシステムはマルチプロセスをサポートしている。
- マルチプロセス（多重プロセス）では、排他制御や、Eventの通知機能、プロセス間の通信が必要となる。
- UNIX系のOSでは、プロセスのforkによって親子関係のあるプロセスが生成され、exitで消滅する。
- CPUの演算機能（CORE/ALUなど）が複数になった場合、並列処理から分散処理が可能となる。
- CORBAは、分散処理を実現する技術（規格）の一つである。
- UNIX系OSでは、PIPEによって、プロセス間の通信が実現されている。