Politechnika Białostocka

Systemy Operacyjne

Dokumentacja Projektu

Wykonujący ćwiczenie:

- **Mateusz Mogielnicki**
- **Dominik Mierzejewski**
- **Przemysław Rutkowski**
- **Jakub Matyszek**

Studia dzienne

Kierunek: Informatyka

Semestr: IV

Grupa zajęciowa: PS6

Prowadzący ćwiczenie: mgr inż. Tomasz Kuczyński

Data wykonania projektu: 8.06.2023r.

Treść Projektu

Temat – czytelnicy i pisarze

Czytelnicy i pisarze. Z czytelni korzysta na okrągło pewna ilość czytelników i pisarzy, przy czym jednocześnie może w niej znajdować się albo dowolna ilość czytelników, albo jeden pisarz, albo nikt - nigdy inaczej. Problem ten ma trzy rozwiązania - z możliwością zagłodzenia pisarzy, z możliwością zagłodzenia czytelników oraz wykluczające zagłodzenie. Napisać:

c) trzy programy symulujące trzy różne rozwiązania tego problemu, przy czym przynajmniej jeden z nich musi korzystać ze zmiennych warunkowych [34 p].

Ilość wątków pisarzy R i czytelników W można przekazać jako argumenty linii poleceń. Zarówno czytelnicy jak i pisarze wkrótce po opuszczeniu czytelni próbują znów się do niej dostać. Program powinien wypisywać komunikaty według poniższego przykładu:

ReaderQ: 11 WriterQ: 10 [in: R:0 W:1] Oznacza to, że w kolejce przed czytelnią czeka 10 pisarzy i 11 czytelników a sama czytelnia zajęta jest przez jednego pisarza. Komunikat należy wypisywać w momencie zmiany którejkolwiek z tych wartości.

Opis Rozwiązań

W ramach tego projektu napisaliśmy trzy różne programy symulujące rozwiązania problemu "czytelnicy i pisarze". Nasze zadanie polegało na zarządzaniu dostępem do czytelni, z której mogą korzystać zarówno czytelnicy, jak i pisarze. Naszym celem było stworzenie programów, które zapewnią optymalne działanie czytelni w zależności od aktualnych potrzeb.

W naszych rozwiązaniach uwzględniliśmy trzy scenariusze:

1) Rozwiązanie z możliwością zagłodzenia pisarzy:

   a) W tym przypadku, stworzyliśmy program, w którym czytelnicy mają pierwszeństwo w dostępie do czytelni. Jeżeli w czytelni są czytelnicy oraz pojawia się pisarz, zostaje on zablokowany i musi czekać, aż wszyscy czytelnicy opuszczą czytelnię. Dopiero wtedy pisarz może wejść i rozpocząć pisanie.

   b) Wykorzystujemy tutaj dwa mutexy – „**readerMutex**” i „**writerMutex**” do synchronizacji dostępu do wspólnych zasobów, które w tym przypadku są dostępem do czytelni.

   c) Czytelnik chcący wejść do czytelni stara się zablokować **readerMutex** metodą „**pthread_mutex_lock(&readerMutex)**”. Następnie sprawdza czy jest już w bibliotece jakaś osoba czytająca, jeżeli nie to zablokowuje **writerMutex**. Potem następuje odblokowanie **readerMutex**, aby kolejne osoby chcące czytać mogły wejść. Przy wyjściu czytelnika następuje sprawdzenie, czy w bibliotece jest jeszcze jakiś czytelnik, jeżeli nie to **writerMutex** zostaje odblokowany metodą „**pthread_mutex_unlock(&writerMutex)**”. Natomiast, gdy pisarz chce wejść do czytelni próbuje on zablokować **writerMutex**, jeżeli mu się uda rozpoczyna pisanie, po zakończeniu zwalniając **mutex** pisarzy. W przeciwnym razie oczekuje na zwolnienie **mutexu**, aby wejść do biblioteki.

2) Rozwiązanie z możliwością zagłodzenia czytelników:

   a) W drugim przypadku skupiliśmy się na pisarzach, zapewniając im pierwszeństwo dostępu do czytelni. W tym przypadku, dopóki jest jakikolwiek pisarz czekający na wejście do czytelni, czytelnicy są zablokowani i muszą czekać do momentu, w którym każdy z oczekujących pisarzy skorzysta z czytelni.

   b) W tym programie wykorzystujemy jeden mutex - „**mutex**” oraz dwie zmienne warunkowe – „**canRead**” oraz „**canWrite**”.

   c) Pisarz chcący wejść do czytelni stara się zablokować **mutex** metodą „**pthread_mutex_lock()**”, następnie sprawdza, czy w czytelni jest jakaś inna osoba, jeżeli jest zatrzymuje się on na blokadzie warunkowej „**pthread_cond_wait()**”, oczekując sygnału na zezwolenie na wejście poprzez „**pthread_cond_signal()**”. Jeżeli biblioteka jest wolna wchodzi on rozpoczynając pisanie, po zakończeniu następuje sprawdzenie, czy są jacyś oczekujący pisarze, jeżeli tak to wysyłany jest sygnał do następnego pisarza, jeżeli nie to następuje wysłanie sygnału funkcją „**pthread_cond_broadcast()**” do wszystkich oczekujących czytelników. Podobnie jest z oczekującymi czytelnikami starają się oni zablokować **mutex** po czym, jeżeli im się udało następuje sprawdzenie, czy w czytelni są jacyś piszący lub oczekujący pisarze. Jeżeli są, to czytelnicy zatrzymują się na blokadzie warunkowej oczekując sygnału, w innym przypadku zaczynają czytanie. Po zakończeniu czytania następuje sprawdzenie czy w czytelni są jeszcze jacyś czytelnicy oraz oczekujący pisarze. Jeśli oba warunki są spełnione to następuje wysłanie sygnału do następnego oczekującego pisarza.

3) Rozwiązanie wykluczające zagłodzenie:

   a) W trzecim rozwiązaniu zaimplementowaliśmy wykluczenie zagłodzenia. W tym przypadku, zarówno czytelnicy, jak i pisarze mają równy dostęp do czytelni, eliminując przy tym zagłodzenie zarówno pisarzy, jak i czytelników. W tym przypadku, jeżeli jakikolwiek z pisarzy chce wejść do czytelni sprawdza czy jest ona pusta, jeżeli tak to może on wejść, w innym przypadku musi czekać. Podobnie jest, gdy to czytelnik chce wejść, musi się on upewnić, że w czytelnia jest pusta lub są w niej czytelnicy.

   b) W tym programie, tak samo jak w rozwiązaniu z możliwością zagłodzenia czytelników wykorzystujemy jeden mutex oraz dwie zmienne warunkowe.

   c) Czytelnik, który chce wejść do czytelni stara się zablokować **mutex**. W przypadku, gdy mu się to udaje następuje sprawdzenie czy są jacyś oczekujący lub piszący pisarze, jeżeli tak to liczba oczekujących czytelników jest zwiększona o jeden oraz czytelnik zostaje zablokowany na blokadzie warunkowej „**pthread_cond_wait()**”, oczekując sygnału zezwolenia na wejście. Jeżeli nie, to liczba czytelników jest zwiększona oraz następuje rozpoczęcie czytania wraz z którym jest zwolniony **mutex** w celu zezwolenia na wejście innym czytelnikom. Po wyjściu czytelnika, sprawdzamy czy w czytelni znajduje się jeszcze jakaś osoba czytająca. Jeżeli warunek jest niespełniony następuje wysłanie sygnału funkcją „**pthread_cond_signal()**” zezwalającego na wejście pisarzowi. Podobnie jest, gdy to pisarz chce wejść do czytelni, stara się on zablokować **mutex**, następnie sprawdzając czy są w niej jakieś osoby czytające lub piszące. Jeżeli, którykolwiek z tych warunków jest spełniony to pisarz zostaje zablokowany warunkowo oczekując na sygnał pozwalający na wejście. Jeżeli natomiast biblioteka jest wolna rozpoczyna pisanie. Po wyjściu następuje sprawdzenie, czy są jacyś oczekujący czytelnicy, jeżeli tak to następuje wysłanie sygnału zezwalającego wszystkim oczekującym czytelnikom na wejście do czytelni funkcją „**pthread_cond_broadcast()**”. W przypadku, gdy nie ma osób chcących czytać zostaje wysłany sygnał do następnego oczekującego pisarza.

Uruchomienie Programów

W celu uruchomienia każdego z poniższych programów należy użyć komendy:

```
./[nazwa programu] -W [liczba czytelników]  -R [liczba pisarzy]
```

1) Rozwiązanie z możliwością zagłodzenia pisarzy:

```
./starvationWriters -W [liczba czytelników]  -R [liczba pisarzy]
```

2) Rozwiązanie z możliwością zagłodzenia czytelników:

```
./starvationReaders -W [liczba czytelników]  -R [liczba pisarzy]
```

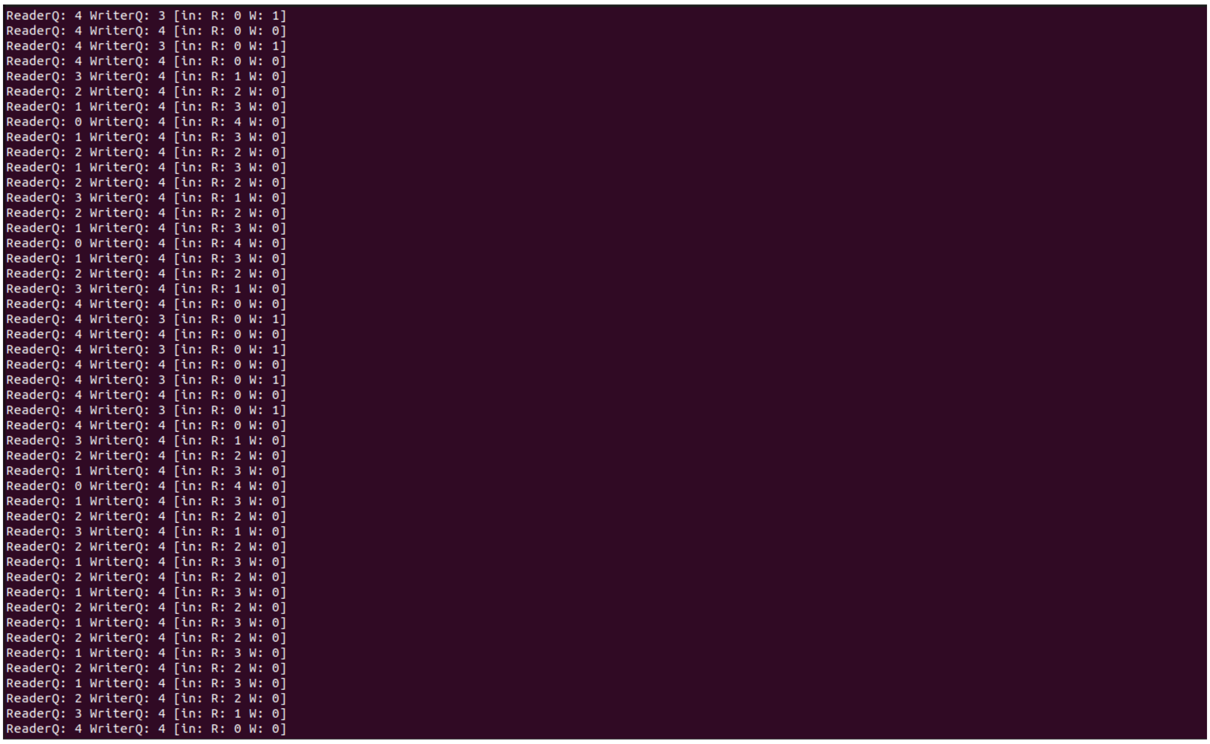
3) Rozwiązanie wykluczające zagłodzenie:

```
./excludingStarvation -W [liczba czytelników]  -R [liczba pisarzy]
```

Przykładowe działanie programów

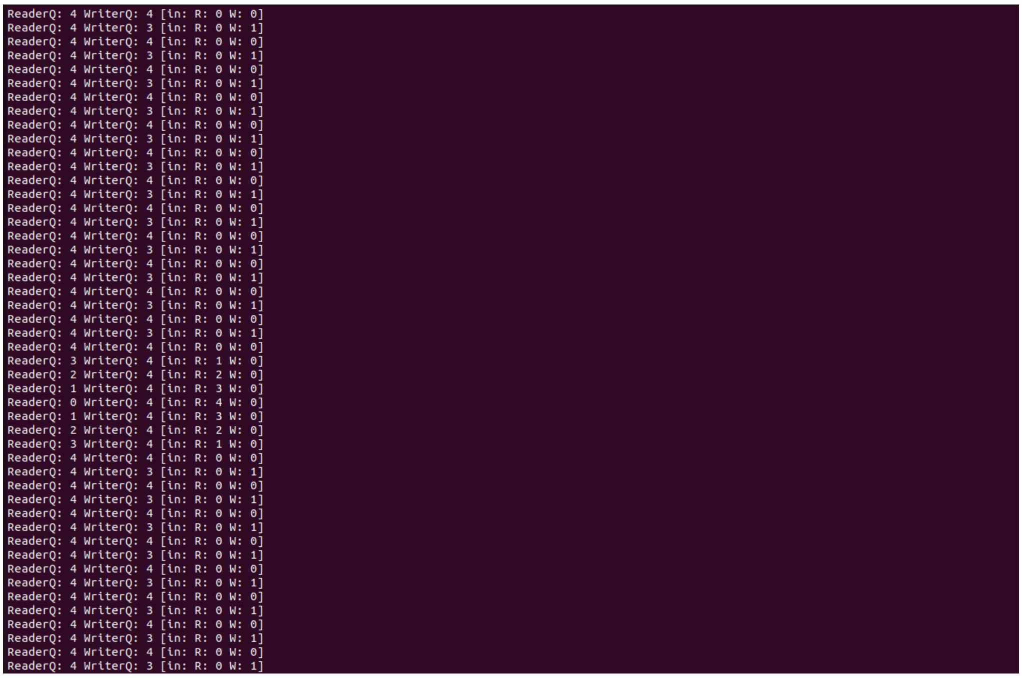
Poniżej można zauważyć poglądowe zrzuty ekranu przedstawiające działanie poszczególnych programów.

1. Rozwiązanie z możliwością zagłodzenia pisarzy:



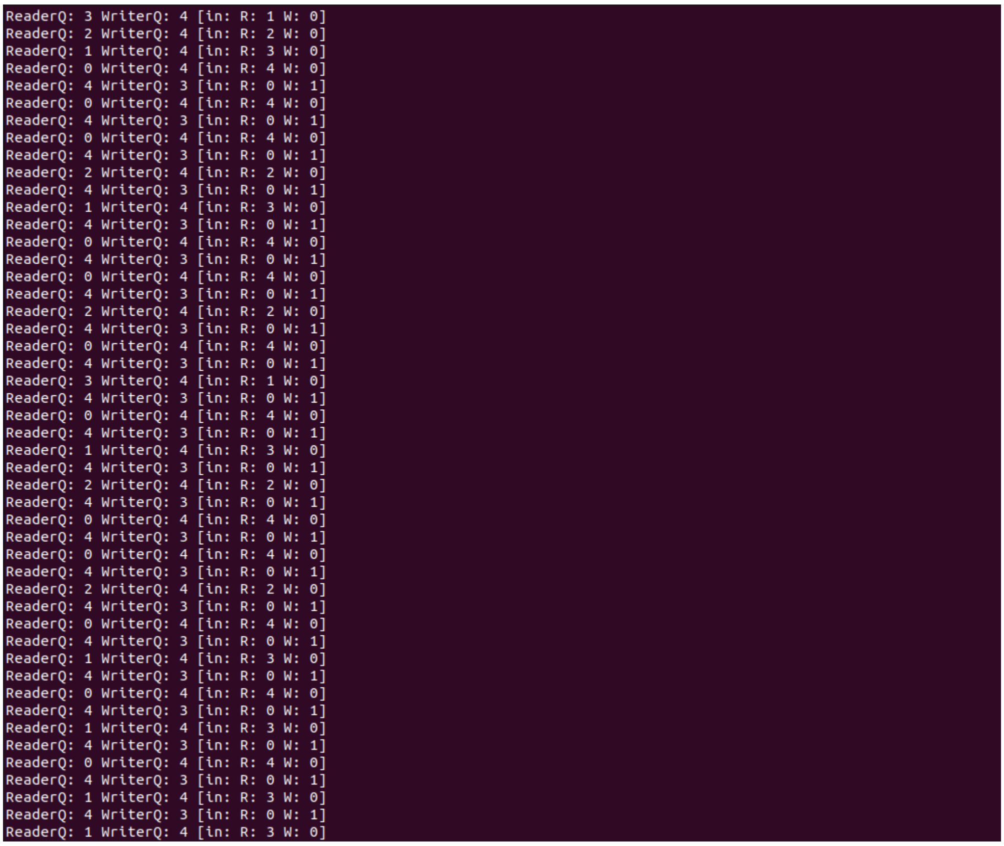
```
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
```

2. Rozwiązanie z możliwością zagłodzenia czytelników:



```
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 4 WriterQ: 4 [in: R: 0 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
```

3. Rozwiązanie wykluczające zagłodzenie:



```
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 3 WriterQ: 4 [in: R: 1 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 2 WriterQ: 4 [in: R: 2 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 0 WriterQ: 4 [in: R: 4 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
ReaderQ: 4 WriterQ: 3 [in: R: 0 W: 1]
ReaderQ: 1 WriterQ: 4 [in: R: 3 W: 0]
```

Podział Pracy

1. Mateusz Mogielnicki
    - excludingStarvation.c
    - poprawki odnośnie innych programów

2. Dominik Mierzejewski
    - starvationWriters.c
    - komentarze w programach

3. Jakub Matyszek
    - Dokumentacja projektu
    - Poprawki w kodzie

4. Przemysław Rutkowski
    - starvationReaders.c
    - dopracowanie innych programów