Systemy Operacyjne

Projekt nr 2: Zarządzanie procesami i pamięcią współdzieloną

Temat: Symulacja wieloprocesowego systemu obsługi restauracji w środowisku Linux

Autor: Yahor Kalesnikau
Nr Indeksu: 282646

Rok akademicki 2025/2026

Spis treści

1 Wstęp

Celem projektu było stworzenie symulacji działania restauracji przy użyciu mechanizmów systemowych rodziny UNIX/Linux. Głównym założeniem było wykorzystanie procesów (funkcja `fork()`) zamiast wątków, co wymusiło zastosowanie pamięci współdzielonej (Shared Memory) oraz mechanizmów synchronizacji międzyprocesowej (IPC).

Program symuluje złożone środowisko, w którym występują zasoby o różnej charakterystyce (stałe, odnawialne, przenośne), a niezależne procesy (Klienci, Dostawca, Pomywacz) rywalizują o dostęp do nich.

1.1 Wymagania i założenia

Symulacja musiała obsługiwać:

- **Zasoby stałe:** Stoły o różnych rozmiarach (2, 4, 6-osobowe).
- **Zasoby odnawialne:** Składniki spożywcze (Warzywa, Mięso, Chleb) oraz naczynia jednorazowe.
- **Zasoby przenośne:** Sztućce metalowe (Widelce, Noże, Łyżki), które przechodzą cykl: *Czyste* $\rightarrow$ *Używane* $\rightarrow$ *Brudne* $\rightarrow$ *Mycie*.
- **Wizualizację:** Graficzny interfejs w terminalu (biblioteka `ncurses`).

2 Architektura Systemu

2.1 Pamięć Współdzielona (Shared Memory)

Ze względu na izolację pamięci procesów w systemie Linux, do komunikacji wykorzystano mechanizm `mmap` z flagami `MAP_SHARED | MAP_ANONYMOUS`. Pozwoliło to na stworzenie struktury `SharedState`, dostępnej dla wszystkich procesów potomnych.

Struktura ta przechowuje:

1. **Muteks międzyprocesowy:** `pthread_mutex_t` skonfigurowany atrybutem `PTHREAD_PROCESS_SHARED`, chroniący sekcję krytyczną (dostęp do magazynu).
2. **Liczniki zasobów:** Ilość wolnych stołów, stan magazynu produktów, ilość czystych i brudnych sztućców.
3. **Statystyki:** Liczniki obsłużonych klientów, odrzuceń oraz zużycia produktów.
4. **Zmienne sterujące:** Flagi kontrolne dla dostawcy i paska postępu.

2.2 Procesy Składowe

System składa się z następujących procesów równoległych:

- **Proces Główny (Generator):** Tworzy procesy potomne (klientów) w losowych odstępach czasu.
- **Proces Wizualizatora:** Odczytuje stan pamięci współdzielonej (z minimalnym blokowaniem) i rysuje interfejs przy użyciu `ncurses`.
- **Proces Dostawcy:** Cyklicznie uzupełnia zapasy w magazynie.
- **Proces Pomywacza:** Przetwarza brudne sztućce z powrotem na czyste.
- **Procesy Klientów:** Krótkotrwałe procesy symulujące konsumpcję posiłku.

3 Szczegółowy Opis Implementacji i Logiki

3.1 Logika Dostawcy (Supplier)

Dostawca jest procesem działającym w pętli nieskończonej. Jego działanie opiera się na dwóch trybach pracy, wybieranych przy starcie programu.

Tryby pracy:

1. **Tryb Stały (Fixed):** Dostawca zawsze przywozi ilość towaru równą 50% maksymalnej pojemności magazynu. Jest to strategia prosta, ale może prowadzić do przepełnienia (nadmiar jest ignorowany) lub niedoborów przy dużym ruchu.
2. **Tryb Inteligentny (Smart):** Dostawca zapamiętuje, ile produktów brakowało podczas poprzedniej wizyty.
   - Przyjazd: Uzupełnia zapasy o wartość `next_order`.
   - Analiza: Oblicza różnicę $Braki = Max - Obecne$.
   - Planowanie: Ustawia zmienną `next_order` na wartość $Braki$ dla przyszłej dostawy.

   Ta strategia pozwala na dynamiczne dostosowanie podaży do popytu.

3.2 Logika Pomywacza (Dishwasher)

Pomywacz działa w tle i reaguje na pojawienie się brudnych naczyń.

- Proces sprawdza liczniki `dirty_forks`, `dirty_knives`, `dirty_spoons`.
- Jeśli > 0, następuje dekrementacja licznika "brudne"i inkrementacja "czyste".
- Operacja jest opóźniona o czas mycia (`dish_speed_us`), co sprawia, że przy dużym ruchu może zabraknąć czystych sztućców, nawet jeśli mamy dużo brudnych.

3.3 Złożona Logika Klienta i Menu

Klienci nie są jednorodni. Zaimplementowano zaawansowaną logikę wyboru zamówienia, która determinuje zużycie zasobów.

Każda nowa grupa losuje typ wizyty w oparciu o parametr `takeout_chance`.

A. Zamówienie na Wynos:

- **Zasoby:** Nie wymaga stołu ani metalowych sztućców.
- **Wymagania:** Dostępność sztućców jednorazowych (`cnt_disposable`) oraz składników.
- **Menu:**
  - Wariant 1: Mięso + Chleb.

  - Wariant 2: Mięso + Warzywa.

- Jeśli zasoby są dostępne, są one natychmiast pobierane, a statystyki aktualizowane. Proces nie musi czekać (symulacja pakowania jest natychmiastowa w kontekście blokowania zasobów).

B. Zamówienie na Sali: Wymaga znalezienia odpowiedniego stołu. Grupa N-osobowa szuka stołu o rozmiarze $\geq N$.

Menu na sali (Zróżnicowanie sztućców): Losowany jest rodzaj posiłku, co wpływa na to, jakie przybory zostaną zablokowane:

1. **Zupa:**
   - Składniki: Warzywa + Chleb.
   - Sztućce: **Tylko Łyżki**. Widelce i noże pozostają wolne dla innych.
2. **Danie Główne:**
   - Składniki: Mięso + Warzywa.
   - Sztućce: **Widelce + Noże**. Łyżki pozostają wolne.

Dzięki temu rozwiązaniu, brak noży nie blokuje klientów chcących zjeść zupę.

4 Automatyzacja Testów i Analiza Wyników

Do weryfikacji działania systemu napisano skrypt w języku Bash, który automatycznie uruchamia symulację z zadanymi parametrami, parsuje wyjście i oblicza średnie wyniki z wielu prób.

Przeprowadzono 4 scenariusze testowe (każdy po 50 powtórzeń, czas trwania próby: 30 sekund).

4.1 Scenariusz 1: Normal Day (Dzień Powszedni)

Konfiguracja: Zrównoważona ilość stołów, duży magazyn, standardowa ilość sztućców.

- **Średnio zamówień na sali:** 19 (Odrzucono: 37)
- **Sriednia ilość obsłużenych ludzi na sali:** 59
- **Średnio zamówień na wynos:** 28 (Odrzucono: 0)
- **Sriednia ilość obsłużenych ludzi na wynos:** 102
- **Zużycie:** Warzywa: 60, Mięso: 140, Chleb: 124, Sztućce jednorazowe: 102.
- **Umyto sztuccow:** 54

Wnioski: System działa stabilnie. 100% skuteczność obsługi na wynos (brak wymogu stołów). Odrzucenia na sali wynikają z naturalnego zajęcia wszystkich stołów. Pomywacz nadąża z myciem naczyń.

4.2 Scenariusz 2: Cutlery Crisis (Kryzys Sztućców)

Konfiguracja: Bardzo mało sztućców metalowych (po 2 sztuki każdego typu), powolne mycie.

- **Średnio zamówień na sali:** 9 (Odrzucono: 49)
- **Sriednia ilość obsłużenych ludzi na sali:** 12
- **Średnio zamówień na wynos:** 28 (Odrzucono: 0)
- **Sriednia ilość obsłużenych ludzi na wynos:** 100
- **Zużycie:** Warzywa: 12, Mięso: 106, Chleb: 107, Sztucce jednorazowe: 100.
- **Umyto sztuccow:** 15

Wnioski: Drastyczny spadek obsługi na sali (z 59 do 12 osób). Jest to dowód na poprawną implementację zależności zasobów. Brak sztućców blokuje stoliki, nawet jeśli są wolne. Co ważne, **kryzys na sali nie wpłynął na obsługę na wynos** (nadal 100 osób obsłużonych), co potwierdza izolację logiczną tych dwóch typów zamówień.

4.3 Scenariusz 3: Fast Food Takeout

Konfiguracja: Tylko po 1 stole, ale ogromny zapas naczyń jednorazowych i bardzo szybki napływ klientów (co 0.1s).

- **Średnio zamówień na sali:** 10 (Odrzucono: 10)
- **Sriednia ilość obsłużenych ludzi na sali:** 33
- **Średnio zamówień na wynos:** 113 (Odrzucono: 64)
- **Sriednia ilość obsłużenych ludzi na wynos:** 371
- **Zużycie:** Warzywa: 46, Mięso: 387, Chleb: 376, Sztucce jednorazowe: 371.
- **Umyto sztuccow:** 41

Wnioski: Test wydajnościowy. System obsłużył ponad 400 osób w 30 sekund. Wąskim gardłem stała się prędkość dostawcy (duże zużycie mięsa), a nie procesy klienta.

4.4 Scenariusz 4: Supply Issues (Problemy z Dostawami)

Konfiguracja: Bardzo powolny dostawca (czas podróży 8s) i mały magazyn.

- **Średnio zamówień na sali:** 7 (Odrzucono: 50)
- **Sriednia ilość obsłużenych ludzi na sali:** 17
- **Średnio zamówień na wynos:** 6 (Odrzucono: 20)
- **Sriednia ilość obsłużenych ludzi na wynos:** 16
- **Zużycie:** Warzywa: 19, Mięso: 24, Chleb: 24, Sztucce jednorazowe: 16.
- **Umyto sztuccow:** 25

Wnioski: Brak produktów paraliżuje pracę całej restauracji – zarówno sali, jak i wynosów. To pokazuje, że "Warzywa"i "Mięso"są zasobami krytycznymi dla obu ścieżek obsługi.

5 Podsumowanie i Wnioski

Stworzono stabilną, wieloprocesową aplikację symulującą działanie restauracji.
Kluczowe osiągnięcia:

1. **Synchronizacja:** Zastosowanie muteksów w pamięci współdzielonej skutecznie zapobiegło zjawisku wyścigu (Race Condition). Dane w logach są spójne.
2. **Zarządzanie zasobami:** Implementacja rozróżnia zasoby blokujące (stoły) od zużywalnych (jedzenie) i odnawialnych (sztućce).
3. **Złożoność biznesowa:** Wprowadzenie różnych typów dań (Zupa vs Drugie) oraz opcji "Na Wynos"uczyniło symulację nietrywialną i bardziej realistyczną.
4. **Narzędzia:** Stworzenie skryptu automatyzującego testy pozwoliło na szybką weryfikację zachowania systemu w warunkach brzegowych.

Projekt ten pozwolił na głębsze zrozumienie mechanizmów IPC w systemie Linux oraz problemów związanych z współbieżnością, takich jak zakleszczenia (deadlock) i zagłodzenie procesów (starvation).