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．(1 5 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トロック状態になる.

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ト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       g++ -Wall main.cpp -pthread -std=c++17
```

以下のコマンドで実行した.

2 実行コマンド

```
1       ./a.out
```

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

```
        Philosopher5 is eating
        Philosopher4 get fork4 DONE
        Philosopher4 get fork3 DONE
        Philosopher4 is eating
        Philosopher1 has finished eating
        Philosopher1 put fork1 DONE
        Philosopher1 put fork5 DONE
        Philosopher1 thinking
        Philosopher2 get fork2 DONE
        Philosopher2 get fork1 DONE
        Philosopher2 is eating
        Philosopher3 has finished eating
        Philosopher3 put fork3 DONE
        Philosopher3 put fork2 DONE
        Philosopher3 thinking
        Philosopher2 has finished eating
        Philosopher2 put fork1 DONE
        Philosopher2 put fork2 DONE
        Philosopher2 thinking
        Philosopher4 has finished eating
        Philosopher4 put fork3 DONE
        PhilosopherPhilosopher4 put fork4 DONE
        Philosopher4 thinking
        1 get fork5 DONE
        Philosopher1 get fork1 DONE
        Philosopher1 is eating
        Philosopher4 get fork4 DONE
        Philosopher4 get fork3 DONE
        Philosopher4 is eating
        Philosopher5 has finished eating
        Philosopher5 put fork5 DONE
        Philosopher5 put fork4 DONE
        Philosopher5 thinking
        Philosopher2 get fork2 DONE
        Philosopher2 get fork1 DONE
        Philosopher2 is eating
        Philosopher1 has finished eating
        Philosopher1 put fork1 DONE
        Philosopher1 put fork5 DONE
        Philosopher1 thinking
        Philosopher3 get fork2 DONE
        Philosopher3 get fork3 DONE
        Philosopher3 is eating
        Philosopher2 has finished eating
        Philosopher2 put fork1 DONE
        Philosopher2 put fork2 DONE
        Philosopher4 has finished eating
        Philosopher4 put fork3 DONE
        Philosopher4 put fork4 DONE
        Philosopher4 thinking
```

```
        Philosopher1 get fork5 DONE
        Philosopher1 get fork1 DONE
        Philosopher1 is eating
        Philosopher5 get fork4 DONE
        Philosopher5 get fork5 DONE
        Philosopher5 is eating
        Philosopher1 has finished eating
        Philosopher1 put fork1 DONE
        Philosopher1 put fork5 DONE
        Philosopher4 get fork4 DONE
        Philosopher4 get fork3 DONE
        Philosopher4 is eating
        Philosopher3 has finished eating
        Philosopher3 put fork3 DONE
        Philosopher3 put fork2 DONE
        Philosopher3 thinking
        Philosopher3 get fork2 DONE
        Philosopher3 get fork3 DONE
        Philosopher3 is eating
        Philosopher5 has finished eating
        Philosopher5 put fork5 DONE
        Philosopher5 put fork4 DONE
        Philosopher5 thinking
        Philosopher3 has finished eating
        Philosopher3 put fork3 DONE
        Philosopher3 put fork2 DONE
        Philosopher4 has finished eating
        Philosopher4 put fork3 DONE
        Philosopher4 put fork4 DONE
        Philosopher5 get fork4 DONE
        Philosopher5 get fork5 DONE
        Philosopher5 is eating
        Philosopher5 has finished eating
        Philosopher5 put fork5 DONE
        Philosopher5 put fork4 DONE
        finish
```

以上の結果より，デット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．実際にデットロック状態をになるようなプログラムを作ってから，デット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.