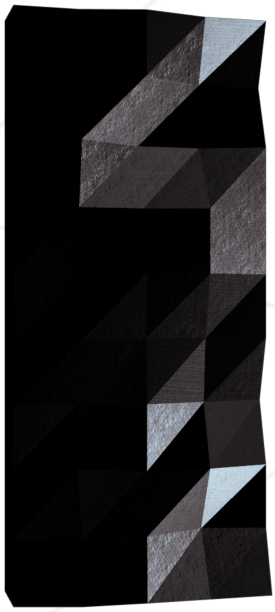

Philosophers

I've never thought philosophy would be so deadly.

Summary: In this project, you will learn the basics of threading a process. You will learn how to make threads. You will discover the mutex.

Version: 9

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

What you should understand to succeed this project:

- One or more philosophers are sitting at a round table either eating, either thinking, either sleeping. While they are eating, they do not think or sleep; while thinking they don't eat or sleep; and, of course, while sleeping, they do not eat or think.
- The philosophers sit at a circular table with a large bowl of spaghetti in the center.
- There are some forks on the table. Serving and eating spaghetti with a single fork is very inconvenient, so the philosophers will eat with two forks, one for each hand.
- Each time a philosopher finishes eating, they will drop their forks and start sleeping. Once they have finished sleeping, they will start thinking. The simulation stops when a philosopher dies.
- Every philosopher needs to eat and they should never starve.
- Philosophers don't speak with each other.
- Philosophers don't know when another philosopher is about to die.
- No need to say that philosophers should avoid dying!

Chapter IV

General Instructions

You have to write a program for the mandatory part and another one for the bonus part but they will have the same basic rules:

- Global variables are forbidden!
- The program 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: is the number of philosophers and also the number of forks.
  - time_to_die: is in milliseconds, if a philosopher doesn't start eating 'time_to_die' milliseconds after starting their last meal or the beginning of the simulation, it dies.
  - time_to_eat: is in milliseconds and is the time it takes for a philosopher to eat. During that time they will need to keep the two forks.
  - time_to_sleep: is in milliseconds and is the time the philosopher will spend sleeping.
  - number_of_times_each_philosopher_must_eat: argument is optional, if all philosophers eat at least 'number_of_times_each_philosopher_must_eat' the simulation will stop. If not specified, the simulation will stop only at the death of a philosopher.
- Each philosopher should be given a number from 1 to 'number_of_philosophers'.
- Philosopher number 1 is next to philosopher number 'number_of_philosophers'. Any other philosopher with the number N is seated between philosopher N - 1 and philosopher N + 1.

About the logs of your program:

- Any change of status of a philosopher must be written as follows (with X replaced with the philosopher number and timestamp_in_ms the current timestamp in milliseconds):
  - 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
- The status printed should not be scrambled or intertwined with another philosopher's status.
- You can't have more than 10 ms between the death of a philosopher and when it will print its death.
- Again, philosophers should avoid dying!

Chapter V

Mandatory part

Program name	`philo`
Turn in files	`philo/`
Makefile	`Yes`
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	`philosopher with threads and mutex`

The specific rules for the mandatory part are:

- Each philosopher should be a thread.
- One fork between each philosopher, therefore if they are multiple philosophers, there will be a fork at the right and the left of each philosopher.
- To avoid philosophers duplicating forks, you should protect the forks state with a mutex for each of them.

Chapter VI

Bonus

Program name	`philo_bonus`
Turn in files	`philo_bonus/`
Makefile	`Yes`
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	`philosopher with processes and semaphore`

For the bonus part, the program takes the same arguments as before and should behave as explained on the General Instructions chapter. The specific rules are:

- All the forks are 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 and the main process should not be a philosopher.