オペレーティングシステム

2022/11/29

海谷 治彦

目次

- 安全性
- 生存性
- 干渉
- デッドロック

互いに邪魔しない点からの性質

- **安全性**(safety): 好ましくない状態にならない
  - 干渉(interfere)がおきない.
  - デッドロック(dead lock)が無い.
- **生存性**(liveness): やろうとしたことは，いつかは処理される.
  - 永遠に待たされることは無い
    - 例えば高層ビルに多数のエレベータがあるが，いつまでたっても来ないとかいうことが無い.

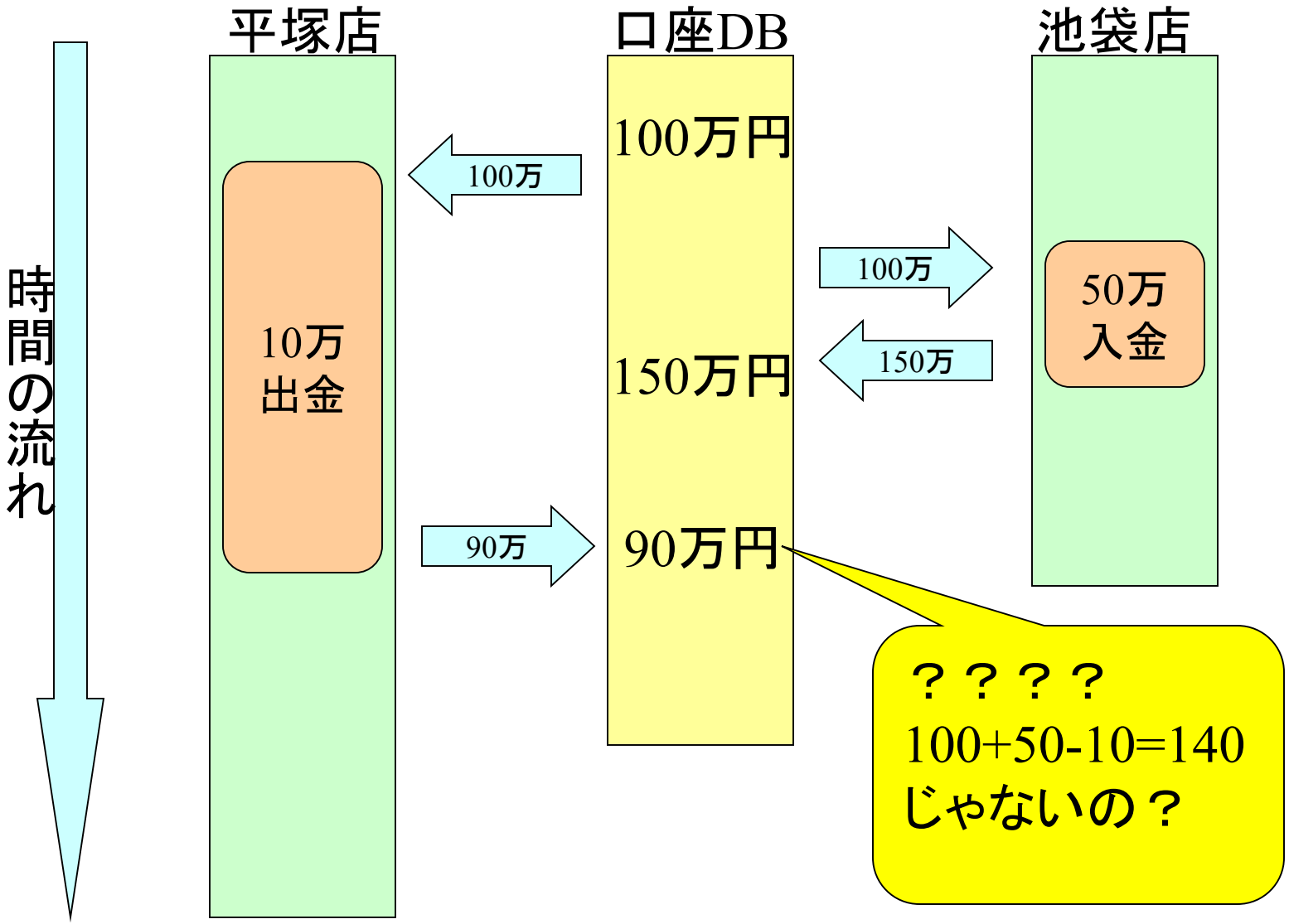
干渉とデットロック

- 干渉(interfere) 並列処理のミスでデータ等の一貫性が失われること.
  - 後述の排他制御である程度回避可能.
- デッドロック(dead lock)
  - 複数の資源を同時に必要とする複数のプロセス(スレッド)が，資源の一部をそれぞれに確保し，残りの資源が空くまで，それぞれが永久に待ってしまうこと.
  - 「哲学者の例題」参照.

排他制御による干渉の予防

- (プロセスだけでなく)並列処理一般に重視しなければいけない問題.
- 要は共有資源を同時に処理しちゃいけない.
- 一般にはロックという方法で排他制御するのが一般的.

有名な干渉の例: アホな銀行



干渉発生のTIPS

- 1つの連続した処理(コレをクリティカルセクションと呼ぶ)が終わる前に，古いデータを読んで，別処理を行っているのがマズい.
- 逐次処理(not並列処理)の場合は問題なし.
- 複数プロセス等による並列処理の場合，データ(変数，ファイル，レコード等)を特定のプロセスに一定期間のみ占有させる必要あり.

干渉防止に関するOSの機能

- OSはプロセスが排他的に動作するための機能を提供している.
  - 割り込みを禁止してクリティカルセクションが細切れにならないようにする等.
- それによって，前述の干渉は防止されている.
- OS授業の定番として，後述の，Lock, セマフォ，モニタと呼ばれる機構を紹介する.
  - これらの機構は実OSや言語に実装されている.

排他制御: 干渉回避の機構

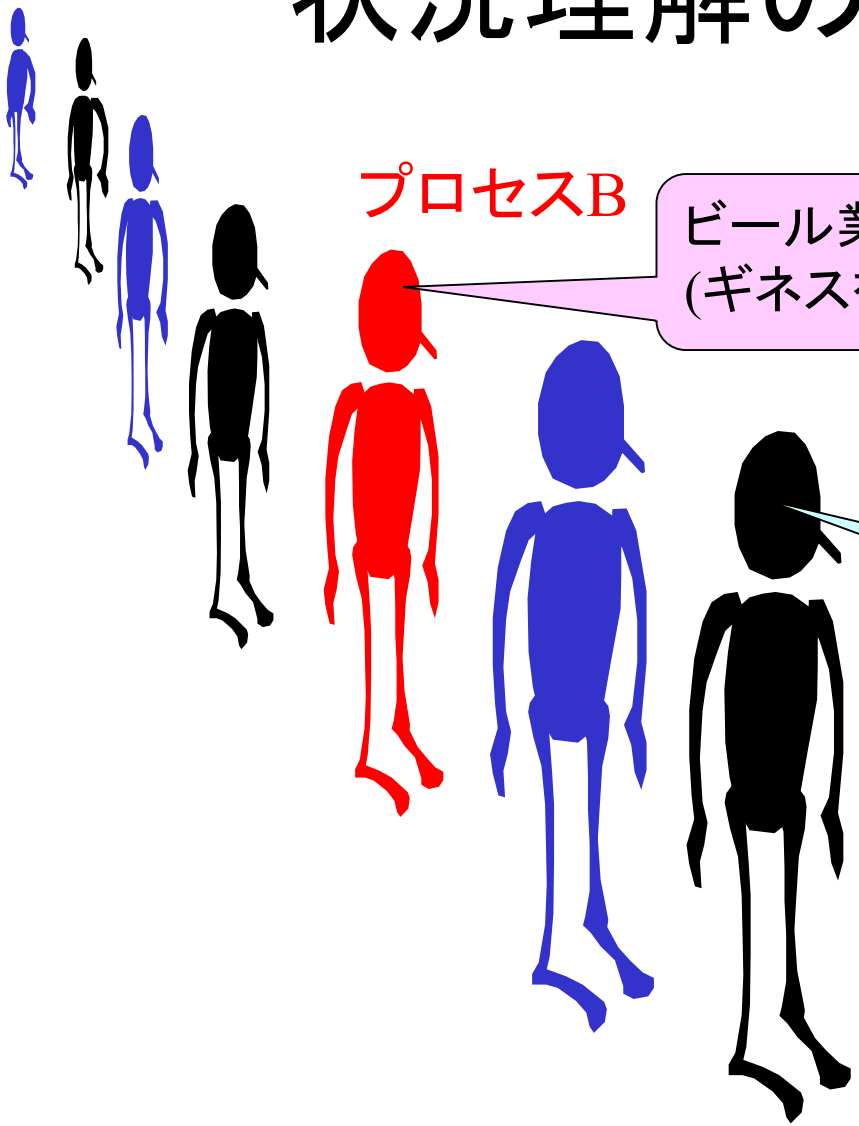
- Lock/unlock
  - 資源(主にデータ等)を特定のプロセスに占有させるためのOSの機構.
  - あるプロセスがlock中は他のプロセスは当該資源にアクセスできない.
- セマフォ (semaphre)
  - やはり資源を特定プロセスに占有させる(利用する)ためのOSの機構.
  - 単純にlock/unlockではなく, 何人利用待ちかを数値としてOSが覚えておく.
  - 加えて, 待ってるプロセスの待ち行列をOSが管理する.
  - 利用中のプロセスの利用が終わったら, (1)待ち人数を減らす, (2)待ってるプロセスに順番が来たことをOSが通知する.

プロセス間の同期

- あるプロセスAがプロセスBの結果に依存して処理を進める場合がある.
- この場合，Aが結果を出したことを，Bがなんらかの方法で知る必要がある.
- この手法として，
  - ポーリング
  - モニタ

  が常套手段としてある.

状況理解のための例題



プロセスB

ビール業者でーす.
(ギネスを補給できます)

プロセスA

ギネスビール
が飲みたい！

ギネス品切れ

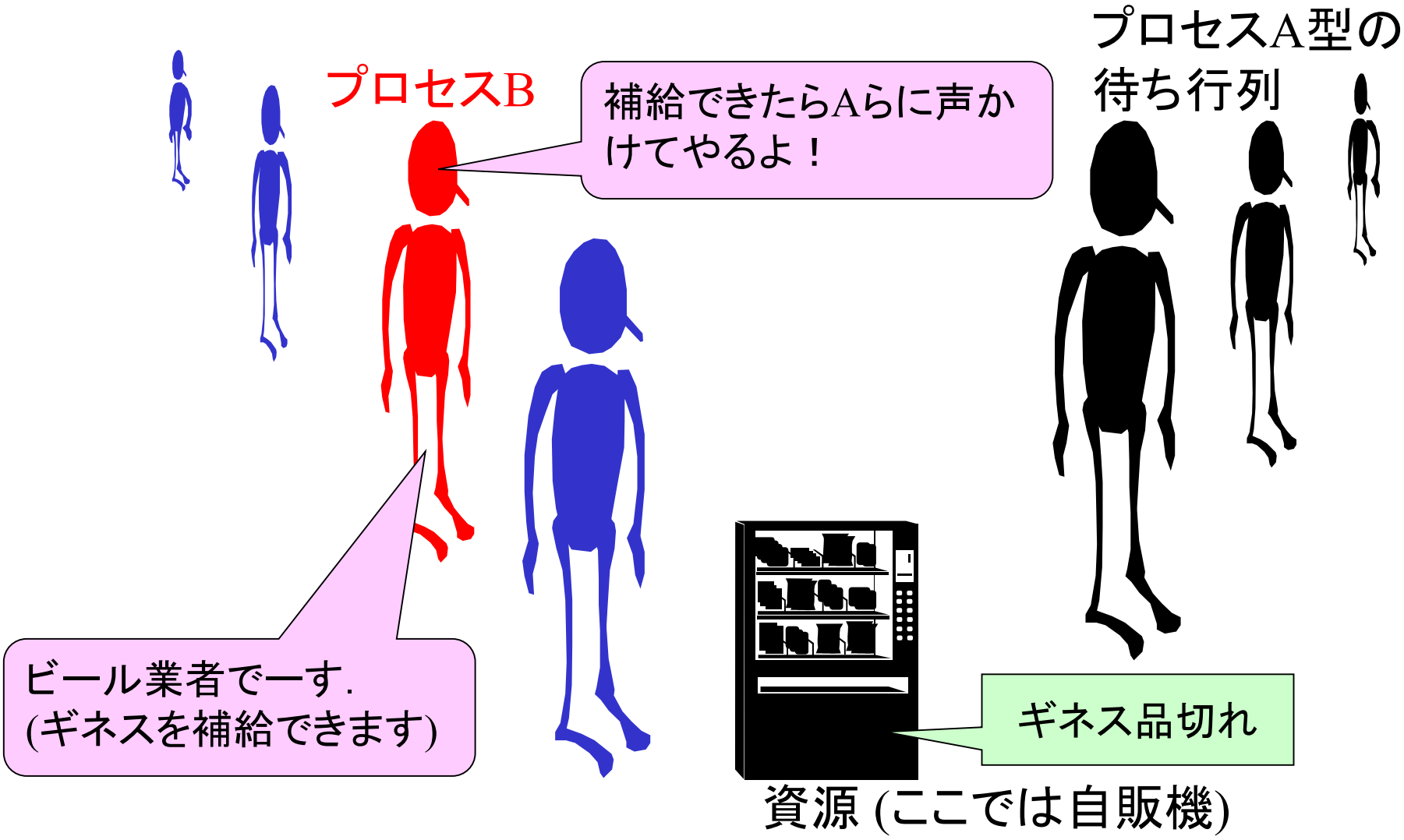
資源 (ここでは自販機)

悪い対処

- プロセスA(酒飲みのおっさん，概ねアイリッシュ)が自販機からどかない．(図8.7a)
  - 他の製品を買う人を阻害．
  - ビール業者の補充作業も阻害．
- プロセスAが数分おきに自販機の様子を見に来る．(図8.7b ポーリング)
  - 他の製品を買う人を邪魔．
  - ビール業者の補充作業にも邪魔．

- 補充されたら来てくれよ，おっさん！

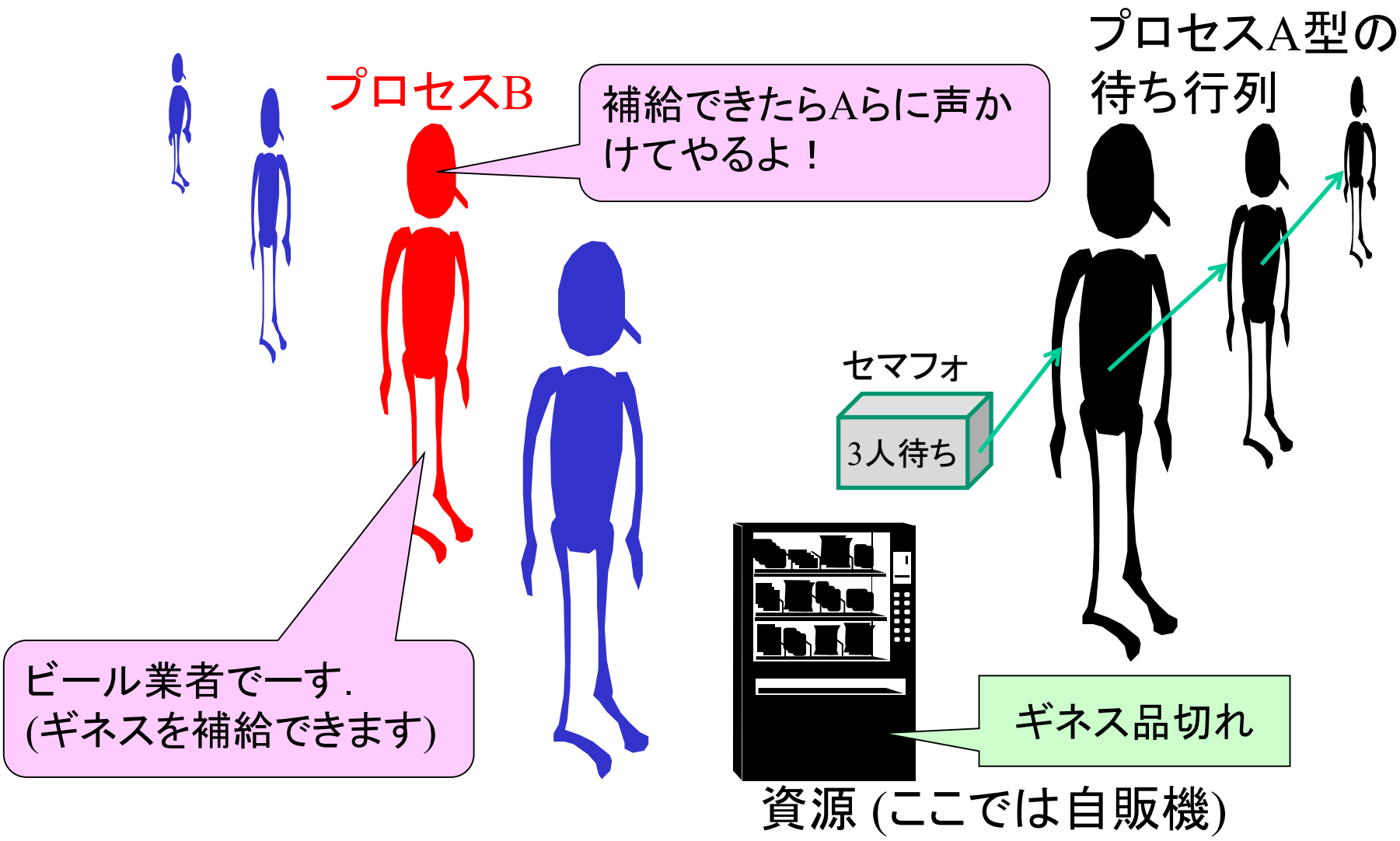
解決法: ギネス待ちの列を作る



OSにおける前述の解決案

- セマフォによる
  - 前述のようにセマフォはunlock待ちプロセス数と，そのプロセスの列を管理している.
  - 加えて待ちプロセスの通知機能もある.
  - よって，コレを直接に用いる.
- モニタによる (後述)

セマフォ


プロセスA型の
待ち行列
プロセスB
補給できたらAらに声かけてやるよ！
セマフォ
3人待ち
ビール業者でーす.
(ギネスを補給できます)
ギネス品切れ
資源 (ここでは自販機)

モニタの機構

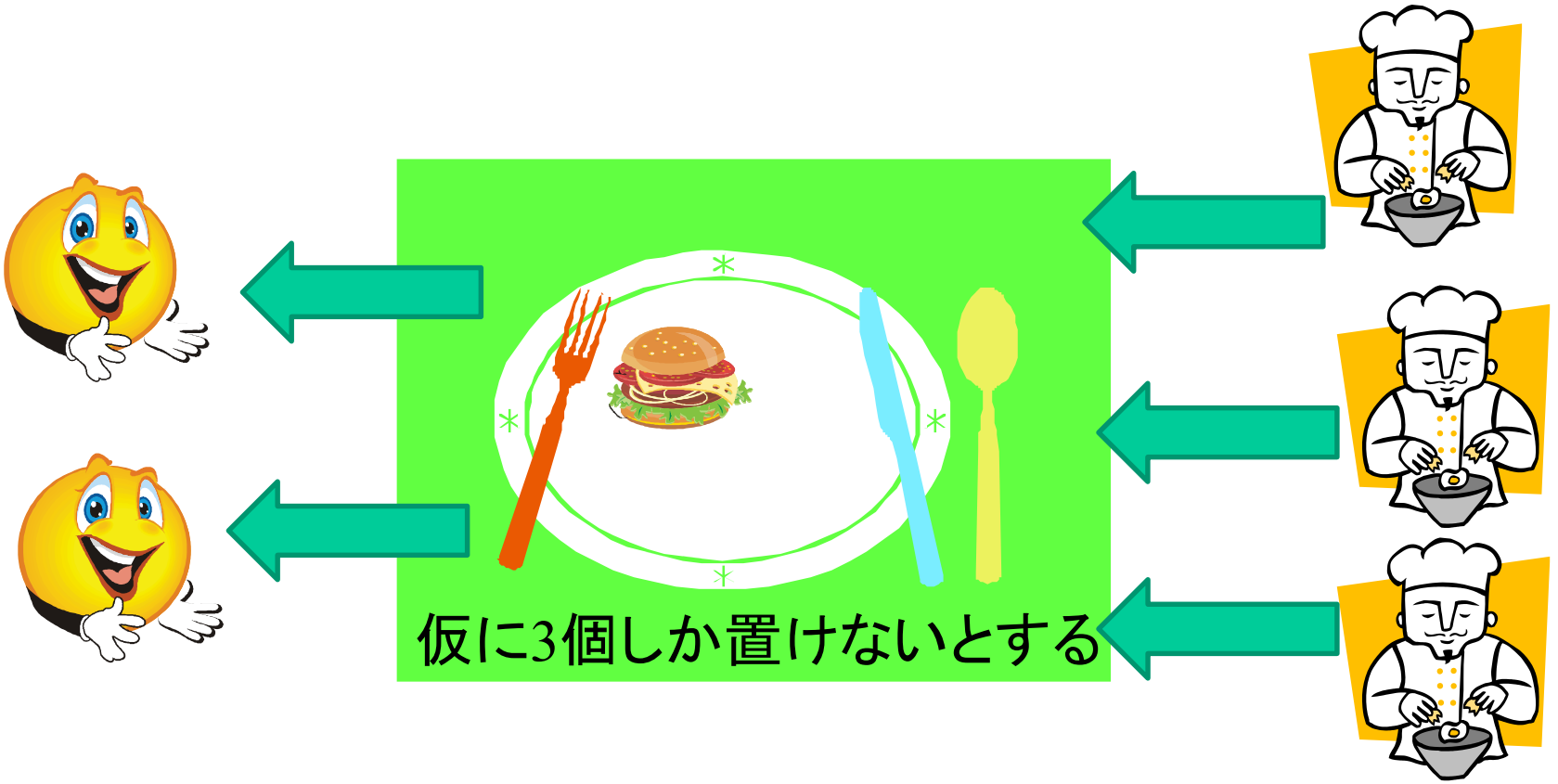
- ある資源を共有したいプロセスが呼び出せる以下の機能をOSが提供する.

- wait() これを呼び出したプロセスは待機状態となり待ち行列に入れられる.
- notify() 待機中のプロセスを1つ再開させる.
  - notifyAll() 待機中のプロセスを全て再開させる.

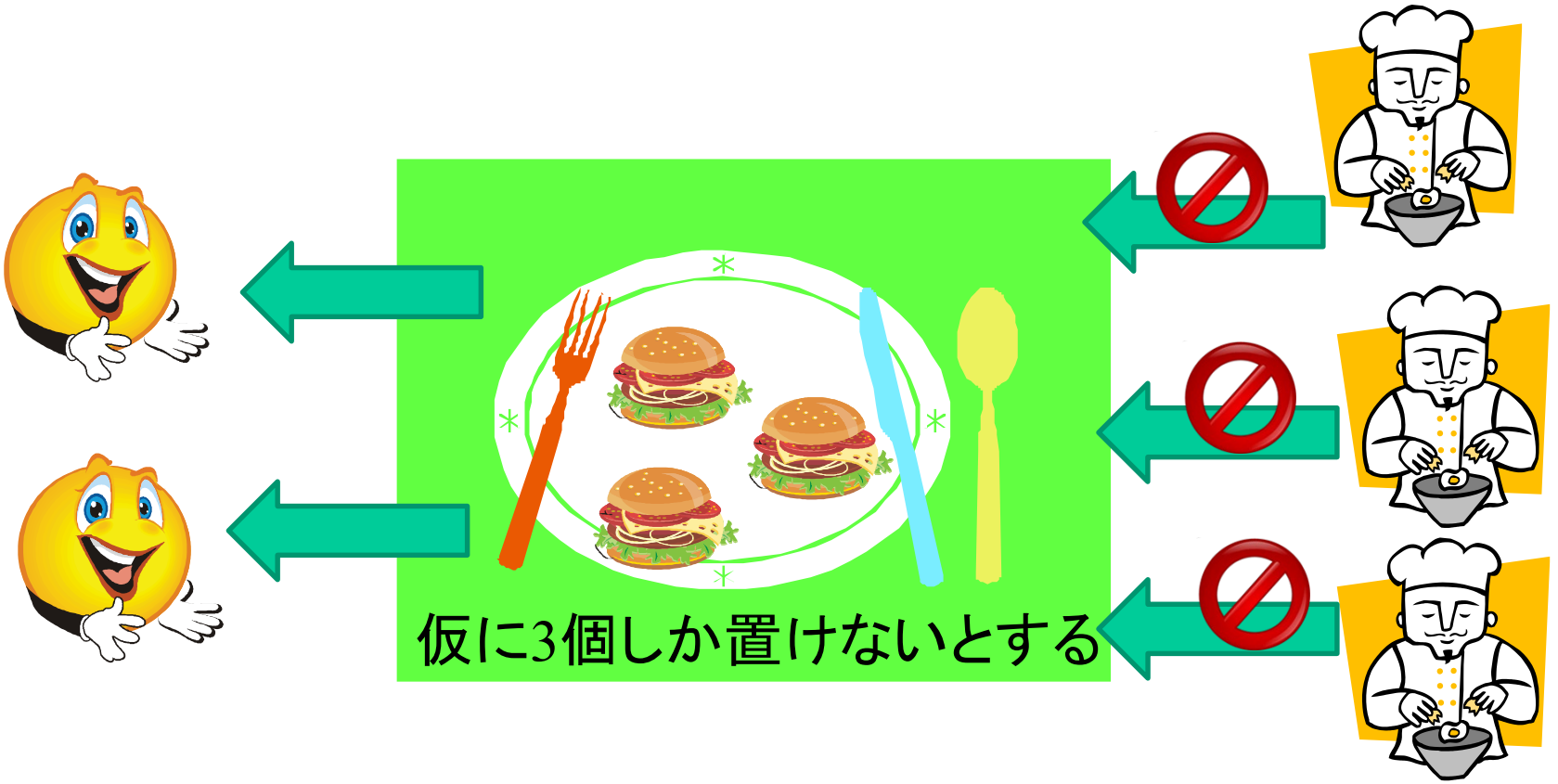
生産者･消費者問題

- モニタ関係の超有名な例題
- 倉庫を介して，生産者(Producer)と消費者(Customer)が商品のやりとりをする.
- 倉庫には商品をおける上限数がある.
- 消費者達は倉庫に商品が無ければ，当然買えない.
- 生産者達は倉庫が満杯の場合，生産を止めざるを得ない.

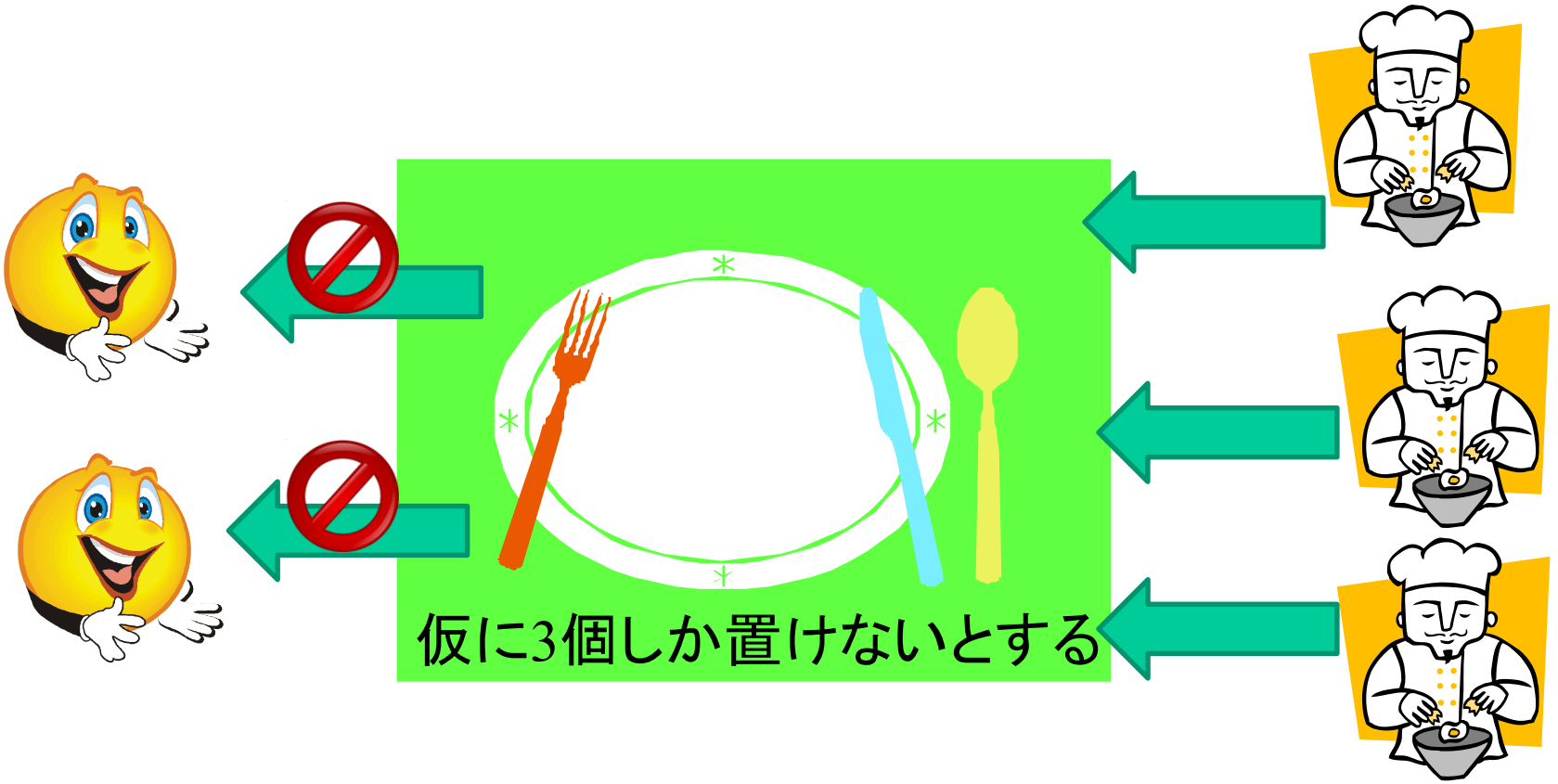
図による例: 両方動ける



図による例: 生産停止

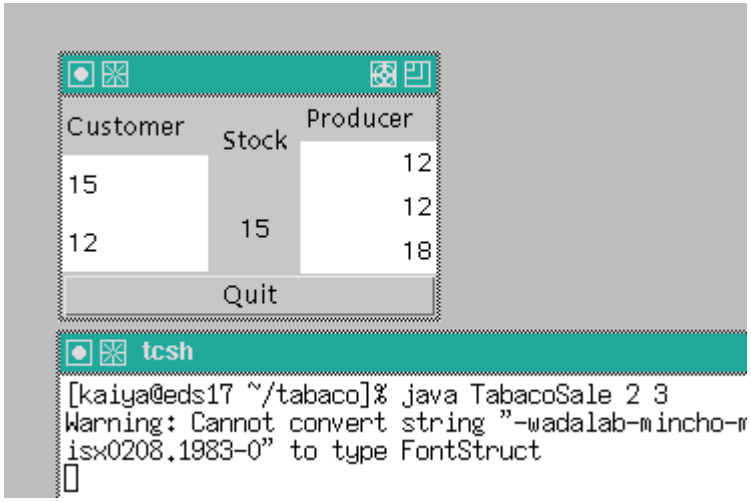


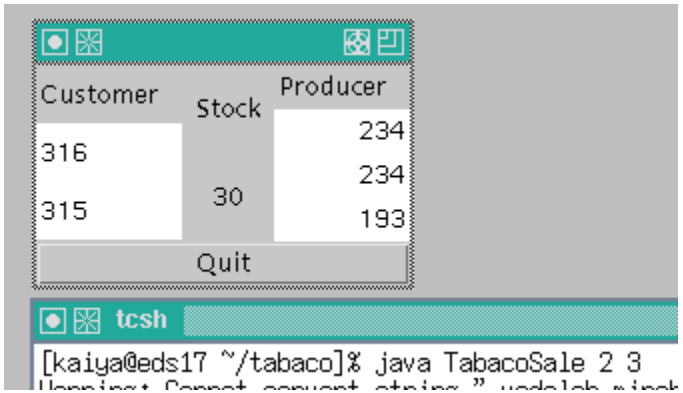
図による例: 消費停止



プログラムでのシミュレーション

所謂，生産者･消費者問題





生産過剰気味の構成にしてあるので，すぐにストック上限(本プログラムでは30に固定)あたりをうろうろする．
しかし，止まりはしない．

倉庫 (Stock.java)

```
class Stock extends IntLabel{
private static final int max=30;

// 中略

// ストックから買う
synchronized void buy(){
    while(empty()){
      try{ wait(); }
       catch(Exception e){}
     }
     super.dec(); // 在庫数を減
     notifyAll();
  }
```

```
// ストックに補給する
synchronized void supply(){
    while(full()){
      try{ wait(); }
       catch(Exception e){}
     }
     super.inc(); // 在庫数を増
     notifyAll();
  }

}
```

コード (Customer, Producer)

```
class Customer extends LabelUpdater{
  Customer(IntLabel l, Stock s){
    super(l,s);
  }

  public void run(){
    while(true){
      stock().buy();
      inc(); // 買った個数を記録
      sleeping(1000);
    }
  }
}
```

```
class Producer extends LabelUpdater{
  Producer(IntLabel l, Stock s){
    super(l, s);
  }

  public void run(){
    while(true){
      stock().supply();
      inc(); // 納品した個数を記録
      sleeping(1000);
    }
  }
}
```

双方Threadのサブ(サブ)クラス

デッドロックについて

- プロセスが同時に複数の資源を必要であるとする.
  - 例えば, スピーカーとタッチパネルの両方を必要とする楽器を模倣するアプリ等を想像してください.
- あるプロセスが一方の資源を, 別のプロセスが他方の資源を確保して, 他が空くのを待つとする.
  - プロセスAがスピーカーを確保し, プロセスBがタッチパネルを確保する.
  - A, Bともに残りが空くのを待つ.
- OSがなんらかの介入をしなければ, プロセス群は永遠に空くのを待ってしまう.

デッドロック回避戦略

- そもそも，同時に必要な資源の一部を自由に確保させるようなことをしない．
  - 例えば，A, B二個の資源が必要な場合，Aが空いてなければ，とりあえずBを押さえることはしない．
- OSがデッドロックになっていることを識別できれば，強制的に確保している資源を解放させる．

古典例: 哲学者の問題

- 数人の哲学者が円形テーブルを囲み食事をとろうとしている.
- それぞれの哲学者の間には1本のフォークがおかれている.
- 哲学者は，自分の左右におかれているフォークが2本そろわないと食事をとることができないことになっている.
- この状況で哲学者同士が言葉等で連絡しあわないとデッドロックが発生する.
  - 実際の人間なら言葉や目配せでけん制し合えるが，コンピュータ内のプロセスはそんなことできない！

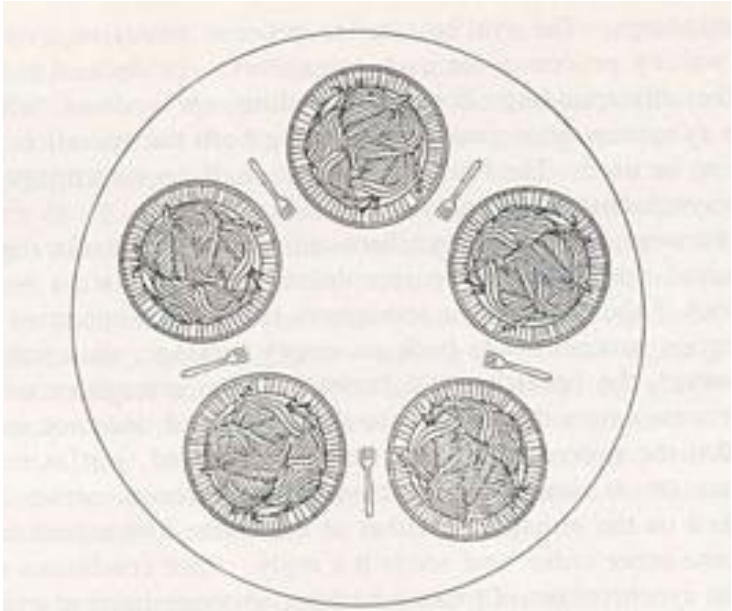


Fig. 2-18. Lunch time in the Philosophy Department.

他例題: 大工の問題

- それぞれN個のノミと木槌をN*2人の大工が共用していたとする.
- とにかく空いてる道具をとって，一方が空くのを待つと，デッドロックが起きる.
  - 以下は N=3 の例 (それぞれ3個の道具，6人の大工)

初期のアンケートについて

今回のアンケートも
よろしくご提出ください