Date . . No.

② スループット

単位時間内に処理されるプロセス（仕事）の数

先の例では  $\frac{3}{18} = \frac{1}{6}$

③ 応答性

対話的なプロセスの場合，キーの入力やマウスのクリックなどの対話的な操作に対して，どれだけ迅速に応答できるかということも重要

実行開始時刻 − 到着時刻　と考えられるかも

④ 公平性

特定のプロセスばかりが実行され，他のプロセスがあまりに実行されない…ことは避けた

時に実行可能なプロセスにいつまでもCPU時間が割り当てられない現象をスタベーション（飢餓状態）とよび，これを避ける必要がある

4.4 スケジューリングのタイミング

スケジューリングが行われるタイミングには以下の2つがある

① プロセスが自主的にCPUを解放するとき

プロセスの実行が終了したとき
ディスクI/Oなどを依頼したとき
他のプロセスにCPU時間（割りあて）を譲るyield()というシステムコールを実行した場合

↓

ready状態のプロセスの選択へ

No. Date

② 強制的に（物理）CPUを横取りするとき

コンピュータの内部では時計が内蔵されており，一定の間隔で割込みを発生させている。
この割込みをタイマ割込みといい，この間隔をクロックティックあるいは単にtickという。

タイマ割込みが起きると，割込みハンドラが起動され，必要があればスケジューラを起動する。
スケジューラは何らかの基準にしたがって現在のプロセスを一時停止させるかどうか決め，必要があればプロセスを停止させる。

このような強制的なCPU時間（物理CPU）の横取りをプリエンプション（preemption）という。

英語の意味 preemption
先取権

具体的なスケジューリングアルゴリズム

I プリエンプション（横取り）なし　非プリエンプティブな方式
non-preemptive

① 到着順，FCFS（First Come First Served），FIFO（First In First Out）方式

プロセスの到着順序に従って処理する

マルチタスクがあまりいかせない…
↳ 応答性の悪いプロセスが出る

Date . . No.

② 最短ジョブ優先方式 (shortest job first)
(横取りなし)

プロセスの実行時間 (処理時間, バースト時間) が最短のものから実行する

平均ターンアラウンド時間を最小にする

	処理時間	
プロセス A	5	同時に到着
プロセス B	2	
プロセス C	3	

プロセスの実行時間が見つもれないといけない

③ 優先度順 (横取りなし)

プロセスに優先度をつけて、優先度順に実行する

優先度の低いプロセスにスタベーションが生じる可能性がある

↳ エイジング (aging) で対応

実行可能状態であるのに長い時間CPU時間が割りあてられなかったプロセスの優先度を少しずつ上げていく

No. Date

II プリエンプティブあり

① ラウンドロビン (round-robin) 方式

複数のプロセスを かわりばんこに実行する方式

各プロセスを一定時間連続して実行する

↳ この時間のことを タイムスライス
タイムクォンタム という

ラウンドロビンスケジューリングの流れ

1. 実行中のプロセスが タイムクォンタムを使いきる
2. スケジューラは、実行中のプロセスを停止し、
レディキューの最後に追加する
3. レディキューの先頭にあるプロセスを取り出し、実行状態にする

タイムスライスを使いきったかどうかは、タイマ割込みの割込みハンドラ内で判定する
現在実行中のプロセスがCPU時間を割りあててからどれだけの時間が経過しているかを調べ、タイムクォンタムを過ぎていたらスケジューラーを起動し、そうでないならプロセスの実行を続ける

タイムクォンタムを∞とする → FCFSと同じ

タイムクォンタムが大きすぎると 対話プロセスの応答性が悪くなる

小さすぎると コンテキスト切替に要するオーバーヘッドが
相対的に大きくなる

実際のOSでは 100msec ぐらいが多い

Date . . No.

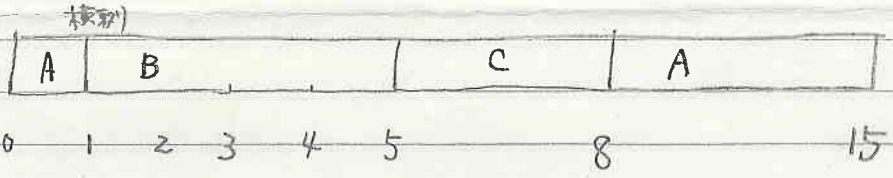
② 横取りありの最短ジョブ優先式

最小残り時間優先方式 (shortest - remaining - Time - first)

スケジューラが動作したタイミング
で各プロセスの処理時間を再計算して，短いものから実行
↓
必要とあれば横取りする

	到着時刻	処理時間
プロセス A	0	8
プロセス B	1	4
プロセス C	3	3

各時刻でスケジューラが動作するとした場合



残り時間					
A	7	7	7		7
B	4	3	2		0
C			3		3

No. Date

(3) 横取りつき優先度有スケジューリング + ラウンドロビン

実行中のプロセスよりも優先度の高いプロセスが実行可能状態となった場合、実行中のプロセスを停止し、優先度の高いプロセスが実行される

常に優先度の高いプロセスが実行されている状態となる

なお同じ優先度のプロセスについてはラウンドロビン方式を用いてスケジューリングする場合が多い

Date . . No.

6 メモリ管理

素朴なメモリ管理 → 飛ばす

どのようなメモリ管理が望ましいのか

① なるべく断片化 (fragmentation) を起こしにくくする

② プロセス間にメモリ保護がある

③ 物理メモリ量を越えるメモリ量を利用可能とする

OSは各プロセスに対して 仮想アドレス空間 (論理アドレス空間) を与える

例えば, 各プロセスに対して 0〜4GBのメモリ空間を割りあてる

各論理空間内のアドレス (仮想アドレス or 論理アドレス) が同じであっても, 実際には別のものとなっている

プロセスは各々, 独立なメモリ空間を有している

↳ プロセスは他のプロセスの仮想アドレス空間にアクセスできない

↳ メモリ保護が達成される

一方, 物理メモリ内の番地を物理アドレスといい, 物理アドレスの空間を物理アドレス空間という

通常のプロセスは 物理アドレス空間に直接アクセスできない