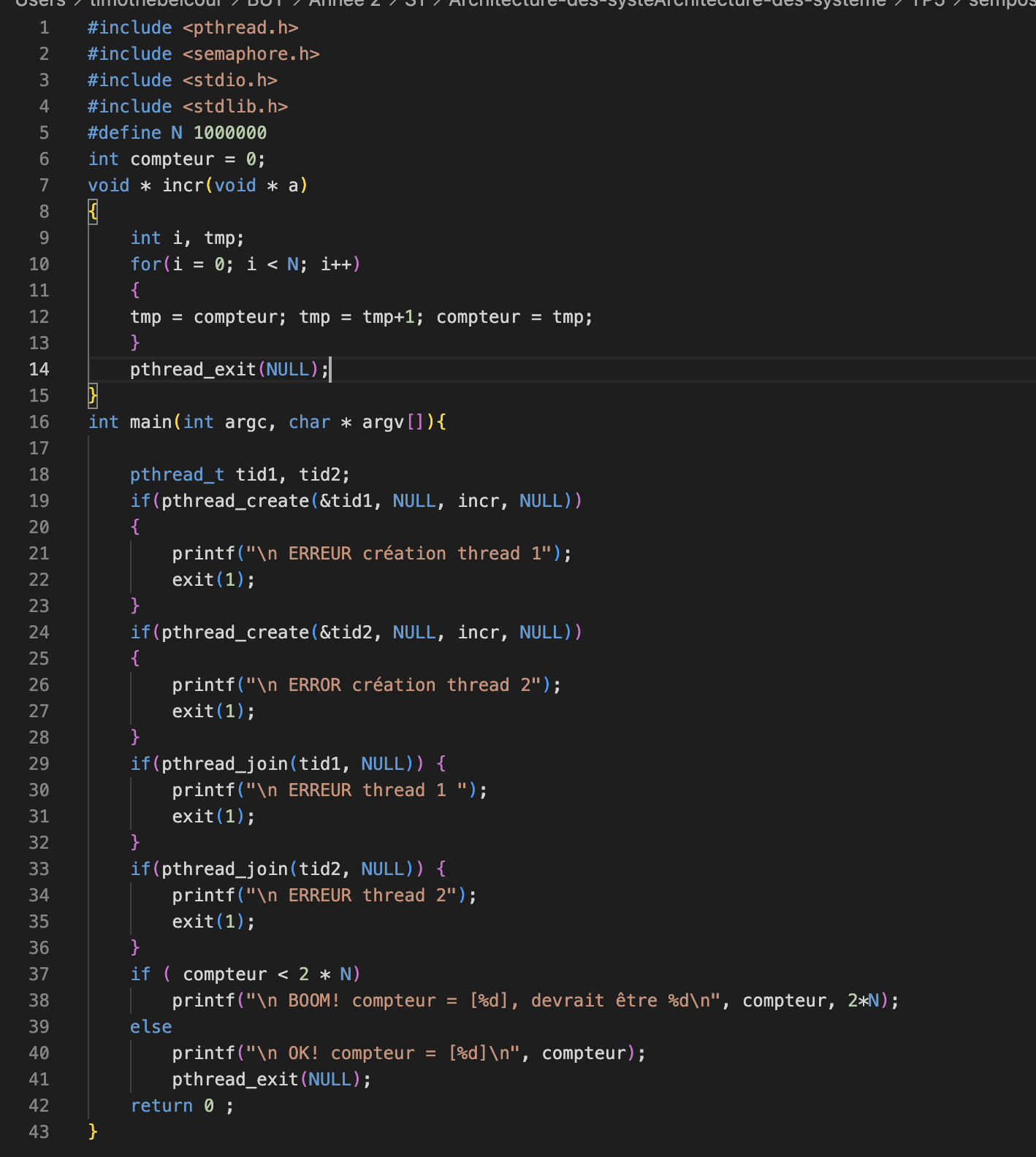
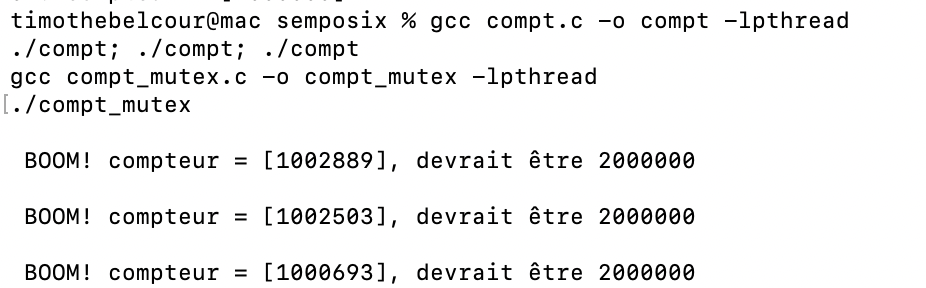
Synchronisation de Threads avec les

Sémaphores POSIX

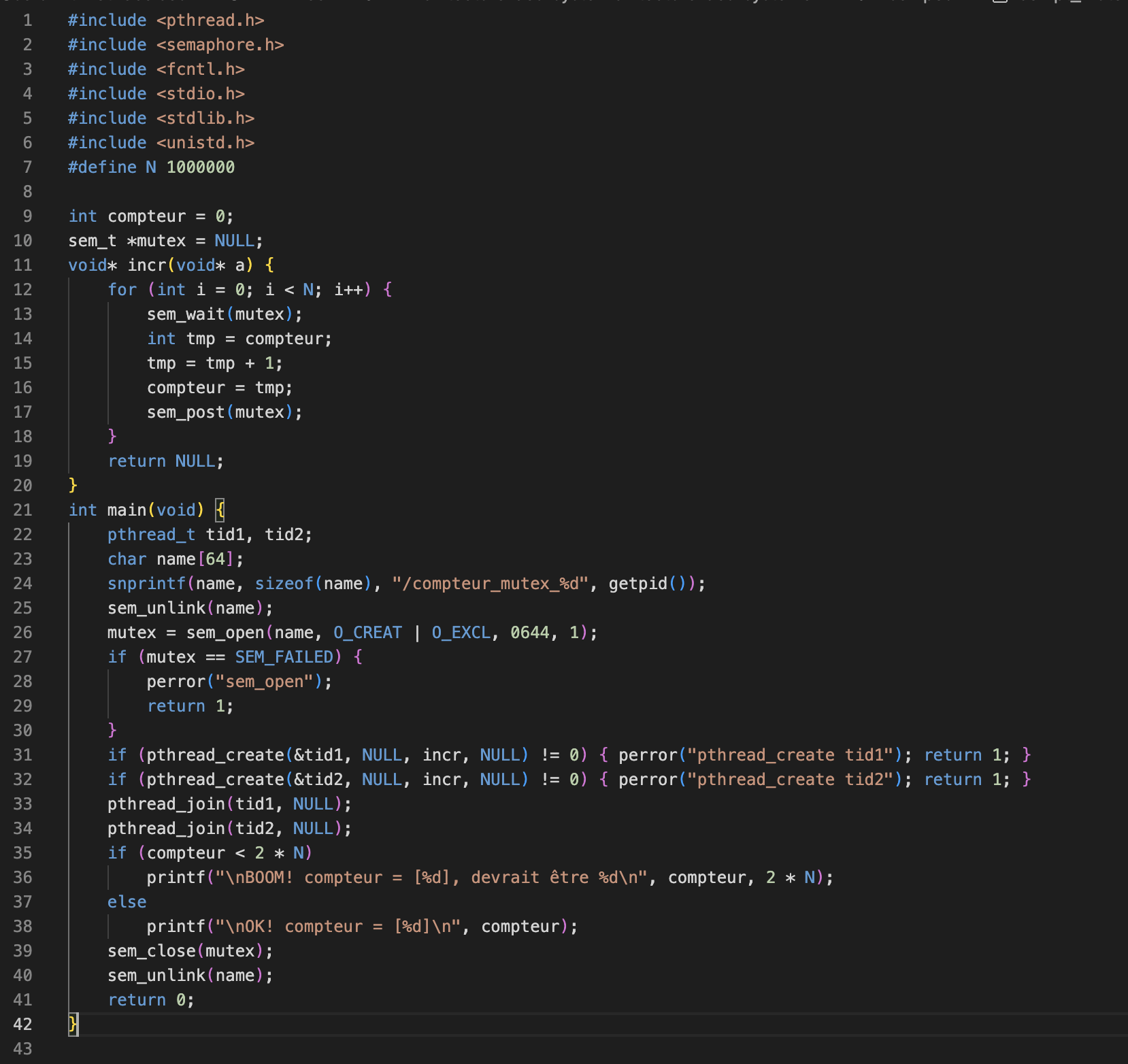
Exercice 1 :

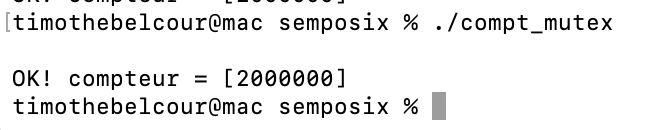




Le programme lance deux threads qui essaient d’augmenter la même variable compteur sans se coordonner.  
Chacun fait : lire la valeur, ajouter 1, réécrire.  
Comme ces trois étapes ne sont pas faites d’un seul bloc, l’autre thread peut s’intercaler entre-temps.  
Exemple : les deux lisent 100, chacun calcule 101 ; le premier écrit 101, puis le second réécrit à nouveau 101.  
On a fait deux incréments, mais la valeur n’a augmenté qu’une fois donc des incréments sont perdus.  
C’est pour cela que le résultat change à chaque exécution et reste en dessous de 2×N (messages “BOOM!”).

Exercice 2 :





On place la section critique (read–modify–write sur compteur) entre sem\_wait et sem\_post d’un sémaphore POSIX, ce qui impose l’exclusion mutuelle : un seul thread à la fois modifie compteur. Dès lors, aucun incrément n’est perdu et la sortie devient déterministe, toujours égale à 2×N (“OK! compteur = 2000000”).

Pour conclure

./compt : résultats variables et < 2×N (BOOM) = attendu (course critique).

./compt\_mutex : toujours 2×N (OK) = attendu (section critique protégée).