オペレーティングシステム

第3回

プロセスの管理とスケジューリング

http://www.info.kindai.ac.jp/OS
E館3階E-331 内線5459
takasi-i@info.kindai.ac.jp

1

オペレーティングシステムの主要概念
プロセス(process), タスク(task)

- 実行中のプログラム
- プログラム実行に必要な情報
  - プログラムコード, データ, スタック, プログラムカウンタ, スタックポインタ, 汎用レジスタ, 開いているファイル

実行 → 停止 → 実行 の繰り返し

再開時に以前の状況を引き継ぐ必要がある

2

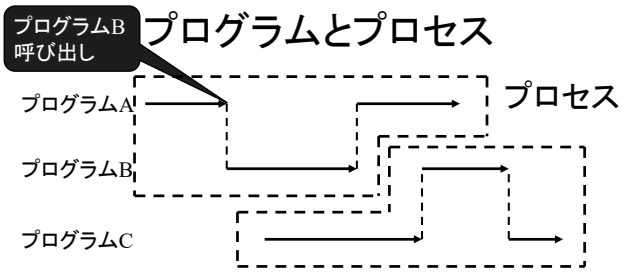
プロセス(Process)

- プロセス(タスク)
  - CPUのスケジューリング対象となる基本単位
  - 実行中のプログラムと実行に必要なデータ
  - 動的な概念(時々刻々と変化するもの)として把握

「プロセス = プログラム」か?

3

プログラムとプロセス



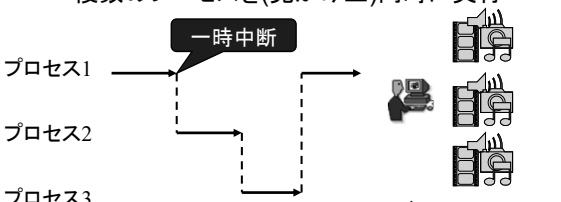
プログラム A→プログラム B→プログラム Aを1つのまとまりとして見た方が便利

左のプログラム B と右のプログラム B は違うものと見た方が便利

4

プロセスの並行処理

- 並行処理(concurrent processing)
  - 複数のプロセスを(見かけ上)同時に実行

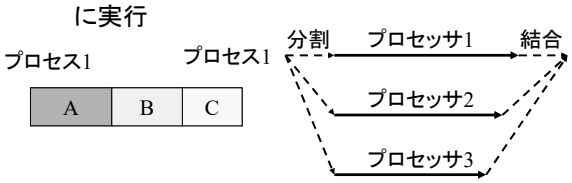


ユーザにとっては
3つのプロセスが
同時に実行されている

5

プロセスの並列処理

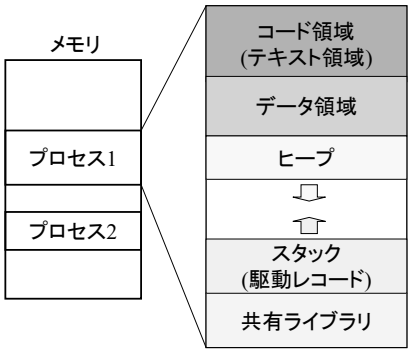
- 並列処理(parallel processing)
  - プロセスを分轄して複数のプロセッサで同時に実行



複数のプロセッサで高速実行
並行処理 ≠ 並列処理

6

プロセスの構造



7

プロセスの構造

- コード領域(code segment),
  テキスト領域(text segment)
  - プログラム命令のコード
    (歴史的な理由からテキストと呼ばれている)
- データ領域(data segment)
  - 静的なデータ

8

プロセスの構造

- ヒープ(heap)
  - プログラム実行時に確保されるメモリ領域
- スタック(stack)
  スタックフレーム(stack frame)
  駆動レコード, 活性レコード(activation record)
  - 関数の引数, 関数の局所変数, 前フレームへのポインタ, 関数呼び出しの戻り番地

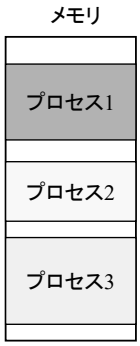
9

プロセッサの状態(processor state)

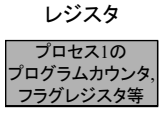
- プロセッサの状態はレジスタが保持
- レジスタ(register)
  - 中央演算装置(CPU)との間で高速にデータ転送を行うことができる記憶装置
  - サイズは小さいが非常に高速
    - プログラムカウンタ
    - フラグレジスタ
    - アキュムレータ
    - スタックレジスタ
    - 割り込みレジスタ

10

プロセッサの状態



プロセス1実行中

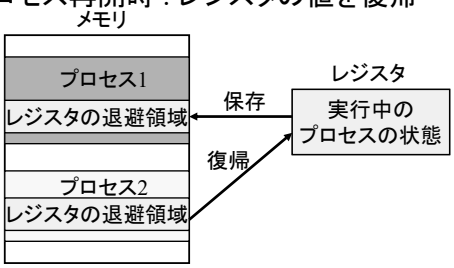


実行中ではないプロセス2,3の
状態も保持しておく必要あり

11

プロセッサの状態

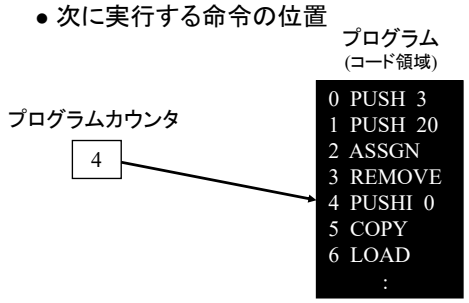
- プロセス中断時 : レジスタの値を保存
- プロセス再開時 : レジスタの値を復帰



12

レジスタ

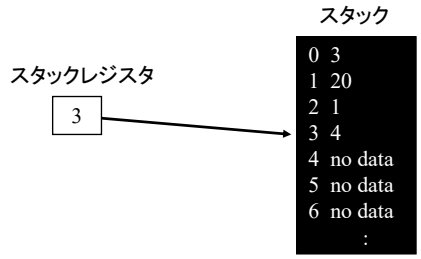
- プログラムカウンタ(program counter)
  - 次に実行する命令の位置



13

レジスタ

- スタックレジスタ(stack register)
  - スタックトップの位置



14

レジスタ

- フラグレジスタ(flag register)
  - 特定の命令を実行した後に自動的に付与
    - OF(overflow flag) : 桁あふれ発生
    - ZF(zero flag) : 演算結果がゼロ
    - SF(sign flag) : 演算結果がマイナス
- アキュムレータ(accumulator)
  - 論理演算, 四則演算の入力と結果を保持
- 割り込みレジスタ(interrupt register)
  - 割り込みに必要なデータを保持

15

プロセス記述子(process descriptor)
プロセス制御ブロック(process control block)

- プロセス記述子(process descriptor),
  プロセス制御ブロック(process control block)
  - プロセスの状態を格納

プロセス記述子

プロセス識別子
プロセスの状態
スケジューリング情報
資源利用情報

16

プロセス記述子(PCB)

- プロセス識別子(process ID)
  - プロセス生成時に付けられる一意な番号
- プロセスの状態
  - 各種レジスタの値
- スケジューリング情報
  - プロセスの優先度
- 資源利用情報
  - 使用しているメモリ領域へのポインタ
  - 開いているファイルへのポインタ

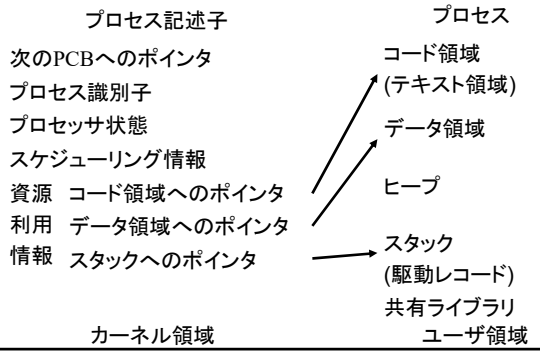
17

プロセス記述子(PCB)

1. 次のPCBへのポインタ
2. プロセス識別子
3. プロセスの状態
4. プロセスの優先度 — スケジューリング情報
5. コード領域へのポインタ — 資源利用情報
6. データ領域へのポインタ
7. スタックへのポインタ
8. プログラムカウンタの退避領域
9. レジスタの退避領域
10. メモリ管理情報
11. 入出力情報

18

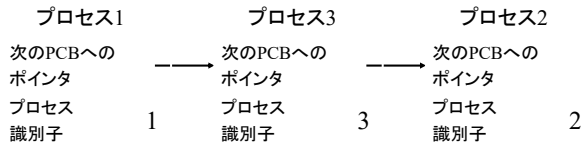
プロセス記述子(PCB)



19

プロセス記述子(PCB)

- プロセス記述子はキューに格納



キューの先頭のプロセスから実行される

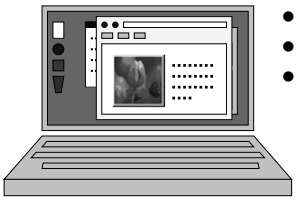
20

プロセスの状態

- 実行中(running)
  - プロセッサを使用している
  - タイムアウト(timeout)で実行可能へ移行
- 実行可能(ready)
  - プロセッサが空くのを待っている
  - ディスパッチ(dispatch)により実行中へ移行
- ブロック(blocked), 待ち(waiting)
  - ただちにプロセッサを使うことはできない
  - 入出力待ち, イベント待ち等

21

プロセスの状態



起動中のプロセス
- テキストエディタ
- メーラ
- ウェブブラウザ

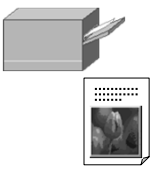
現在ウェブブラウザを使用中

ウェブブラウザ : 実行中
テキストエディタ, メーラ : 実行可能

22

プロセスの状態

類似例 : プリンタの場合



実行中(running) = 印刷中

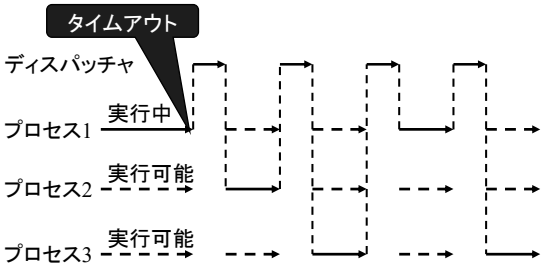
実行可能(ready) = 印刷待ち

ブロック(blocked) = 紙切れ

.....

23

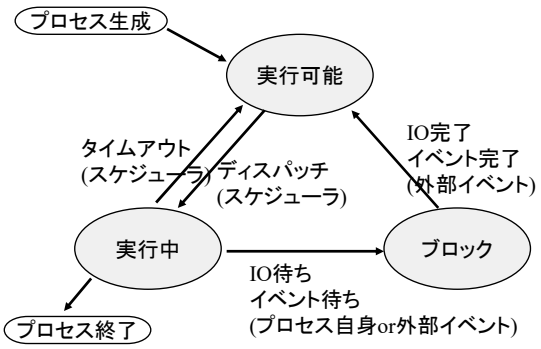
プロセスの状態遷移



各プロセスの状態を頻繁に遷移することにより
見かけ上同時に複数のプロセスを実行できる

24

プロセスの状態遷移



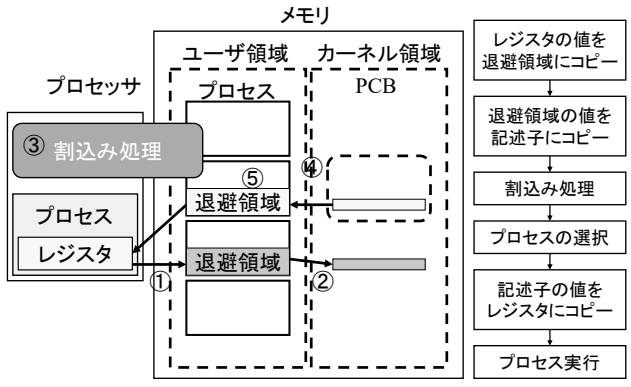
25

プロセスの状態遷移

- 実行中から実行可能への移行
  1. レジスタの値をメモリの退避領域にコピー
  2. 1で退避された値をプロセス記述子の退避領域にコピー
  3. 割込みに対応した処理 — 割込みハンドラ
  4. 次に実行するプロセスを決定 — スケジューラ
  5. 5で選択したプロセス記述子のレジスタ退避領域の値をレジスタにコピー
  6. プログラムカウンタの示す行からプロセスの実行開始 — ディスパッチャ

26

プロセスの状態遷移



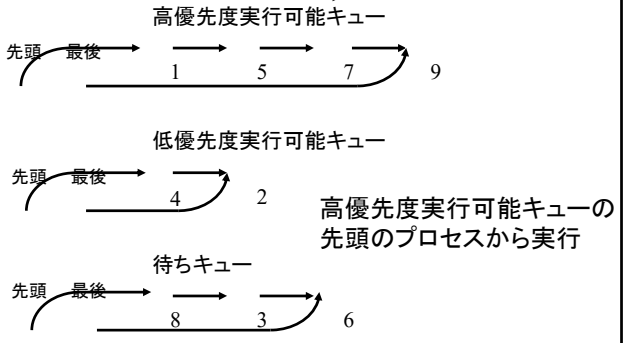
27

実行可能キュー(ready queue) 待ちキュー(waiting queue)

- 実行可能キュー(ready queue)
  - 実行可能状態のプロセスのキュー
  - キューの先頭のプロセスから順に実行
  - 優先順位別に複数キューの場合もある
- 待ちキュー(waiting queue)
  - ブロック状態のプロセスのキュー
  - 待ち状態が解消されると実行可能キューへ

28

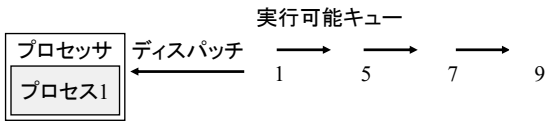
実行可能キュー, 待ちキュー



29

実行可能キュー

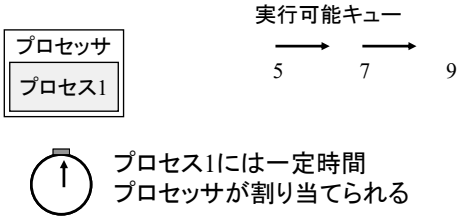
実行可能キューの先頭のプロセスから実行



30

実行可能キュー

実行可能キューの先頭のプロセスから実行

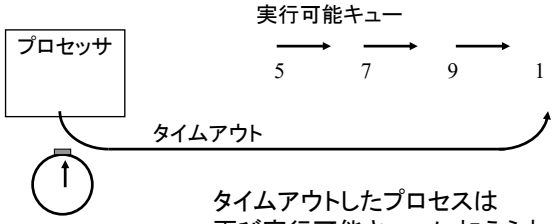


プロセス1には一定時間
プロセッサが割り当てられる

31

実行可能キュー

実行可能キューの先頭のプロセスから実行

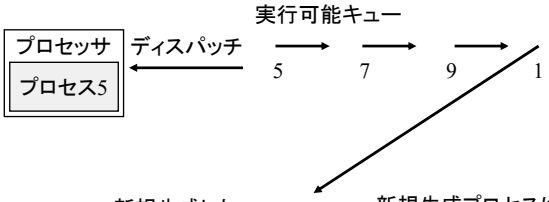


タイムアウトしたプロセスは
再び実行可能キューに加えられる

32

実行可能キュー

実行可能キューの先頭のプロセスから実行



33

スケジューリング(scheduling)

- スケジューリング(scheduling)
  - 次にどのプロセスを実行するかを決定

プロセッサ

実行可能状態のプロセス

この中のどれを次に実行する？

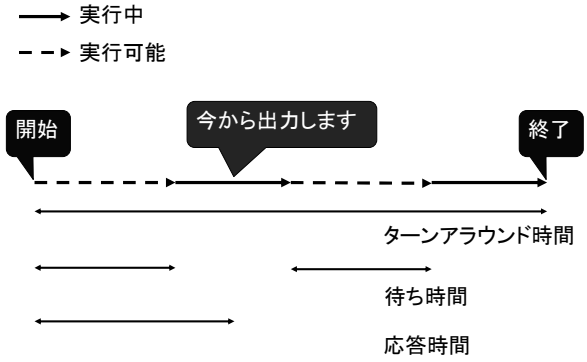
34

スケジューリング

- スケジューリングアルゴリズム選択の指標
  - CPU利用率(CPU utilization)
    - CPUの動作時間 / システム稼働時間
  - スループット(throughput)
    - CPUが単位時間当たりに完了するプロセス数
  - ターンアラウンド時間(turnarround time)
    - プロセス実行要求から完了するまでの時間
  - 待ち時間(waiting time)
    - プロセスが完了するまでに実行可能キューで待つ時間
  - 応答時間(response time)
    - プロセス実行要求から応答開始までの時間

35

スケジューリング



36

スケジューリング

- 良いスケジューリングアルゴリズム

CPU利用率	最大
スループット	最大
ターンアラウンド時間	最小
待ち時間	最小
応答時間	最小

しかしこれら全てを同時に満たすのは難しい

37

スケジューリングアルゴリズム

- 実行するプロセスの決定の仕方
  - 到着順(first come first service)
  - ラウンドロビン(round robin)
  - 処理時間順(shortest processing time first)
  - 残余処理時間順(shortest remaining time first)
  - 優先度順(priority dispatching)
  - 多重フィードバック(multiple feedback)

38

スケジューリングアルゴリズム
到着順(FCFS)

- 到着順(first come first service, FCFS)
  - プロセスの到着順に処理
  - 処理中のプロセスが終わるまで実行

長所
- 公平・簡単

短所
- 処理に時間のかかるプロセスが他のプロセスの実行を妨げる
- 優先度の高いプロセスが先に実行されない

39

スケジューリングアルゴリズム
到着順(FCFS)

プロセス	到着順位	処理時間
1	1	10
2	2	5
3	3	20

40

スケジューリングアルゴリズム
ラウンドロビン(RR)

- ラウンドロビン(round robin, RR)
  - プロセスの到着順に処理
  - 一定時間が過ぎると処理中のプロセスはタイムアウト, キューの末尾へ

タイムスライス(time slice), 定時間(time quantum)
多くの場合 1/60 sec (16.7ms)

長所
- 各プロセスに公平に時間が割り当てられる

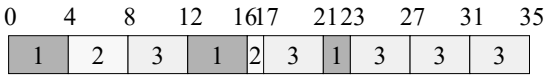
短所
- プロセスが入力待ち等でブロック状態になってもプロセッサが開放されない

41

スケジューリングアルゴリズム
ラウンドロビン(RR)

(タイムスライス : 4)

プロセス	到着順位	処理時間
1	1	10
2	2	5
3	3	20



42

スケジューリングアルゴリズム
処理時間順(SPT)

- 処理時間順(shortest processing time first, SPT)
  - プロセスの処理時間の短い順に処理
    - 実行可能のプロセスと処理時間を比較

長所
- 処理時間の短いプロセスのターンアラウンド時間が改善される

短所
- 処理時間の予測が必要
- プロセスに割り当てられる時間が不公平

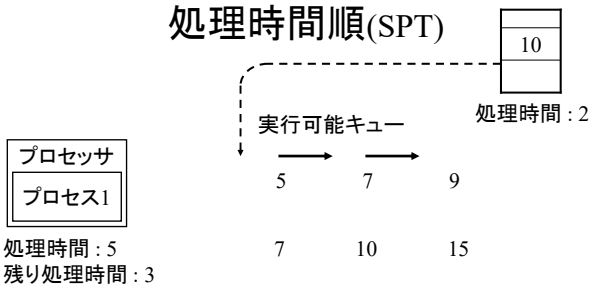
43

スケジューリングアルゴリズム
処理時間順(SPT)

プロセス	到着順位	処理時間
1	1	10
2	2	5
3	3	20

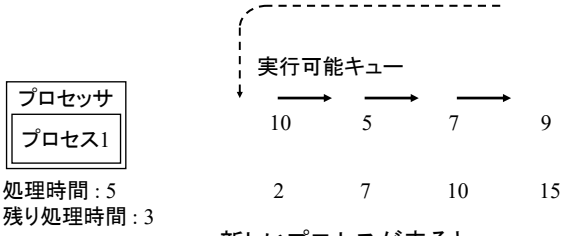
44

スケジューリングアルゴリズム
処理時間順(SPT)



45

スケジューリングアルゴリズム
処理時間順(SPT)



新しいプロセスが来ると
処理時間順に並ぶように
実行可能キューに加える

46

スケジューリングアルゴリズム
残余処理時間順(SRT)

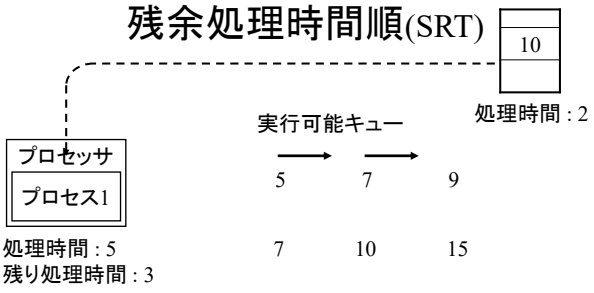
- 残余処理時間順(shortest remaining time first, SRT)
  - プロセスの残り処理時間の短い順に処理
    - 実行中のプロセスと処理時間を比較

長所
- 処理時間の短いプロセスのターンアラウンド時間が改善される

短所
- 処理時間の予測が必要
- プロセスに割り当てられる時間が不公平

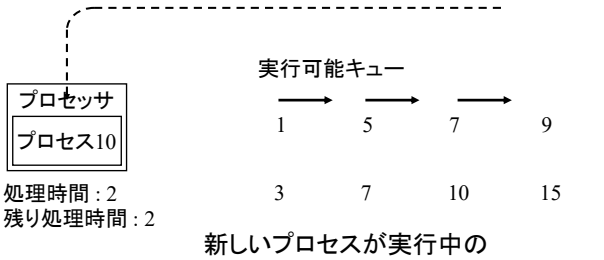
47

スケジューリングアルゴリズム
残余処理時間順(SRT)



48

スケジューリングアルゴリズム
残余処理時間順(SRT)

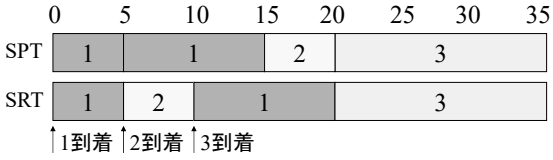


新しいプロセスが実行中の
プロセスよりも処理時間が短いならば
実行中のプロセスと入れ替える

49

スケジューリングアルゴリズム
処理時間順と残余処理時間順

プロセス	到着順位	到着時刻	処理時間
1	1	0	15
2	2	5	5
3	3	10	15



50

スケジューリングアルゴリズム
優先度順

- 優先度順(priority dispatching)
  - 各プロセスに優先度を割り当て、優先度の高いものから処理
    - デバイスハンドラ, 処理時間の短いプロセス, 重要なプロセスの優先度を高くする

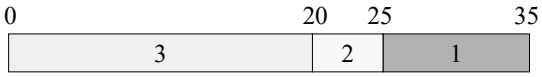
長所
- 優先度の高いプロセスが先に処理される

短所
- 優先度の低いプロセスが無限に待たされる可能性
- プロセスに割り当てられる時間が不公平

51

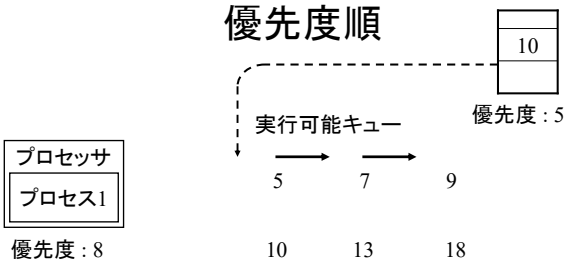
スケジューリングアルゴリズム
優先度順

プロセス	到着順位	処理時間	優先順位
1	1	10	15
2	2	5	12
3	3	20	9



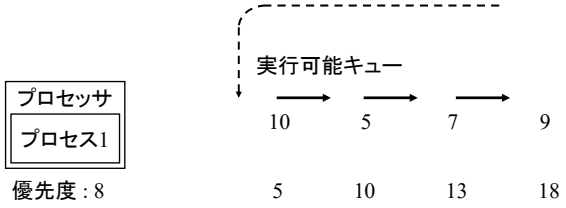
52

スケジューリングアルゴリズム
優先度順



53

スケジューリングアルゴリズム
優先度順



新しいプロセスが来ると
優先度順に並ぶように
実行可能キューに加える
(実行中のプロセスと入れ替える場合もある)

54

スケジューリングアルゴリズム
多重フィードバック

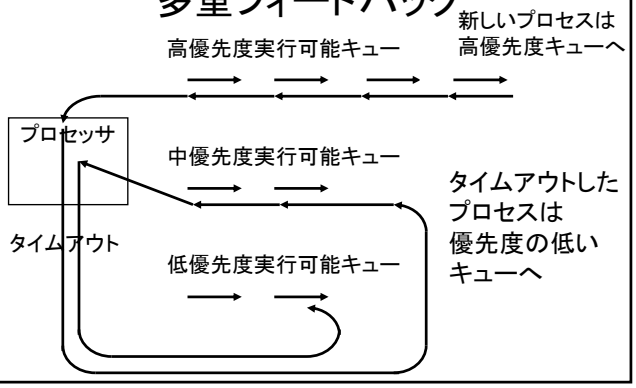
- 多重フィードバック(multiple feedback)
  - キューを多重化し、優先度の高いキューから先に実行する
    - 新しく到着したプロセス
      - ⇒ 優先度の高いキューに
    - 一度処理してタイムアウトしたプロセス
      - ⇒ 優先度の低いキューに

長所
- 処理時間の短いプロセスが先に処理される

短所
- 処理時間の長いプロセスが無限に待たされる可能性

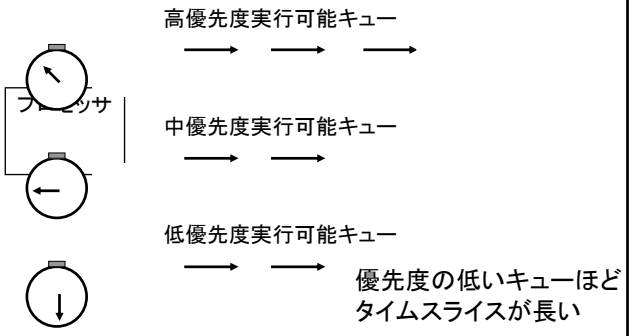
55

スケジューリングアルゴリズム
多重フィードバック



56

スケジューリングアルゴリズム
多重フィードバック



優先度の低いキューほど
タイムスライスが長い

57

スケジューリングの例

スケジューリング例

プロセス	処理時間
1	10
2	5
3	20

到着順は以下の6通り
1→2→3　1→3→2
2→1→3　2→3→1
3→1→2　3→2→1

このプロセスを到着順で処理すると？

58

プロセス	処理時間
1	10
2	5
3	20

スケジューリングの例
到着順の場合

到着順	1番目			2番目			3番目			TA平均
	プロセス	開始	終了	プロセス	開始	終了	プロセス	開始	終了	
1,2,3	1	0	10	2	10	15	3	15	35	20
1,3,2	1	0	10	3	10	30	2	30	35	25
2,1,3	2	0	5	1	5	15	3	15	35	18
2,3,1	2	0	5	3	5	25	1	25	35	22
3,1,2	3	0	20	1	20	30	2	30	35	28
3,2,1	3	0	20	2	20	25	1	25	35	27

59

プロセス	処理時間
1	10
2	5
3	20

スケジューリングの例
到着順の場合

到着順	TA平均
1,2,3	20
1,3,2	25
2,1,3	18
2,3,1	22
3,1,2	28
3,2,1	27

到着順は、処理時間が短いプロセスが先に来るとTA時間が短くなるが、
処理時間が長いプロセスが先に来るとTA時間が長くなる

60

プロセス	処理時間
1	10
2	5
3	20

スケジューリングの例
処理時間順の場合

到着順	1番目 プロセス	1番目 開始	1番目 終了	2番目 プロセス	2番目 開始	2番目 終了	3番目 プロセス	3番目 開始	3番目 終了	TA 平均
1,2,3										
1,3,2										
2,1,3	2	0	5	1	5	15	3	15	35	18
2,3,1										
3,1,2										
3,2,1										

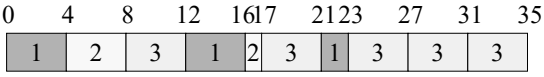
61

プロセス	処理時間
1	10
2	5
3	20

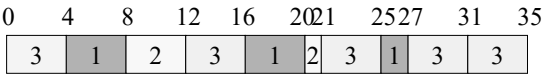
スケジューリングの例
ラウンドロビンの場合

(タイムスライス : 4)

1→2→3の順で到着した場合　平均TA時間 : 25



3→1→2の順で到着した場合　平均TA時間 : 28



62

プロセス	処理時間
1	10
2	5
3	20

スケジューリングの例

到着順	平均TA時間 到着順	処理時間順	ラウンドロビン
1,2,3	20	18	25
1,3,2	25	18	26
2,1,3	18	18	24
2,3,1	22	18	25
3,1,2	28	18	28
3,2,1	27	18	26

63

スケジューリング
横取り(preemption)

- 横取り(preemption)
  - プロセッサを他のプロセスから奪い取ること
- 横取り可能(preemptive)
  - プロセッサが実行中のプロセスを中断して他のプロセスを実行できる
- 横取り不可能(non-preemptive)
  - プロセスが終了するまで他のプロセスは実行できない

64

スケジューリング
横取り(preemption)

スケジューリング法	横取り
到着順	横取り不可能
ラウンドロビン	横取り可能
処理時間順	横取り不可能
残余処理時間順	横取り可能
優先度順	どちらも可
多重フィードバック	横取り可能

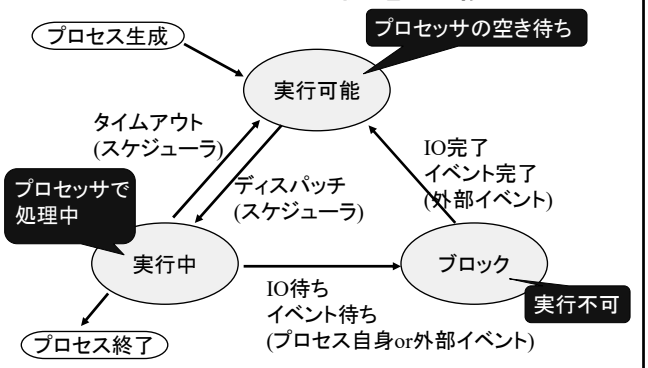
65

まとめ
プロセス記述子

- プロセス記述子 (PCB)
  - プロセスの管理を行う
    - プロセス識別子
    - プロセスの状態
    - スケジューリング情報
    - 資源利用情報
  - カーネル領域に置かれる
  - 処理順にキューに格納される

66

まとめ
プロセスの状態遷移



67

まとめ
スケジューリング

スケジューリング法	長所	短所	横取り
到着順	公平 簡単	処理時間を無視 優先順位を無視	×
ラウンドロビン	公平	優先順位を無視	○
処理時間順	平均処理時間が短い	処理時間の予測が必要 長い処理は待たされる	×
残余処理時間順			○
優先度順	高優先度は早く処理	低優先度は待たされる	△
多重フィードバック	高優先度は早く処理	低優先度は待たされる	○

68