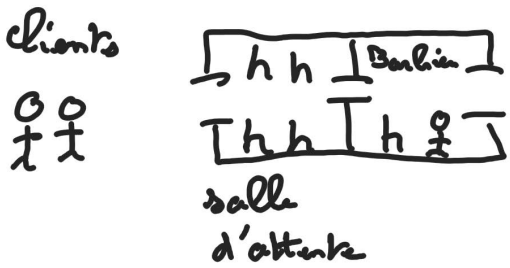
1. Le problème du barbier



Code des activités

Client

```
boucle
        a) entrer dans la salle d'att
        -- lit les revues
        b) s'asseoir dans le fauteuil du barbier
        -- rêvasse pdt le rasage
        c) partir
        --- va travailler
finBoucle
```

Barbier

```
boucle
        débuter rasage
        -- raser le client
        terminer rasage
        -- va au bistrot
finBoucle
```

Interface du moniteur	Condition d'acceptation	Variables d'état	Prédicats d'acceptation
entree_SA	la salle d'att n'est pas pleine	nbPlacesLibres : nat	nbPlacesLibres > 0
SAsseoir	le fauteuil est libre	fauteuilLibre : bool	fauteuilLibre
Partir	le client n'a plus de barbe	barbePresente : bool	¬barbePresente
Débuter_rasage	il y a un client barbu dans le fauteuil		barbePresente (^ ¬fauteuilLibre)
Terminer_rasage	---		true

Invariants

inv 0 ≤ nbPlacesLibres ≤ N
inv barbePresente => ¬fauteuilLibre

Opération

```
si ¬CA alors
        VC.attendre
finSi
màj des variables d'état
déblocage
```

Variables Conditions

Salle
Fauteuil
ClientBarbu
Rasé

Codage

entrée

```
tantQue ¬(nbPlacesLibres > 0) faire
        Salle.wait
finTantQue
{précondition : nbPlacesLibres > 0}
nbPlacesLibres--
{nbPlacesLibres >= 0}
```

SAsseoir

```
tantQue ¬(fauteuilLibre) faire
        Fauteuil.wait
finTantQue
{fauteuilLibre}
fauteuilLibre <- false
nbPlacesLibres++
barbePresente <- true
{¬fauteuilLibre ^ barbePresente ^ (nbPlacesLibres > 0)}
Salle.signal
ClientBarbu.signal
```

Partir

```
tantQue barbePresente faire
        Rasé.wait
finTantQue
{¬barbePresente}
fauteuilLibre <- true
{fauteuilLibre}
Fauteuil.signal
```

DebutRasage

```
tantQue ¬barbePresente faire
        ClientBarbu.wait
finTantQue
```

TerminerRasage

```
barbePresente <- false
{barbePresente}
Rasé.signal
```

⚡ **Exécution des opérations en exclusion mutuelle**

<- retour

#TD/SC