Systemy operacyjne

Salon fryzjerski – projekt numer 17.

Opis problemu.

W salonie pracuje **F** fryzjerów (**F > 1**) i znajduje się w nim **N** foteli (**N < F**). Salon jest czynny w godzinach od **Tp** do **Tk**. Klienci przychodzą do salonu w losowych momentach czasu. W salonie znajduje się poczekalnia, która może pomieścić **K** klientów jednocześnie.

Każdy klient rozlicza usługę z fryzjerem przekazując mu kwotę za usługę przed rozpoczęciem strzyżenia. Fryzjer wydaje resztę po zakończeniu obsługi klienta, a w przypadku braku możliwości wydania reszty, klient musi zaczekać, aż fryzjer znajdzie resztę w kasie. Kasa jest wspólna dla wszystkich fryzjerów. Płatność może być dokonana banknotami o nominałach **10 zł**, **20 zł** i **50 zł**.

Zasada działania **fryzjera** (*cyklicznie*):

1. Wybiera klienta z poczekalni (ewentualnie czeka, jeśli go jeszcze nie ma).
2. Znajduje wolny fotel.
3. Pobiera opłatę za usługę, umieszcza ją we wspólnej kasie, wydaje resztę (w przypadku braku możliwości jej wydania informuje klienta)
4. Realizuje usługę.
5. Zwalnia fotel.

Zasada działania **klienta** (*cyklicznie*):

1. Zarabia pieniądze (gdy znajduje się po za salonem)
2. Przychodzi do salonu fryzjerskiego i siada w poczekalni czekając na fryzjera - jeżeli jest miejsce, jeżeli nie opuszcza salon i wraca do zarabiania pieniędzy.
3. Gdy jest wezwany przez fryzjera płaci za próbuje płacić za usługę (jeżeli nie ma pieniędzy opuszcza salon).
4. Jest strzyżony.
5. Opuszcza salon i wraca do zarabiania pieniędzy.

Zasada działania **kierownika**:

1. W losowym momencie czasu może zwolnić fryzjera (Sygnał 1).
2. W losowym momencie ewakuuje z poczekalni klientów, którzy wracają do zarabiania pieniędzy (Sygnał 2).

Fryzjer.

Inicjalizacja procesu fryzjera:

Na początku ustawiam nazwę procesu **prctl(PR_SET_NAME, "fryzjer", 0, 0, 0)**, co umożliwi mi łatwą identyfikację procesu w systemie (np. w przypadku wywołania ps). Proces fryzjera działa w nieskończonej pętli, symulując jego ciągłą dostępność w salonie fryzjerskim.

Oczekiwanie na klienta:

Fryzjer czeka na klienta korzystając z semaforów:

- **operacja_semaforowa(semafor, 2, -1)** – fryzjer blokuje semafor obsługujący kolejkę klientów, co oznacza, że przygotowuje się do przyjęcia klienta.
- **operacja_semaforowa(semafor, 1, -1)** – fryzjer blokuje semafor obsługujący zasoby pracy fryzjera, sygnalizując, że jest gotowy do pracy.

Odbiór informacji o kliencie:

Fryzjer odbiera wiadomość z kolejki komunikatów przy użyciu funkcji **msgrcv**:

- Wiadomość zawiera informacje o kliencie, takie jak jego identyfikator (**wiad.id_klienta**) oraz czas, jaki fryzjer będzie poświęcał na strzyżenie (**wiad.czas**).
- W przypadku błędu odbioru wiadomości wypisywany jest komunikat diagnostyczny wraz z kodem błędu **errno** i jego opisem.

Proces strzyżenia:

Po odebraniu informacji o kliencie, fryzjer symuluje strzyżenie:

- Wyświetlana jest informacja o rozpoczęciu strzyżenia z podaniem identyfikatora fryzjera oraz klienta.
- Czas trwania strzyżenia symulowany jest za pomocą funkcji **sleep(wiad.czas)**.

Zakończenie strzyżenia i informowanie klienta:

- Po zakończeniu strzyżenia fryzjer wysyła wiadomość zwrotną do klienta przy użyciu **msgsnd**, informując o zakończeniu usługi.
- W przypadku błędu wysyłania wiadomości również wypisywany jest odpowiedni komunikat diagnostyczny.

Zarządzanie kasą salonu:

Po zakończeniu obsługi klienta, fryzjer wyświetla aktualny stan kasy salonu:

- Wyliczana jest suma wszystkich banknotów w kasie za pomocą funkcji **oblicz_sume_banknotow**.
- Wyświetlane są informacje o liczbie banknotów każdego nominału oraz całkowita suma pieniędzy w kasie.

Odblokowanie zasobów:

Po zakończeniu obsługi klienta fryzjer zwalnia zablokowany wcześniej semafor zasobów fryzjera, używając **operacja_semaforowa(semafor, 1, 1)**.

Klient.

Inicjalizacja procesu klienta:

Proces klienta rozpoczyna działanie poprzez zainicjalizowanie swoich zasobów:

- Funkcja **inicjalizuj_klienta** ustawia początkowe parametry klienta, takie jak posiadane banknoty.
- Ustawienie nazwy procesu na "klient" za pomocą **prctl(PR_SET_NAME, "klient", 0, 0, 0)**, co ułatwia identyfikację w systemie.
- Klient przygotowuje się do obsługi sygnałów ewakuacyjnych, ustawiając obsługę sygnału **SIGUSR1** na funkcję **obsluz_sygnal**.

Obsługa sygnału ewakuacji

Jeśli klient odbierze sygnał **SIGUSR1**, funkcja **obsluz_sygnal** ustawia zmienną ewakuacja na wartość **1**, co inicjuje specjalne zachowanie:

- Klient zwalnia semafory salonu (semafor 0 i semafor 1), symulując swoją ewakuację.
- Klient odczekuje losowy czas (1–5 sekund), zanim spróbuje ponownie podjąć swoje działania.

Losowe oczekiwanie

Klient losowo odczekuje 1–3 sekundy pomiędzy próbami wejścia do salonu, symulując różne czasy przybycia.

Sprawdzanie stanu ewakuacji salonu

Klient sprawdza wartość semafora **semafor 4**: jeżeli wartość wynosi **0**, salon jest w trakcie ewakuacji. Klient odczekuje losowy czas (1–5 sekund) przed kolejną próbą.

Wejście do salonu

Klient próbuje wejść do salonu za pomocą operacji na semaforze **semafor 0**: jeżeli salon jest pełny (operacja blokuje się), klient opuszcza salon, odczekuje losowy czas (1–5 sekund), a następnie zarabia losowy banknot o losowym nominale.

Czekanie na fotel

Po wejściu do salonu klient sygnalizuje swoją obecność, zwiększając wartość semafora **semafor 2**. Klient czeka na dostępność fotela, co jest symulowane przez dalsze operacje semaforów i logikę salonu.

Sprawdzenie środków na opłatę za usługę

Klient oblicza sumę posiadanych banknotów: jeżeli kwota jest niewystarczająca do zapłaty za usługę (**KOSZT_USLUGI**) klient opuszcza salon, zwalniając zarezerwowane zasoby.

Płatność za usługę

Klient podejmuje próbę zapłaty za usługę przy użyciu funkcji **zaplac**:

- W przypadku sukcesu środki są przekazywane do kasy salonu.
- W przypadku problemów z płatnością klient opuszcza salon, zwalniając zajęte zasoby.

Obsługa strzyżenia

Klient wysyła wiadomość do fryzjera za pomocą kolejki komunikatów:

- Wiadomość zawiera identyfikator klienta oraz czas trwania usługi.

Klient oczekuje na wiadomość zwrotną od fryzjera, potwierdzającą zakończenie strzyżenia.

Zwolnienie zasobów salonu

Po zakończeniu obsługi klient zwalnia zajęte zasoby:

- Zwiększa wartość semaforów **semafor 1** i **semafor 0**, sygnalizując dostępność kolejnych zasobów.

Losowy czas do ponownej wizyty

Klient odczekuje losowy czas (1–30 sekund) przed rozpoczęciem kolejnej iteracji w symulacji.

Kierownik.

Inicjalizacja procesu kierownika

Funkcja rozpoczyna swoją pracę od:

- Ustawienia nazwy procesu na "kierownik" za pomocą **prctl(PR_SET_NAME, "kierownik", 0, 0, 0)**, co ułatwia identyfikację procesu w systemie.
- Inicjalizacji generatora liczb losowych przy użyciu **srand(time(NULL))**, aby zapewnić losowość działań kierownika.

Główna pętla działania

Kierownik działa w pętli nieskończonej, wykonując swoje obowiązki w regularnych odstępach czasu:

- Kierownik losowo odczekuje od 5 do 15 sekund przed podjęciem kolejnej akcji.

Losowe wybieranie akcji

W każdej iteracji pętli kierownik losowo wybiera jedną z dwóch możliwych akcji:

- **Akcja 0** – Zwolnienie fryzjera
- **Akcja 1** – Ewakuacja klientów

Zwolnienie fryzjera

Jeśli wybrana zostanie akcja zwolnienia fryzjera:

- Kierownik sprawdza, czy są aktywni fryzjerzy (**aktywni_fryzjerzy > 0**).
- Losuje jednego z aktywnych fryzjerów (**rand() % aktywni_fryzjerzy**) i oblicza jego PID.
- Wysyła sygnał **SIGTERM** do procesu fryzjera za pomocą funkcji **kill**:

W przypadku powodzenia:

- Wyświetla komunikat informujący o zwolnieniu fryzjera.
- Zmniejsza licznik aktywnych fryzjerów (**aktywni_fryzjerzy--**).
- Jeśli wszyscy fryzjerzy zostali zwolnieni (**aktywni_fryzjerzy == 0**):

  Wyświetla komunikat o zakończeniu symulacji.
  Wysyła sygnał **SIGINT** do wszystkich procesów (**kill(0, SIGINT)**), co kończy symulację.

W przypadku błędu (np. nieistniejącego PID) wyświetla odpowiedni komunikat.

Ewakuacja klientów

Jeśli wybrana zostanie akcja ewakuacji klientów:

- Kierownik wyświetla komunikat o rozpoczęciu ewakuacji i czyści kolejkę komunikatów klientów za pomocą **wyczysc_kolejke(kolejka)**.
- Zmniejsza wartość semafora **semafor 4** o 1, co blokuje dostęp klientów do salonu.
- Ustawia zmienną globalną **ewakuacja** na **1**, co sygnalizuje klientom konieczność ewakuacji.
- Symuluje trwanie ewakuacji przez odczekanie 5 sekund.
- Po zakończeniu ewakuacji:
  - Ustawia zmienną **ewakuacja** na **0**.
  - Przywraca dostęp klientów do salonu, zwiększając wartość semafora **semafor 4** o 1.
- Wyświetla komunikat informujący o zakończeniu ewakuacji.

Mechanizmy synchronizacji i zarządzania.

Semafory:

- Semafor 0 - zarządzanie dostępnością miejsc w poczekalni
- Semafor 1 - zarządzanie dostępnością foteli fryzjerskich
- Semafor 2 - synchronizuje komunikację między klientem a fryzjerem (gotowość do strzyżenia).
- Semafor 3 - zapewnia wyłączny dostęp do zasobów współdzielonych (kasa)
- Semafor 4 - zapewnia blokowanie przychodzenia do salonu w trakcie ewakuacji

Sygnały:

- SIGTERM – Sygnał wysyłany do procesu fryzjera w celu zakończenia jego działania.
- SIGINT – Sygnał wysyłany do wszystkich procesów w celu zakończenia symulacji.
- SIGUSR1 – Sygnał wysyłany podczas ewakuacji klientów z salonu.

Kolejka komunikatów:

- Wykorzystana do komunikacji między procesami klientów i fryzjerów.
- Klient przesyła komunikat zawierający swoje dane, a fryzjer odsyła potwierdzenie zakończenia usługi.

Pamięć współdzielona:

- Wykorzystywana do przechowywania stanu kasy z dostępem kontrolowanym przez semafor 3.

Pliki nagłówkowe oraz main.

funkcje.h:

Definiowanie stałych:

- Parametry salonu: liczba fryzjerów, klientów, foteli, miejsc w poczekalni.
- Parametry finansowe: nominały banknotów, początkowy stan kasy, koszt usługi.
- Parametry czasu: godziny otwarcia i zamknięcia, długość godziny w sekundach.

Struktury danych:

- **Kasa**: stan kasy (liczba banknotów różnych nominałów).
- **Klient**: środki finansowe klienta.
- **Wiadomosc**: komunikacja klient-fryzjer (ID klienta, czas usługi).

Deklaracje funkcji:

- Zarządzanie zasobami (**inicjalizuj_zasoby**, **zwolnij_zasoby**).
- Obsługa płatności (**zaplac**, **dodaj_banknoty**).
- Pomocnicze (**oblicz_sume_banknotow**, **wyczysc_kolejke**).

funkcje.c:

Inicjalizacja zasobów:

- Tworzenie semaforów, pamięci współdzielonej (kasa) i kolejki komunikatów (poczekalnia).
- Ustawianie początkowych wartości semaforów i kasy.

Operacje zarządzające:

- Zarządzanie semaforami (**operacja_semaforowa**).
- Obsługa płatności: sprawdzanie środków, dodawanie pieniędzy, wydawanie reszty.

Zarządzanie komunikacją i pamięcią:

- Przechowywanie danych w pamięci współdzielonej (kasa).
- Czyszczenie kolejki komunikatów (podczas ewakuacji).

Zwalnianie zasobów:

- Usuwanie semaforów, pamięci współdzielonej i kolejki przy zakończeniu symulacji.

main.c:

Plik ten jest głównym modułem programu, który inicjalizuje symulację, uruchamia procesy i kontroluje ich działanie. Program wykonuje takie działania jak:

- Tworzy semafory, pamięć współdzieloną i kolejki komunikatów.
- Uruchamia procesy (**fork()**) – kierownik, fryzjerzy, klienci.
- Monitoruje czas trwania symulacji (godziny otwarcia, zamknięcia salonu)

Po zakończeniu symulacji (brak fryzjerów lub zakończenie czasu działania salonu) wysyła sygnał **SIGINT** aby zakończyć procesy i zwolnić zasoby.

Użyte konstrukcje w projekcie.

1. **Tworzenie i obsługa plików.**

Zrealizowane.

Zapis stanu kasy po zakończeniu trwania symulacji.

2. **Tworzenie procesów.**

Zrealizowane.

- **fork()** – do tworzenia procesów klientów, fryzjerów i kierownika.
- **exit()** – do zamykania procesów.

3. **Tworzenie i obsługa wątków.**

Nie zrealizowane.

4. **Obsługa sygnałów.**

Zrealizowane.

- **signal()** – do obsługi sygnałów SIGINT (kończenie symulacji) i SIGTERM (zwalnianie fryzjerów).
- **kill()** – do wysyłania sygnałów do procesów (np. kierownik wysyła SIGTERM).

5. **Synchronizacja procesów.**

Zrealizowane.

- **ftok()** – do generowania kluczy dla semaforów.
- **semctl()** – do ustawiania wartości semaforów.
- **semop()** – do operacji na semaforach (rezerwacja i zwalnianie zasobów).

6. **Łącza nazwane i nienazwane**

Nie zrealizowane.

7. **Segmenty pamięci dzielonej**

Zrealizowane.

- **ftok()** – do generowania kluczy.
- **shmget()** – do tworzenia segmentów pamięci dzielonej (kasa).
- **shmat()** – do dołączania pamięci dzielonej.
- **shmdt()** – do odłączania pamięci dzielonej.
- **shmctl()** – do zarządzania pamięcią dzieloną.

8. Kolejki komunikatów.

Zrealizowane.

- **msgget()** – do tworzenia kolejek komunikatów.
- **msgsnd()** – do wysyłania wiadomości.
- **msgrcv()** – do odbierania wiadomości.
- **msgctl()** – do zarządzania kolejkami.

9. Gniazda

Nie zrealizowane.

Testy programu.

Wariant pierwszy:

Normalna praca typowego salonu fryzjerskiego.

```
#define FRYZJERZY 5
#define KLIENCI 20
#define POCZEKALNIA 8
#define FOTELE 3

#define LICZBA_NOMINALOW 3
#define KOSZT_USLUGI 30
#define KASA_STARTOWA_BANKNOTY {20, 20, 20}
#define KLIENT_STARTOWE_BANKNOTY {5, 5, 5}

#define CZAS_OTWARCIA 8
#define CZAS_ZAMKNIECIA 16
#define GODZINA 20  // Reprezentacja jednej godziny w systemie (np. 1 sekunda = 1 godzina)
```

Uruchomienie programu:

```
Symulacja czasu pracy salonu: 8 godzin będzie trwać 160 sekund.
```

Analiza obsługi losowego klienta, wybieram pierwszego który się pojawił czyli: 3441991

```
Klient 3441991 przychodzi do salonu.
Klient 3441991 czeka na fotel.
Klient PID: 3441991 - Kwota do zapłaty: 30, suma klienta: 400, reszta do wydania: 370
Klient PID: 3441991 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3441991 płaci 30 zł za usługę.
Fryzjer 3441983 strzyże klienta 3441991.
```

Po jakimś czasie fryzjer kończy usługę i wyświetla stan kasy.

```
Fryzjer 3441983 zakończył strzyżenie klienta 3441991.
Stan kasy:
10 zł: 60 banknotów
20 zł: 52 banknotów
50 zł: 4 banknotów
Łączna suma w kasie: 1840 zł
```

Po pewnym czasie klient wraca do salonu i jest strzyżony przez innego fryzjera, możemy również zaobserwować jego mniejszy stan konta, co jest logiczne ponieważ musiał uiścić opłatę za poprzednie strzyżenie (miał pecha i nie udało mu się zarobić po za salonem).

```
Klient 3441991 przychodzi do salonu.
Klient 3441991 czeka na fotel.
Klient PID: 3441991 - Kwota do zapłaty: 30, suma klienta: 370, reszta do wydania: 340
Klient PID: 3441991 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3441991 płaci 30 zł za usługę.
Fryzjer 3441987 zakończył strzyżenie klienta 3441996.
Stan kasy:
10 zł: 119 banknotów
20 zł: 66 banknotów
50 zł: 1 banknotów
Łączna suma w kasie: 2560 zł
Fryzjer 3441987 strzyże klienta 3441991.
```

Również usługę udaje się zrealizować i zakończyć sukcesem

```
Fryzjer 3441987 zakończył strzyżenie klienta 3441991.
Stan kasy:
10 zł: 119 banknotów
20 zł: 70 banknotów
50 zł: 0 banknotów
```

Analiza losowego fryzjera, wybieram również pierwszego jaki się pojawił czyli: 3568386.

```
Fryzjer 3568386 strzyże klienta 3568406.
```

Następny rekord:

```
Fryzjer 3568386 zakończył strzyżenie klienta 3568406.
```

I kolejne:

```
Fryzjer 3568386 strzyże klienta 3568391.
Fryzjer 3568386 zakończył strzyżenie klienta 3568391.
Fryzjer 3568386 strzyże klienta 3568395.
Fryzjer 3568386 zakończył strzyżenie klienta 3568395.
```

I tak dalej… Możemy zaobserwować poprawne działanie fryzjera – nie strzyże on kilku klientów na raz, oraz nie zdarza się tak, aby nie zakończył strzyżenia a brał kolejnego klienta.

Może się też zdarzyć tak, że fryzjer zostaje zwolniony:

```
Kierownik: Zwolniono fryzjera o PID 3441986.
```

Wtedy możemy podglądnąć w ps, że zostaje on procesem zombie (możemy monitorować zwolnionych fryzjerów)

```
3568386 pts/144  00:00:00 fryzjer <defunct>
```

A na końcu działania symulacji zostaje poprawnie zabity.

Dla zobrazowania przerywam działanie symulacji wciskając CTRL+C

```
^Cleczycki.krzysztof.151458@torus:~/so_project$
```

Jak możemy zauważyć wszystkie procesy zostały poprawnie zabite:

```
^Cleczycki.krzysztof.151458@torus:~/so_project$ ps
    PID TTY          TIME CMD
1381719 pts/144  00:00:01 bash
3624607 pts/144  00:00:00 ps
```

Analiza działania kierownika:

Jest on uprawniony do losowego zwalniania fryzjerów:

```
Kierownik: Zwolniono fryzjera o PID 3637189.
```

Może on także przeprowadzać ewakuację w salonie – wyrzuca klientów z poczekalni oraz zakańcza strzyżenie klientów z foteli i również ich wyprasza.

```
Fryzjer 3637192 zakończył strzyżenie klienta 3637206.
Stan kasy:
10 zł: 120 banknotów
20 zł: 49 banknotów
50 zł: 4 banknotów
Łączna suma w kasie: 2380 zł
Fryzjer 3637190 zakończył strzyżenie klienta 3637208.
Stan kasy:
10 zł: 120 banknotów
20 zł: 49 banknotów
50 zł: 4 banknotów
Łączna suma w kasie: 2380 zł
Kierownik: Ewakuacja klientów - czyszczenie kolejki komunikatów.
Kolejka komunikatów została wyczyszczona.
Kierownik: Dostęp do salonu zablokowany.
Fryzjer 3637191 zakończył strzyżenie klienta 3637200.
Stan kasy:
10 zł: 120 banknotów
20 zł: 49 banknotów
50 zł: 4 banknotów
Łączna suma w kasie: 2380 zł
Kierownik: Ewakuacja zakończona - odblokowanie dostępu.
Klient 3637211 przychodzi do salonu.
```

W momencie rozpoczęcia ewakuacji jeden z fryzjerów dokańcza usługę strzyżenia (dwóch fryzjerów zakończyło strzyc zaraz przed samą ewakuacją i nie podjęli się kolejnych klientów). Kierownik czyści poczekalnie (kolejkę komunikatów) i po pewnym czasie odblokowuje możliwość przychodzenia do salonu (w momencie ewakuacji nie da się do niego wejść – odpowiada za to semafor 4).

```
Kierownik: Ewakuacja zakończona - odblokowanie dostępu.
Klient 3637211 przychodzi do salonu.
Klient 3637211 czeka na fotel.
Klient PID: 3637211 - Kwota do zapłaty: 30, suma klienta: 390, reszta do wydania: 360
Klient PID: 3637211 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3637211 płaci 30 zł za usługę.
Fryzjer 3637193 strzyże klienta 3637211.
Klient 3637201 przychodzi do salonu.
Klient 3637201 czeka na fotel.
Klient PID: 3637201 - Kwota do zapłaty: 30, suma klienta: 370, reszta do wydania: 340
Klient PID: 3637201 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3637201 płaci 30 zł za usługę.
Fryzjer 3637192 strzyże klienta 3637201.
Klient 3637209 przychodzi do salonu.
Klient 3637209 czeka na fotel.
Klient PID: 3637209 - Kwota do zapłaty: 30, suma klienta: 370, reszta do wydania: 340
Klient PID: 3637209 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3637209 płaci 30 zł za usługę.
Fryzjer 3637190 strzyże klienta 3637209.
```

Po odblokowaniu możliwości przychodzenia do salonu, poczekalnia wypełnia się i momentalnie trzech fryzjerów podejmuje klientów (co oznacza, że podczas ewakuacji usługa nie jest wykonywana).

Gdy kierownik zwolni wszystkich fryzjerów, symulacja dobiega końca:

```
Kierownik: Zwolniono fryzjera o PID 3701761.
Kierownik: Wszyscy fryzjerzy zostali zwolnieni. Kończę symulację.
```

Po zakończeniu symulacji wszystkie zasoby zostają poprawnie zwolnione:

```
leczycki.krzysztof.151458@torus:~/so_project$ ipcs

------ Message Queues --------
key        msqid      owner     perms      used-bytes   messages
0x50000979 1376259    antosiewic 666        0            0
0x410007b9 131080     ciesielczy 666        0            0
0x44000157 3211279    sarnecki.p 666        0            0
0x00000000 3145757    zawartka.z 666        0            0
0x00000000 3145758    zawartka.z 666        0            0
0x00000000 3145759    zawartka.z 666        0            0
0x0000000f 1998880    moses.mich 777        0            0
0x00000000 3145761    zawartka.z 666        0            0
0x00000000 3145762    zawartka.z 666        0            0
0x00000000 3145763    zawartka.z 666        0            0
0x00000000 3145764    zawartka.z 666        0            0
0x00000000 3145765    zawartka.z 666        0            0
0x00000000 3145766    zawartka.z 666        0            0
0x00000000 3145767    zawartka.z 666        0            0
0x000002be 3244091    jaros.anto 600        8            1

------ Shared Memory Segments --------
key        shmid      owner     perms      bytes      nattch     status
0x50000979 19529736   antosiewic 666        24         0
0x53000157 40501267   sarnecki.p 666        4          0
0x00000000 19431445   ciesla.bar 666        24         4          dest
0x00000000 19955753   kowalska.a 666        100        1          dest
0x00000030 16449581   janus.luka 666        28         0
0x00000000 40435763   zawartka.z 666        44         1          dest
0x000004d3 33030205   krzeminski 666        52         0
0x000004d4 33030206   krzeminski 666        40         0
0x00000000 40403007   zawartka.z 666        44         1          dest

------ Semaphore Arrays --------
key        semid      owner     perms      nsems
0x50000979 1179652    antosiewic 666        1
0x010002be 2228244    jaros.anto 600        1
0x0000002f 720917     janus.luka 666        5
0x020002be 2228246    jaros.anto 600        2
0x0000000d 1310751    skiba.dami 777        1
0x53000157 2195493    sarnecki.p 666        3
0x410002b8 2097210    antosiewic 666        2
```

```
leczycki.krzysztof.151458@torus:~/so_project$ jobs -l
leczycki.krzysztof.151458@torus:~/so_project$
```

```
leczycki.krzysztof.151458@torus:~/so_project$ ps
    PID TTY          TIME CMD
1381719 pts/144  00:00:01 bash
3738350 pts/144  00:00:00 ps
```

Symulacja również może zakończyć się z powodu zakończenia czasu pracy salonu:

```
Czas symulacji upłynął. Zakończono symulację z_powodu zakończenia czasu pracy salonu.
```

Wariant drugi

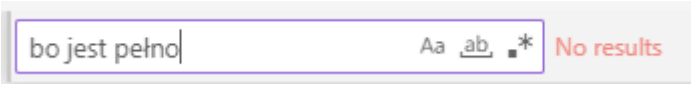
Poczekalnia większa, niż liczba klientów.

```
#define FRYZJERZY 5
#define KLIENCI 20
#define POCZEKALNIA 30
#define FOTELE 3

#define LICZBA_NOMINALOW 3
#define KOSZT_USLUGI 30
#define KASA_STARTOWA_BANKNOTY {20, 20, 20}
#define KLIENT_STARTOWE_BANKNOTY {5, 5, 5}

#define CZAS_OTWARCIA 8
#define CZAS_ZAMKNIECIA 16
#define GODZINA 20  // Reprezentacja jednej godziny w systemie (np. 1 sekunda = 1 godzina)
```

Podczas trwania całej symulacji, nie zdarzyło się, że brakłoby miejsca w poczekalni (co jest logiczne, bo klientów jest 20, a poczekalnia ma rozmiar 30)



Wariant trzeci

Jednomiejscowa poczekalnia, duża liczba fryzjerów i foteli.

```
#define FRYZJERZY 20
#define KLIENCI 20
#define POCZEKALNIA 1
#define FOTELE 20

#define LICZBA_NOMINALOW 3
#define KOSZT_USLUGI 30
#define KASA_STARTOWA_BANKNOTY {20, 20, 20}
#define KLIENT_STARTOWE_BANKNOTY {5, 5, 5}

#define CZAS_OTWARCIA 8
#define CZAS_ZAMKNIECIA 16
#define GODZINA 20  // Reprezentacja jednej godziny w systemie (np. 1 sekunda = 1 godzina)
```

Pomimo takiej konfiguracji, nie jesteśmy „wepchnąć” wszystkich możliwych klientów na fotele fryzjerskie i zapewnić im usługi strzyżenia. Zgodnie z moimi obserwacjami jednocześnie odbywa się jedno strzyżenie

```
Symulacja czasu pracy salonu: 8 godzin będzie trwać 160 sekund.
Klient 3750367 przychodzi do salonu.
Klient 3750367 czeka na fotel.
Klient PID: 3750367 - Kwota do zapłaty: 30, suma klienta: 400, reszta do wydania: 370
Klient PID: 3750367 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3750367 płaci 30 zł za usługę.
Fryzjer 3750349 strzyże klienta 3750367.
Klient 3750369 przychodzi do salonu.
Klient 3750369 opuszcza salon, bo jest pełno.
Klient 3750371 przychodzi do salonu.
Klient 3750371 opuszcza salon, bo jest pełno.
Klient 3750372 przychodzi do salonu.
Klient 3750372 opuszcza salon, bo jest pełno.
Klient 3750373 przychodzi do salonu.
Klient 3750373 opuszcza salon, bo jest pełno.
Klient 3750384 przychodzi do salonu.
Klient 3750384 opuszcza salon, bo jest pełno.
Klient 3750385 przychodzi do salonu.
Klient 3750385 opuszcza salon, bo jest pełno.
Fryzjer 3750349 zakończył strzyżenie klienta 3750367.
Stan kasy:
10 zł: 25 banknotów
20 zł: 24 banknotów
50 zł: 18 banknotów
Łączna suma w kasie: 1630 zł
Klient 3750375 przychodzi do salonu.
Klient 3750375 czeka na fotel.
Klient PID: 3750375 - Kwota do zapłaty: 30, suma klienta: 400, reszta do wydania: 370
Klient PID: 3750375 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3750375 płaci 30 zł za usługę.
Fryzjer 3750350 strzyże klienta 3750375.
Klient 3750377 przychodzi do salonu.
Klient 3750377 opuszcza salon, bo jest pełno.
Klient 3750378 przychodzi do salonu.
Klient 3750378 opuszcza salon, bo jest pełno.
Klient 3750383 przychodzi do salonu.
Klient 3750383 opuszcza salon, bo jest pełno.
Klient 3750369 zarabia banknot 50 zł.
Klient 3750368 przychodzi do salonu.
Klient 3750368 opuszcza salon, bo jest pełno.
Klient 3750374 przychodzi do salonu.
Klient 3750374 opuszcza salon, bo jest pełno.
Fryzjer 3750350 zakończył strzyżenie klienta 3750375.
```

Wariant czwarty:

Brak fryzjerów na start.

```
#define FRYZJERZY 0
#define KLIENCI 20
#define POCZEKALNIA 3
#define FOTELE 3

#define LICZBA_NOMINALOW 3
#define KOSZT_USLUGI 30
#define KASA_STARTOWA_BANKNOTY {20, 20, 20}
#define KLIENT_STARTOWE_BANKNOTY {5, 5, 5}

#define CZAS_OTWARCIA 8
#define CZAS_ZAMKNIECIA 16
#define GODZINA 20  // Reprezentacja jednej godziny w systemie (np. 1 sekunda = 1 godzina)
```

Symulacja wystartowała

```
Symulacja czasu pracy salonu: 8 godzin będzie trwać 160 sekund.
Klient 3758685 przychodzi do salonu.
Klient 3758685 czeka na fotel.
Klient PID: 3758685 - Kwota do zapłaty: 30, suma klienta: 400, reszta do wydania: 370
Klient PID: 3758685 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3758685 płaci 30 zł za usługę.
Klient 3758691 przychodzi do salonu.
Klient 3758691 czeka na fotel.
Klient PID: 3758691 - Kwota do zapłaty: 30, suma klienta: 400, reszta do wydania: 370
Klient PID: 3758691 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3758691 płaci 30 zł za usługę.
Klient 3758693 przychodzi do salonu.
Klient 3758694 przychodzi do salonu.
Klient 3758694 czeka na fotel.
Klient PID: 3758694 - Kwota do zapłaty: 30, suma klienta: 400, reszta do wydania: 370
Klient PID: 3758694 - Transakcja zakończona sukcesem. Reszta wydana w pełni.
Klient 3758694 płaci 30 zł za usługę.
Klient 3758693 opuszcza salon, bo jest pełno.
Klient 3758697 przychodzi do salonu.
Klient 3758697 opuszcza salon, bo jest pełno.
Klient 3758701 przychodzi do salonu.
Klient 3758700 przychodzi do salonu.
Klient 3758701 opuszcza salon, bo jest pełno.
Klient 3758700 opuszcza salon, bo jest pełno.
Klient 3758704 przychodzi do salonu.
Klient 3758704 opuszcza salon, bo jest pełno.
Klient 3758686 przychodzi do salonu.
Klient 3758686 opuszcza salon, bo jest pełno.
Klient 3758687 przychodzi do salonu.
Klient 3758687 opuszcza salon, bo jest pełno.
Klient 3758692 przychodzi do salonu.
Klient 3758692 opuszcza salon, bo jest pełno.
Klient 3758701 zarabia banknot 20 zł.
```

Jednak jest problem, trzech pierwszych klientów zostało skasowanych, lecz nie mogą zostać obsłużeni bo nie mają przez kogo.

Po drobnych poprawkach w kodzie:

```
if (aktywni_fryzjerzy == 0) {
    printf(KIEROWNIK_COLOR "Kierownik: Salon nie może zostać otwarty z powodu braku pracowników. Kończę symulację.\n" PID_COLOR);
    kill(0, SIGINT);
    exit(0);
}
```

Uruchamiam symulację:

```
leczycki.krzysztof.151458@torus:~/so_project$ ./salon
Symulacja czasu pracy salonu: 8 godzin będzie trwać 160 sekund.
Kierownik: Salon nie może zostać otwarty z powodu braku pracowników. Kończę symulację.
```

Teraz działa poprawnie.