1. Założenia projektowe.

Celem projektu było stworzenie symulacji działania sklepu dyskontowego w środowisku Linuxa. Projekt zakładał wykorzystanie procesów oraz komunikacji międzyprocesowej.

Kluczowe założenia projektu:

- brak centralizacji (każda kasa, klient, obsługa są osobnymi procesami)
- synchronizacja dostępu do zasobów za pomocą semaforów
- wykorzystane fork() i exec() do uruchamiania modułów
- wymiana danych za pomocą kolejek komunikatów
- współdzielenie stanu sklepu przy użyciu pamięci dzielonej

2. Ogólny opis kodu.

Kod został podzielony na niezależne pliki źródłowe, co zapewnia czytelność oraz modularność:

- main.c – inicjalizuje zasoby komunikacji międzyprocesowej, uruchamia procesy pracowników oraz generatora klientów
- kierownik.c – monitoruje stan kolejek przy użyciu pamięci dzielonej i reaguje na sygnały systemowe, decydując o otwarciu/zamknięciu kas bądź ewakuacji
- kasjer.c – proces obsługujący logikę kasowania produktów przy kasie stacjonarnej, komunikujący się z klientem przy pomocy kolejki komunikatów
- kasa_samoobslugowa.c – proces obsługujący logikę kasowania produktów przy kasie samoobsługowej, komunikujący się z klientem przy pomocy kolejki komunikatów
- klient.c – proces symulujący zachowanie klienta; wybór produktów i kolejki
- obsluga.c – proces reagujący na awarie/alkohol przy kasach samoobsługowych
- utils.c, common.h – biblioteki pomocnicze zawierające definicje struktur, funkcje logowania i obsługi kolorów w terminalu

3. Co udało się zrobić.

Zrealizowano wszystkie wymagania funkcjonalne i niefunkcjonalne:

- pełna separacja procesów przy użyciu funkcji execv
- obsługa sygnałów: sterowanie ręczne kasami (SIGUSR1 – otwieranie kasy stacjonarnej 2, SIGUSR2 – zamykanie danej kasy stacjonarnej, SIGQUIT – ogłoszenie ewakuacji przez kierownika)
- implementacja logiki biznesowej: wchodzenie klientów, zajmowanie kas, obsługa awarii i alkoholu
- walidacja danych wejściowych wpisywanych przez użytkownika

- generowanie logów (raport.txt) oraz paragonów (paragony.txt)

4. Napotkane problemy.

- Zombie procesy: Początkowo procesy potomne pozostawały w stanie zombie. Dodano obsługę waitpid z flagą WNOHANG w pętli generatora oraz zbiorcze czyszczenie w funkcji sprzataj().
- Zakleszczenie przy wyborze kolejki: zidentyfikowałem problem w procesie klienta, gdzie proces trzymał opuszczony semafor podczas funkcji usleep, a następnie próbował opuścić go ponownie co prowadziło do zawieszenia procesu. Rozbiłem sekcje krytyczną na wiele mniejszych bloków co rozwiązało problem.
- Przy zamykaniu kas występował problem, w którym kasjer kończył pracę natychmiast po zakończeniu pracy mimo, że w kolejce stali klienci. Wprowadziłem tryb wygaszania, który pozwala obsłużyć klientów w kolejce do końca przed ostatecznym zamknięciem.
- Wyścig o dostęp do terminala: ze względu na asynchroniczność procesów, komunikaty logów pojawiały się czasami w nielogicznej kolejności (np. najpierw był komunikat, że kasa S1 obsługuje klienta, a potem, że się otwiera). Przebudowałem sekcje krytyczne w pętli głównej kasjera co zapewniło chronologiczną kolejność wyświetlania komunikatów.

5. Dodane elementy specjalne.

- kolorowanie wyjścia terminala – zastosowanie kodów ANSI dla łatwiejszego wizualnego odróżnienia procesów w logach
- dynamiczna identyfikacja – w logach wyświetlany jest zarówno porządkowy numer klienta jak i jego PID
- drukowanie paragonów w osobnym pliku – paragon każdego z klientów zostaje zapisany w pliku paragony.txt z dokładną listą zakupów, ceną za artykuł i zapłaconą sumą

6. Przeprowadzone testy i weryfikacja (brak większych problemów)

- Stress test: Symulacja jednoczesnego wejścia 200 klientów w bardzo krótkim czasie. System poprawnie obsłużył kolejkowanie bez naruszenia pamięci współdzielonej i bez powstawania procesów zombie.
- Test nagłej ewakuacji: Weryfikacja przy pełnym obciążeniu sklepu. Potwierdzono, że wszystkie procesy potomne zakończyły się poprawnie, a zasoby IPC są dokładnie czyszczone.
- Test przebiegły pomyślnie. Jedyną trudnością jest losowość symulacji – czasy są generowane losowo, co sprawia, że w testach manualnych czasami trzeba poczekać dłużej na wystąpienie konkretnego zdarzenia np. awarii.

7. Linki do kodu.

Poniżej znajdują się odnośniki do fragmentów kodu w repozytorium GitHub, obrazujące użycie wymaganych funkcji systemowych.

a. tworzenie i obsługa plików:
   - funkcje: fopen, fprintf, fclose
   - linki:

     https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/utils.c#L40-L44

b. tworzenie procesów:
   - funkcje: fork, execv, exit, wait
   - linki:

     https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L104-L118

     https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L36

c. obsługa sygnałów:
   - funkcje: signal, kill
   - linki:

     https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/kierownik.c#L24-L26

     https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L31

d. synchronizacja procesów (semafory):
   - funkcje: semget, semop, semctl
   - linki:

     https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L174

     https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L235

     https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L40

e. segmenty pamięci dzielonej:
   - funkcje: shmget, shmat, shmctl
   - linki:

https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L164-L167

https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L39

f. kolejki komunikatów:
   - funkcje: msgget, msgsnd, msgrcv, msgctl
   - linki:

https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/klient.c#L12

https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/klient.c#L193

https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/klient.c#L218

https://github.com/Sz0p3ro/SO-2025-2026/blob/e31d4b0b96aee0cfccb5acf948f34e2ac1536d82/main.c#L41