Semaine 7

1

2

v1 = n (n producteurs)
v2 = 0

4

```
void main() {
  sprod = CS(n);
  sprod = CS(0);
  CT(production);
  CT(consommation);

  SHARED int id;
  SHARED int ir;

  MutexP = CS(1);
  MutexC = CS(1);

  MutexCase[n] = CS(1)
}
```

```
void deposer(MESS* message) {
  int I;
  P(MutexP);
  I = id;
  id = (id + 1) % n;
  V(MutexP);
  T[I] = message;
}
```

```
void retirer(MESS* message) {
  int J;
  P(MutexC);
  J = ir;
  ir = (ir + 1) % n;
  V(MutexC);
  message = T[J];
}
```

Trucs bien:
- Deux consommateurs ne peuvent pas écrire en même temps dans une même case
- Deux producteurs ne peuvent pas écrire en même temps dans une même case

Problème: un producteur et un consommateur peuvent écrire en même temps dans une même case

Solution: rajouter N mutex:

```
void deposer(MESS* message) {
  int I;
```

```
  P(MutexP);
  I = id;
  id = (id + 1) % n;
  V(MutexP);
  P(MutexCase[I])
  T[I] = message;
  V(MutexCase[I])
}
```

```
void retirer(MESS* message) {
  int J;
  P(MutexC);
  J = ir;
  ir = (ir + 1) % n;
  V(MutexC);
  P(MutexCase[J])
  message = T[J];
  V(MutexCase[J])
}
```

Problème: Cas où interruption juste avant `P(MutexCase[I])` :
Un consommateur peut consommer une case vide

Solution: (inversion de deux lignes)

```
void deposer(MESS* message) {
  int I;
  P(MutexP);
  I = id;
  id = (id + 1) % n;
  P(MutexCase[I])
  V(MutexP);
  T[I] = message;
  V(MutexCase[I])
}
```

```
void retirer(MESS* message) {
  int J;
  P(MutexC);
  J = ir;
  ir = (ir + 1) % n;
  P(MutexCase[J])
  V(MutexC);
  message = T[J];
  V(MutexCase[J])
}
```

2

1

Écrivain: s'il y a au moins un autre processus sur le fichier Lecteur: s'il y a uniquement des écrivains sur la case

```
void main() {
  SEME = CS(1);
  Mutexcpt = CS(1);
  SHARED cpt = 0;
}
```

```
void OuvreEcriture() {
  P(SEME);
}
```

```
void OuvreEcriture() {
  V(SEME);
```

```
void OuvreLecture() {
  P(Mutexcpt)
  if (cpt == 0)
      P(SEME);
  cpt++;
  V(Mutexcpt)
}
```

```
void FermeLecture() {
  P(Mutexcpt)
  if (cpt == 0)
      V(SEME);
  cpt--;
  V(Mutexcpt)
}
```