Concurrent Programming Practical 4: Dining Philosophers

It is recommended that you read the material from the section of the course notes on the Dining Philosophers before you attempt this practical. The code from the lectures is on the course website. The aim of the practical is to investigate some variants of the Dining Philosophers, which aim to avoid deadlocks. You should implement the first three variants below, and as many of the optional variants you have the inclination to work on.

Variant 1: a right-handed philosopher
In the standard version of the dining philosophers, all the philosophers are left-handed: they pick up their left fork first. Implement a variant where one of the philosophers is right-handed, i.e. she picks up her right fork first.

Variant 2: using a butler
Now consider a variant using an extra process, which represents a butler. The butler makes sure that no more than four philosophers are ever simultaneously seated.

Variant 3: asking the forks using a protocol
Now consider a variant where, if a philosopher is unable to obtain her second fork, she puts down her first fork, and re-tries later. This can be solved straighforwardly by implementing a bidirectional protocol between forks and philosophers: a philosopher should be able ask a fork to commit itself to her, and be answered positively only if the fork has actually committed to her; negatively if the fork has already committed to its other potential user. [Hint: the protocol will need two channels; it will simplify your program structure to package them in a class.]

Optional, but desirable
For each of the variants you implement include a way of counting the number of times each philosopher has eaten, and of presenting all this information periodically in addition to, or instead of, the detailed reports of the actions of forks and philosophers. Think of this as insurance against starvation! The same code should suffice for all variants.

Variant (optional: for fun): pile the forks in the middle of the table
In this variant, forks that are not being used sit in a pile in the middle of the table, and philosophers can use any two of them to eat.

Variant (optional: timeouts)
Now consider a variant where, if a philosopher is unable to obtain her second fork, she puts down her first fork, and re-tries later. Hint: This can be solved

straighforwardly by using the deadlined ‘writeBefore‘ method in the channel from a philosopher to one of her forks.[1]

Reporting

Your report should be in the form of a well-commented program, together with a brief discussion of any design considerations and of your results.

Bernard Sufrin
4th **October, 2017(4.1554)**

[1]It may be worth doing some simple experiments in advance to see how deadlined writes/reads interact with alternations. My own solution used a deadline that was roughly twice the eating time.