

W projekcie implementujemy program zarówno ze zmiennymi warunkowymi jak również z jedynie mutexami i semaforami.

Oczekiwana ilość punktów: 34

W zadaniu:

- Stworzono 2 mutexy:

- **mutexWRoom** – Kontroluje wchodzenie i wychodzenie z kolejki.
- **mutexBarber** – Kontroluje czas pracy fryzjera.

- Stworzono 1 zmienną warunkową:

- **barberSleeping** – pozwala na budzenie fryzjera.

- Stworzono 3 listy:

- **allClients** – Przechowuje wszystkich klientów.
- **resignedClients** – Przechowuje klientów którzy zrezygnowali z usługi fryzjera.
- **clientsInWRoom** – Przechowuje klientów aktualnie przebywających w poczekalni

- Zainicjowano 9 zmiennych globalnych:

- **int maxSeatsInWRoom** = 10 – Ilość miejsc w poczekalni, może być modyfikowana opcją -s.
- **int numberOfClients** = 30 – Całkowita ilość klientów którzy odwiedzają fryzjera podczas działania programu, może być zmodyfikowana opcją -c.
- **int freeSeatsInWRoom** = 10 – Aktualna ilość wolnych miejsc w poczekalni.
- **int maxClippingTime** = 6 – Maksymalny czas strzyżenia, może być zmodyfikowana opcją -t.
- **int maxClientArrivalTime** = 30 – Maksymalny czas do przybycia ostatniego klienta, może być zmodyfikowany opcją -m.
- **int resignedCounter** = 0 – Licznik klientów którzy zrezygnowali z usługi fryzjera.
- **int clientOnSeat** = -1 – Numer aktualnie obsługiwanego klienta, -1 oznacza że fryzjer śpi.
- **bool bDebug** = false – Flaga oznaczająca że program ma być uruchomiony w trybie debug, może być uruchomiona opcją -d.
- **bool finished** = false – Flaga mówiąca fryzjerowi czy pozostali jeszcze jacyś klienci.

Wątek klienta:

void *Client(void *cNumber)

***cNumber** – Dostarcza funkcji numer klienta w wątku.

Wątek zaczyna od przeczekania losowej ilości czasu, następnie blokuje mutex „mutexWRoom” w celu zdecydowania wejścia do poczekalni, jeśli poczekalnia jest pełna to klient rezygnuje i raportuje aktualny stan zakładu.

```
if (freeSeatsInWRoom <= 0)
{
    //-----Leaving the building
    resignedCounter++;
    printf("Res:%d WRoom: %d/%d [in: %d]\t Number %d resigned\n", resignedCounter, maxSeatsInWRoom - freeSeatsInWRoom,
maxSeatsInWRoom, clientOnSeat, clientNumber);
    pthread_mutex_unlock(&mutexWRoom);
    if (bDebug == true)
    {
        Append(&resignedClients, clientNumber, 0);
        Print(resignedClients, clientsInWRoom);
    }
}
```

```
}
```

Jeśli klient zdecyduje się wejść do poczekalni dołącza on do listy poczekalni i raportuje stan zakładu. Aby sprawdzić czyja jest aktualnie kolej wszyscy klienci sprawdzają czy mają pierwsze miejsce na liście.

```
freeSeatsInWRoom--;
Append(&clientsInWRoom, clientNumber, 0);
printf("Res:%d WRoom: %d/%d [in: %d]\t Number %d joined queue\n", resignedCounter, maxSeatsInWRoom -
freeSeatsInWRoom, maxSeatsInWRoom, clientOnSeat, clientNumber);
if (bDebug == true)
{
    Print(resignedClients, clientsInWRoom);
}
pthread_mutex_unlock(&mutexWRoom);

while(clientsInWRoom->clientNumber != clientNumber) {}
```

Kiedy nadchodzi kolej klienta blokuje on mutex „mutexBarber” w celu powiadomienia o tym że już ktoś idzie skorzystać z usługi klienta, zaraz potem blokuje on mutex „mutexWRoom” w celu bezpiecznego wyjścia z kolejki. Po opuszczeniu kolejki odblokowuje on kolejno mutexy „mutexWRoom” i „mutexBarber”. Teraz klient budzi fryzjera a klient usuwany jest z listy kolejki.

```
//-----Entering barber's room
pthread_mutex_lock(&mutexBarber);

//-----Leaving queue
pthread_mutex_lock(&mutexWRoom);
freeSeatsInWRoom++;
clientOnSeat = clientNumber;
pthread_mutex_unlock(&mutexWRoom);

pthread_mutex_unlock(&mutexBarber);

//-----Waking up the barber
pthread_cond_signal(&barberSleeping);

Pop(&clientsInWRoom);
```

Wątek fryzjera:

void *Barber()

Fryzjer na początku przygotowuje się do pracy poprzez zadeklarowanie zmiennej „clippingTime” przechowującej czas strzyżenia aktualnego klienta, następnie blokuje on mutex „mutexBarber”. Po przygotowaniu do pracy wchodzi on w pętlę która kończy się po przejściu wszystkich klientów. W pętli fryzjer zaczyna od czekania na zmienną „barberSleeping” co przedstawia jego spanie, po obudzeniu strzyże on klienta przez losowy czas, wypisuje raport o stanie zakładu oraz oznacza fotel do strzyżenia jako wolny.

```
int clippingTime;
pthread_mutex_lock(&mutexBarber);
while (finished == false)
{
    //-----Going to sleep
    pthread_cond_wait(&barberSleeping, &mutexBarber);

    //-----Clipping hair
    clippingTime = rand() % maxClippingTime + 1;
    sleep(clippingTime);
    printf("Res:%d WRoom: %d/%d [in: %d]\t Number %d has been serviced\n", resignedCounter, maxSeatsInWRoom -
freeSeatsInWRoom, maxSeatsInWRoom, clientOnSeat, clientOnSeat);
    if (bDebug == true)
    {
        Print(resignedClients, clientsInWRoom);
    }
}
```

```
//-----Clearing the seat  
clientOnSeat = -1;  
}
```