Andrzej Gierlak, 236411 Wrocław 15.03.2021

Grupa: E07-20j Poniedziałek 13-15 TP

Prowadzący: Dr inż. Dominik Żelazny

Systemy Operacyjne 2
Projekt 1: Problem Głodujących Filozofów

1 Wstęp

Celem projektu było utworzenie aplikacji wielowątkowej rozwiązującej zagadnienie ucztujących filozofów.

Przedstawienie problemu: Pięciu filozofów siedzi przy stole i każdy wykonuje jedną z dwóch czynności – albo je, albo rozmyśla. Stół jest okrągły, przed każdym z nich znajduje się miska ze spaghetti, a pomiędzy każdą sąsiadującą parą filozofów leży widelec, a więc każda osoba ma przy sobie dwie sztuki – po swojej lewej i prawej stronie. Ponieważ jedzenie potrawy jest trudne przy użyciu jednego widelca, zakłada się, że każdy filozof korzysta z dwóch. Dodatkowo nie ma możliwości skorzystania z widelca, który nie znajduje się bezpośrednio przed daną osobą.

2 Rozwiązanie problemu synchronizacji

W celu uniknięcia deadlocków i livelocków, zaimplementowano rozwiązanie za pomocą arbitra/kelnera. Wątek (filozof) nie podnosi widelców, dopóki nie uzyska zgody od kelnera. Kelner obsługuje żądania wątków po kolei. Przez to zmniejszona jest równoległość obliczeń, gdyż kelner musi zarządzać wszystkimi procesami. Kiedy kelner pozwala filozofowi jeść podnosi wtedy oba widelce. W tym rozwiązaniu nie jest istotne który widelec podnosi najpierw. W programie zaimplementowano proste menu rozwijane przyciskiem F9, a opcjami do wyboru jest wyłączenie wątków i zamknięcie aplikacji. Obecnie trzeba to robić w odpowiedniej kolejności, czyli najpierw zatrzymywać wątki(wtedy żaden wątek filozofa nie rozpocznie nowego cyklu życia), a dopiero potem zamknąć GUI, co zkaończy samą aplikację.

3 Rozwiązanie problemu zagłodzenia

Po zastosowaniu kelnera, mógłby zaistnieć przypadek zagłodzenia procesu. W "pechowym" scenariuszu, proces obsługiwany przez kelnera ciągle natrafiałby na odmowę dostępu do zasobów, gdyż jego sąsiedzi mogli mieć małe czasy myślenia (thiking, filozofowania) i długie czasy jedzenia. Żeby umożliwić bardziej sprawiedliwy dostęp do zasobów, ustalono dodatkowe warunki, czy kelner pozwala danemu filozofowi jeść lub nie. Dodaną regułę można przedstawić w postaci zdania: "Nie pozwól jeść sąsiadom najgłodniejszego procesu", gdzie najgłodniejszy proces jest tym, który jest na szczycie kolejki (kolejność pod względem "głodu"). Kiedy jest obsługiwany proces inny niż najbardziej głodujący lub jego sąsiedzi, kolejka aktualizowana jest tylko w fragmencie (obsługiwany idzie na koniec kolejki, a znajdujący się za nim filozofowie, podchodzą o jedno miejsce do przodu).

4 Wykorzystane technologie

Projekt realizowałem w języku c# na platformie .Net (5.0). W celu zaprezentowania symulacji (GUI) użyłem biblioteki Terminal.Gui Version="0.90.3", która powinna działać zarówno na systemach windows, mac i linux (testowano tylko na systemie windows 10). Zawiera sterowniki dla Curses, Windows Console i .Net Console. Wykorzystałem wątki Thread, a w celu synchronizacji użyłem mutexy Mutex i semafory SemaphoreSlim, wszystko z biblioteki System.Threading. SemaphoreSlim to wersja zwykłego Semaphore, która jest ograniczona zasięgiem tylko do aplikacji, co było wystarczające do tego projektu.

5 Przykładowe screeny z GUI

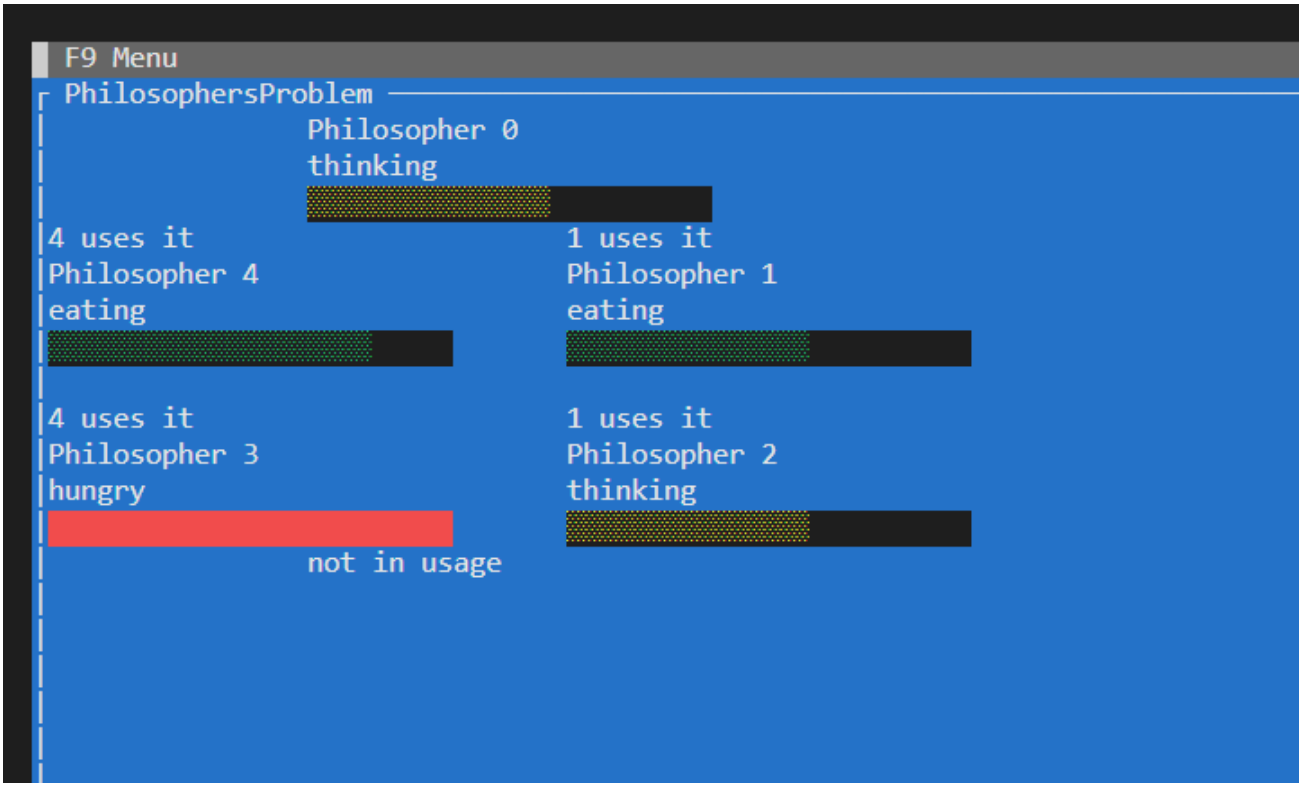

Figure 1: Działająca aplikacja

6 Linki

Biblioteka graficzna Terminal.GUI
Github Projektu