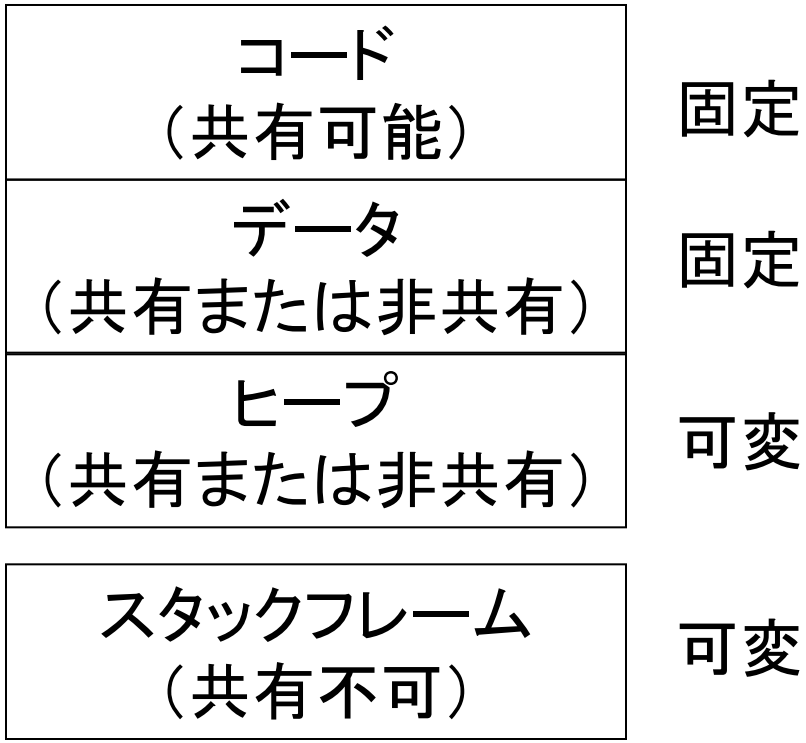
オペレーティングシステム論
(Theory of Operating Systems)
第三回

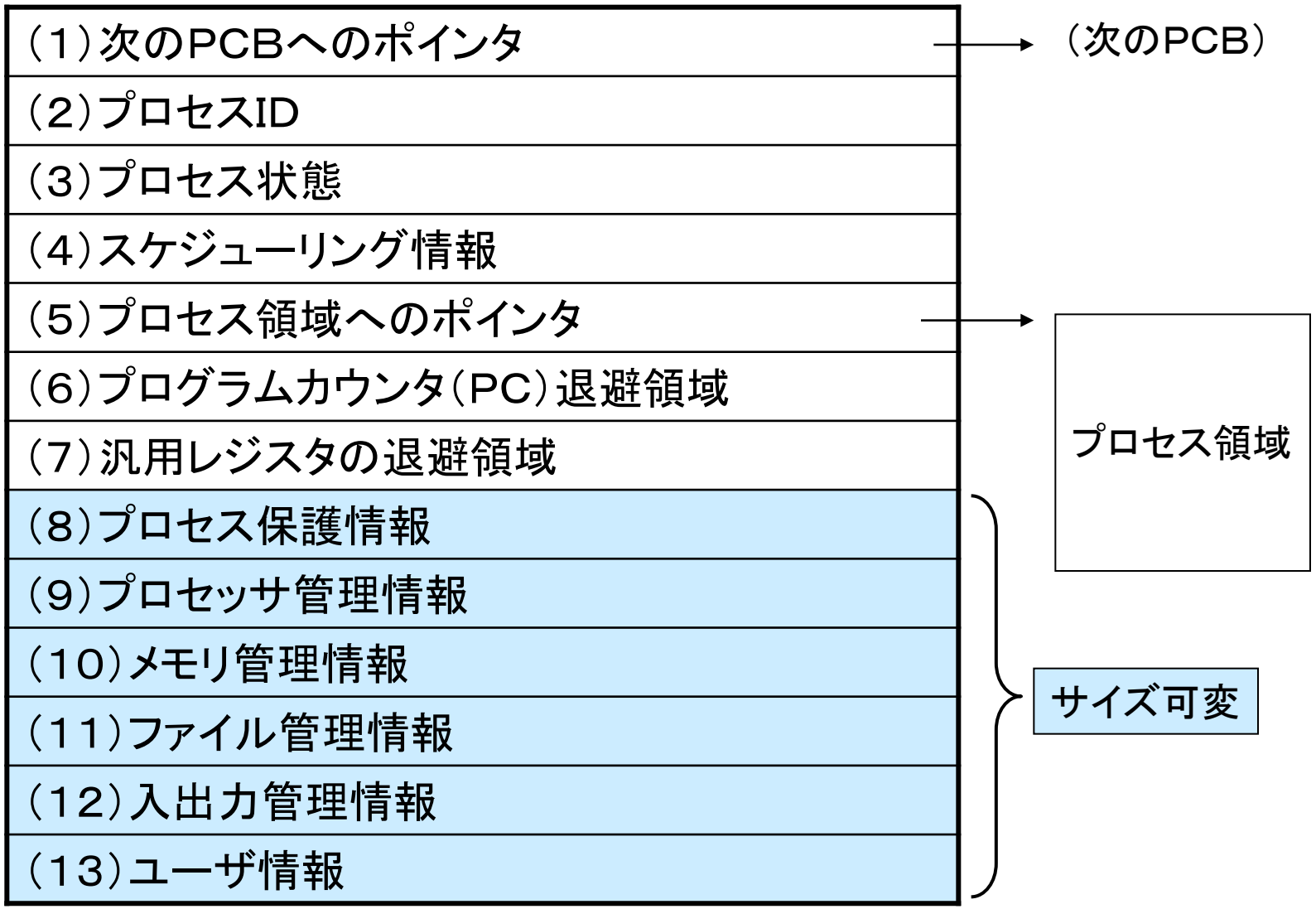
ソフトウェア情報学部
基盤ソフトウェア講座
澤本　潤

プロセススケジューリング（1）

プロセス領域



プロセス制御ブロック（PCB）とプロセスコンテキスト



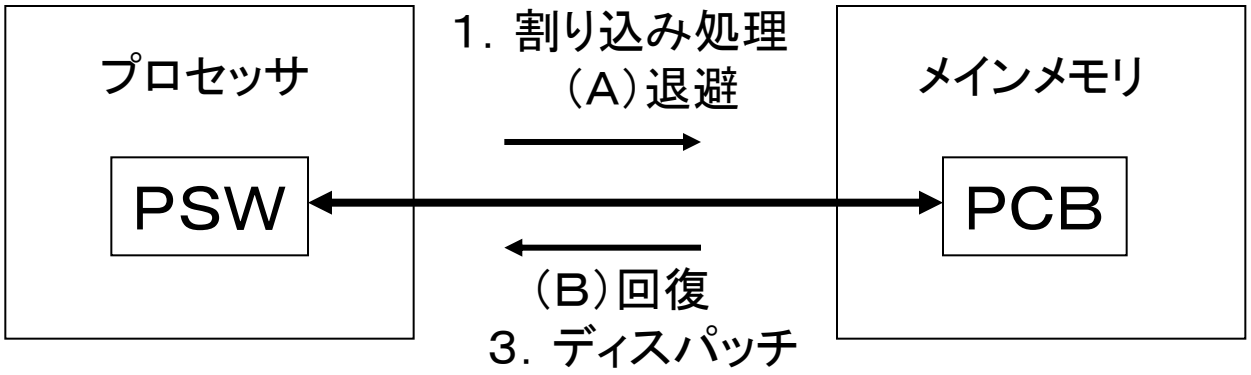
プロセッサ状態ワード（PSW）

内部装置であるプロセッサとメインメモリの
ハードウェア状態を示す。
PCB：論理的なプロセスコンテクスト
PSW：物理的なプロセスコンテクスト

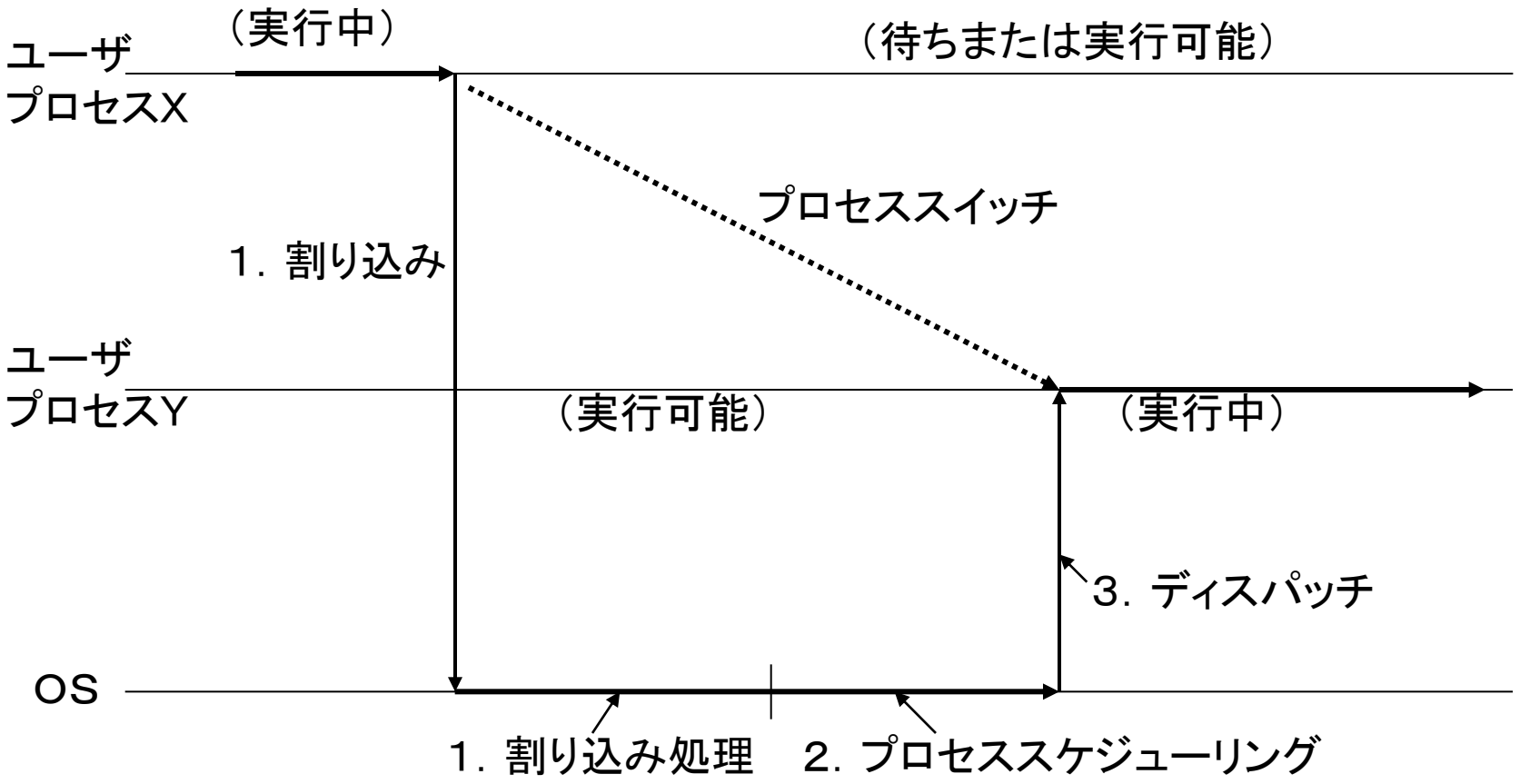
(a)プロセッサ状態
(b)コンディション（条件）
(c)プログラムカウンタ（PC）
(d)汎用レジスタ
(e)メモリイメージ
(f)メインメモリについての管理情報
(g)割り込みの優先度

プロセスコンテクストスイッチの動作

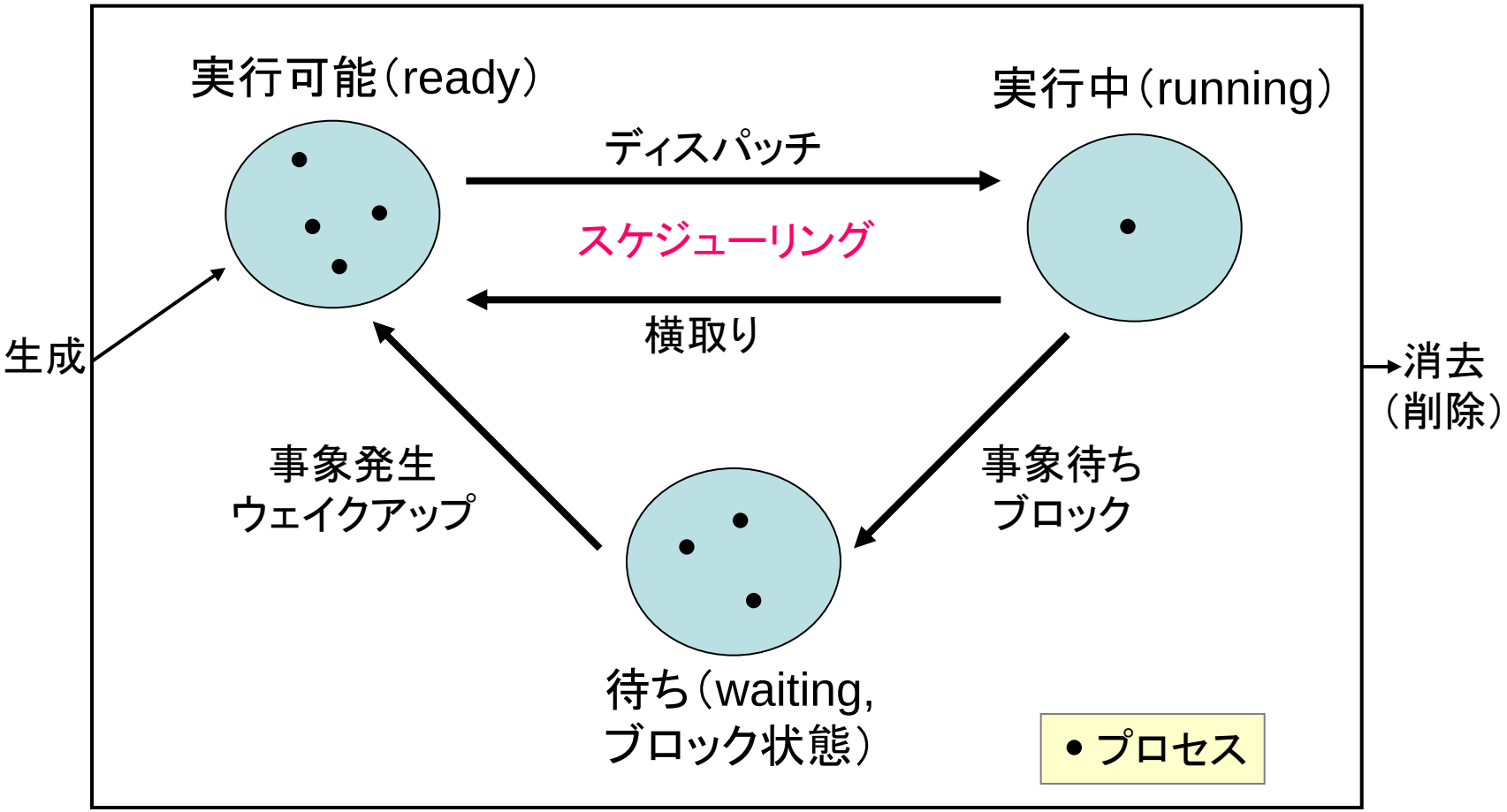
1. 割り込み処理（ユーザプロセス⇒OS）
2. プロセススケジューリング（狭義）
3. プロセスディスパッチ（OS⇒ユーザプロセス）



プロセススイッチ


（実行中）
（待ちまたは実行可能）
ユーザ
プロセスX
プロセススイッチ
1．割り込み
ユーザ
プロセスY
（実行可能）
（実行中）
3．ディスパッチ
OS
1．割り込み処理
2．プロセススケジューリング

プロセスの状態と状態遷移図



プロセススケジューリングとは？

プロセススケジューリング

- ユーザプロセスが共用あるいは共有するハードウェア資源（プロセッサ、メインメモリ、入出力装置など）を時間的あるいは空間的に有効利用する

スケジューラ (scheduler)

1つ以上のプロセスが実行可能状態であるとき、
その中から どれを先に実行するかを決定する部分

スケジューラで用いるアルゴリズム
→ スケジューリング・アルゴリズム

プロセスの性質

1. プロセッサバウンド（CPUバウンド）:
2. I／Oバウンド:

判定する定量的な指標例
1.プロセッサ時間
2.入出力時間
3.入出力命令数
4.ブロック要因

優先取得可能スケジューリング (preemptive scheduling)

論理的に実行可能なプロセスを、一時的に中断できるような戦略によるスケジューリング

⇕

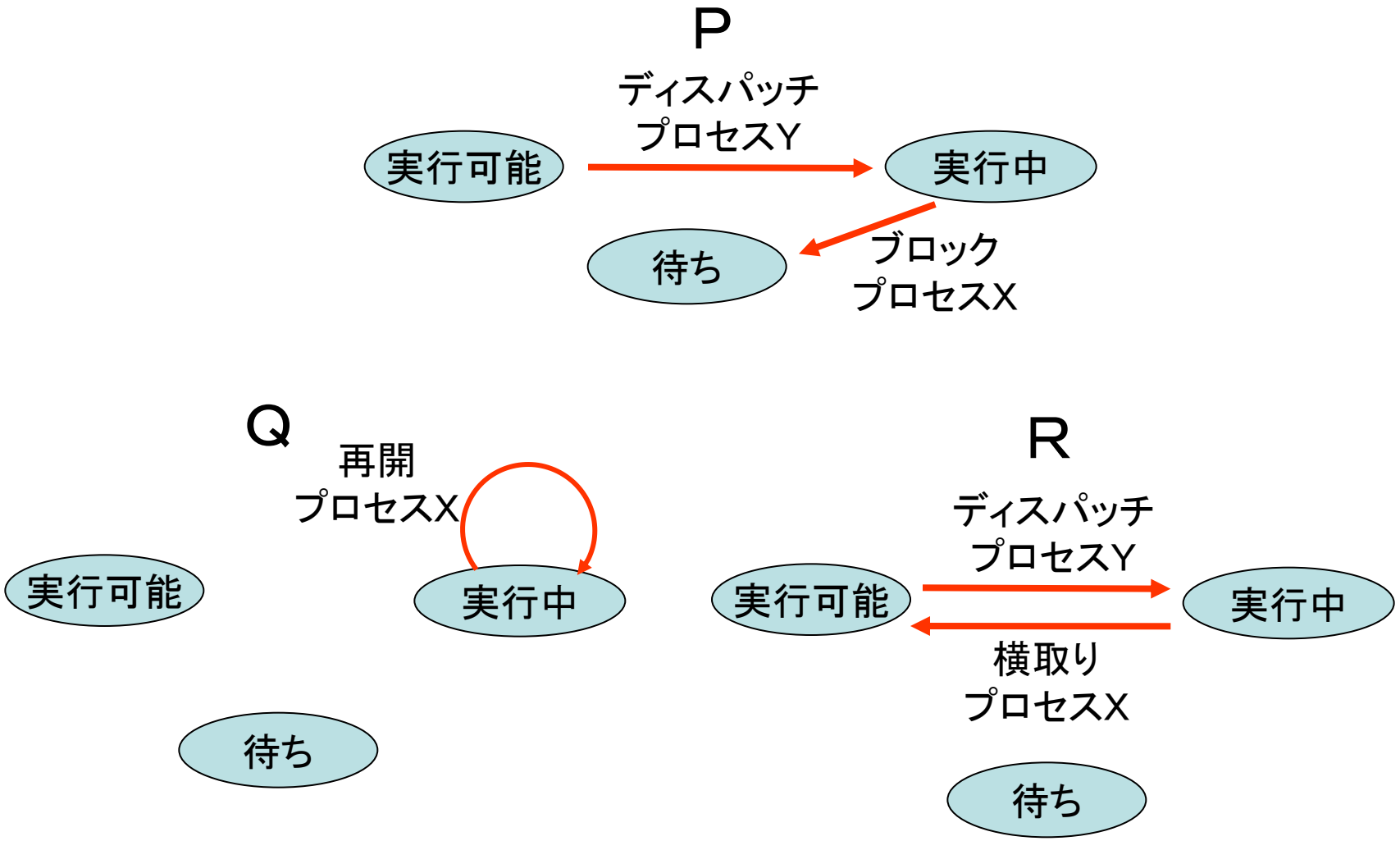
優先取得不可能スケジューリング (nonpreemptive scheduling)

※ preempt：【動】 先買権によって獲得する。先取する。
《文語》(…の) 代わりをする。 (…に) 代わる。
（テレビ・ラジオの定期番組を）さし替える。 私物化する。

※ 先買権（せんばいけん）：他人に先んじて物または権利を買い取る権利。

... CPUで実行する権利を横取りできるかどうか。

プロセススケジューリングによるプロセス状態遷移



スケジューラの行おうとすること

以下の基準について、ポリシーを決定する。

1. 公平さ (fairness) ..... 各プロセスを公平に
2. 効率 ..... CPU を休み無く利用（稼働率を上げる）
3. 応答時間 (response time)
   ..... 会話型ユーザへの応答を素早く
4. ターンアラウンド ..... バッチユーザへの出力を素早く
5. スループット ..... 単位時間あたりの処理ジョブ数を多く

上の目標は、それぞれ矛盾する。

スケジューリングアルゴリズムの選定指標

- プロセッサ（CPU）利用率（使用率）
  - CPUが稼動している割合
- スループット
  - 単位時間当たりに処理できたプロセスの数
- ターンアラウンド時間
  - プロセスの投入から完了までの時間
- 待ち時間
  - 実行待ち列で待っている時間
- 応答時間
  - プロセスが投入されてから実行開始までの時間

スケジューリングアルゴリズム

- 優先取得不可能(non-preemptive)横取りなし
  - 到着順 (First-Come-First-Served, FCFS)
  - 最短要求時間順(Shortest-Job-First, SJF)
  - 優先度順
- 優先取得可能(preemptive)　横取りあり
  - 優先度順
    - 最短実行時間
    - 最小残余時間(Shortest-Remaining-Time-First)
    - I／Oバウンド優先、エージングなど
  - ラウンドロビン(Round-Robin,RR)

横取りなし到着順 (FCFS)

- プロセスが、A、B、Cの順でほぼ同時に到着する
- 実行可能キューFIFOを利用
- プロセス特性を活用できない―短所
- チャートを書き、平均ターンアラウンド時間と平均応答時間を求めよ

プロセス	CPU時間
A	10
B	15
C	5

A ― B ― C の順

(10+25+30)／3 = 21.6 （平均ターンアラウンド時間）
(0+10+25)／3 = 11.6 （平均応答時間）

B ― A ― C の順

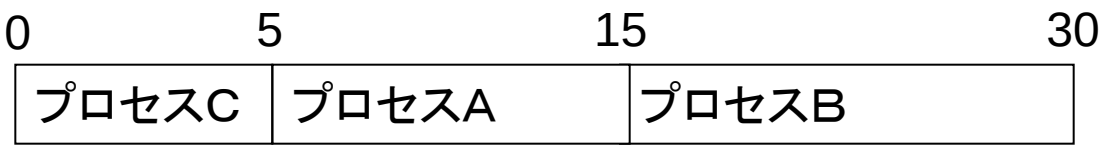
(15+25+30)／3 = 23.3 （平均ターンアラウンド時間）
(0+15+25)／3 = 13.3 （平均応答時間）

横取りなし最短要求時間順(SJF)

- プロセスが、A、B、Cの順でほぼ同時に到着する
- 平均ターンアラウンド時間と平均待ち時間が最小（最良）となる
- 各プロセスのプロセッサ（CPU）時間が既知でなければならない―非現実的な短所
- チャートを書き、平均ターンアラウンド時間と平均応答時間を求めよ

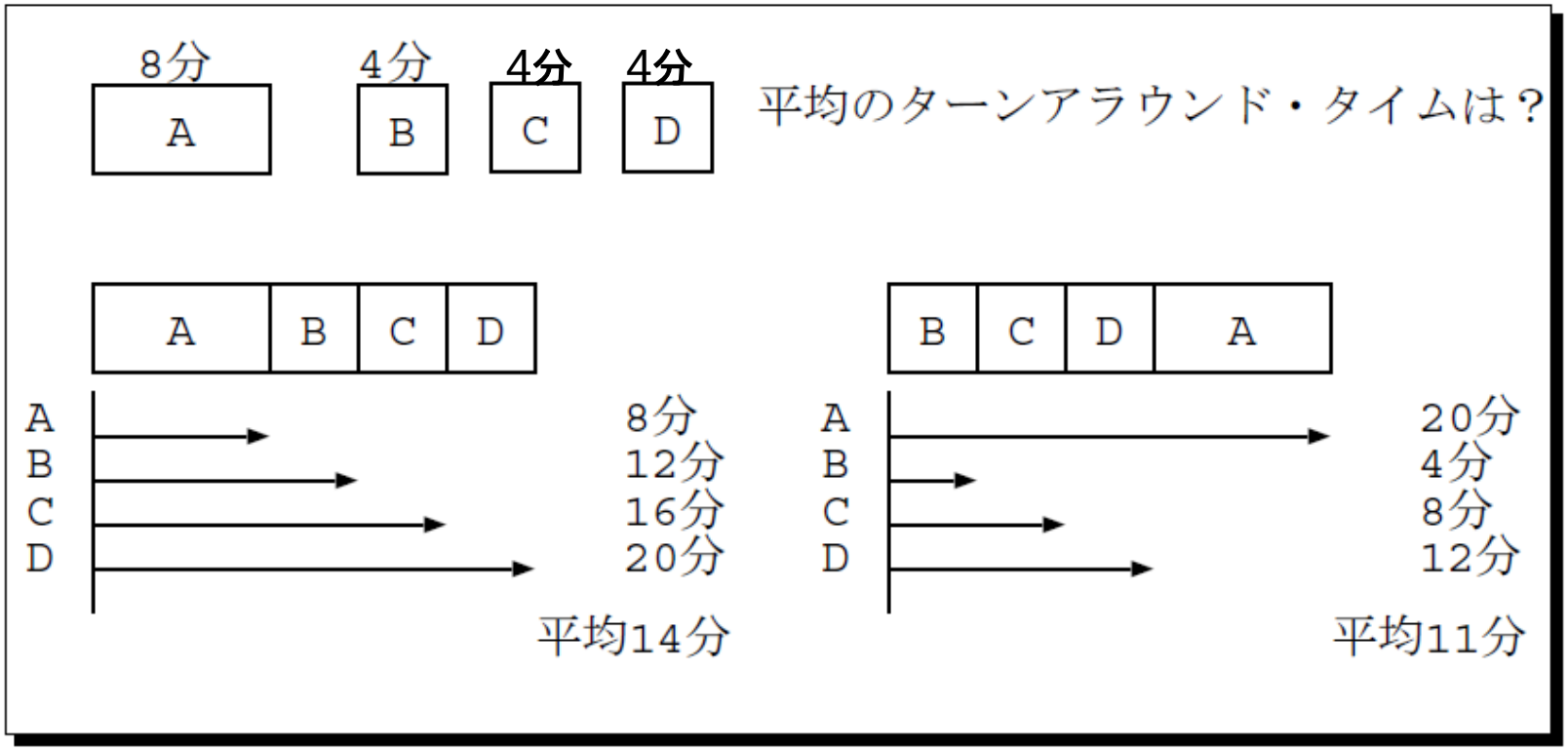
プロセス	CPU時間
A	10
B	15
C	5

C － A － B の順



(5+15+30)／3 = 16.6 （平均ターンアラウンド時間）
(0+5+15)／3 = 6.6 （平均応答時間）

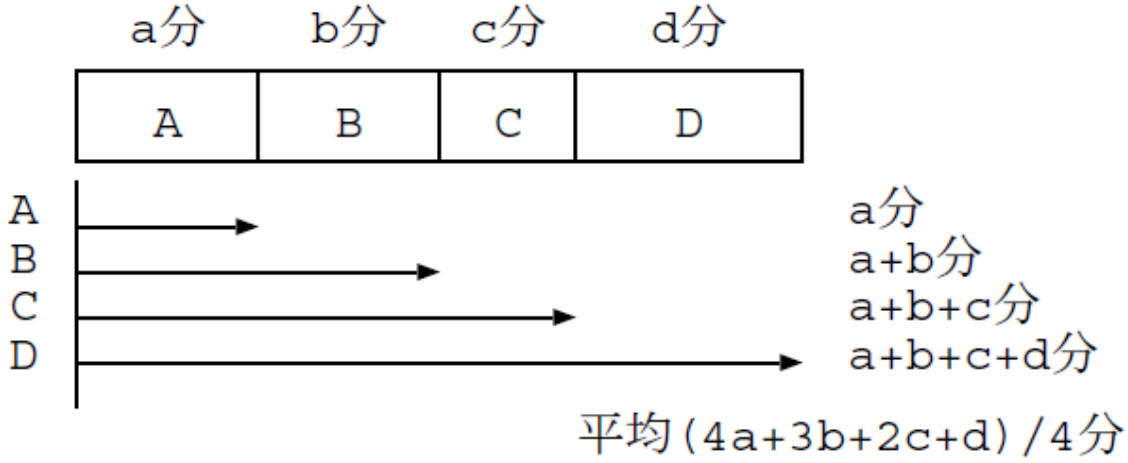
最短ジョブ優先(shortest job first)



実行時間が前もって分かっていることが前提

最短ジョブ優先(shortest job first)

一般には以下のようになる。



平均ターンアラウンド・タイムを最小にするには、

$$a \leq b \leq c \leq d$$

横取りなし優先度順

- プロセスが、同時に到着する。優先順は以下（5が優先順高）
- チャートを書き、平均ターンアラウンド時間と平均応答時間を求めよ

プロセス	優先順	CPU時間
1	3	10
2	1	6
3	4	2
4	5	4
5	2	8

(4+6+16+24+30)／5 = 16 （平均ターンアラウンド時間）

(0+4+6+16+24)／5= 10 （平均応答時間）

優先順位(priority)スケジューリング

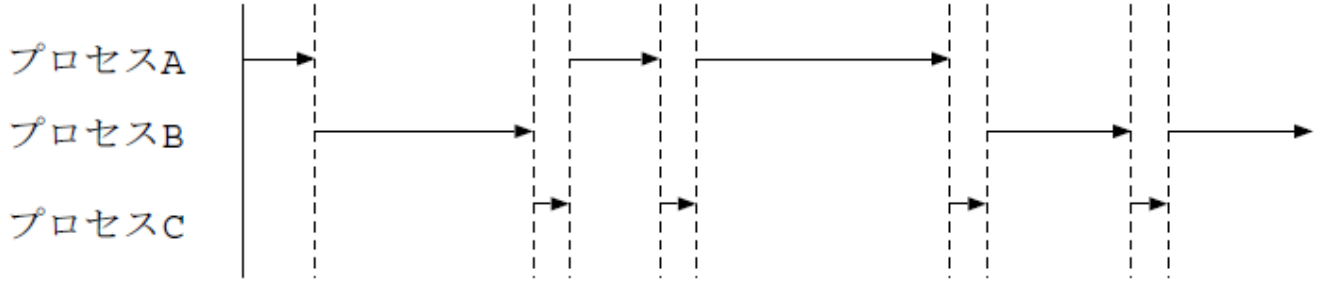
各プロセスに優先順位が割り当てられ、
優先順位の高いプロセスがスケジュールされる。
↓

応用例）

1. 静的に優先順位を、大将は100、大佐90、少佐80、大尉70、士官60 のように割り当てる。
   ※ ← 数値の大きいほど優先順位が高い
2. クロック割り込みごとに、現プロセスの優先順位を下げる。
3. 重要性に応じて、動的に優先順位を変える。

優先順位スケジューリングの応用例

例えば、レスポンスを速くするために、
入出力の必要なプロセスほど動かしたい
→「ブロックするプロセスほど重要なプロセス」とする



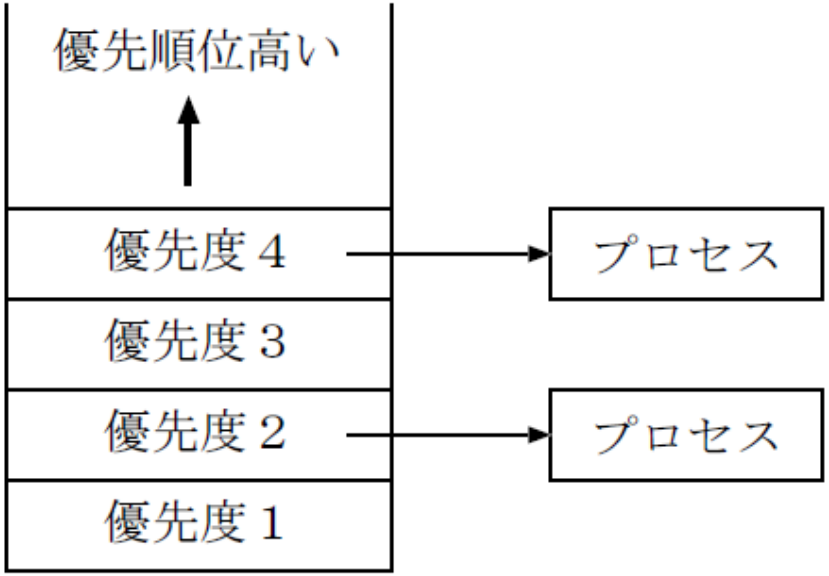
直前のクォンタムでCPUを使用した割合(f)に応じて
優先度$(1/f)$を与える。

quantum=100msecのところ、

- 2msecしかCPUを使わなければ 優先度 $1/0.02 = 50$
- 50msecの間CPUを使ったら 優先度 $1/0.5 = 2$

優先順位スケジューリングの実装

優先順位に対して、配列をメンテナンスする。

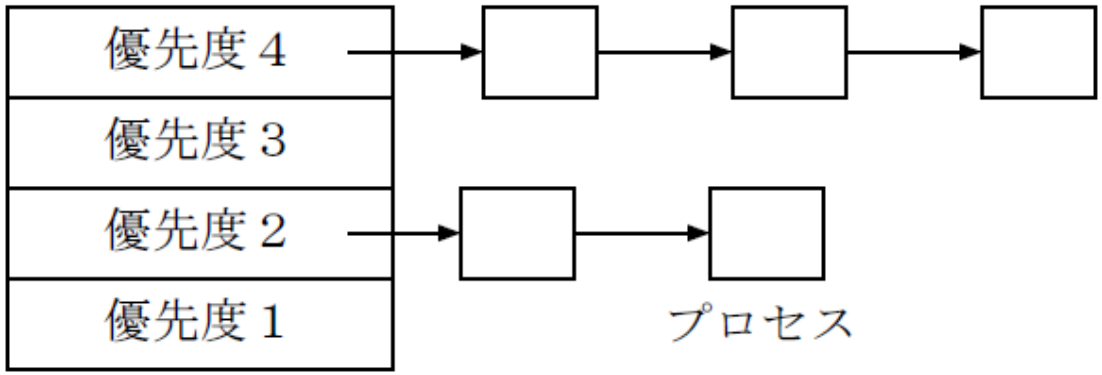


優先順位が同じプロセスが複数あったら、どうするか？

異なる優先順位は、いくつまであれば足りるか？

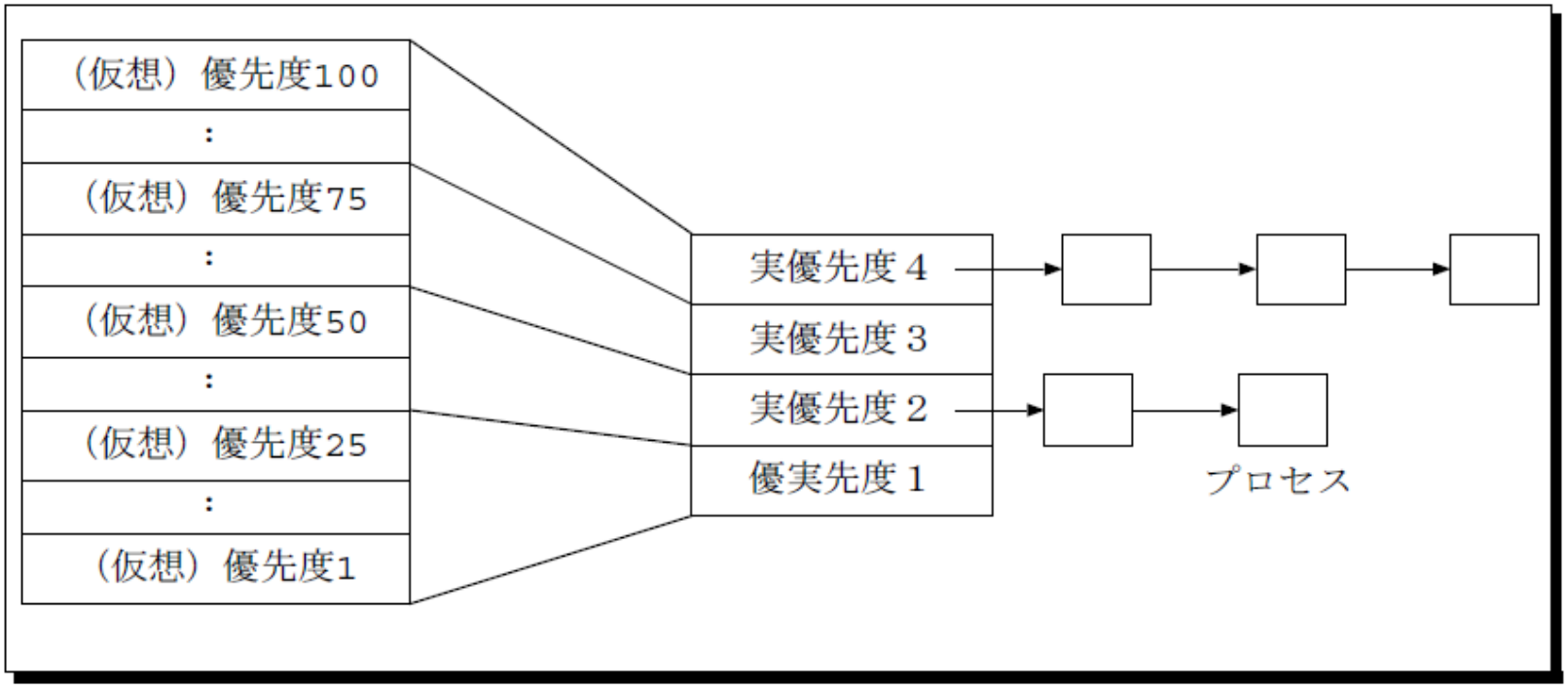
... 優先順位 10000 まであれば ....？

優先順位とラウンドロビンの併用



同じ優先順位のプロセスを、ラウンドロビンで動かす。

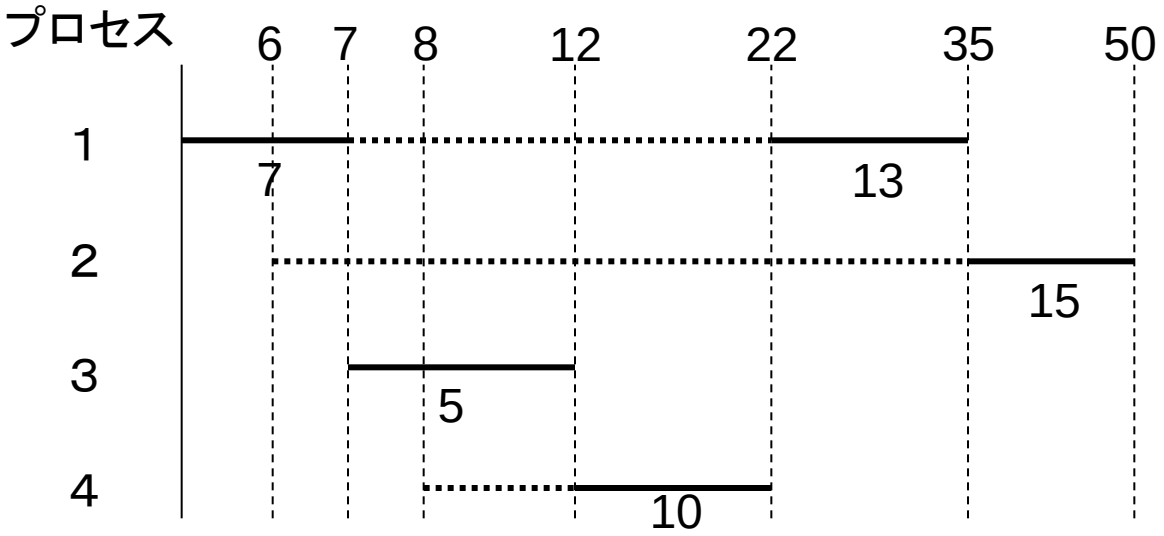
優先順位とラウンドロビンの併用



横取りあり　最小残余時間

- チャートを書き、平均ターンアラウンド時間と平均待ち時間を求めよ

プロセス	CPU時間（秒）	
1	20	
2	15	（プロセス1の6秒後に到着）
3	5	（プロセス1の7秒後に到着）
4	10	（プロセス1の8秒後に到着）



平均ターンアラウンド時間
(35+44+5+14)／4 = 24.5

平均応答時間
(0+29+0+4)／4 = 8.25

扱うプロセスの実行は、予測不可能である。

..... あるプロセスを実行させたら、

何分動くか分からない。

↑

たいていのコンピュータでは、組み込みの電子タイマやクロックで、定期的に割り込みがかかるようにしてある。

1秒間に50～60回 (50～60 Hz)

→クロック割り込み毎にOSが起動され、

現在実行中のプロセスの継続の可否を決定する。

最短ジョブ優先は、最小の平均応答時間になる。

↓

会話型プロセスにも応用できれば、効果が期待できる。

最短ジョブ優先方式 → **最短プロセス優先方式**

会話型プロセスの動作は、...

コマンド待ち→実行→コマンド待ち→実行→...

↑

コマンドの実行単位で個々のジョブとみなす。

問題点

実行可能なプロセスの中で、

どうやって最短のものを選ぶか？

一般には実行時間があらかじめ分からないので、最短ジョブ優先方式は実現できない。

↓

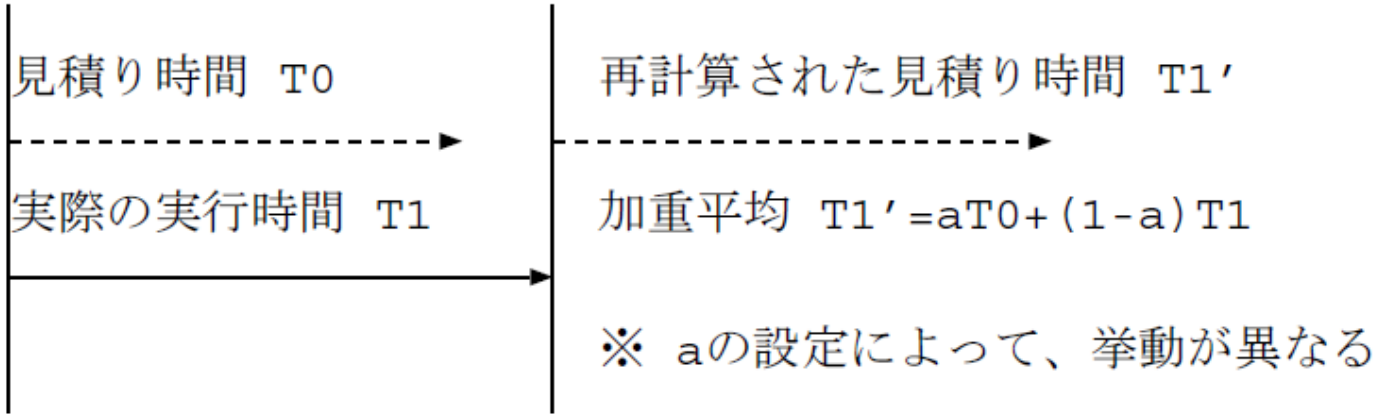
先のことが分からないなら、過去から予想するしかない。

↓

そのために、エイジングという技術を使う

エイジング (aging)

現在の計測値と以前の見積もりから、次の値を見積もる。

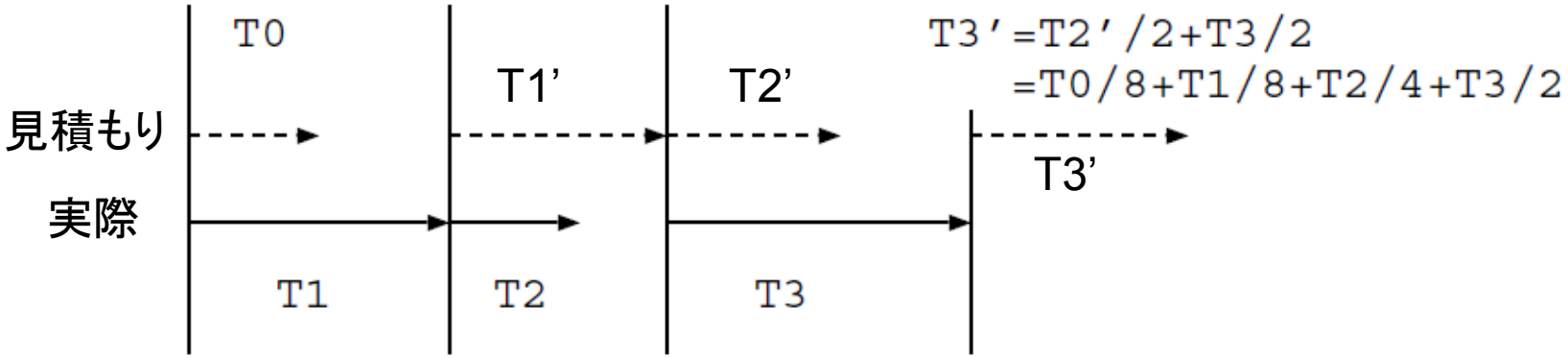


※ age : 【名】 年齢。 【動】 (人を) ふけさせる。(ものを) 古びさせる。
(ワイン・チーズを) ねかす。熟成させる。 Worry and illness age a man.

例）$a = \frac{1}{2}$とするとき

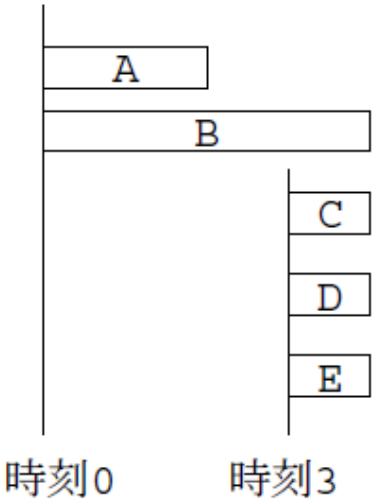
T1'=T0/2+T1/2

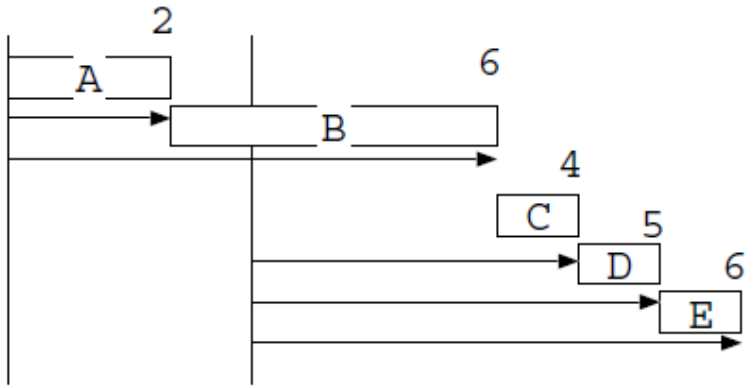
T2'=T1'/2+T2/2
=T0/4+T1/4+T2/2



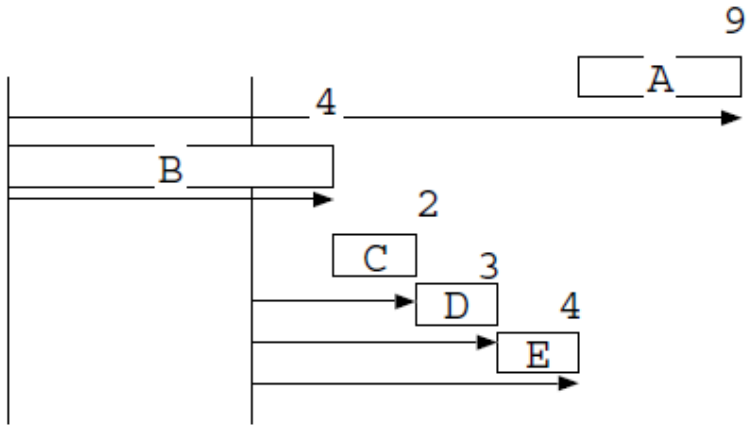
※ $a = \frac{1}{2}$が最も実装しやすい。

注）同時にReadyでないプロセスでは、最適になるとは限らない。





平均4.6

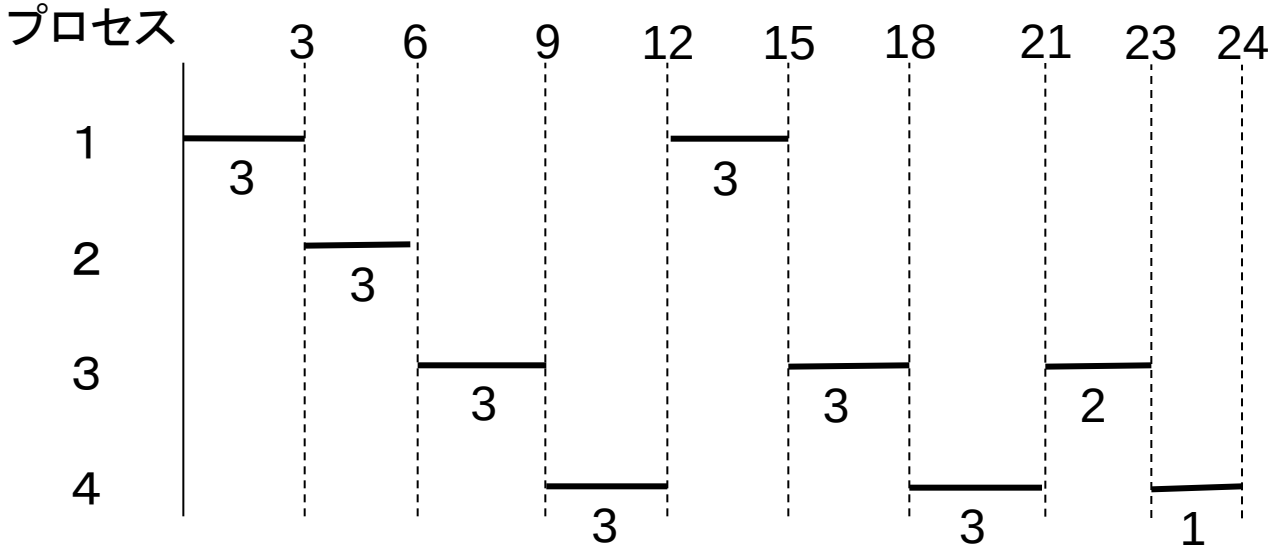


平均4.4

横取りあり　ラウンドロビン

- プロセスは、同時に到着する
- クォンタムを3とする
- ガントチャートを書き、平均ターンアラウンド時間と平均応答時間を求めよ

プロセス	CPU時間
1	6
2	3
3	8
4	7

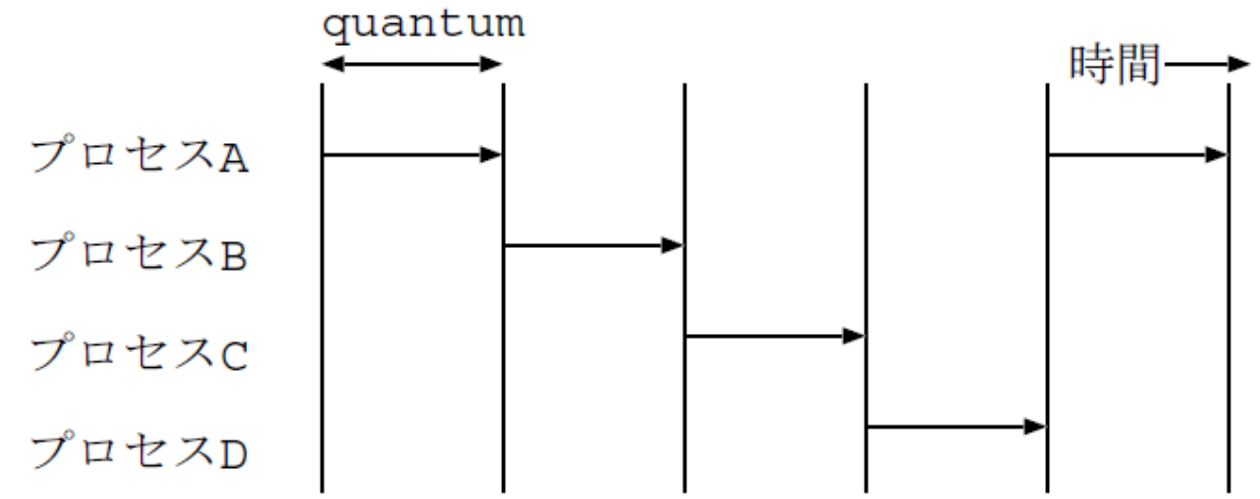


(15+6+23+24)／4 = 17（平均ターンアラウンド時間）

(0+3+6+9)／4 = 4.5（平均応答時間）

ラウンドロビン (round robin) スケジューリング

最も古く、単純で、公平かつ広く用いられている
アルゴリズム

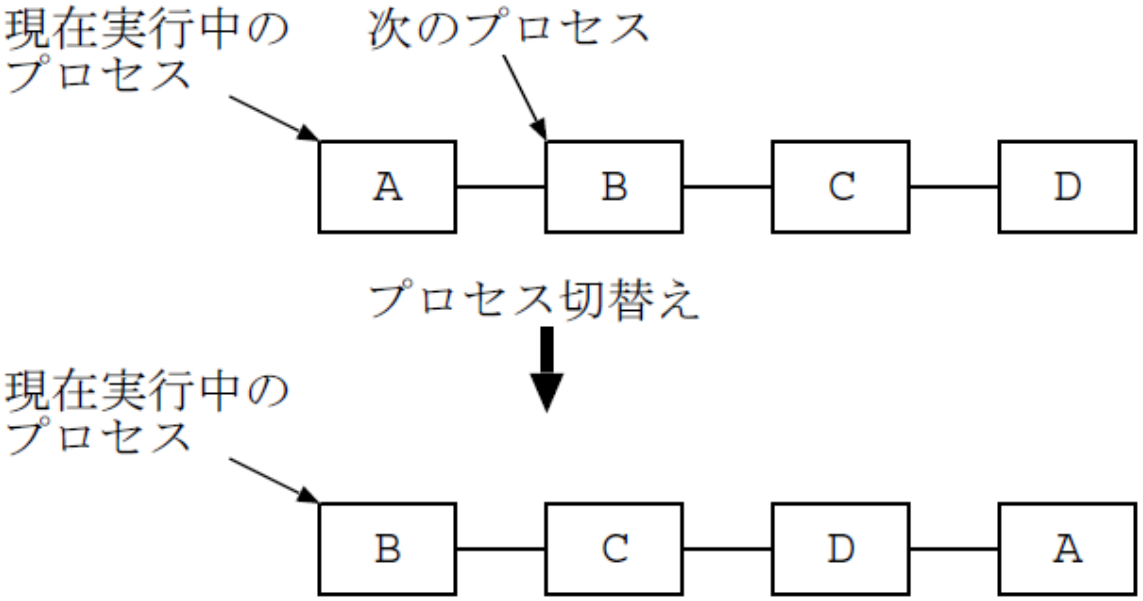


- 各プロセスは、
  実行許可された時間 – クォンタム (quantum)
  が与えられる。

※ round robin ... くるくる巡回する コマドリ ?

ラウンドロビンの実装

実装も単純で、実行可能プロセスを
リスト管理することだけ。

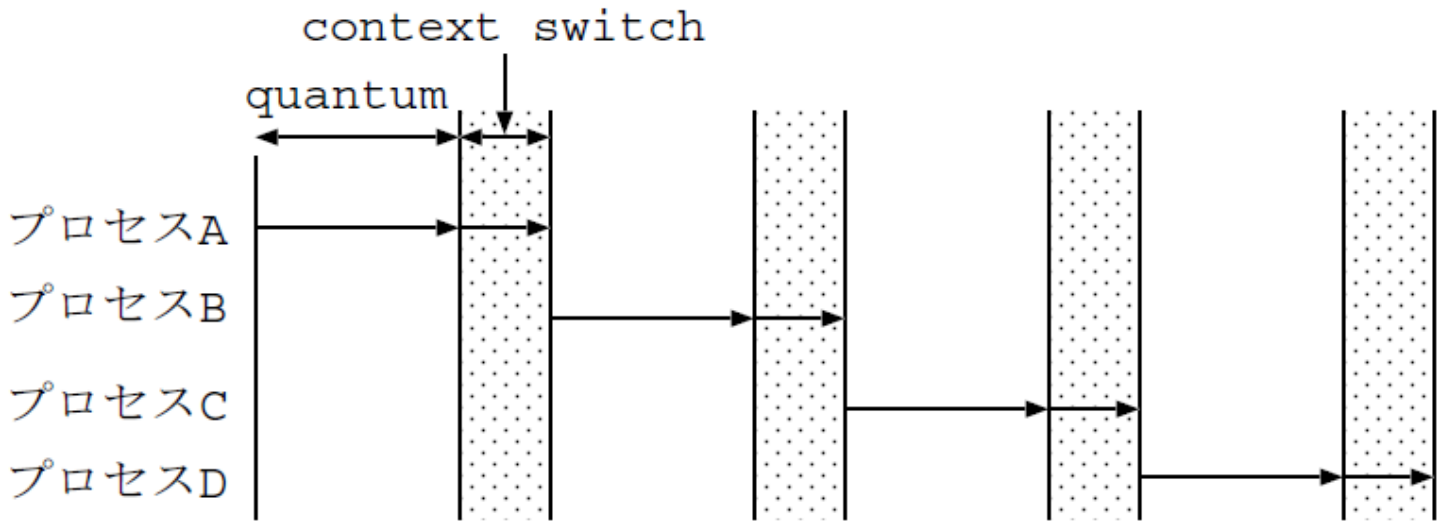


ラウンドロビンの問題点

クォンタムの長さの定め方によって、
システムの性質が変わる。
↑
なぜなら... (どんなスケジューリングであっても)
プロセス切り替え (process switch)
(あるいは **コンテキスト切り替え (context switch)** と呼ぶ)
にはコスト
(レジスタ等のセーブ/ロード、表やリストの更新...等々の処理時間)
がかかるから ....

※ context : 【名】 (文章の) 前後関係。文脈。(ある事柄の) 状況。環境

プロセス切り替え – クォンタムが短いとき

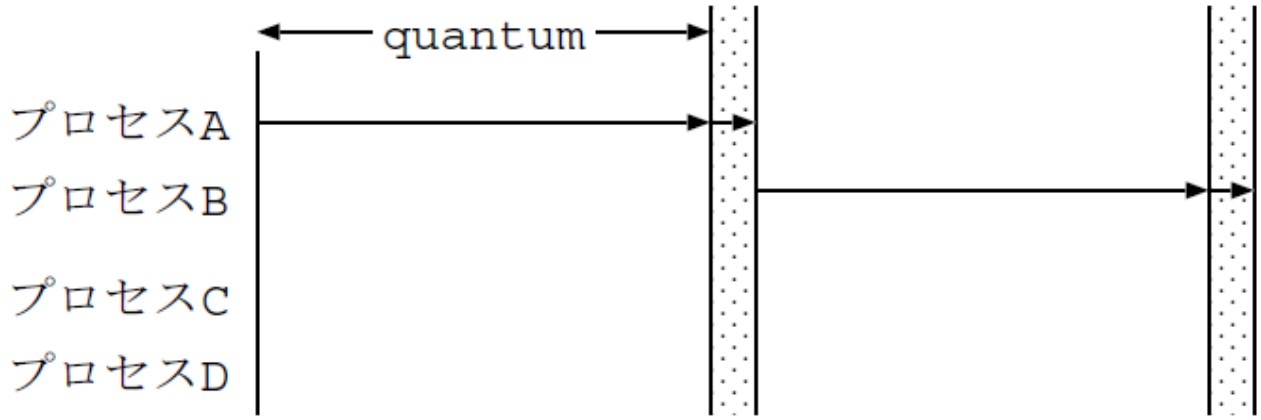


context switch : 5msec / quantum : 20msec

とすると、20%がオーバーヘッドとなる。

クォンタムを長くすれば、効率は改善するだろうか？

プロセス切り替え – クォンタムが長いとき



context switch : 5msec / quantum : 500msec
とすると、オーバーヘッドは1%以下となる。

しかし、..... 応答時間は長くなる。

ラウンドロビンスケジュールのまとめ

クォンタムを

非常に短くとると、プロセス切り替えが多発して
CPUの効率を落す。

非常に長くとると、プロセスの応答時間が長くなる。

↓

普通は、適当な長さの合理的なクォンタムで妥協する。

ラウンドロビンでは、すべてのプロセスは平等

↑

暗黙の仮定

⇕

重要な仕事のプロセスは、速く結果を返すべきでは？

多重レベルスケジューリング

- プロセスを特性別に分け、それぞれを複数の独立する実行可能キューで管理し、それぞれの特性に適した個別のスケジューリングアルゴリズムを適用する
  - 会話型タスクには、ラウンドロビン
  - リアルタイムタスクには、優先度順
- 多重レベルフィードバックスケジューリング
  - プロセスの実行可能キュー間の移動を許す多重レベルスケジューリング

演習クイズ（第３回）

- 優先取得不可能(non-preemptive)横取りなし
  - 到着順 (First-Come-First-Served, FCFS)
  - 最短要求時間順(Shortest-Job-First, SJF)
  - 優先度順
- 優先取得可能(preemptive)　横取りあり
  - 優先度順
    - 最小残余時間(Shortest-Remaining-Time-First)
  - ラウンドロビン(Round-Robin,RR)

演習クイズ（第3回）

- 優先取得不可能(non-preemptive)
  - 到着順 (First-Come-First-Served, FCFS)
    - プロセスが、1、2、3の順でほぼ同時に到着する
    - チャートを書け
    - FCFSの場合の平均ターンアラウンド時間を求めよ

プロセス	CPU時間
1	24
2	3
3	3

演習クイズ（第３回）

- 優先取得不可能(non-preemptive)
  - 到着順 (First-Come-First-Served, FCFS)
    - では、プロセスが、２、３、１の順でほぼ同時に到着する場合
    - チャートを書け
    - FCFSの場合の平均ターンアラウンド時間を求めよ

プロセス	CPU時間
１	２４
２	３
３	３

演習クイズ（第3回）

- 優先取得不可能(non-preemptive)
  - 最短要求時間順(Shortest-Job-First, SJF)
    - プロセスが、同時に到着する場合
    - チャートを書け
    - SJFの場合の平均ターンアラウンド時間と平均応答時間を求めよ

プロセス	CPU時間
1	6
2	3
3	8
4	7

演習クイズ（第３回）

- 優先取得不可能(non-preemptive)
  - 優先度順
    - プロセスが、同時に到着する場合、優先順は以下（５が優先順高）
    - チャートを書け
    - 平均ターンアラウンド時間と平均応答時間を求めよ

プロセス	優先順	CPU時間
１	３	１０
２	１	６
３	４	２
４	５	４
５	２	８

演習クイズ（第3回）

- 優先取得可能(preemptive)
  - 最小残余時間(Shortest-Remaining-Time-First)
    - チャートを書け
    - 平均ターンアラウンド時間と平均応答時間を求めよ

プロセス	CPU時間（秒）
1	20
2	15 （プロセス1の6秒後に到着）
3	5 （プロセス1の7秒後に到着）
4	10 （プロセス1の8秒後に到着）

演習クイズ（第3回）

- 優先取得可能(preemptive)
  - ラウンドロビン(Round-Robin,RR)
    - プロセスは、同時に到着する
    - クォンタムを3とする
    - チャートを書け
    - RRの場合の平均ターンアラウンド時間と平均応答時間を求めよ

プロセス	CPU時間
1	6
2	3
3	8
4	7