Raport projekt UL Adam Furtak

Ogólne założenia

Projekt w założeniu składa się z 4 rodzajów procesów, które komunikują się między sobą

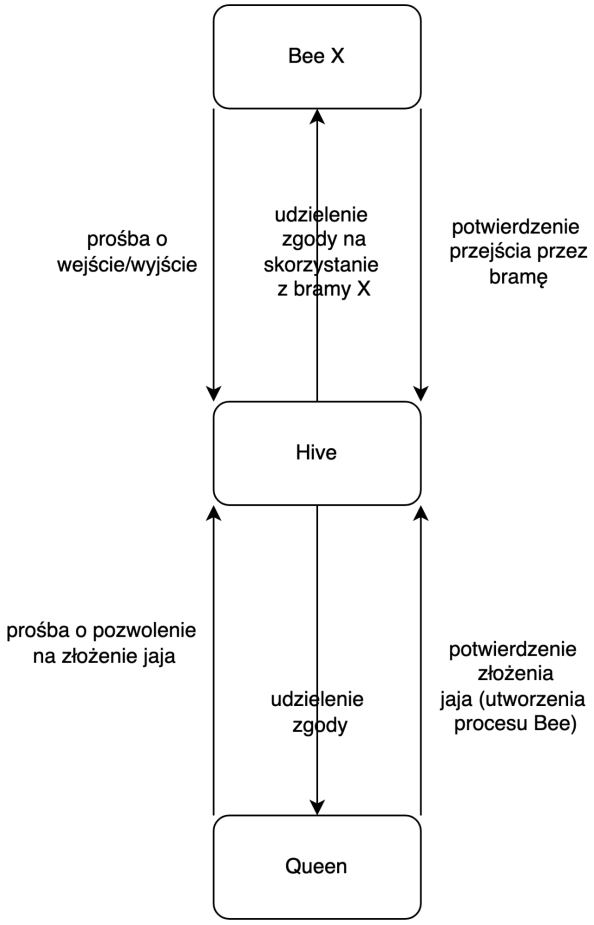
1. hive
2. bee
3. queen
4. logger_server

Podczas gdy istnieje jeden proces hive i jeden proces queen, w jednym momencie istnieje wiele niezależnych procesów bee.

Proces hive podczas swojego startu tworzy procesy potomne queen i bee na podstawie pliku konfiguracyjnego, podanego jako parametr wywołania.

Pomiędzy procesem hive, queen oraz procesami bee, jest utworzona kolejka komunikatów służąca do koordynacji pracy ula w kontekście wchodzących i wychodzących pszczół przez poszczególne wejścia oraz składania jaj przez królową.

Ogólny szkic protokołu komunikacji wygląda następująco:



Bee trzyma informację o swoim stanie: czy znajduje się na zewnątrz ula, wewnątrz ula i tym podobne.

Nad wszystkim znajduje się biblioteka do logowania, która wysyła logi do serwera logowania, który następnie wypisuje je na standardowe wyjście lub przekierowuje do pliku. Komunikacja pomiędzy procesami a serwerem logowania odbywa się poprzez współdzieloną pamięć i jest koordynowana za pomocą semaforów. Dla wygody użytkownika (w tym przypadku programisty) dodano bibliotekę z wygodnym interfacem do wysyłania logów na serwer.

Co udało się zrobić?

1. Zaimplementowano system logowania, pod warunkiem, że serwer logowania zostaje uruchomiony w odseparowaniu od pozostałych procesów. Pozostałe procesy wymagają uruchomionego procesu serwera logowania, w przeciwnym przypadku po zapełnieniu bufora, będą czekać na możliwość zapisania logów.
2. Zaimplementowano synchronizację dostępu do wspólnych zasobów pomiędzy procesami - wejścia do ula oraz miejsca w ulu.

3. Dodano mechanizm tworzenia nowych robotnic przez królową w porozumieniu z procesem ula pilnującym dostępu do miejsca w ulu
4. Zaimplementowano protokół komunikacji pomiędzy trzema rodzajami procesów, oparty na kolejce komunikatów odwzorowujący model działania klient i serwer z dodatkowym wysłaniem potwierdzenia do klientów.
5. Zaimplementowano współdzielone biblioteki ułatwiające implementację komunikacji pomiędzy procesami
6. Dodano walidację danych wejściowych w postaci pliku konfiguracyjnego odczytywanego przez proces hive
7. Zaimplementowano bezpieczną obsługę (async safe) sygnałów

Z czym są problemy

1. Procesy zawieszają się w oczekiwaniu na wysłanie loga do serwera w przypadku gdy nie istnieje proces serwera logowania
2. Mechanizm wymiany informacji pomiędzy hive i bee oraz queen wchodzi w nieprawidłowy stan w przypadku gdy zakomentować wszystkie użycia sleep(x), z powodu na “zapychanie” się kolejki zdarzeń
3. Przekazanie sygnału sigint do dzieci z hive nie zawsze działa za pierwszym razem skutkując zakończeniem symulacji

Nie udało się zaimplementować

1. Dodania/usunięcia dodatkowych ramek do ula za pomocą sygnałów

Przeprowadzone testy

1. Zwykłe uruchomienie symulacji i sprawdzenie czy wszystkie mechanizmy komunikacji międzyprocesowej zostały odpowiednio wyczyszczone
2. Testowanie nieprawidłowych plików konfiguracyjnych
3. Zakomentowanie “sleepów” w procesach queen i bee
    a. skutkuje wejściem symulacji w nieprawidłowy stan zakleszczenia
4. Uruchomienie symulacji bez serwera logowania
    a. symulacja będzie czekać na wczytanie logów przez serwer, po zapełnieniu dostępnego bufora
5. Uruchomienie symulacji, następnie zakończenie procesu serwera logów
    a. podobnie jak wyżej, symulacja będzie czekać na zwolnienie miejsca do zapisania logów w pamięci współdzielonej

Ważniejsze fragmenty kodu

1.