B4 - Concurrent Programming

B-CCP-400

Panoramix

One refill at a time!

1.4

EPITECH.

Panoramix

binary name: panoramix
language: C
compilation: via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Context

The year is 50 BC. Gaul is entirely occupied by the Romans. Well, not entirely… One small village of indomitable Gauls still holds out against the invaders. And life is not easy for the Roman legionaries who garrison the fortified camps of Totorum, Aquarium, Laudanum and Compendium…

In this village the druid prepares a cooking pot of magic potions. The villagers get a serving of potion and then procede to beat roman soldiers. When the pot is empty, the villager wakes up the druid and waits until the druid has refilled the pot.

Project

The purpose of this project is to study examples of threads and mutex programming and the algorithmic difficulties that go with it.

To do that, you must create a program that simulate the described behavior.

Terminal

```
~/B-CCP-400> ./panoramix
USAGE: ./panoramix <nb_villagers> <pot_size> <nb_fights> <nb_refills>
```

- `nb_villagers`: indicates the number of villagers (must be >0);
- `pot_size`: indicates the maximum number that can be contained in the cooking pot (must be >0);
- `nb_fights`: indicates the maximum number of fights a villager will engage in (must be >0);
- `nb_refills`: incicates the maximum number of time the druid will refill the pot (must be >0).

The program must follow these rules:

- You **MUST** make use of semaphores;
- You **MUST** make use of mutexes;
- Each villager (and the druid) **MUST** run in its own thread;
- The cooking pot is full at the start of the program:
- A villager's thread must stop when all the fights have been done (and hopefully won);
- The druid's thread must stop when he has no ingredients left (aka: `nb_refills`) to refill the pot;
- The program must stop gracefully when all the villager's threads have stopped.

VILLAGER

Each villager is identified by an `id` (a number unique to the villager). It will fight `nb_fights` time before leaving the battlefield. Before each fight, it must take a serving of magical potion from the pot (if the pot is empty, it must inform the druid and wait until the pot is refilled).

A villager cannot fight without having drank some potion first!

You **MUST** display the following messages on the standard output, when the corresponding event occurs:

- When a villager thread starts

```
Villager <id>: Going into battle!
```

- When a villager tries to drink some potion

```
Villager <id>: I need a drink... I see <nb_servings_left> servings left.
```

- When a villager call the druid for a refill

```
Villager <id>: Hey Pano wake up! We need more potion.
```

- When a villager is fighting

```
Villager <id>: Take that roman scum! Only <fights> left.
```

- When a villager has done all is fights

```
Villager <id>: I'm going to sleep now.
```

There are cases when villagers may wait endlessly. Can you see which ones? We're not speaking about deadlocks…

DRUID

The druid will wait to be called by a villager; then it will refill the pot with `pot_size` servings. When `nb_refills` have been done, the druid has run out of ingredients and it's thread must stop.

It may be a good idea to inform the villagers that the pot has been refilled.

You **MUST** display the following messages on the standard output, when the corresponding event occurs:

- When the druid thread starts

```
Druid: I'm ready... but sleepy...
```

- When the druid is called to refill the pot

```
Druid: Ah! Yes, yes, I'm awake! Working on it! Beware I can only make <refills> more
    refills after this one.
```

- When the druid runs out of ingredients

```
Druid: I'm out of viscum. I'm going back to... zZz
```

EXAMPLES

Threads run asynchronously, so the outputs will not always be exactly in the same order but it must always follow all the rules previously indicated.

Terminal

```
~/B-CCP-400> ./panoramix -1 3 2 1 >/dev/null; echo $?
USAGE: ./panoramix <nb_villagers> <pot_size> <nb_fights> <nb_refills>
Values must be >0.
84
```

Terminal

```
~/B-CCP-400> ./panoramix 3 5 3 1
Druid: I'm ready... but sleepy...
Villager 2: Going into battle!
Villager 1: Going into battle!
Villager 0: Going into battle!
Villager 2: I need a drink... I see 5 servings left.
Villager 0: I need a drink... I see 4 servings left.
Villager 0: Take that roman scum! Only 2 left.
Villager 1: I need a drink... I see 3 servings left.
Villager 2: Take that roman scum! Only 2 left.
Villager 1: Take that roman scum! Only 2 left.
Villager 0: I need a drink... I see 2 servings left.
Villager 0: Take that roman scum! Only 1 left.
Villager 2: I need a drink... I see 1 servings left.
Villager 1: I need a drink... I see 0 servings left.
Villager 1: Hey Pano wake up! We need more potion.
Druid: Ah! Yes, yes, I'm awake! Working on it! Beware I can only make 0 more
refills after this one.
Druid: I'm out of viscum. I'm going back to... zZz
Villager 1: Take that roman scum! Only 1 left.
Villager 2: Take that roman scum! Only 1 left.
Villager 0: I need a drink... I see 4 servings left.
Villager 1: I need a drink... I see 3 servings left.
Villager 0: Take that roman scum! Only 0 left.
Villager 2: I need a drink... I see 2 servings left.
Villager 1: Take that roman scum! Only 0 left.
Villager 2: Take that roman scum! Only 0 left.
Villager 0: I'm going to sleep now.
Villager 1: I'm going to sleep now.
Villager 2: I'm going to sleep now.
```