PROJEKT 2 - DOKUMENTACJA

Problem śpiącego fryzjera

PROWADZĄCY: DR INŻ. MARCIN KOŹNIEWSKI

WYKONAŁ: ERYK SUREL

Ogólny opis

Projekt został zrealizowany w 2 wersjach :

- Mutex - wykorzystująca mutexy i semafory,
- Cond - wykorzystujące zmienne warunkowe.

Sposób uruchamiania

./mutex

Argumenty wymagane:

- -k - liczba klientów, którzy przyjdą do fryzjera,
- -r - liczba krzeseł w poczekalni,

Argumenty opcjonalne:

- -c - maksymalna wartość opóźnienia klienta,
- -f - maksymalny czas strzyżenia pojedynczego klienta,
- -d - tryb debugowania.

./cond

Argumenty wymagane:

- -k - liczba klientów, którzy przyjdą do fryzjera,
- -r - liczba krzeseł w poczekalni,

Argumenty opcjonalne:

- -c - maksymalna wartość opóźnienia klienta,
- -f - maksymalny czas strzyżenia pojedynczego klienta,
- -d - tryb debugowania.

Zaimplementowane mechanizmy synchronizacji

Dla uproszczenia oraz skrócenia fragmentów, nieistotne części kodu zostały zastąpione <code>.

Mutex

- Klient

```
        sem_wait(&clientsSem);

        pthread_mutex_lock(&waitingRoom);

        freeSeatsAmount++;

        sem_post(&barberSem);

        pthread_mutex_unlock(&waitingRoom);

        cut();

      <code>

        pthread_mutex_unlock(&barberSeat);
```

- Fryzjer

```
pthread_mutex_lock(&waitingRoom);
if (freeSeatsAmount > 0) {
    freeSeatsAmount--;
    if (isDebug == 1)
        addToWList(clientId, clientTime);
   <code>
    if (isDebug == 1)
    {
        printWList();
    }
    sem_post(&clientsSem);
    pthread_mutex_unlock(&waitingRoom);
    sem_wait(&barberSem);
    pthread_mutex_lock(&barberSeat);
    clientOnSeatId = clientId;
   <code>
    if (isDebug == 1)
    {
        deleteNode(&waitingClients, clientId);
        printWList();
    }
```

```
    }
    else {
        pthread_mutex_unlock(&waitingRoom);
        rejectedClientsCounter++;
       <code>
        if (isDebug == 1) addToRList(clientId, clientTime);
    }
```

Cond

- Fryzjer

```
    pthread_mutex_lock(&barberMutex);
    while (isEnd == 0)
    {
        pthread_cond_wait(&wakeupBarberCond, &barberMutex);
        pthread_mutex_unlock(&barberMutex);
        if (isEnd == 0)
        {
            cut();
       <code>
            pthread_cond_signal(&shearEndCond);
        }
        else {
            if (isDebug == 1) printf("Fryzjer zakonczyl prace\n");
```

```
        }
    }
```

- Klient

```
    pthread_mutex_lock(&waitingRoom);
    if (freeSeatsAmount > 0) {
        freeSeatsAmount--;
        if (isDebug == 1)
            addToWList(clientId, clientTime);
       <code>
        pthread_mutex_unlock(&waitingRoom);
        pthread_mutex_lock(&barberSeatMutex);
        if (isSeatBusy == 1)
            pthread_cond_wait(&isBarberAvailableCond, &barberSeatMutex);
        isSeatBusy = 1;
        pthread_mutex_unlock(&barberSeatMutex);
        pthread_mutex_lock(&waitingRoom);
        freeSeatsAmount++;
        clientOnSeatId = clientId;
       <code>
        if (isDebug == 1)
        {
            deleteNode(&waitingClients, clientId);
```

```
        }
    <code>
        pthread_mutex_unlock(&waitingRoom);
        pthread_cond_signal(&wakeupBarberCond);
        pthread_mutex_lock(&clientFinished);
        pthread_cond_wait(&shearEndCond, &clientFinished);
        isEnd = 0;
        pthread_mutex_unlock(&clientFinished);
        pthread_mutex_lock(&barberSeatMutex);
        isSeatBusy = 0;
        pthread_mutex_unlock(&barberSeatMutex);
        pthread_cond_signal(&isBarberAvailableCond);
    }
    else {
        pthread_mutex_unlock(&waitingRoom);
        rejectedClientsCounter++;
        if (isDebug == 1)
            addToRList(clientId, clientTime);
    <code>
    }
```