オペレーティングシステム 第4章 スケジューリング

https://github.com/tctsigemura/OSTextBook

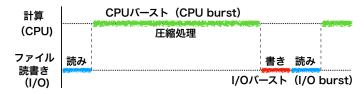
評価基準

- スループット (Throughput)
- ターンアラウンド時間(Turnaround time)
- レスポンス時間(Response time)
- 締め切り (Deadline)
- その他 (公平性, 省エネ, 予測性など)

スケジューリングの目標

コンピュータの種類	重視する性能
メインフレーム(バッチ処理)	スループット,ターンアラウンド時間
ネットワークサーバ	レスポンス時間,スループット
デスクトップパソコン	レスポンス時間
モバイルデバイス	レスポンス時間,省エネルギー
組込み制御	締め切り

CPUバウンドプロセス



- 動画圧縮の例
- I/O バウンドプロセス (エクセル)

I/O バウンドプロセス



• スプレッドシートの例

FCFS スケジューリング(1)

プロセス	到着時刻	CPU バースト時間 (ms)			
P_1	0	100	-		
P_2	0	20			
P_3	0	10			
		P_1		P_2	P_3
0			10	00	120 130

- *P*₁, *P*₂, *P*₃ の順に実行
- 平均ターンアラウンド時間((100 + 120 + 130)/3 = 117 ms)

FCFS スケジューリング(2)

プロセス	到着時刻	CPU バースト時間 (ms)	
$\overline{P_1}$	0	100	-
P_2	0	20	
P_3	0	10	
	P_2 P_3	P_1	
0	20 3	0	130

- P₂, P₃, P₁の順に実行
- 平均ターンアラウンド時間 ((20+30+130)/3=60 ms)

SJF スケジューリング

プロセス	到着時刻	CPU バースト時間 (ms)	
P_1	0	100	-
P_2	0	20	
P_3	0	10	
P_3	P_2	P_1	
0	10 3	0	130

• 平均ターンアラウンド時間 ((10+30+130)/3=57 ms)

SJF スケジューリング(比較のため)

プロセス	到着時刻	CPU バース	ト時間 (ms)		
P_1	0	60)		
P_2	10	40)		
P_3	60	30)		
	P_1		P_3	P_2	
0		60	0	90	130

- SJF はプリエンプションなし
- 平均ターンアラウンド時間 $(((60-0)+(90-10)+(130-60))/3=70~{\rm ms})$

SRTF スケジューリング

プロセス	到着時刻	CPUバー	スト時間 (ms)		
P_1	0		60	_	
P_2	10		40		
P_3	60		30		
P_1	P_2	P_1	P_1 P_3		P_1
0	10	50	60	90	130

- SRTF はプリエンプションあり
- 平均ターンアラウンド時間 (((130-0)+(50-10)+(90-60))/3=67 ms)

RRスケジューリング(1)

プロセ	ス	到着	時刻	CI	PU 🗡	ニース	ト時	間 (m	s)					
P_1			0			6	0							
P_2		1	.0			4	0							
P_3		6	50			3	0							
	P_1	P_2	P_1	P_2	P_1	P_2	P_1	P_3	P_2	P_1	P_3	P_1	P_3	
0	1	0 2	0 3	0 4	.0 5	0 6	0 7	0 8	0 9	0 10	00 1:	10 12	20 13	30

- クォンタムタイム= 10ms
- 平均ターンアラウンド時間 (((120-0)+(90-10)+(130-60))/3=90)

RRスケジューリング(2)

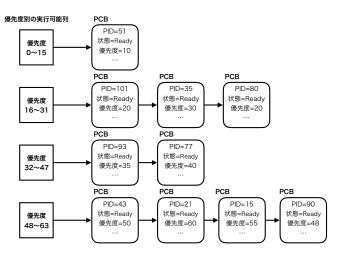
プロセス	到着時刻	CPUバ	バースト時間 (r	ns)			
P_1	0		60				
P_2	10		40				
P_3	60		30				
	P_1		P_2		P_1	P_3	
0		5	50	9	00 10	00	130

- クォンタムタイム= 50ms
- 平均ターンアラウンド時間 (((100-0)+(90-10)+(130-60))/3=83 ms)

優先度順スケジューリング

- 静的・動的
- スタベーション
- ・エージング

FB スケジューリング



• エージング

TacOS のスケジューラ

```
public void schProc(PCB proc) {
    int r = setPri(DI|KERN);
                                                // 割り込み禁止、カーネル
    int enice = proc.enice;
    PCB head = readyQueue.next;
                                                // 実行可能列から
5
                                                    優先度がより低い
    while (head.enice <= enice)
6
      head = head.next:
                                                // プロセスを探す
                                                // 見つけたプロセスの
    insProc(head,proc);
8
    setPri(r):
                                                    直前に挿入する
9
                                                // 割り込み状態を復元する
```

TacOS の実行可能列(参考)

- yield
- dispatch
- 実行可能列

