Raport - T7 Michał Kalinowski

Projekt Systemy Operacyjne

Politechnika Krakowska, Informatyka NST, III semestr Michał Kalinowski

1. Cel i założenia projektowe

Celem projektu jest odwzorowanie zachowań owadów w ulu przy użyciu mechanizmów systemu operacyjnego:

- Współdzielenie zasobów (pamięć współdzielona, semafory).
- Wielowątkowość: pszczoły-robotnice w osobnych wątkach.
- Ograniczona pojemność ula (capacity) – jednocześnie tylko określona liczba pszczół/jaj.
- Możliwość dynamicznej zmiany pojemności ula przez pszczelarza (poprzez sygnały SIGUSR1 / SIGUSR2).
- Symulacja cyklu życiowego pszczół (pszczoła „umiera" po ustalonej liczbie wizyt w ulu).
- Wieloprocesowość: tworzenie procesów (królowa queen, wylęgający hatch).

Kluczowe założenia:

- Dwa procesy potomne:
  - queen – składa jaja (jeśli w ulu jest miejsce).
  - hatch – wylęga pszczoły z jaj.
- Wątek „pszczelarza" (listening na sygnały SIGUSR1 i SIGUSR2).
- N (zadana liczba) wątków „robotnic" na starcie.
- Pamięć współdzielona (System V) trzyma strukturę EggQueue z informacjami:
  - occupant_count (aktualna liczba pszczół + jaj)
  - capacity (maks. liczba rezydentów),
  - kolejka jaj do wylęgania.
- Semafory:
  - Posix: ul_wejscie (liczbowe ograniczenie dostępności miejsc w ulu),
  - wejscie1_kierunek, wejscie2_kierunek (kontrola wchodzenia/wychodzenia),

    - SysV: do blokowania dostępu do kolejki jaj (sem_id).
- Obsługa sygnałów SIGUSR1, SIGUSR2, SIGINT (zatrzymanie programu).
- Kontrola:
    - Pszczoły wchodzą do ula, śpią tam określoną liczbę sekund, wychodzą, śpią poza ulem i tak w kółko, aż visits_left wyczerpie się.
    - Królowa co pewien czas próbuje złożyć jajo (jeśli nie ma miejsca, czeka).
    - Proces wylęgania co pewien czas „wyjmuje" jajo i tworzy nowy wątek pszczoły.
    - Wątek pszczelarza pozwala powiększyć/zmniejszyć ul za pomocą sygnałów.

2. Opis ogólny kodu (architektura)

Kod podzielony jest na kilka plików:

- main.c – punkt startowy, czyta N i P, inicjuje pamięć współdzieloną, semafory, uruchamia procesy queen i hatch, wątek pszczelarza oraz startowe wątki pszczół. Obsługuje również SIGINT.
- bee.[ch] – logika tworzenia i życia pszczół (wątki).
- queen.[ch] – proces queen (składanie jaj) i hatch (wylęganie).
- hive.[ch] – logika „wejścia/wyjścia" z ula, wyświetlanie stanu ula, zmiana pojemności
- egg.[ch] – struktura i operacje na kolejce jaj (EggQueue), zarządzanie pamięcią współdzieloną, semafor System V.
- beekeeper.[ch] – wątek pszczelarza, odbieranie sygnałów SIGUSR1 / SIGUSR2 i wywołanie adjust_hive_capacity().
- cleanup.[ch] – funkcja sprzątająca (zamyka procesy, wątki i zwalnia zasoby).
- error_handling.[ch] – definicje błędów i funkcja handle_error().

Co się udało zrobić:

- Działa poprawne tworzenie procesów i wątków: queen, hatch, bee.
- Bezpieczne zarządzanie tablicą wątków pszczół (z powiększaniem realloc).
- Obsługa pamięci współdzielonej, semaforów POSIX i SysV.
- Dynamiczna modyfikacja capacity (sygnały SIGUSR1, SIGUSR2).

- Mechanizm ograniczający kilkukrotną nieudaną próbę utworzenia wątku (jeśli errno == EAGAIN).
- Kolorowe logi i wyświetlanie stanu ula.

Z czym były problemy:

- Kolejność logów (asynchronizacja wątków i procesów) powoduje „przeplatanie" komunikatów.
- Zmniejszenie capacity nie wyrzuca pszczół już znajdujących się w ulu – część pszczół może więc przekroczyć nowy limit do czasu, aż same wyjdą.
- „Ostatnie pszczoły" mogą jeszcze chwilę działać po wywołaniu cleanup(), bo w praktyce czekamy aż wszystkie wątki się zakończą (anulowanie i pthread_join).

Linki do fragmentów kodu:

- **Tworzenie procesów:**

fork():

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/main.c#L158C4-L158C24

wait():

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/cleanup.c#L45C9-L45C37

exit():

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/bee.c#L63C14-L63C36

- Tworzenie i obsługa wątków:

pthread_create():

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/bee.c#L47C5-L47C52

pthread_join():

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/cleanup.c#L74C9-L74C44

pthread_mutex_lock():

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/cleanup.c#L66

pthread_mutex_unlock():

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/cleanup.c#L80C1-L80C43

- Obsługa sygnałów:

sigaction()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/main.c#L90C1-L90C44

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/beekeeper.c#L44

kill()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/cleanup.c#L44C8-L44C34

- Semafory SysV:

semget()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/egg.c#L33C5-L33C70

semctl()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/egg.c#L61

semop()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/egg.c#L151

- Semafory POSIX:

seminit()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/main.c#L134C1-L143C39

sem_trywait()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/hive.c#L31C8-L34C14

- Obsługa pamięci współdzielonej:

shmget()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/egg.c#L33

shmctl()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/egg.c#L74C1-L74C40

- Obsługa błędów

perror()

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/main.c#L92

errno() (EAGAIN)

https://github.com/Elleass/SO-Projekt/blob/1df356fa0ab6aa656ef2e6ecd1eddb7ce0083702/src/bee.c#L55

Testy:

1. Test wysokich wartości (ponad maksymalną wartość wątków)

Cel: Sprawdzenie czy w przypadku przekroczenia limitu dopuszczalnych wątków program wyłączy się w sposób przewidywalny.

```
...
Failed to create bee thread (errno=11)
Failed to create bee thread (errno=11)
Failed to create bee thread (errno=11)
Repeated thread creation failures. Stopping program.
Pszczoła 4448:  wchodzi do ula wejściem 1.
Wolna przestrzeń: 3526 (zajęte: 474). Bees: 474, Eggs: 0
Pszczoła 4449:  czeka, oba wejścia są zajęte.
Pszczoła 4214 startuje w wątku.
Pszczoła 4214 kończy życie.
```

```
Pszczoła 4216 startuje w wątku.
Pszczoła 4216 kończy życie.
Pszczoła 4452 startuje w wątku.
Pszczoła 4452 kończy życie.
Pszczoła 4032 startuje w wątku.
Pszczoła 4032 kończy życie.
Pszczoła 4453:  czeka, oba wejścia są zajęte.
Pszczoła 4218 startuje w wątku.
...
Queen process terminated.
Terminating hatch process (PID: 131278)...
Hatch process terminated.
Canceling beekeeper thread...
Beekeeper thread terminated.
.
Pszczoła 4623 kończy życie.
Pszczoła 4041 startuje w wątku.
Pszczoła 4041 kończy życie.
Pszczoła 4655 kończy życie.
Bee threads cleaned up.
Destroying semaphores...
Semaphores destroyed.
Destroying egg queue...
Egg queue destroyed.
Cleanup completed. Exiting.
```

Po trzykrotnym wywołaniu się: Failed to create bee thread (errno=11) program wciąż tworzy zakolejkowane pszczoły które od razu umierają, dzieje się tak ponieważ program wpierw musi zamknąć procesy queen oraz hatch które mogły by tworzyć nowe pszczoły, następnie wykonuje się ciąg dalszy funkcji cleanup co pozwala zamknąć program w pełni kontrolowany sposób.

2. Test prawie maksymalnej wartości (czy program usuwa martwe pszczoły) [Zaaktualizowane]

Cel: Sprawdzenie, czy martwe pszczoły wciąż zajmują miejsce wątków i czy program osiągnie maksymalną wartość wątków z powodu nowych pszczół.
Przykładowe parametry wejściowe:
-N=4750(łącznie pszczół “robotnic”)
-P=2000początkowa maksymalna pojemność ula, P<N/2P<N/2>

```
Pszczoła 4826: startuje w wątku.
[DEBUG] Bee 4826 acquired wejscie1_kierunek
Pszczoła 4826: wchodzi do ula wejściem 1.
[DEBUG] Bee 4826 posted wejscie1_kierunek
Wolna przestrzeń: 2199 (zajęte: 1). Bees: 1, Eggs: 0
Królowa: Złożono jajo ID: 4827 (wykluje się za 5 s)
Pszczoła 4825: kończy życie.
[DEBUG] Bee 4826 acquired wejscie1_kierunek
Pszczoła 4826: wychodzi z ula wejściem 1.
[DEBUG] Bee 4826 posted wejscie1_kierunek
Wolna przestrzeń: 2199 (zajęte: 1). Bees: 0, Eggs: 1
Wykluwanie: Jajko ID: 4827 wykluło się w pszczołę!
[DEBUG] occupant_count=0, egg_count=0 ⇒ bee_count=0
Pszczoła 4827: startuje w wątku.
[DEBUG] Bee 4827 acquired wejscie1_kierunek
Pszczoła 4827: wchodzi do ula wejściem 1.
[DEBUG] Bee 4827 posted wejscie1_kierunek
Wolna przestrzeń: 2199 (zajęte: 1). Bees: 1, Eggs: 0
Królowa: Złożono jajo ID: 4828 (wykluje się za 5 s)
```

Wniosek: ID pszczoły wykroczyło po za limit z poprzedniego testu co pokazuję, że pszczoła po śmierci zwalnia swoje miejsce pozwalają progamowi działać w nieskończoność

3. Test drastycznej zmiany pojemności ula [Zaaktualizowane]

Cel: sprawdzenie zachowania ula w przypadku zmniejszenia jego pojemności do minimalnej wartości
Przykładowe parametry wejsciowe:
N = 100
P = 49

```
Pszczoła 20: czeka - ul pełny.
Zmniejszanie pojemności ula z 6 do 3
Zmniejszanie pojemności ula z 3 do 1
[DEBUG] Bee 11 acquired wejscie1_kier
.
.
.
Pszczoła 51: czeka, oba wejścia są zajęte.
Pszczoła 69: wychodzi z ula wejściem 1.
```

```
[DEBUG] Bee 69 posted wejscie1_kierunek
Ul jest pełny. | Bees: 14, Eggs: 0
[DEBUG] Bee 68 acquired wejscie2_kierunek
[DEBUG] Bee 68 posted wejscie2_kierunek
Pszczoła 68: czeka - ul pełny.
[DEBUG] Bee 81 acquired wejscie1_kierunek
[DEBUG] Bee 81 posted wejscie1_kierunek
Pszczoła 81: czeka - ul pełny.
[DEBUG] Bee 26 acquired wejscie1_kierunek
Pszczoła 26: wychodzi z ula wejściem 1.
[DEBUG] Bee 26 posted wejscie1_kierunek
Ul jest pełny. | Bees: 13, Eggs: 0
[DEBUG] Bee 91 acquired wejscie1_kierunek
.
.
.
Pszczoła 83: wychodzi z ula wejściem 1.
[DEBUG] Bee 83 posted wejscie1_kierunek
Wolna przestrzeń: 1 (zajęte: 0). Bees: 0, Eggs: 0
[DEBUG] Bee 95 acquired wejscie1_kierunek
Pszczoła 95: wchodzi do ula wejściem 1.
[DEBUG] Bee 95 posted wejscie1_kierunek
Ul jest pełny. | Bees: 1, Eggs: 0
Pszczoła 97: czeka, oba wejścia są zajęte.
[DEBUG] Bee 80 acquired wejscie2_kierunek
[DEBUG] Bee 80 posted wejscie2_kierunek
Pszczoła 80: czeka - ul pełny.
```

Wniosek: Pszczoły znajdujące się w ulu w trakcie wywołania jego zmniejszenia nie są z niego wyrzucane, jednak nowe pszczoły nie są dopuszczane do wejścia. Po opuszczeniu ula przez nadmiarowe pszczoły ul funkcjonuje już poprawnie.

4. Test długiego czasu działania programu

Cel: Sprawdzenie czy po długim czasie działania programu symulacja wciąż będzie działać:
Przykładowe parametry wejściowe:
N = 1000
P = 499

```
Wolna przestrzeń: 498 (zajęte: 1). Bees: 0, Eggs: 1
Pszczoła 1085: kończy życie.
```

```
Wykluwanie: Jajko ID: 1087 wykluło się w pszczołę!
[DEBUG] occupant_count=0, egg_count=0 ⇒ bee_count=0
Pszczoła 1087: startuje w wątku.
[DEBUG] Bee 1087 acquired wejscie1_kierunek
Pszczoła 1087: wchodzi do ula wejściem 1.
[DEBUG] Bee 1087 posted wejscie1_kierunek
Wolna przestrzeń: 498 (zajęte: 1). Bees: 1, Eggs: 0
Królowa: Złożono jajo ID: 1088 (wykluje się za 5 s)
[DEBUG] Bee 1087 acquired wejscie1_kierunek
Pszczoła 1087: wychodzi z ula wejściem 1.
[DEBUG] Bee 1087 posted wejscie1_kierunek
Wolna przestrzeń: 499 (zajęte: 0). Bees: 0, Eggs: 0
Wykluwanie: Jajko ID: 1088 wykluło się w pszczołę!
[DEBUG] occupant_count=0, egg_count=0 ⇒ bee_count=0
Pszczoła 1088: startuje w wątku.
[DEBUG] Bee 1088 acquired wejscie1_kierunek
Pszczoła 1088: wchodzi do ula wejściem 1.
```

Wniosek: Po godzinie działania program wciąż funkcjonuje bez zarzutu.

Problem: Brak możliwości dostosowania liczby jaj składanych przez królową co powoduje powolne powstawanie nowych pszczół

5. Test krótkiej żywotności pszczół

pszczoła ginie po pierwszym wyjściu z ula. Cel: sprawdzenie , czy occupant_count jest poprawnie dekrementowany i czy wątki „umierają” właściwie.

```
Pszczoła 9:  wychodzi z ula wejściem 1.
Wolna przestrzeń: 1 (zajęte: 3). Bees: 3, Eggs: 0
Pszczoła 8:  wychodzi z ula wejściem 1.
Wolna przestrzeń: 2 (zajęte: 2). Bees: 2, Eggs: 0
Pszczoła 5:  wychodzi z ula wejściem 1.
Wolna przestrzeń: 3 (zajęte: 1). Bees: 1, Eggs: 0
Pszczoła 6:  wychodzi z ula wejściem 1.
Wolna przestrzeń: 4 (zajęte: 0). Bees: 0, Eggs: 0
Pszczoła 0:  wchodzi do ula wejściem 1.
Wolna przestrzeń: 3 (zajęte: 1). Bees: 1, Eggs: 0
Pszczoła 4:  wchodzi do ula wejściem 1.
Wolna przestrzeń: 1 (zajęte: 3). Bees: 3, Eggs: 0
Pszczoła 7:  wchodzi do ula wejściem 2.
Wolna przestrzeń: 1 (zajęte: 3). Bees: 3, Eggs: 0
```

```
Pszczoła 0: kończy życie.
Pszczoła 3: kończy życie.
Pszczoła 1: kończy życie.
Pszczoła 2: kończy życie.
Pszczoła 0: wychodzi z ula wejściem 1.
Wolna przestrzeń: 2 (zajęte: 2). Bees: 2, Eggs: 0
Pszczoła 4: wychodzi z ula wejściem 1.
Wolna przestrzeń: 3 (zajęte: 1). Bees: 1, Eggs: 0
Pszczoła 7: wychodzi z ula wejściem 1.
Wolna przestrzeń: 4 (zajęte: 0). Bees: 0, Eggs: 0
Pszczoła 9: kończy życie.
Pszczoła 8: kończy życie.
Pszczoła 5: kończy życie.
Pszczoła 6: kończy życie.
Krolowa: Zlozono jajo ID: 1 (wykluje sie za 5 s)
Wykluwanie: Jajko ID: 1 wyklulo sie w pszczole!
Pszczoła 1: startuje w wątku.
Pszczoła 1: wchodzi do ula wejściem 1.
Wolna przestrzeń: 3 (zajęte: 1). Bees: 1, Eggs: 0
Pszczoła 0: kończy życie.
Pszczoła 4: kończy życie.
Pszczoła 7: kończy życie.
```

occupant_count spada do zera, gdy wszyscy wyjdą.
Pszczoły z krótkim życiem nie powracają do ula drugi raz.
Królowa i proces wylęgania nadal mogą generować nowe pszczoły (z nowymi ID).
Napotkany problem: Kolejność pojawiania się logów nie jest precyzyjna przez co zachowanie programu może stawać się nie czytelne. Zanim krolowa zlozy pierwsze jajo pszczoly umieraja przez co krolowa tworzy pszczole z ID martwej pszczoly.

6. Napotkane błędy i ich rozwiąznie

Problem: Nowo wyklute pszczoły nie mogą wejść do ula z powodu fałszywego odczytu dostępności wejść.

Przyczyna: Semafory lokalne (z pshared=0) nie były faktycznie współdzielone między procesem głównym a procesami potomnymi, więc nowo wyklute pszczoły (działające w innym procesie) nie widziały aktualnego stanu semaforów.

Rozwiązanie: Przeniesienie semaforów do pamięci współdzielonej i inicjalizacji z pshared=1 powoduje, że wszyscy uczestnicy (proces główny, procesy queen/hatch) operują na tych samych strukturach semaforów i widzą ich prawidłowe wartości.

https://github.com/Elleass/SO-Projekt/blob/54bb9364f6eb03b2179726c68cd89e8c1dce6b84/src/main.c#L141C4-L155C1

7. Podsumowanie

- Projekt pozwala na sprawdzenie mechanizmów synchronizacji i komunikacji międzyprocesowej (pamięć współdzielona, semafory, sygnały).
- Udało się zrealizować wszystkie główne założenia: proces królowej i wylęgania, wątki pszczół, wątek pszczelarza reagujący na sygnały.
- Obsługa wielu wątków i procesów generuje asynchronię w logach, co jest normalne w aplikacjach wielowątkowych.
- Główne ograniczenie to brak wyrzucania pszczół z ula, jeśli zmniejszymy capacity „w locie". Można to uznać za rozszerzenie na przyszłość.
- Testy pokazały, że mechanizmy zatrzymania (SIGINT) i sprzątania zasobów działają stabilnie