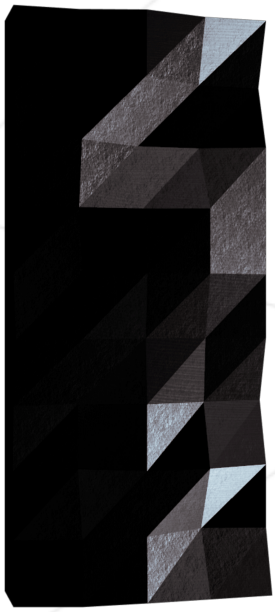

Philosophers

I never thought philosophy would be so deadly

Summary:
In this project, you will learn the basics of threading a process.
You will see how to create threads and you will discover mutexes.

Version: 10

Contents

Chapter I

Introduction

Philosophy (from Greek, *philosophia*, literally "love of wisdom") is the study of general and fundamental questions about existence, knowledge, values, reason, mind, and language. Such questions are often posed as problems to be studied or resolved. The term was probably coined by Pythagoras (c. 570 – 495 BCE). Philosophical methods include questioning, critical discussion, rational argument, and systematic presentation.

Classic philosophical questions include: Is it possible to know anything and to prove it? What is most real? Philosophers also pose more practical and concrete questions such as: Is there a best way to live? Is it better to be just or unjust (if one can get away with it)? Do humans have free will?

Historically, "philosophy" encompassed any body of knowledge. From the time of Ancient Greek philosopher Aristotle to the 19th century, "natural philosophy" encompassed astronomy, medicine, and physics. For example, Newton's 1687 Mathematical Principles of Natural Philosophy later became classified as a book of physics.

In the 19th century, the growth of modern research universities led academic philosophy and other disciplines to professionalize and specialize. In the modern era, some investigations that were traditionally part of philosophy became separate academic disciplines, including psychology, sociology, linguistics, and economics.

Other investigations closely related to art, science, politics, or other pursuits remained part of philosophy. For example, is beauty objective or subjective? Are there many scientific methods or just one? Is political utopia a hopeful dream or hopeless fantasy? Major sub-fields of academic philosophy include metaphysics ("concerned with the fundamental nature of reality and being"), epistemology (about the "nature and grounds of knowledge [and]... its limits and validity"), ethics, aesthetics, political philosophy, logic and philosophy of science.

Chapter II

Common Instructions

- Your project must be written in C.
- Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check and you will receive a `0` if there is a norm error inside.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a `0` during the evaluation.
- All heap allocated memory space must be properly freed when necessary. No leaks will be tolerated.
- If the subject requires it, you must submit a `Makefile` which will compile your source files to the required output with the flags `-Wall`, `-Wextra` and `-Werror`, use cc, and your Makefile must not relink.
- Your `Makefile` must at least contain the rules `$(NAME)`, `all`, `clean`, `fclean` and `re`.
- To turn in bonuses to your project, you must include a rule `bonus` to your Makefile, which will add all the various headers, librairies or functions that are forbidden on the main part of the project. Bonuses must be in a different file `_bonus.{c/h}` if the subject does not specify anything else. Mandatory and bonus part evaluation is done separately.
- If your project allows you to use your `libft`, you must copy its sources and its associated `Makefile` in a `libft` folder with its associated Makefile. Your project's `Makefile` must compile the library by using its `Makefile`, then compile the project.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done

after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter III

Overview

Here are the things you need to know if you want to succeed this assignment:

- One or more philosophers sit at a round table.
  There is a large bowl of spaghetti in the middle of the table.
- The philosophers alternatively **eat**, **think**, or **sleep**.
  While they are eating, they are not thinking nor sleeping;
  while thinking, they are not eating nor sleeping;
  and, of course, while sleeping, they are not eating nor thinking.
- There are also forks on the table. There are **as many forks as philosophers**.
- Because serving and eating spaghetti with only one fork is very inconvenient, a philosopher takes their right and their left forks to eat, one in each hand.
- When a philosopher has finished eating, they put their forks back on the table and start sleeping. Once awake, they start thinking again. The simulation stops when a philosopher dies of starvation.
- Every philosopher needs to eat and should never starve.
- Philosophers don't speak with each other.
- Philosophers don't know if another philosopher is about to die.
- No need to say that philosophers should avoid dying!

Chapter IV

Global rules

You have to write a program for the mandatory part and another one for the bonus part (if you decide to do the bonus part). They both have to comply with the following rules:

- Global variables are forbidden!
- Your(s) program(s) should take the following arguments:
  `number_of_philosophers time_to_die time_to_eat time_to_sleep [number_of_times_each_philosopher_must_eat]`
  - `number_of_philosophers`: The number of philosophers and also the number of forks.
  - `time_to_die` (in milliseconds): If a philosopher didn't start eating `time_to_die` milliseconds since the beginning of their last meal or the beginning of the simulation, they die.
  - `time_to_eat` (in milliseconds): The time it takes for a philosopher to eat. During that time, they will need to hold two forks.
  - `time_to_sleep` (in milliseconds): The time a philosopher will spend sleeping.
  - `number_of_times_each_philosopher_must_eat` (optional argument): If all philosophers have eaten at least `number_of_times_each_philosopher_must_eat` times, the simulation stops. If not specified, the simulation stops when a philosopher dies.
- Each philosopher has a number ranging from `1` to `number_of_philosophers`.
- Philosopher number `1` sits next to philosopher number `number_of_philosophers`. Any other philosopher number `N` sits between philosopher number `N` - 1 and philosopher number `N` + 1.

About the logs of your program:

- Any state change of a philosopher must be formatted as follows:
  - `timestamp_in_ms X has taken a fork`
  - `timestamp_in_ms X is eating`
  - `timestamp_in_ms X is sleeping`
  - `timestamp_in_ms X is thinking`
  - `timestamp_in_ms X died`

  Replace `timestamp_in_ms` *with the current timestamp in milliseconds and* `X` *with the philosopher number.*
- A displayed state message should not be mixed up with another message.
- A message announcing a philosopher died should be displayed no more than 10 ms after the actual death of the philosopher.
- Again, philosophers should avoid dying!

Your program must not have any **data races.**

Chapter V

Mandatory part

Program name	`philo`
Turn in files	`Makefile, *.h, *.c, in directory philo/`
Makefile	`NAME, all, clean, fclean, re`
Arguments	`number_of_philosophers time_to_die time_to_eat time_to_sleep [number_of_times_each_philosopher_must_eat]`
External functs.	`memset, printf, malloc, free, write, usleep, gettimeofday, pthread_create, pthread_detach, pthread_join, pthread_mutex_init, pthread_mutex_destroy, pthread_mutex_lock, pthread_mutex_unlock`
Libft authorized	`No`
Description	`Philosophers with threads and mutexes`

The specific rules for the mandatory part are:

- Each philosopher should be a thread.
- There is one fork between each pair of philosophers. Therefore, if there are several philosophers, each philosopher has a fork on their left side and a fork on their right side. If there is only one philosopher, there should be only one fork on the table.
- To prevent philosophers from duplicating forks, you should protect the forks state with a mutex for each of them.

Chapter VI

Bonus part

Program name	`philo_bonus`
Turn in files	`Makefile, *.h, *.c, in directory philo_bonus/`
Makefile	`NAME, all, clean, fclean, re`
Arguments	`number_of_philosophers time_to_die time_to_eat time_to_sleep [number_of_times_each_philosopher_must_eat]`
External functs.	`memset, printf, malloc, free, write, fork, kill, exit, pthread_create, pthread_detach, pthread_join, usleep, gettimeofday, waitpid, sem_open, sem_close, sem_post, sem_wait, sem_unlink`
Libft authorized	`No`
Description	`Philosophers with processes and semaphores`

The program of the bonus part takes the same arguments as the mandatory program. It has to comply with the requirements of the *Global rules* chapter.

The specific rules for the bonus part are:

- All the forks are put in the middle of the table.
- They have no states in memory but the number of available forks is represented by a semaphore.
- Each philosopher should be a process. But the main process should not be a philosopher.

`The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.`

Chapter VII

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.

Mandatory part directory: `philo/`

Bonus part directory: `philo_bonus/`