

オペレーティングシステム中間課題

B223349 西澤英志

2023 年 2 月 6 日

1 プログラムの説明

1.1 作成したプログラム

哲学者の食事問題を解くプログラムを作成した。

1.2 クリティカルセクションと排他制御について行ったこと

今回作成したプログラムでは、フォークをリソースとして、哲学者がリソースを使用するという形になっている。哲学者がフォークを取得する部分をクリティカルセッションとしていて、優先で取得できるフォークがすでに取得されている場合は使用可能になるまで待機状態になる。

本プログラムで排他制御を行わなかった場合、それぞれの哲学者は以下のように振る舞う。

1. 左のフォークが得られるまで思考し、フォークを取得する。
2. 右のフォークが得られるまで思考し、フォークを取得する。
3. 食事をする。(15秒のランダムな時間待機)
4. 右のフォークを置く。
5. 左のフォークを置く。

この場合、全哲学者が左のフォークを持ち、右のフォークが得られなくなるのでデッドロック状態になる。

この問題の解決方法として、本プログラムでは哲学者の位置の応じて、右もしくは左のフォークを優先するようにした。具体的には、哲学者の番号が奇数のときは左のフォークを先に取得するようにし、偶数のときは右のフォークを先に取得するようにした。これにより、全哲学者が左のフォークを持ち、右のフォークが得られなくなることがなくなるのでデッドロック状態を回避することが可能になる。

2 動作環境

表 1 動作環境

OS	ubuntu22.04 LTS
CPU	Intel® Core™ i7-8700K CPU @ 3.70GHz × 12
メモリ	32.0GB
コンパイラ	g++ version 11.3.0
使用言語	C++17

3 実行手順

プログラムのコンパイルコマンドを以下に示す .

	1 コンパイル
1	<code>g++ -Wall main.cpp -pthread -std=c++17</code>

以下のコマンドで実行した .

	2 実行コマンド
1	<code>./a.out</code>

4 実行結果

ループ数 3(食事の回数が 3 回) で哲学者が 5 人の条件でプログラムの実行したときの結果を以下に示す .

	3 実行結果
1	<code>f11@desktop-f11<TUThumbleERv3>:~/Git/The_dining_philosophers\$./a.out</code>
2	<code>start</code>
3	<code>fork 1</code>
4	<code>fork 2</code>
5	<code>fork 3</code>
6	<code>fork 4</code>
7	<code>fork 5</code>
8	<code>Philosopher 1 is reading..</code>
9	<code>Philosopher 2 is reading..</code>
10	<code>Philosopher 3 is reading..</code>
11	<code>Philosopher1 thinking</code>
12	<code>Philosopher2 thinking</code>
13	<code>Philosopher 4 is reading..</code>
14	<code>Philosopher3 thinking</code>
15	<code>Philosopher 5 is reading..</code>
16	<code>Philosopher5 thinking</code>
17	<code>Philosopher4 thinking</code>
18	<code>Philosopher2 get fork2 DONE</code>
19	<code>Philosopher2 get fork1 DONE</code>
20	<code>Philosopher2 is eating</code>
21	<code>Philosopher3 get fork2 DONE</code>
22	<code>Philosopher3 get fork3 DONE</code>
23	<code>Philosopher3 is eating</code>
24	<code>Philosopher1 get fork5 DONE</code>
25	<code>Philosopher1 get fork1 DONE</code>
26	<code>Philosopher1 is eating</code>
27	<code>Philosopher2 has finished eating</code>
28	<code>Philosopher2 put fork1 DONE</code>
29	<code>Philosopher2 put fork2 DONE</code>
30	<code>Philosopher2 thinking</code>
31	<code>Philosopher5 get fork4 DONE</code>
32	<code>Philosopher5 get fork5 DONE</code>

33 Philosopher5 is eating
 34 Philosopher4 get fork4 DONE
 35 Philosopher4 get fork3 DONE
 36 Philosopher4 is eating
 37 Philosopher1 has finished eating
 38 Philosopher1 put fork1 DONE
 39 Philosopher1 put fork5 DONE
 40 Philosopher1 thinking
 41 Philosopher2 get fork2 DONE
 42 Philosopher2 get fork1 DONE
 43 Philosopher2 is eating
 44 Philosopher3 has finished eating
 45 Philosopher3 put fork3 DONE
 46 Philosopher3 put fork2 DONE
 47 Philosopher3 thinking
 48 Philosopher2 has finished eating
 49 Philosopher2 put fork1 DONE
 50 Philosopher2 put fork2 DONE
 51 Philosopher2 thinking
 52 Philosopher4 has finished eating
 53 Philosopher4 put fork3 DONE
 54 PhilosopherPhilosopher4 put fork4 DONE
 55 Philosopher4 thinking
 56 1 get fork5 DONE
 57 Philosopher1 get fork1 DONE
 58 Philosopher1 is eating
 59 Philosopher4 get fork4 DONE
 60 Philosopher4 get fork3 DONE
 61 Philosopher4 is eating
 62 Philosopher5 has finished eating
 63 Philosopher5 put fork5 DONE
 64 Philosopher5 put fork4 DONE
 65 Philosopher5 thinking
 66 Philosopher2 get fork2 DONE
 67 Philosopher2 get fork1 DONE
 68 Philosopher2 is eating
 69 Philosopher1 has finished eating
 70 Philosopher1 put fork1 DONE
 71 Philosopher1 put fork5 DONE
 72 Philosopher1 thinking
 73 Philosopher3 get fork2 DONE
 74 Philosopher3 get fork3 DONE
 75 Philosopher3 is eating
 76 Philosopher2 has finished eating
 77 Philosopher2 put fork1 DONE
 78 Philosopher2 put fork2 DONE
 79 Philosopher4 has finished eating
 80 Philosopher4 put fork3 DONE
 81 Philosopher4 put fork4 DONE
 82 Philosopher4 thinking

```
83     Philosopher1 get fork5 DONE
84     Philosopher1 get fork1 DONE
85     Philosopher1 is eating
86     Philosopher5 get fork4 DONE
87     Philosopher5 get fork5 DONE
88     Philosopher5 is eating
89     Philosopher1 has finished eating
90     Philosopher1 put fork1 DONE
91     Philosopher1 put fork5 DONE
92     Philosopher4 get fork4 DONE
93     Philosopher4 get fork3 DONE
94     Philosopher4 is eating
95     Philosopher3 has finished eating
96     Philosopher3 put fork3 DONE
97     Philosopher3 put fork2 DONE
98     Philosopher3 thinking
99     Philosopher3 get fork2 DONE
100    Philosopher3 get fork3 DONE
101    Philosopher3 is eating
102    Philosopher5 has finished eating
103    Philosopher5 put fork5 DONE
104    Philosopher5 put fork4 DONE
105    Philosopher5 thinking
106    Philosopher3 has finished eating
107    Philosopher3 put fork3 DONE
108    Philosopher3 put fork2 DONE
109    Philosopher4 has finished eating
110    Philosopher4 put fork3 DONE
111    Philosopher4 put fork4 DONE
112    Philosopher5 get fork4 DONE
113    Philosopher5 get fork5 DONE
114    Philosopher5 is eating
115    Philosopher5 has finished eating
116    Philosopher5 put fork5 DONE
117    Philosopher5 put fork4 DONE
118    finish
```

以上の結果より、デッドロック状態に陥ることがなく 1 から 5 番までのすべての哲学者が食事を 3 回終えることができていることがわかる。

5 工夫した点・感想

セマフォと fork の取得と fork を置く関数を struct にし、哲学者を class にまとめることでグローバル変数やグローバル関数を使わないようなプログラムになるよう工夫した。実際にデッドロック状態になるようなプログラムを作ってから、デッドロックが起きないように改良をしていったので哲学者の食事問題についての理解が深まった。

今回の課題でスレッドプログラムの作成方法に関して知識を少し深めることができた。他の IPC 問題についても実装してみてスレッドプログラムについての理解をより深めていきたいと感じた。

6 参考文献

参考文献

- [1] Akira Takahashi ,” std::thread”,cpprefjp - C++ 日本語リファレンス,<https://cpprefjp.github.io/reference/thread/thread.html>,2022 年 11 月 29 日.