The math behind 42's Philosophers

by mofrim aka fmaurer

Abstract

This document is about the math behind 42's Philosophers Project. This is how i came to understand it while i was working on the project. So there might still be something left or even wrong.

The Basics

Let

$$\begin{aligned} t_{\text{die}} &:= \texttt{time to die}, \\ t_{\text{eat}} &:= \texttt{time to eat}, \\ t_{\text{sleep}} &:= \texttt{time to sleep}, \\ n &:= \texttt{number of philosophers aka philno}. \end{aligned}$$

Most evidently philosophers must die if

$$t_{\text{die}} \leq t_{\text{eat}} \quad \text{or} \quad t_{\text{die}} \leq t_{\text{sleep}}.$$

The normal way of philosopher life goes like:

$$\texttt{eat} \rightarrow \texttt{sleep} \rightarrow \texttt{think} \rightarrow \texttt{repeat}.$$

So it is also immediately clear that philosophers will die if

$$t_{\text{die}} \leq t_{\text{eat}} + t_{\text{sleep}},$$

because they will always at least have to eat **and** sleep until they can start their next meal. So if a philo starts at time 0 eats for t_{eat}, sleeps for t_{sleep}, even skips thinking completely, his time consumption since last meal start will be

$$t_{\text{eat}} + t_{\text{sleep}} = t_{\text{die}} \quad \Rightarrow \quad \text{☠}.$$

So we have the first contraint rule on the possible combinations of parameters for the Philosophers Problem:

Rule i:

$$\boxed{t_{\text{die}} \geq t_{\text{eat}} + t_{\text{sleep}}}$$

If the cdmline args to our `philo` programm do not respect this rule, it is certain that our philos will die.

For further insights we have to look at the 2 possible cases independently: even and uneven number of philos. The uneven case will turn out to be much more interesting....

Even number of Philos

The big question: *Is* Rule i *already enough?* Let's have a look at the following two figures showing the life or death of 2 philos:

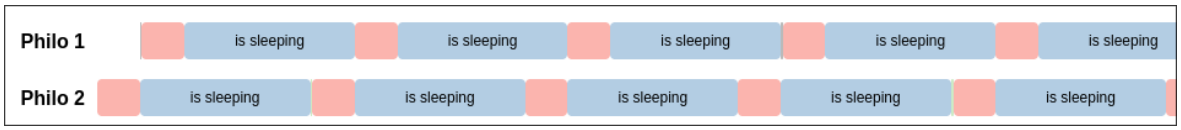

Figure 1: `./philo 2 120 20 80`

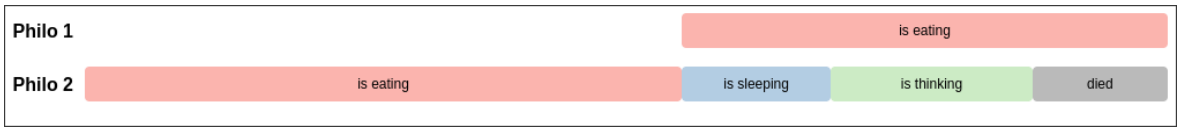

Figure 2: `./philo 2 120 80 20`

The only difference is that t_{sleep} and t_{eat} have been switched. In fig. Everything goes fine, but in fig. 2nd Philo dies only after one meal. This happens because he dies, while the Philo 1 is still eating. This would not have happened if

Rule ii:

$$\boxed{t_{\text{die}} \geq 2 \cdot t_{\text{eat}}}$$

But is this enough now?! Of course not! Take for example a session with [`./philo 2 40 20 80`] the condition $t_{\text{die}} > 2 \cdot t_{\text{eat}}$ is still met, but the Philos die in their sleep. So, the safe Rule for an even number of philos can only be:

Rule I: If $n \in 2\mathbb{Z}$

$$t_{\text{die}} > \max\left(2 \cdot t_{\text{eat}},\ t_{\text{eat}} + t_{\text{sleep}}\right).$$

Wait, what? Why max? Take again the example with

[`./philo 2 120 20 80`] and [`./philo 2 120 80 20`]

In the former the conditions are both met, $t_{\text{die}} > 2 \cdot t_{\text{eat}}$ and $t_{\text{die}} > t_{\text{eat}} + t_{\text{sleep}}$. In the latter Rule ii is violated. If we did not take the max of both as our value for t_{die} then one will always be violated. So, this is the only way. It follows, that even a smaller value of $t_{\text{die}} = 101$ will work for the example [`./philo 2 120 20 80`].

Uneven philno

Now what's the difference with an uneven number of philos? The fundamental difference can be shown by the example of 3 philosophers. The same pattern will be true for all odd n.

With $n = 3$ we also have 3 forks on the table. As a consequence only one philo can start eating, while the 2 others have to wait. Let's call the philos A, B and C. So, philo A eats, B & C wait. But then again only one of B & C can eat next! Say, that B is eating next while C is still waiting for his 2nd fork (technically: his `pthread_mutex_lock` is blocking). Finally after B has finished his meal, C will eat. But at this time the `last_meal_start` time of A will be $t = 0$, which is the beginning of the simulation **and** he lived already $2 \cdot t_{\text{eat}}$ without starting another meal. And now C still has to eat! So, if now $t_{\text{die}} \leq 3 \cdot t_{\text{eat}}$, philo A will die during C's meal. Of course Rule i still must be obeyed because stuff like `./philo 5 601 200 401` will still kill your philos. So it follows in the uneven case:

Rule II: If $n \in 2\mathbb{Z} + 1$

$$t_{\text{die}} > \max\left(3 \cdot t_{\text{eat}},\ t_{\text{eat}} + t_{\text{sleep}}\right).$$

I only gave the example for $n = 3$ but it is easy to see that with an odd philno there will always 2 philos waiting at the beginning of simulation not eating in the first round of meals. In the 2nd round of meals only one of them is able to eat because they share one fork. This leads to the neighbor of the philo who still can't eat in round 2, but has eaten in the 1st round, not havening his next meal before $3 \cdot t_{\text{eat}}$ after his first meal started, qed.

Syncronization

Now what if all philos grab exactly one fork at the very beginning or any later moment in the simulation? *Deadlock! Philo-mass-starving!*

The answer to this problem is: Syncronization.

Even philno

Here one easy syncro is very obvious: Just let half of the philos sleep at beginning of simulation for a little while, then the other half will have just enough forks to eat. Take $t_{\text{eat}}/2$ f.ex., that's it.

Odd philno

For one philo there is not much hope, having only one fork... ☠

But also for odd $n > 1$ there is at least one major pitfall (which, in my case, only in the bonus part happened to be more likely) and this is due to the

> **Fact:** The POSIX standard does not guarantee that `pthread_mutex_lock` or `sem_wait` work with the **FIFO = First In First Out** principle. Meaning: it is not guranteed that the first process who called `pthread_mutex_lock`/`sem_wait` will be the first to be unblocked (in that case increment the semaphore or get to lock the mutex)! For deeper info see: https://github.com/torvalds/linux/.../Documentation/locking/mutex-design.rst

In fig.3 you can find a diagram showing such an event where normally it should have been philo 1's turn after 2 and 3 ate, but the scheduler decided differently...

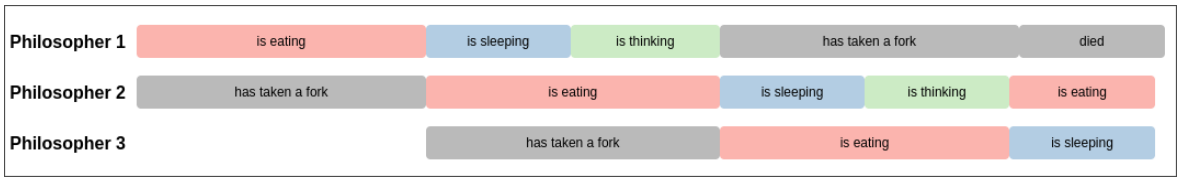

Figure 3: A deadly meal...

So, what is the solution to this dilemma???

...to guarantee a fixed well chosen time to think t_{think} after each meal in case we have an odd number of philos!

What is this *well chosen* t_{think}?

From fig.3 we can tell that if philo 2 would have thought a little bit longer there would have been no conflict with philo 1. But t_{think} has to be a derived parameter! Meaning: we are only alllowed to specify t_{die} and t_{sleep} on the command line. So we can only set t_{think} from a given set of parameters which of course still have to obey Rule II in order for our philos to be able to survive! So if

$$\max\left(3 \cdot t_{\text{eat}},\ t_{\text{eat}} + t_{\text{sleep}}\right) \quad = \quad 3 \cdot t_{\text{eat}}$$

then:

$$t_{die} > 3 \cdot t_{eat} = t_{eat} + t_{sleep} + t_{think} \quad \Rightarrow \quad \boxed{t_{think} = 2 \cdot t_{eat} - t_{sleep}}$$

but if

$$\max\left(3 \cdot t_{eat},\ t_{eat} + t_{sleep}\right) \quad = \quad t_{eat} + t_{sleep}$$

then we get

$$3 \cdot t_{eat} < t_{eat} + t_{sleep} \quad \Rightarrow \quad 2 \cdot t_{eat} - t_{sleep} < 0,$$

meaning: there is no more room to think. But in this case the long sleeping times already guarantee that there will be no fork ressource problems.

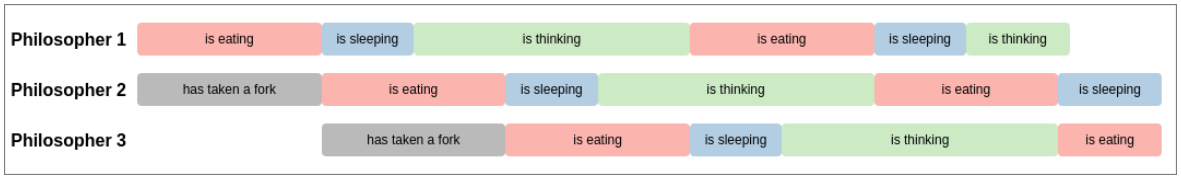

Figure 4: The solution with $t_{think} = 2 \cdot t_{eat} - t_{sleep}$

In fig.4 a philo dinner with well chosen t_{think} is shown and in fig.5 a dinner with $3 \cdot t_{eat} < t_{eat} + t_{sleep}$ is shown (except for the numerical glitch in the first block of philo 1 ;). It is obvious that the patterns with a syncro like this align in a way that 2 philos are always either busy sleeping or thinking when the 3rd needs to eat again. So there is no problem!

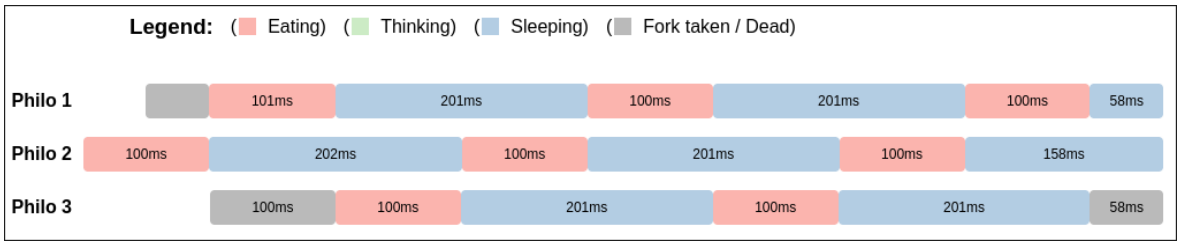

Figure 5: Odd long sleep.

Peace 🖖.

P.s.: For the diagrams i used (a modified version of) Romain's very nice Philosophers Visualizer. Thx a lot for creating this, Romain!