OS 演習課題第 3 回

坪井正太郎（101830245）

2020 年 12 月 28 日

問題 1

1

(a)

応答時間

- A:13
- B:20
- C:23

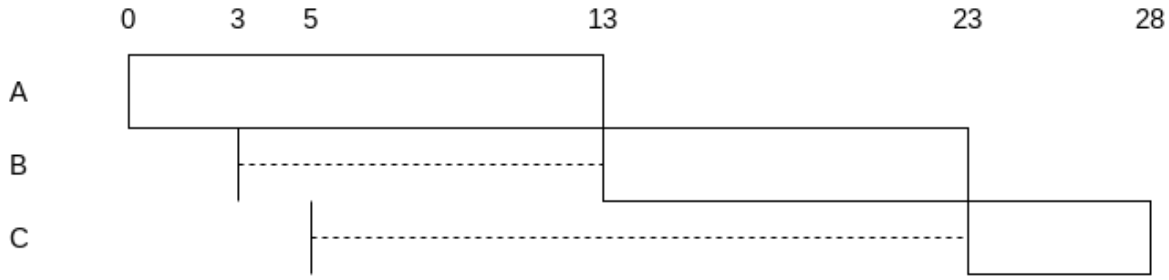

図 1　到着順タイムチャート

(b)

応答時間

- A:18
- B:25
- C:5

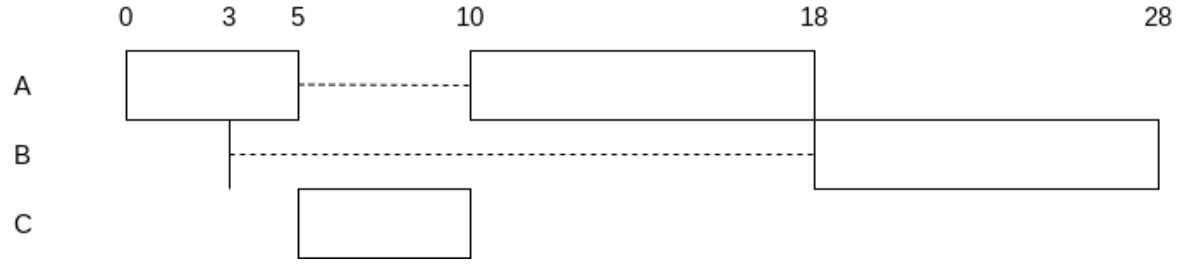

図 2　実行時間最短タイムチャート

(c)

応答時間

- A:27
- B:25
- C:15

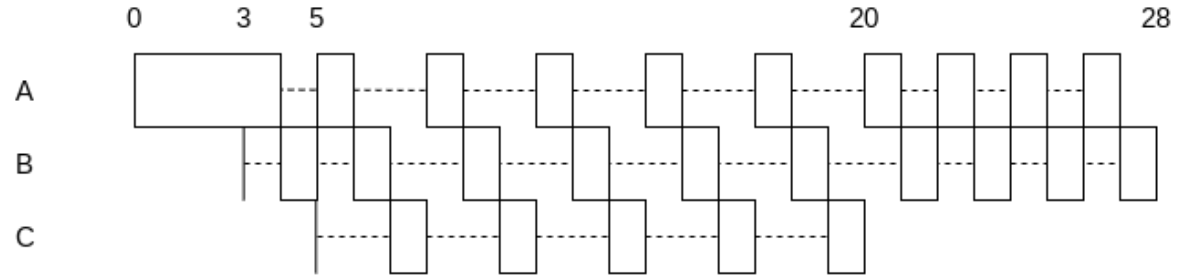

図 3　ラウンドロビンタイムチャート

1

- 到着順
  - 単純な制御で、オーバーヘッドがないが、実行に時間がかかるプロセスが合った場合、その後に入るプロセスの応答が悪化する。
  - それぞれのプロセスの実行時間が短い場合に、実装が単純なため有利である。
- 実行時間最短
  - プロセスの実行が完了するまでの時間が平均して最小になるが、それぞれのプロセスの実行時間を事前に知ることが難しい。
  - 各プロセスの実行時間が分かっている機器の制御では有利にはたらく。
- ラウンドロビン
  - 公平に実行されるが、プロセスの切替によるオーバーヘッドが大きい。
  - 多数の利用者が利用するシステムでは、リソースを公平に分配できるので都合がいい。

問題 2

(a)

ソースコード 1　fifo.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct que {
5    int remain;
6    int index;
7    struct que *next;
8    struct que *prev;
9  } Que;
10
11 Que *Top = NULL;
12 Que *End = NULL;
13
14 void push(int remain, int index) {
15   Que *t = malloc(sizeof(Que));
16   t->remain = remain;
17   t->index = index;
18
19   if (Top == NULL) {
20     Top = t;
21     End = t;
22   } else {
23     t->next = Top;
24     Top->prev = t;
25     Top = t;
26   }
27 }
28
29 Que *pop() {
30   Que *t = End;
31   if (End == Top) {
32     Top = NULL;
33     End = NULL;
34   } else {
35     End = End->prev;
36   }
37
38   return t;
39 }
```

```
40
41 int main(int argc, char const *argv[]) {
42   int n;
43   scanf("%d", &n);
44
45   int *process_time = malloc(sizeof(int) * n);
46   int *arrival = malloc(sizeof(int) * n);
47   int *result = malloc(sizeof(int) * n);
48
49   for (int i = 0; i < n; i++) {
50     scanf("%d %d", &process_time[i], &arrival[i]);
51   }
52
53   printf("process start\n");
54
55   Que *prosessing = NULL;
56   int finished = 0;
57   int time = -1;
58
59   while (finished < n) {
60     time++;
61     printf("%d : ", time);
62     for (int i = 0; i < n; i++)
63       if (arrival[i] == time) push(process_time[i], i);
64
65     if (prosessing == NULL) prosessing = pop();
66
67     if (prosessing != NULL) {
68       prosessing->remain--;
69       printf("processing %d, remain %d\n", prosessing->index,
70              prosessing->remain);
71
72       if (prosessing->remain == 0) {
73         // プロセスが完了したら応答時間を計測
74         finished++;
75         result[prosessing->index] = time + 1 - arrival[prosessing->index];
76         prosessing = pop();
77       }
78     }
79   }
80
81   printf("\n");
82   for (int i = 0; i < n; i++) printf("process %d : %d\n", i, result[i]);
83   printf("\ntotal : %d\n", time + 1);
84
85   return 0;
```

```
86 }
```

実行コマンドとその結果

ソースコード 2　fifo.c 実行結果

```
1  $ gcc fifo.c
2  $ ./a.out
3  $ 3
4  $ 13 0
5  $ 10 3
6  $ 5 5
7  process start
8  0 : processing 0, remain 12
9  1 : processing 0, remain 11
10 2 : processing 0, remain 10
11 3 : processing 0, remain 9
12 4 : processing 0, remain 8
13 5 : processing 0, remain 7
14 6 : processing 0, remain 6
15 7 : processing 0, remain 5
16 8 : processing 0, remain 4
17 9 : processing 0, remain 3
18 10 : processing 0, remain 2
19 11 : processing 0, remain 1
20 12 : processing 0, remain 0
21 13 : processing 1, remain 9
22 14 : processing 1, remain 8
23 15 : processing 1, remain 7
24 16 : processing 1, remain 6
25 17 : processing 1, remain 5
26 18 : processing 1, remain 4
27 19 : processing 1, remain 3
28 20 : processing 1, remain 2
29 21 : processing 1, remain 1
30 22 : processing 1, remain 0
31 23 : processing 2, remain 4
32 24 : processing 2, remain 3
33 25 : processing 2, remain 2
34 26 : processing 2, remain 1
35 27 : processing 2, remain 0
36
37 process 0 : 13
38 process 1 : 20
39 process 2 : 23
40
41 total : 28
```

(c)

ソースコード 3　roundrobin.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct que {
5    int remain;
6    int index;
7    struct que *next;
8    struct que *prev;
9  } Que;
10
11 Que *Top = NULL;
12 Que *End = NULL;
13
14 void push(int remain, int index) {
15   Que *t = malloc(sizeof(Que));
16   t->remain = remain;
17   t->index = index;
18
19   if (Top == NULL) {
20     Top = t;
21     End = t;
22   } else {
23     t->next = Top;
24     Top->prev = t;
25     Top = t;
26   }
27 }
28
29 void pushP(Que *que) {
30   if (Top == NULL) {
31     Top = que;
32     End = que;
33   } else {
34     que->next = Top;
35     Top->prev = que;
36     Top = que;
37   }
38 }
39
40 Que *pop() {
41   Que *t = End;
```

```
42    if (End == Top) {
43      Top = NULL;
44      End = NULL;
45    } else {
46      End = End->prev;
47    }
48
49    return t;
50  }
51
52  int main(int argc, char const *argv[]) {
53    int n;
54    scanf("%d", &n);
55
56    int *process_time = malloc(sizeof(int) * n);
57    int *arrival = malloc(sizeof(int) * n);
58    int *result = malloc(sizeof(int) * n);
59
60    for (int i = 0; i < n; i++) {
61      scanf("%d %d", &process_time[i], &arrival[i]);
62    }
63
64    printf("process start\n");
65
66    Que *prosessing = NULL;
67    int finished = 0;
68    int time = -1;
69
70    while (finished < n) {
71      time++;
72      printf("%d : ", time);
73      for (int i = 0; i < n; i++)
74        if (arrival[i] == time) push(process_time[i], i);
75
76      if (prosessing == NULL) prosessing = pop();
77
78      if (prosessing != NULL) {
79        prosessing->remain--;
80        printf("processing %d, remain %d\n", prosessing->index,
81               prosessing->remain);
82
83        if (prosessing->remain == 0) {
84          // プロセスが完了したら応答時間を計測
85          finished++;
86          result[prosessing->index] = time + 1 - arrival[prosessing->index];
87        } else {
```

```
88          // 一回計算したらキューに入ってるやつをとりだしてくる
89          pushP(prosessing);
90        }
91        prosessing = pop();
92      }
93    }
94
95    printf("\n");
96    for (int i = 0; i < n; i++) printf("process %d : %d\n", i, result[i]);
97    printf("\ntotal : %d\n", time + 1);
98
99    return 0;
100 }
```

ソースコード 4　roundrobin.c 実行結果

```
1   $ gcc roundrobin.c
2   $ ./a.out
3   $ 3
4   $ 13 0
5   $ 10 3
6   $ 5 5
7   process start
8   0 : processing 0, remain 12
9   1 : processing 0, remain 11
10  2 : processing 0, remain 10
11  3 : processing 0, remain 9
12  4 : processing 1, remain 9
13  5 : processing 0, remain 8
14  6 : processing 1, remain 8
15  7 : processing 2, remain 4
16  8 : processing 0, remain 7
17  9 : processing 1, remain 7
18  10 : processing 2, remain 3
19  11 : processing 0, remain 6
20  12 : processing 1, remain 6
21  13 : processing 2, remain 2
22  14 : processing 0, remain 5
23  15 : processing 1, remain 5
24  16 : processing 2, remain 1
25  17 : processing 0, remain 4
26  18 : processing 1, remain 4
27  19 : processing 2, remain 0
28  20 : processing 0, remain 3
29  21 : processing 1, remain 3
30  22 : processing 0, remain 2
31  23 : processing 1, remain 2
```

```
24 : processing 0, remain 1
25 : processing 1, remain 1
26 : processing 0, remain 0
27 : processing 1, remain 0

process 0 : 27
process 1 : 25
process 2 : 15

total : 28
```

問題 1 の結果と比較しても、正しく出力されている