Introduction à la programmation concurrente

Lecteurs-rédacteurs

Yann Thoma, Fiorenzo Gamba

Février 2021

Reconfigurable and Embedded Digital Systems Institute
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Introduction

Enoncé du problème

- Plusieurs threads doivent accéder à une ressource

Enoncé du problème

- Plusieurs threads doivent accéder à une ressource
- Threads lecteurs
    - Ne peuvent que lire les données

Enoncé du problème

- Plusieurs threads doivent accéder à une ressource
- Threads lecteurs
    - Ne peuvent que lire les données
- Threads rédacteurs
    - Peuvent modifier les données

Contraintes du problème

1. Plusieurs lecteurs peuvent lire simultanément les données;

Contraintes du problème

1. Plusieurs lecteurs peuvent lire simultanément les données;
2. Les rédacteurs s'excluent mutuellement;

Contraintes du problème

1. Plusieurs lecteurs peuvent lire simultanément les données;
2. Les rédacteurs s'excluent mutuellement;
3. Les lecteurs et les rédacteurs s'excluent mutuellement.

Question

- Pourquoi pas une simple section critique pour protéger la ressource?

Types de solutions

1. Priorité aux lecteurs (famine possible des rédacteurs)

Types de solutions

1. Priorité aux lecteurs (famine possible des rédacteurs)
2. Priorité aux lecteurs si un lecteur a déjà accès à la ressource

Types de solutions

1. Priorité aux lecteurs (famine possible des rédacteurs)
2. Priorité aux lecteurs si un lecteur a déjà accès à la ressource
3. Priorité aux rédacteurs (famine possible des lecteurs)

Types de solutions

1. Priorité aux lecteurs (famine possible des rédacteurs)
2. Priorité aux lecteurs si un lecteur a déjà accès à la ressource
3. Priorité aux rédacteurs (famine possible des lecteurs)
4. Accès aux données selon l'ordre d'arrivée. Toutes les demandes de lecteur qui se suivent sont satisfaites en même temps.

Classe abstraite

```
class AbstractReaderWriter {
public:
    AbstractReaderWriter(){};
    virtual ~AbstractReaderWriter(){};
    virtual void lockReading()   = 0;
    virtual void lockWriting()   = 0;
    virtual void unlockReading() = 0;
    virtual void unlockWriting() = 0;
};
```

Toutes les sources de cette présentation sont regroupées dans un même projet accessible depuis le lien de cette page

Priorité aux lecteurs

Priorité aux lecteurs - Règles

- Un lecteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0

Priorité aux lecteurs - Règles

- Un lecteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0
- Un rédacteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0
  - ET le nombre de lecteurs en cours de lecture vaut 0
  - ET le nombre de lecteurs en attente de la ressource vaut 0

Variables et sémaphores nécessaires

- Une variable `nbReaders`

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutexReaders`, qui est en charge de protéger l'accès à la variable `nbReaders`.

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutexReaders`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- `writer`, qui permet au premier lecteur qui accède la ressource de bloquer les futurs rédacteurs. Il permet également au rédacteur accédant la ressource de bloquer les lecteurs pendant l'écriture.

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutexReaders`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- `writer`, qui permet au premier lecteur qui accède la ressource de bloquer les futurs rédacteurs. Il permet également au rédacteur accédant la ressource de bloquer les lecteurs pendant l'écriture.
- `mutexWriters`, qui permet au rédacteur accédant la ressource de bloquer les autres rédacteurs. De ce fait, un seul rédacteur peut être en attente du sémaphore `writer`, empêchant ainsi un rédacteur de brûler la priorité à un lecteur.

Priorité au lecteurs

```
class ReaderWriterPrioReaders :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders;
  PcoSemaphore mutexWriters;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterPrioReaders() :
    mutexReaders(1),
    mutexWriters(1),
    writer(1),
    nbReaders(0){}

  void lockReading() {




  }
```

```
  void unlockReading() {




  }

  void lockWriting() {


  }

  void unlockWriting() {


  }
};
```

Priorité au lecteurs

```
class ReaderWriterPrioReaders :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders;
  PcoSemaphore mutexWriters;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterPrioReaders() :
    mutexReaders(1),
    mutexWriters(1),
    writer(1),
    nbReaders(0){}

  void lockReading() {









  }
```

```
  void unlockReading() {









  }

  void lockWriting() {

    writer.acquire();
  }

  void unlockWriting() {
    writer.release();

  }
};
```

Priorité au lecteurs

```
class ReaderWriterPrioReaders :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders;
  PcoSemaphore mutexWriters;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterPrioReaders() :
    mutexReaders(1),
    mutexWriters(1),
    writer(1),
    nbReaders(0){}

  void lockReading() {

    nbReaders++;
    if (nbReaders==1) {
      writer.acquire();
    }

  }
```

```
  void unlockReading() {

    nbReaders -= 1;
    if (nbReaders==0) {
      writer.release();
    }

  }

  void lockWriting() {

    writer.acquire();
  }

  void unlockWriting() {
    writer.release();

  }
};
```

Priorité au lecteurs

```
class ReaderWriterPrioReaders :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders;
  PcoSemaphore mutexWriters;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterPrioReaders() :
    mutexReaders(1),
    mutexWriters(1),
    writer(1),
    nbReaders(0){}

  void lockReading() {
    mutexReaders.acquire();
    nbReaders++;
    if (nbReaders==1) {
      writer.acquire();
    }
    mutexReaders.release();
  }
```

```
  void unlockReading() {
    mutexReaders.acquire();
    nbReaders -= 1;
    if (nbReaders==0) {
      writer.release();
    }
    mutexReaders.release();
  }

  void lockWriting() {

    writer.acquire();
  }

  void unlockWriting() {
    writer.release();

  }
};
```

Priorité au lecteurs

```
class ReaderWriterPrioReaders :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders;
  PcoSemaphore mutexWriters;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterPrioReaders() :
    mutexReaders(1),
    mutexWriters(1),
    writer(1),
    nbReaders(0){}

  void lockReading() {
    mutexReaders.acquire();
    nbReaders++;
    if (nbReaders==1) {
      writer.acquire();
    }
    mutexReaders.release();
  }

  void unlockReading() {
    mutexReaders.acquire();
    nbReaders -= 1;
    if (nbReaders==0) {
      writer.release();
    }
    mutexReaders.release();
  }

  void lockWriting() {
    mutexWriters.acquire();
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
    mutexWriters.release();
  }
};
```

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

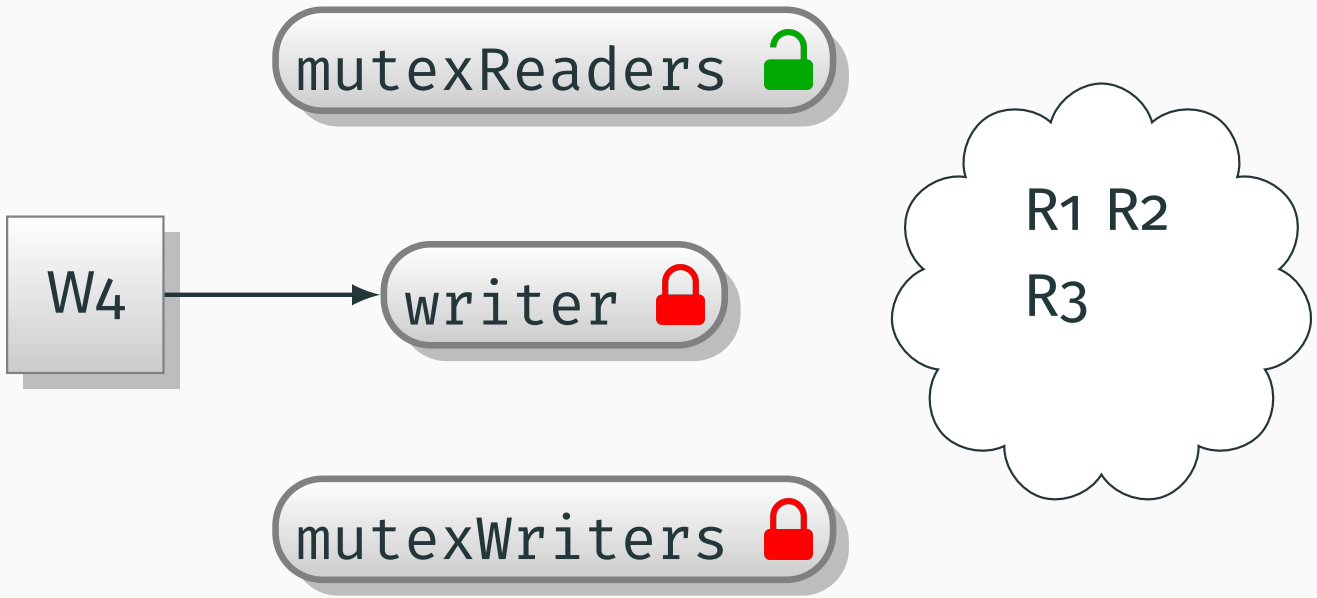
- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

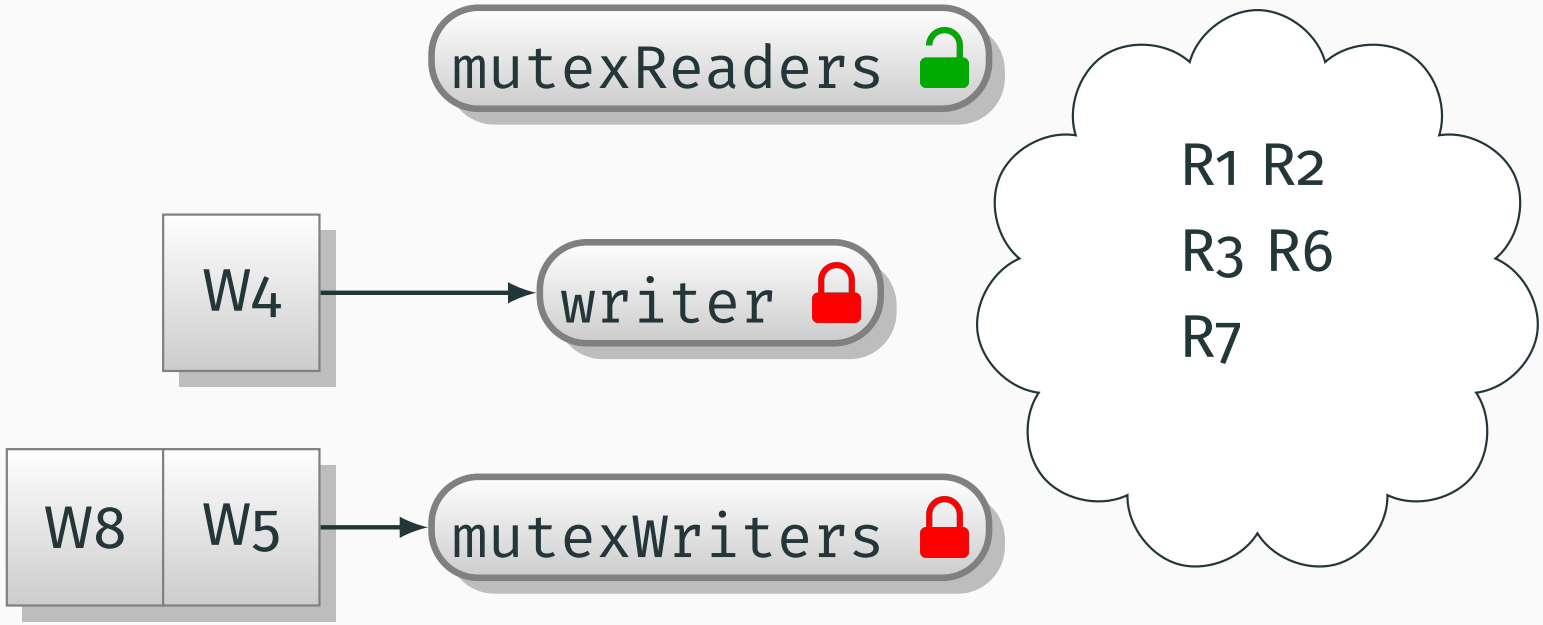
- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

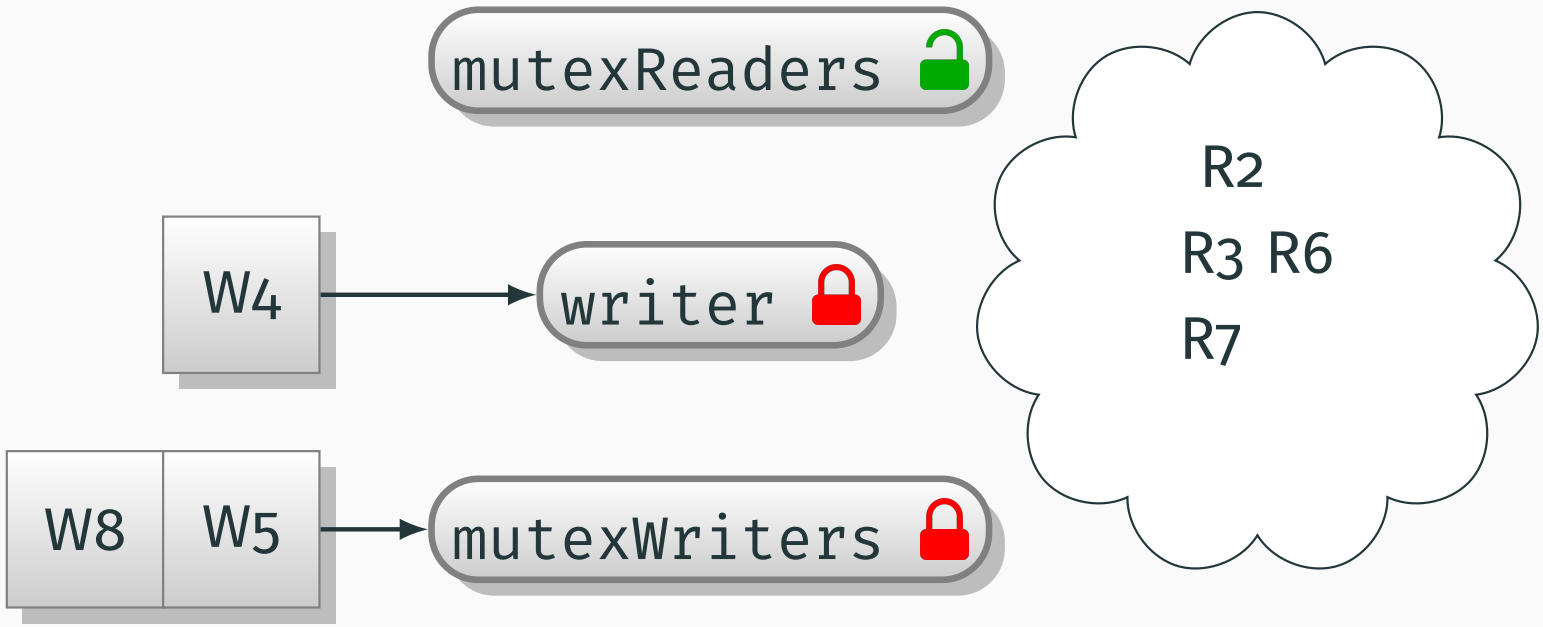
Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



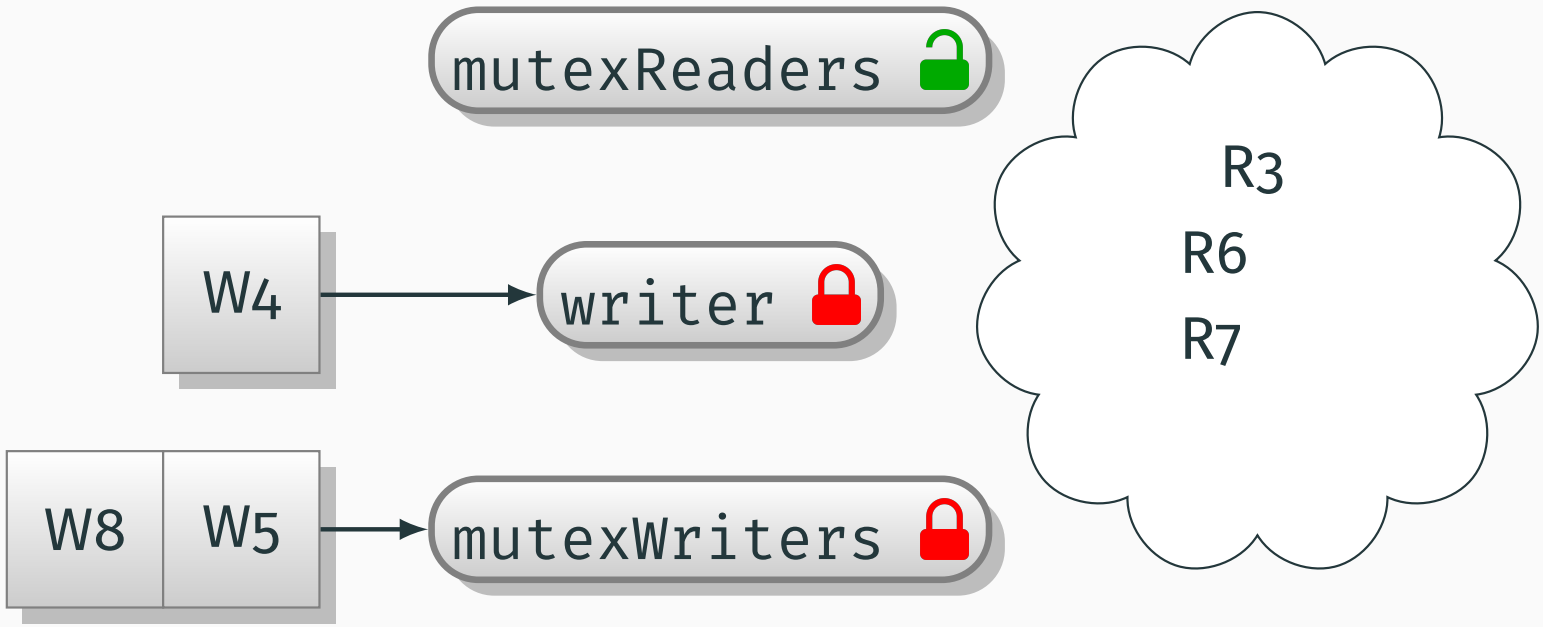
Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

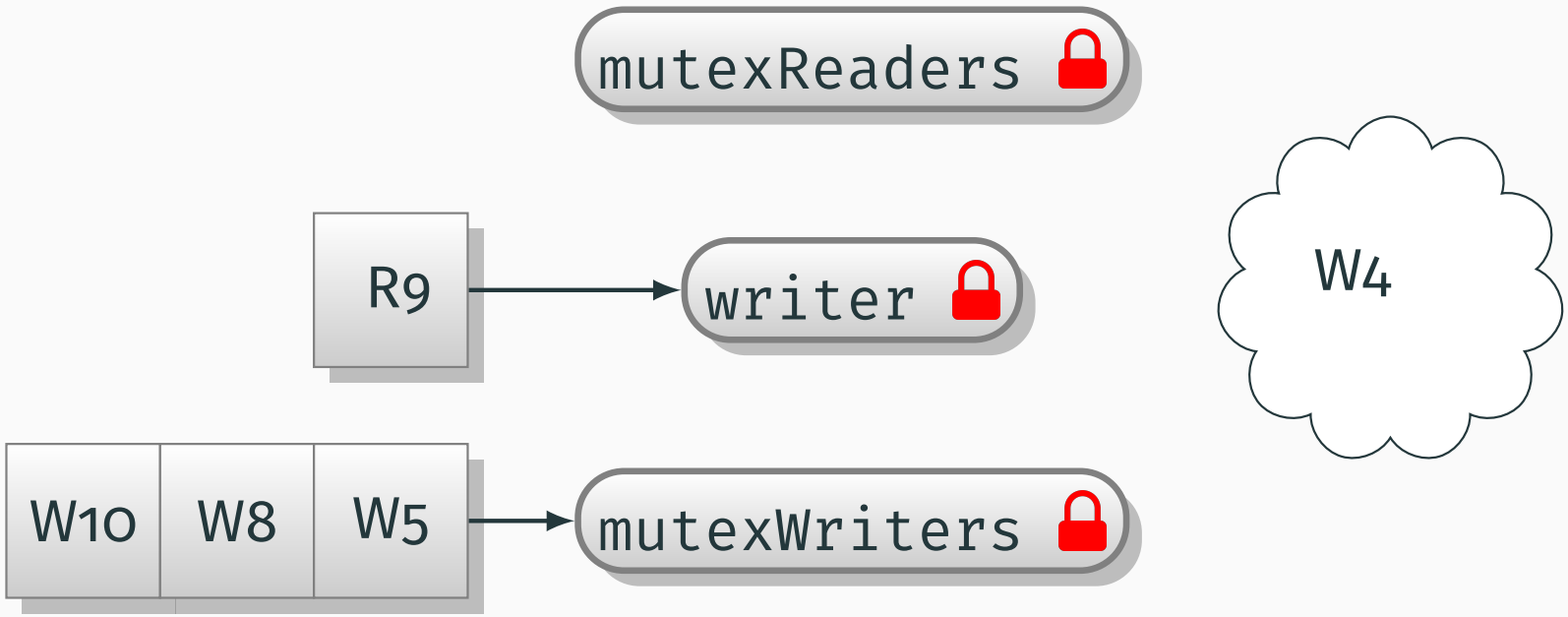
- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

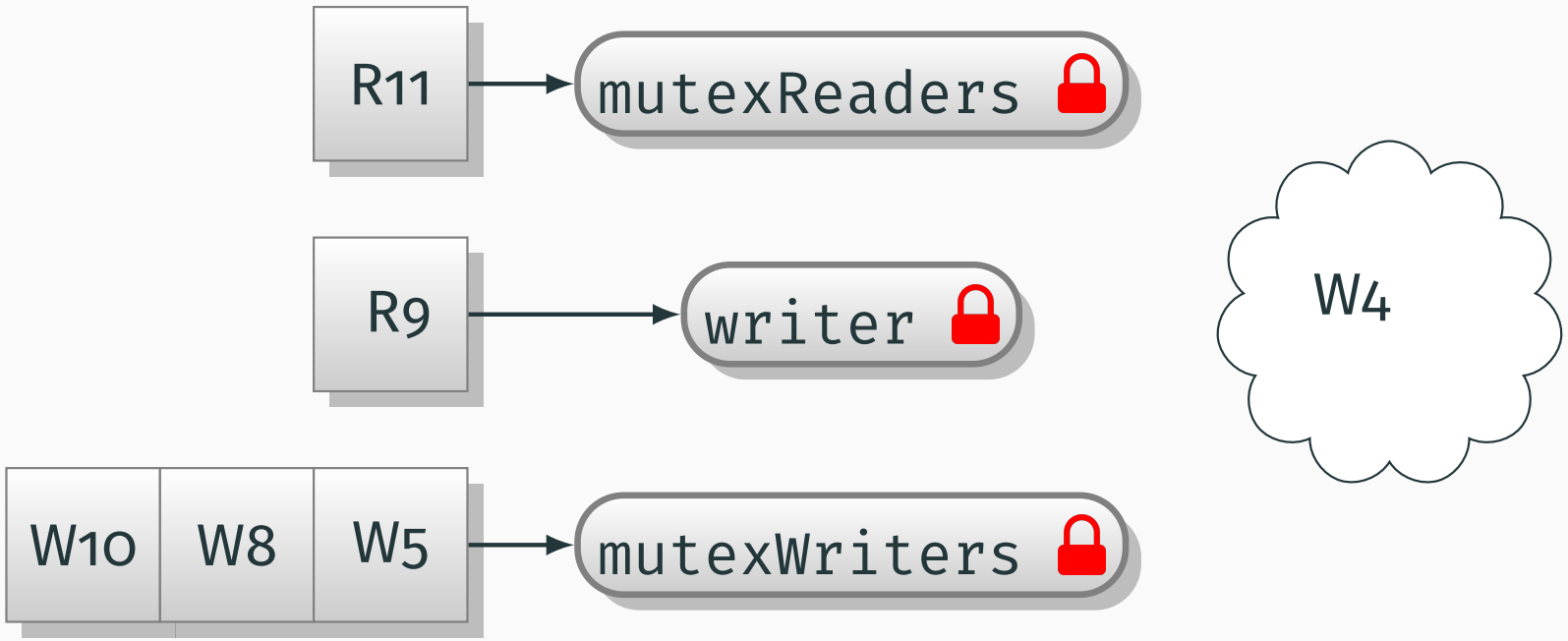
Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



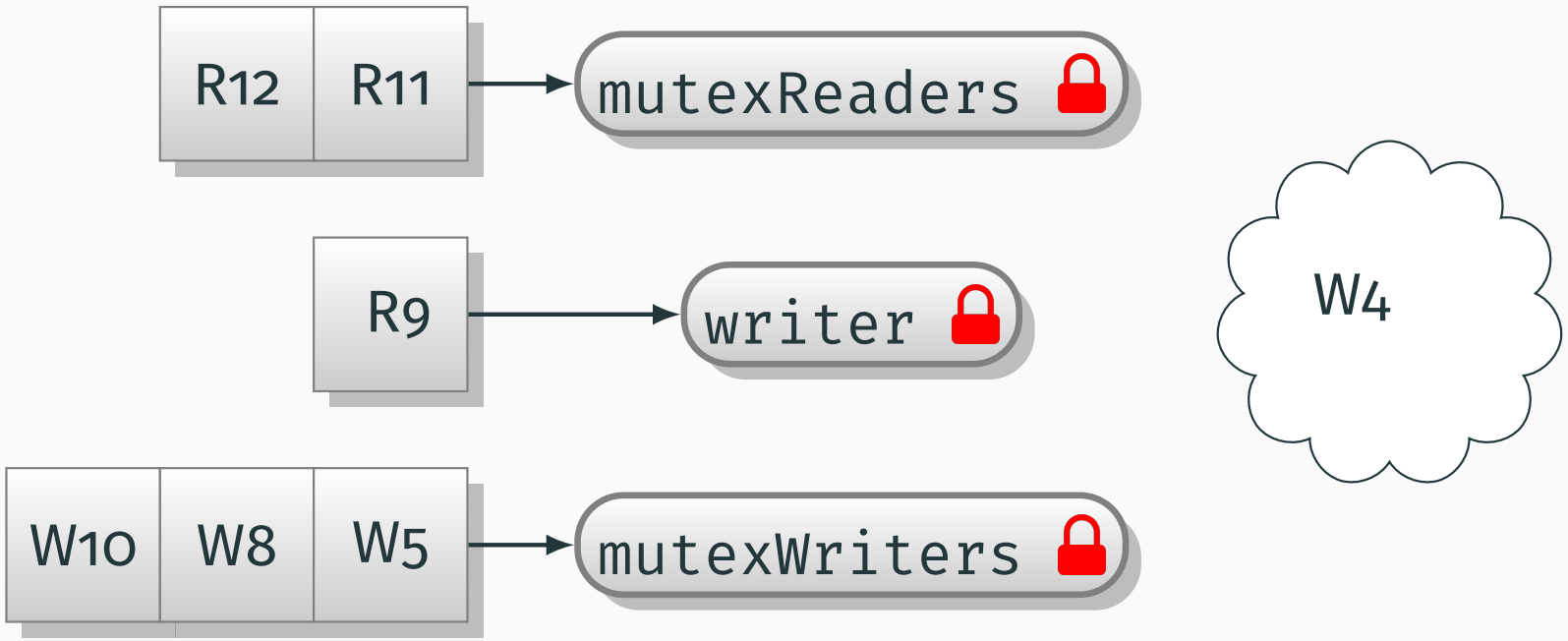
Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12



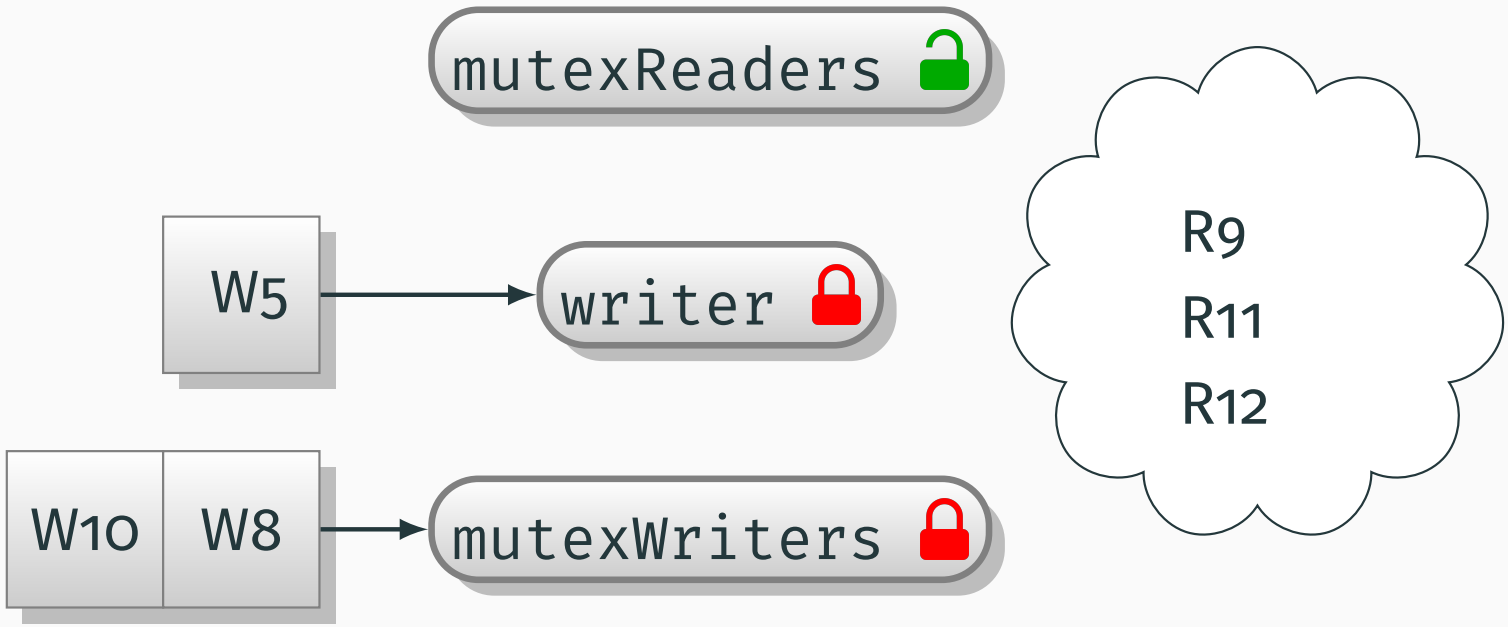
Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

Priorité aux lecteurs si lecture en cours

Même solution mais sans `mutexWriters`:

```
class ReaderWriterPrioReading : public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders;
  PcoSemaphore writer;
  int nbReaders;
public:
  ReaderWriterPrioReading() :
    mutexReaders(1),
    writer(1),
    nbReaders(0){}
```

Priorité aux lecteurs si lecture en cours

```
  void lockReading() {









  }
  void unlockReading() {









  }
  void lockWriting() {
  }
  void unlockWriting() {
  }
};
```

Priorité aux lecteurs si lecture en cours

```
  void lockReading() {









  }

  void unlockReading() {









  }

  void lockWriting() {
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
  }
};
```

Priorité aux lecteurs si lecture en cours

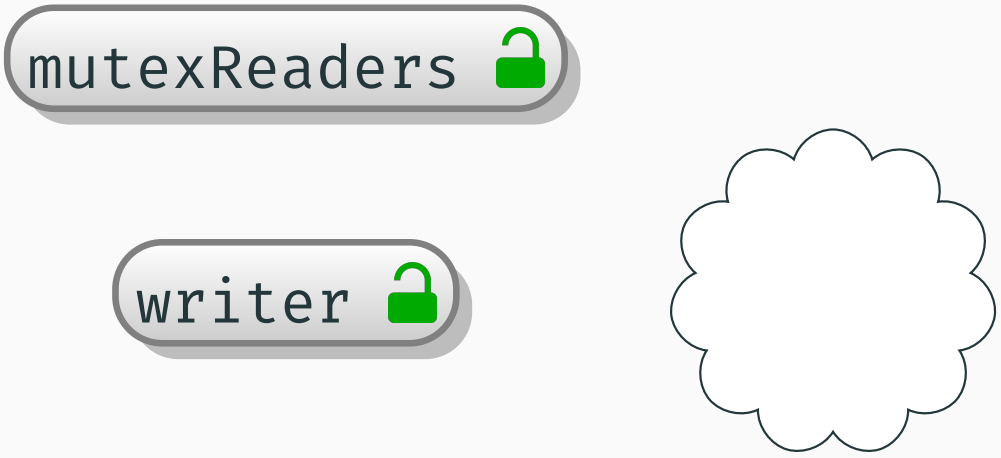
```
  void lockReading() {

    nbReaders++;
    if (nbReaders==1) {
      writer.acquire();
    }

  }

  void unlockReading() {

    nbReaders -= 1;
    if (nbReaders==0) {
      writer.release();
    }

  }

  void lockWriting() {
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
  }
};
```

Priorité aux lecteurs si lecture en cours

```
  void lockReading() {
    mutexReaders.acquire();
    nbReaders++;
    if (nbReaders==1) {
      writer.acquire();
    }
    mutexReaders.release();
  }

  void unlockReading() {
    mutexReaders.acquire();
    nbReaders -= 1;
    if (nbReaders==0) {
      writer.release();
    }
    mutexReaders.release();
  }

  void lockWriting() {
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
  }
};
```

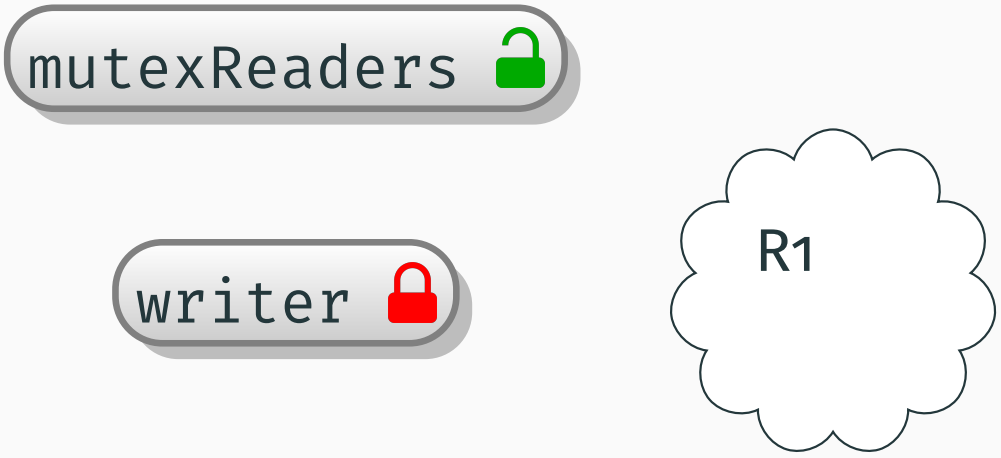
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



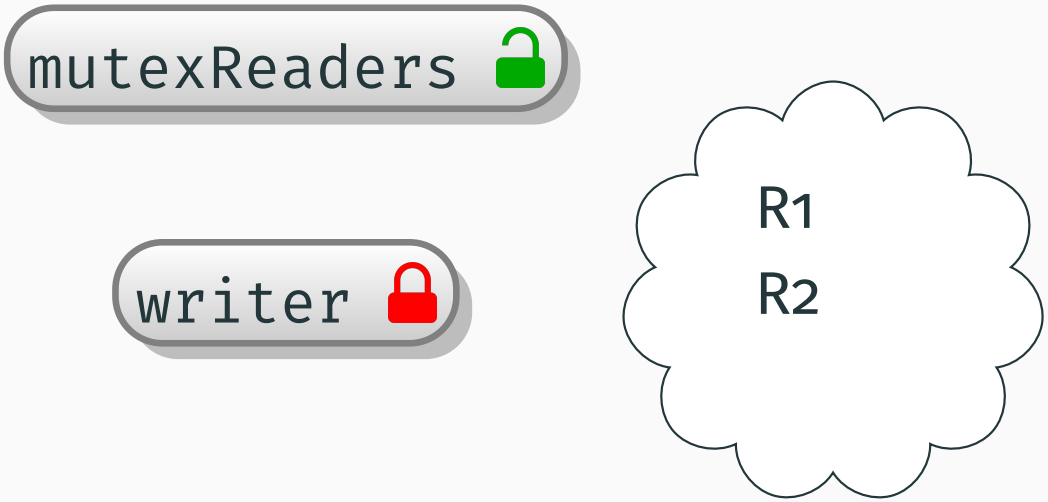
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



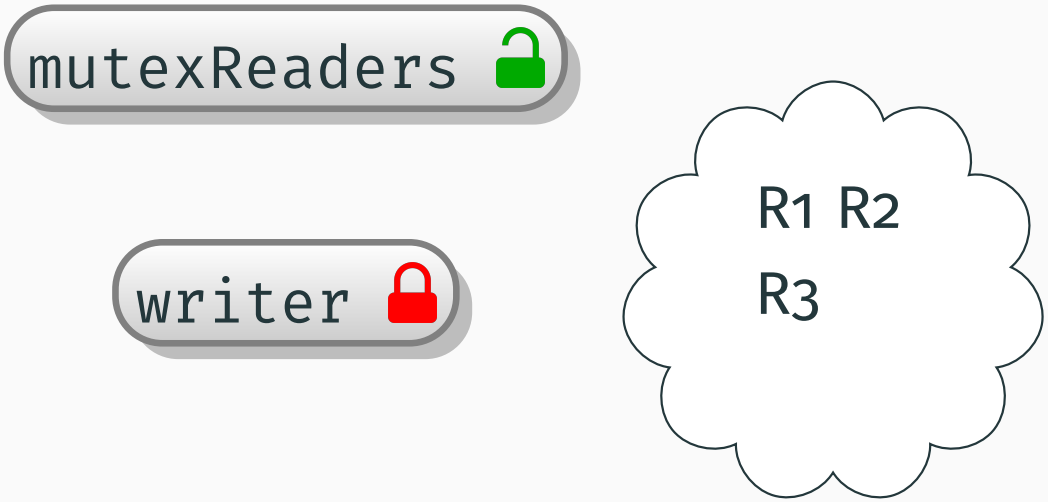
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12



Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12

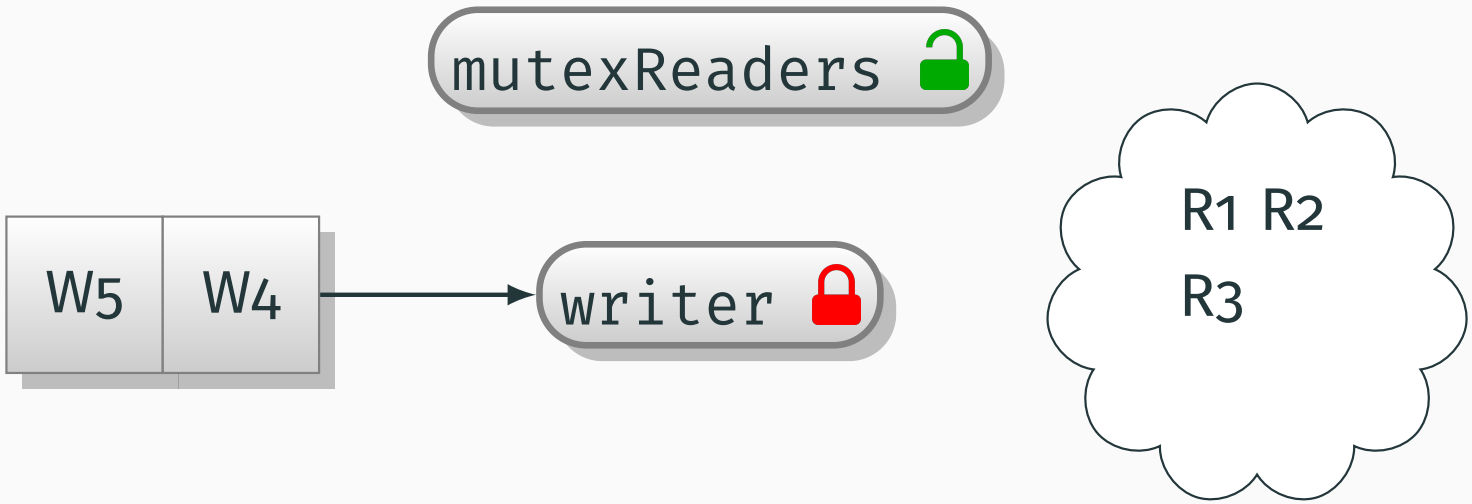


Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*

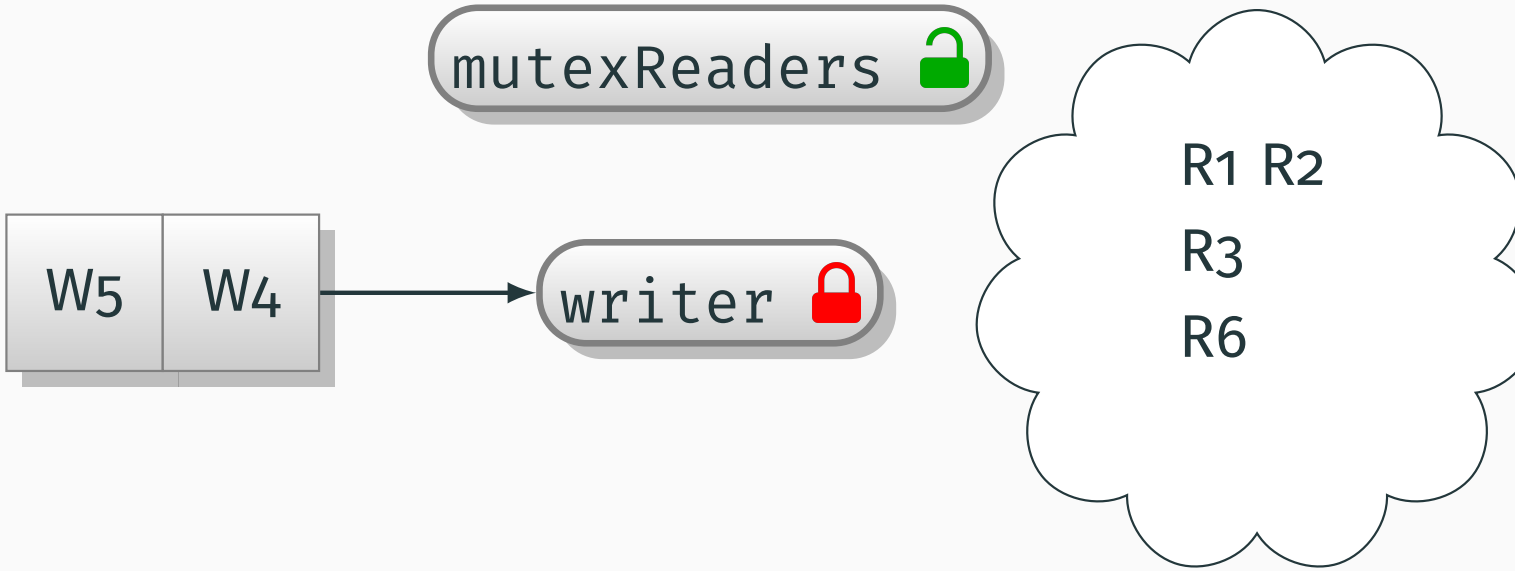
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12



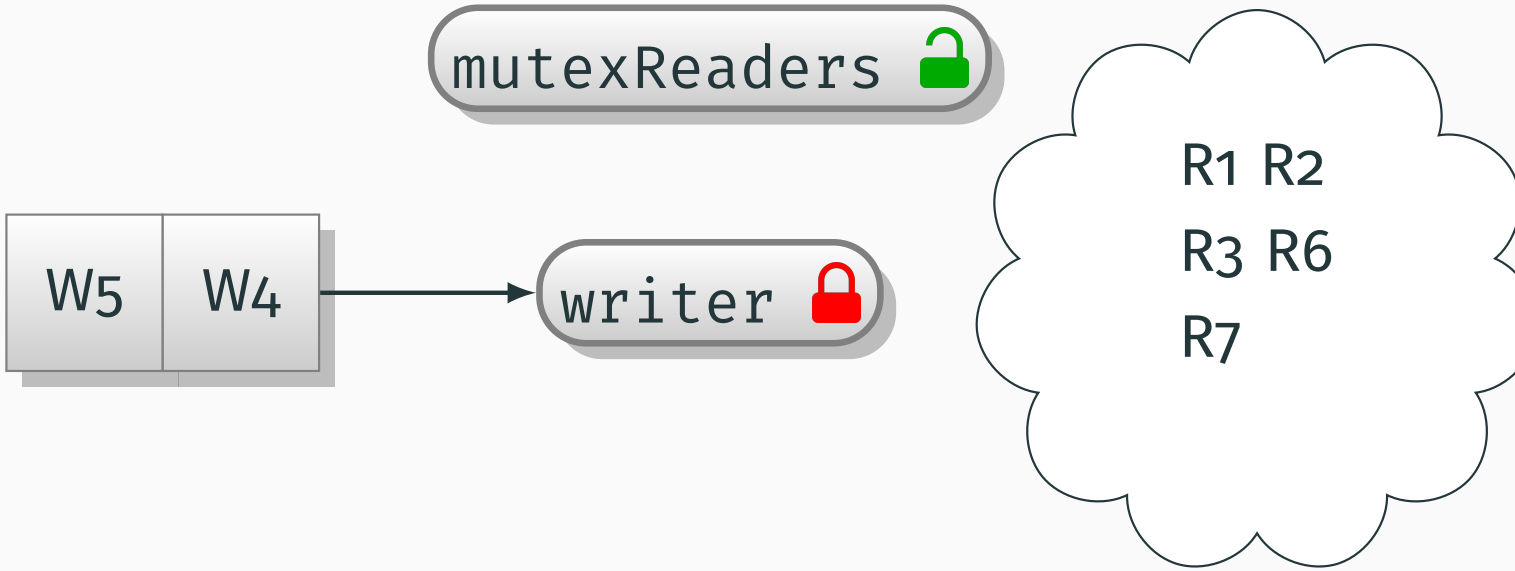
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



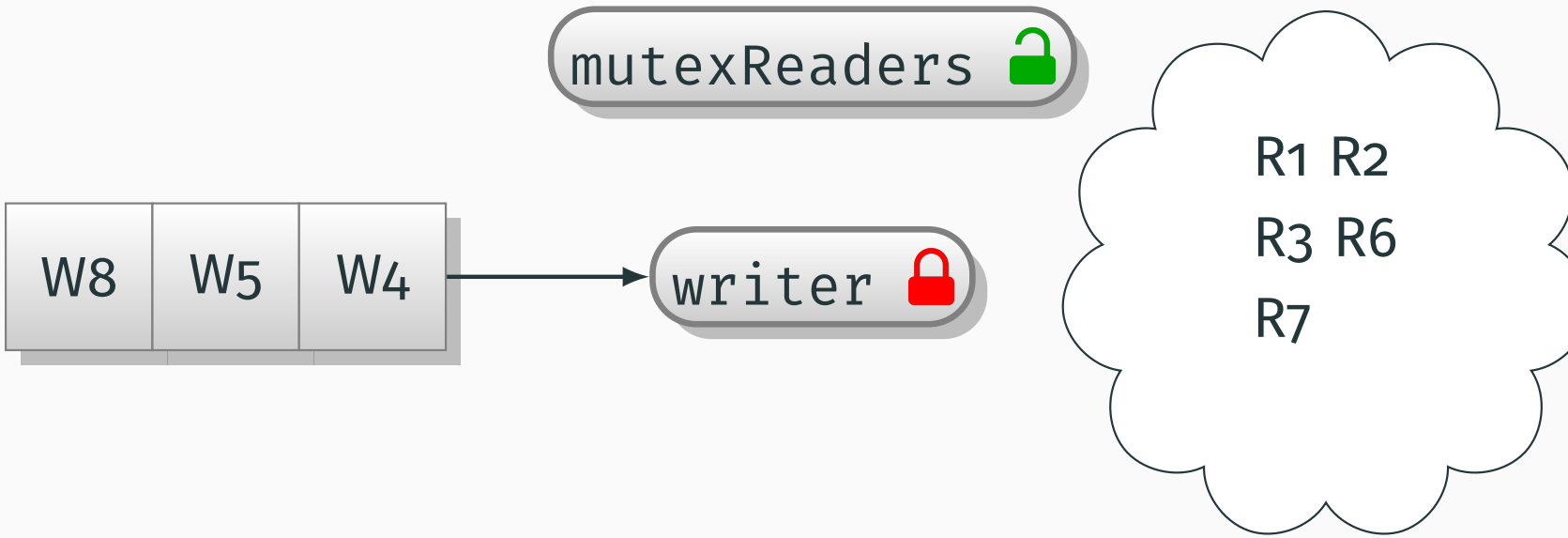
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12



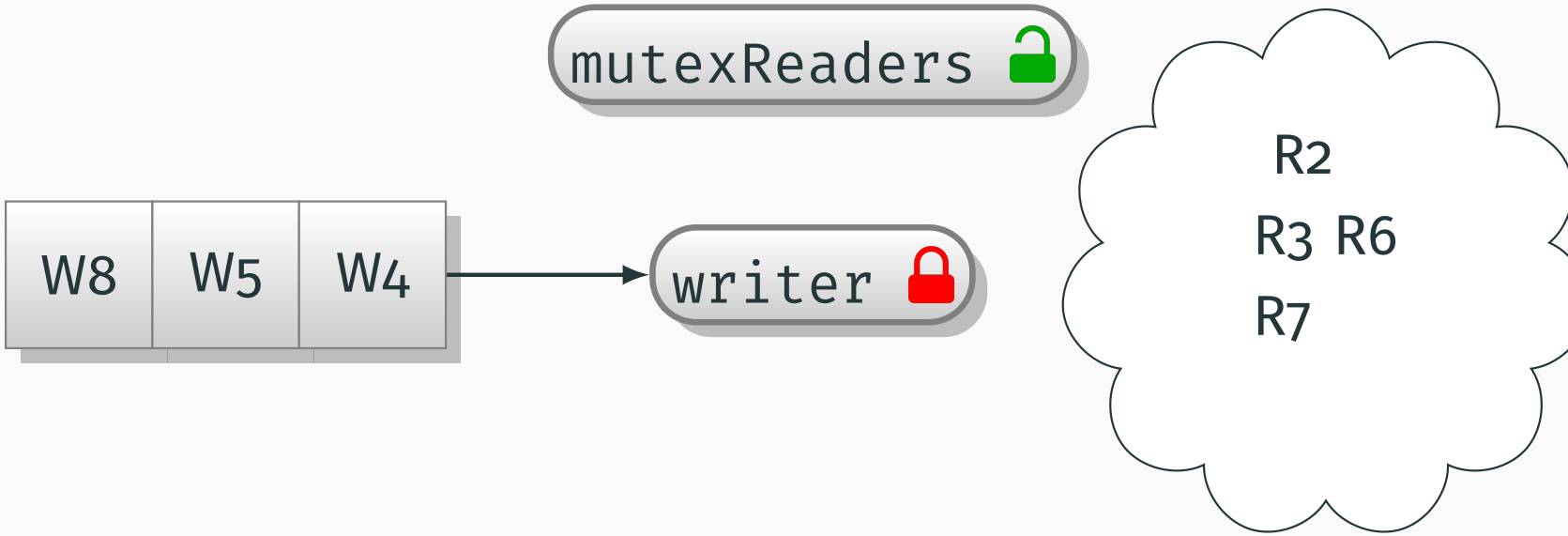
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



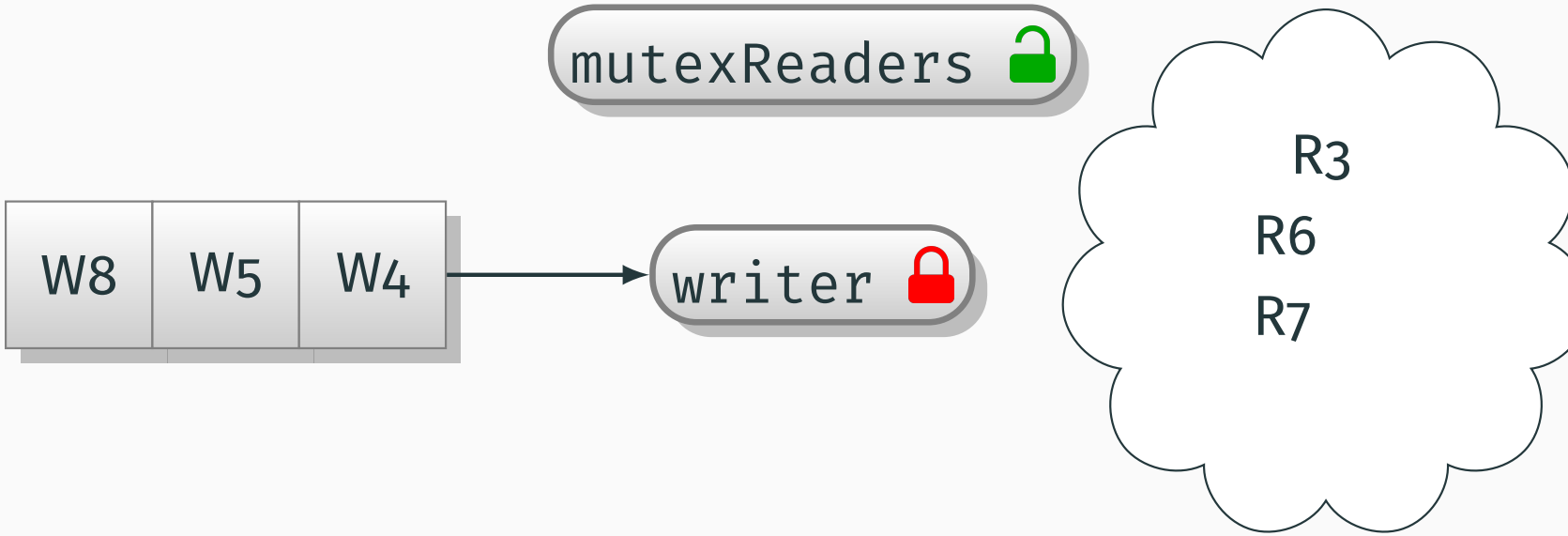
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



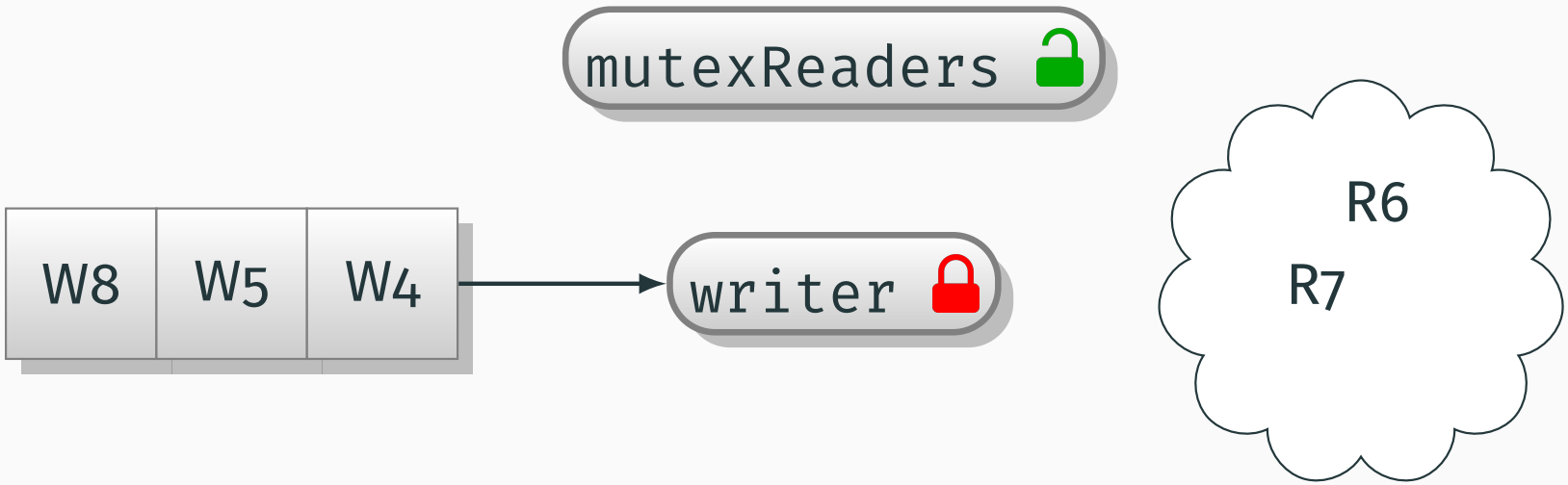
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



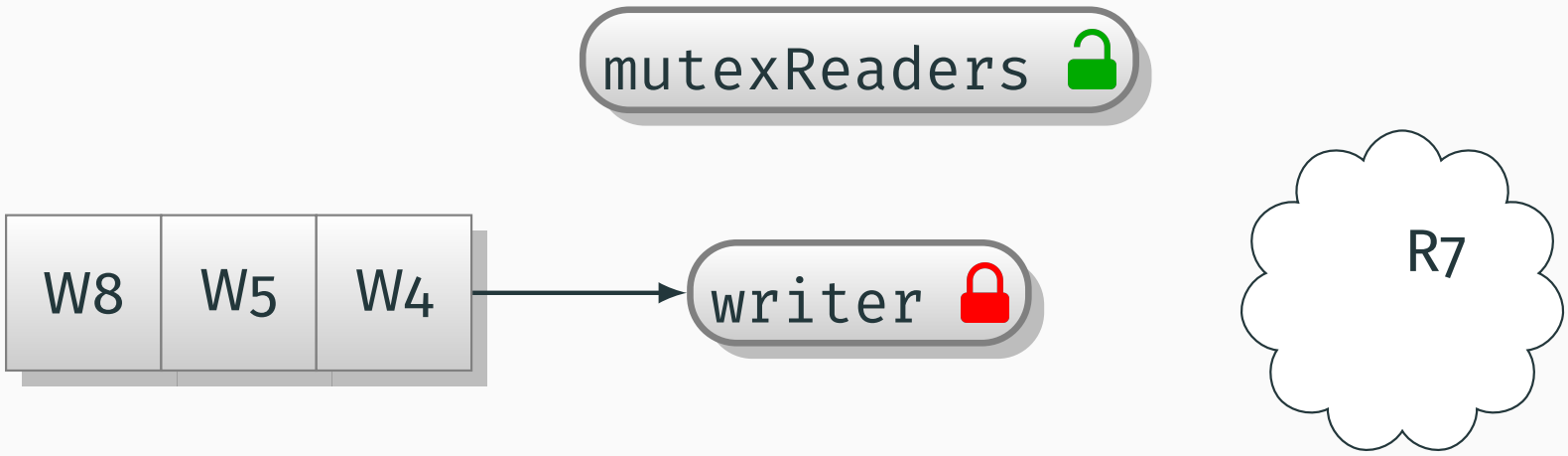
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12



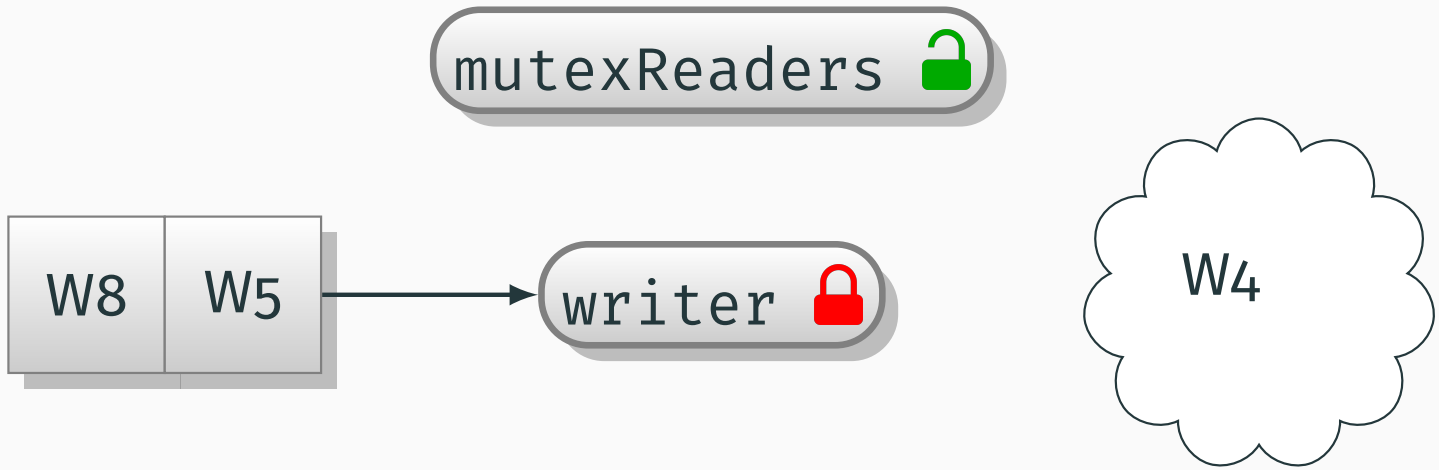
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12



Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$

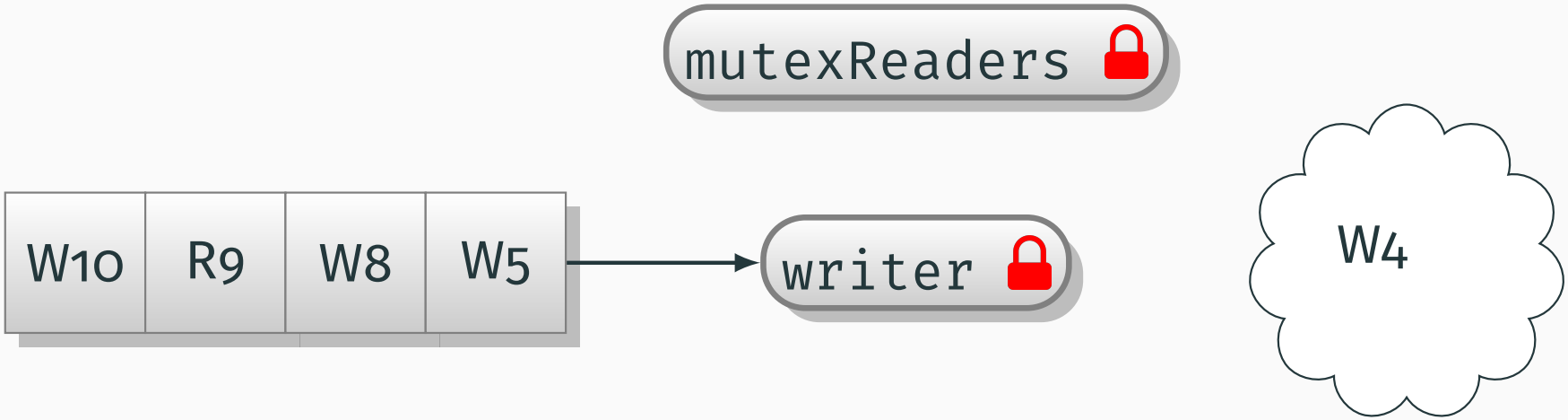


Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12

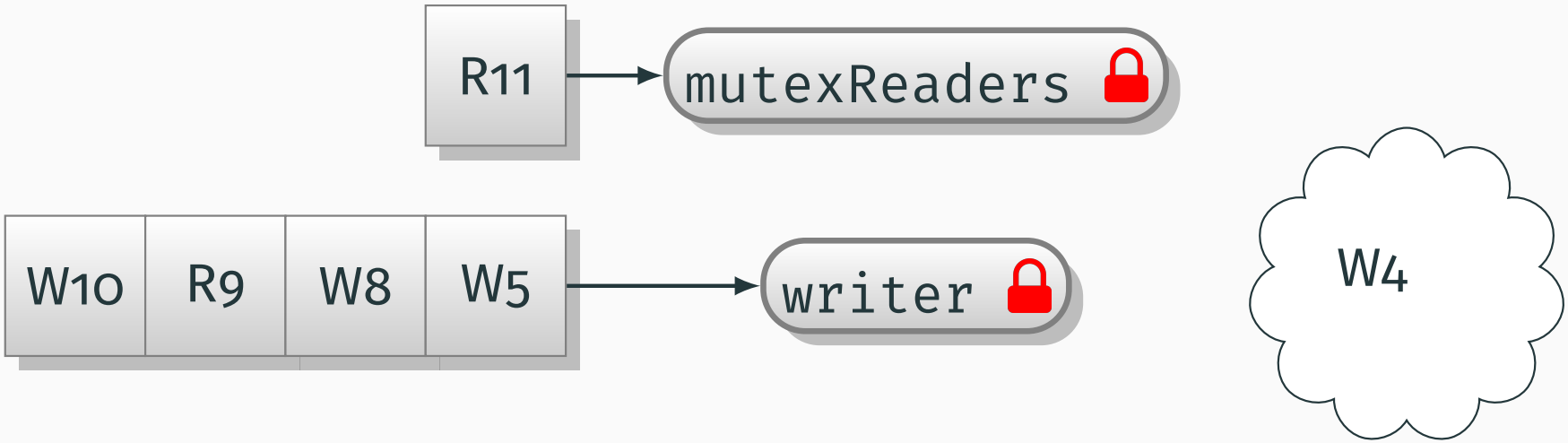
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12



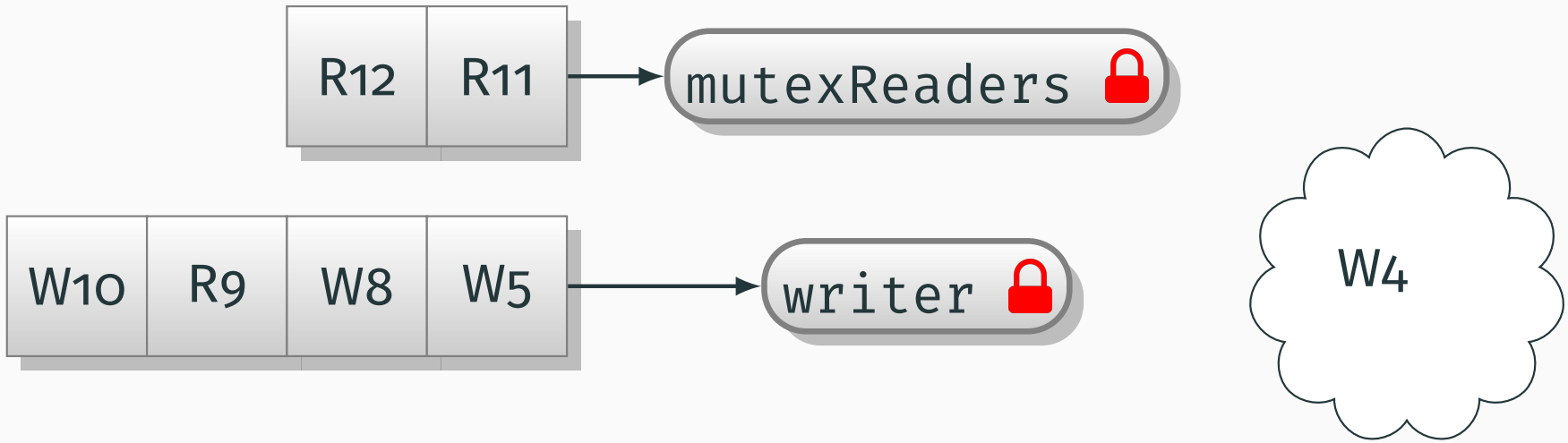
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



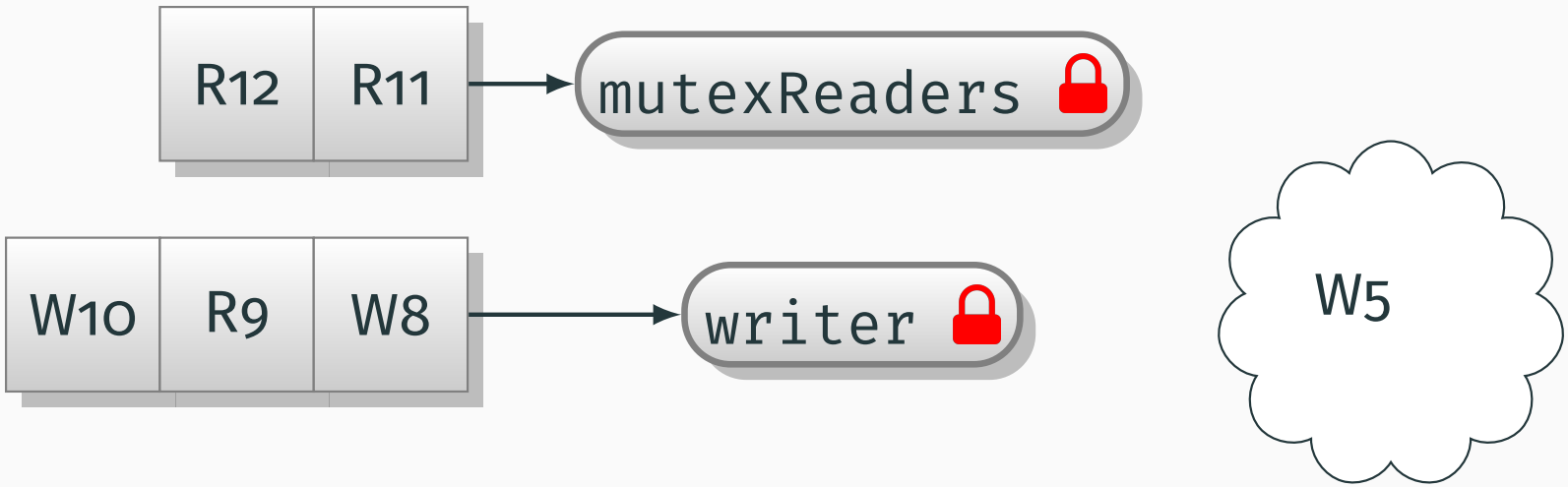
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



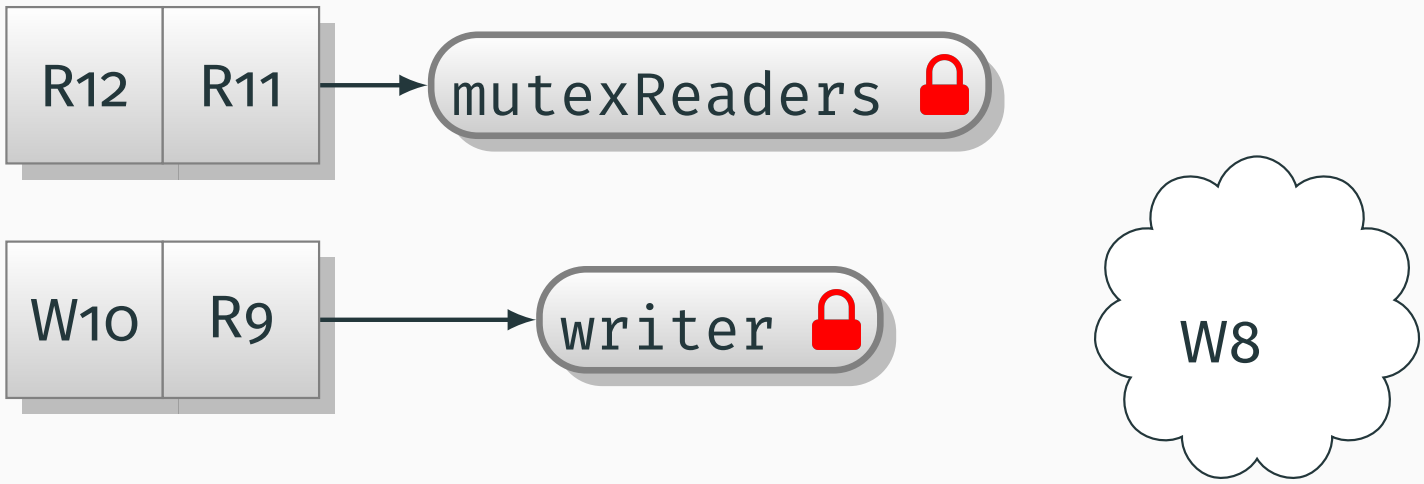
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



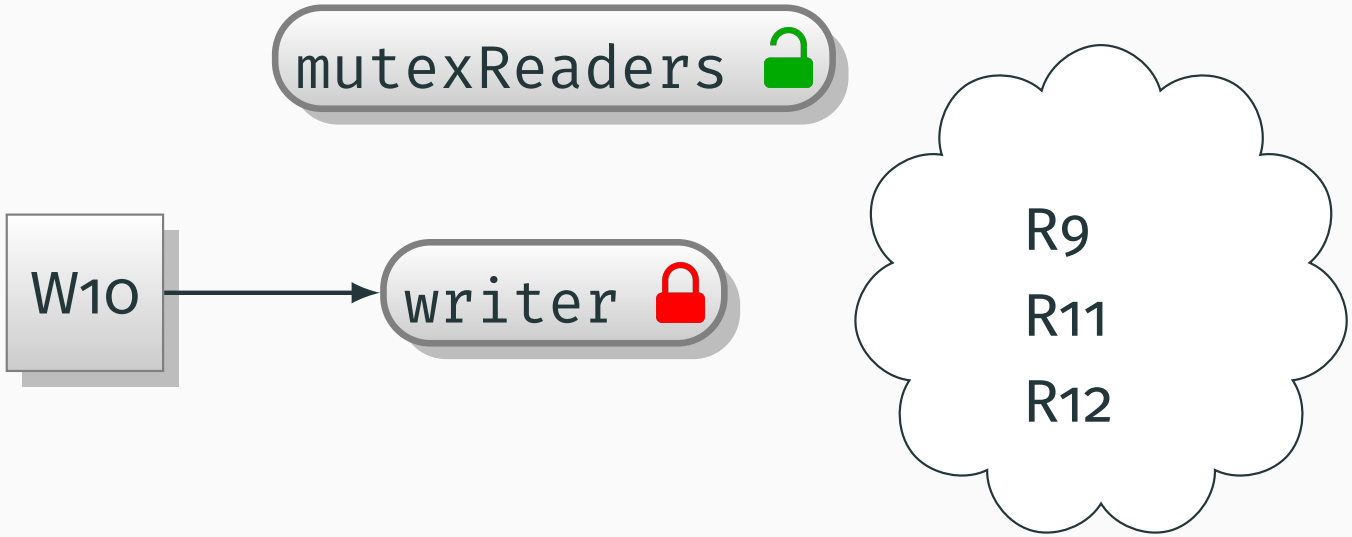
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*



Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12



Priorité aux lecteurs si lecture en cours - Exemple

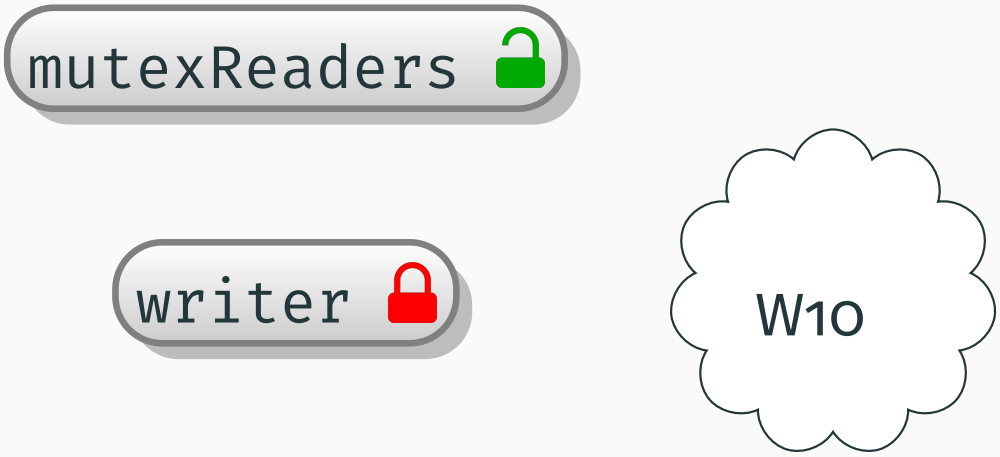
- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R9*, *W10*, *R11*, *R12*

Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de *R*1, *R*2, *R*3, *W*4, *W*5, *R*6, *R*7, *W*8
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de *R*9, *W*10, *R*11, *R*12

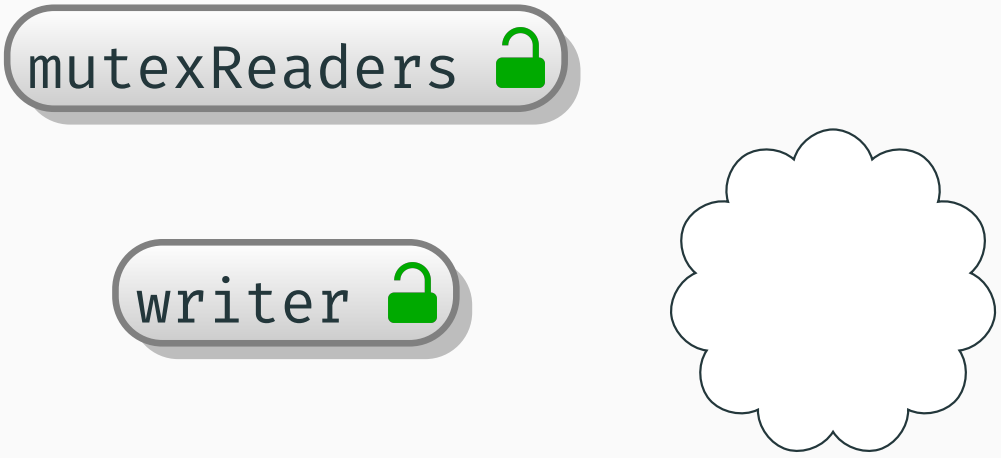
Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



Priorité aux lecteurs si lecture en cours - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$
  - Tous les lecteurs quittent l'accès à la ressource
  - Arrivée de $R9$, $W10$, $R11$, $R12$



Priorité aux lecteurs: algorithme général

- Algorithme selon un autre modèle

```
class ReaderWriterPrioReader2 :
      public AbstractReaderWriter {
protected:
  PcoSemaphore mutex;
  PcoSemaphore readerBlocker;
  PcoSemaphore writerBlocker;
  int nbReaders;
  int nbReadersWaiting;
  int nbWritersWaiting;
  bool oneWriter;
```

```
public:

  ReaderWriterPrioReader2() :
    mutex(1),
    nbReaders(0),
    nbReadersWaiting(0),
    nbWritersWaiting(0),
    oneWriter(false) {}
```

Algorithme général

```
void lockReading() {
  mutex.acquire();
  if (oneWriter) {
    nbReadersWaiting++;
    mutex.release();
    readerBlocker.acquire();
  }
  else {
    nbReaders++;
    mutex.release();
  }
}

void unlockReading() {
  mutex.acquire();
  nbReaders--;
  if (nbReaders==0) {
    if (nbWritersWaiting>0) {
      oneWriter=true;
      nbWritersWaiting--;
      writerBlocker.release();
    }
  }
  mutex.release();
}
```

```
void lockWriting() {
  mutex.acquire();
  if (oneWriter || (nbReaders>0)
      || (nbReadersWaiting>0)) {
    nbWritersWaiting++;
    mutex.release();
    writerBlocker.acquire();
  }
  else {
    oneWriter=true;
    mutex.release();}}

void unlockWriting() {
  mutex.acquire();
  oneWriter=false;
  if (nbReadersWaiting>0) {
    for(int i=0;i<nbReadersWaiting;i++)
      readerBlocker.release();
    nbReaders=nbReadersWaiting;
    nbReadersWaiting=0;
  }
  else {
    if (nbWritersWaiting>0) {
      oneWriter=true;
      nbWritersWaiting--;
      writerBlocker.release();
    }
  }
  mutex.release();
}
```

Priorité égale

Priorité égale - Règles

- Un lecteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0

Priorité égale - Règles

- Un lecteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0
- Un rédacteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0
  - ET le nombre de lecteurs en cours de lecture vaut 0

Variables et sémaphores nécessaires

- Une variable `nbReaders`

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutex`, qui est en charge de protéger l'accès à la variable `nbReaders`.

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutex`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- `writer`, qui permet au premier lecteur qui accède la ressource de bloquer les futurs rédacteurs.
  ⚠ Il ne sera pas utilisé par les rédacteurs pour bloquer les lecteurs.

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutex`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- `writer`, qui permet au premier lecteur qui accède la ressource de bloquer les futurs rédacteurs.
  ⚠ Il ne sera pas utilisé par les rédacteurs pour bloquer les lecteurs.
- $\Rightarrow$ `fifo`, une file d'attente dans laquelle passent tous les lecteurs et rédacteurs.

Priorité égale

```
class ReaderWriterEqual : public AbstractReaderWriter {
protected:
  PcoSemaphore mutex;
  PcoSemaphore fifo;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterEqual() :
    mutex(1),
    fifo(1),
    writer(1),
    nbReaders(0) {}

  void lockReading() {




  }
```

```
  void unlockReading() {




  }

  void lockWriting() {


  }

  void unlockWriting() {


  }
};
```

Priorité égale

```
class ReaderWriterEqual : public AbstractReaderWriter {
protected:
  PcoSemaphore mutex;
  PcoSemaphore fifo;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterEqual() :
    mutex(1),
    fifo(1),
    writer(1),
    nbReaders(0) {}

  void lockReading() {
    fifo.acquire();




    fifo.release();
  }
```

```
  void unlockReading() {




  }

  void lockWriting() {
    fifo.acquire();

  }

  void unlockWriting() {

    fifo.release();
  }
};
```

Priorité égale

```
class ReaderWriterEqual : public AbstractReaderWriter {
protected:
  PcoSemaphore mutex;
  PcoSemaphore fifo;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterEqual() :
    mutex(1),
    fifo(1),
    writer(1),
    nbReaders(0) {}

  void lockReading() {
    fifo.acquire();

    nbReaders++;
    if (nbReaders==1) {
      writer.acquire();
    }

    fifo.release();
  }

  void unlockReading() {

    nbReaders -= 1;
    if (nbReaders==0) {
      writer.release();
    }

  }

  void lockWriting() {
    fifo.acquire();
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
    fifo.release();
  }
};
```

Priorité égale

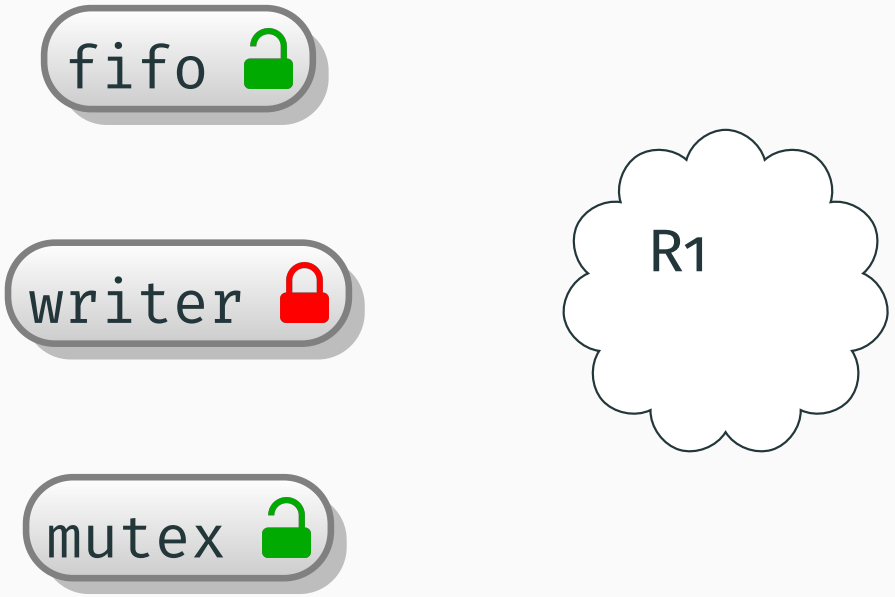
```
class ReaderWriterEqual : public AbstractReaderWriter {
protected:
  PcoSemaphore mutex;
  PcoSemaphore fifo;
  PcoSemaphore writer;
  int nbReaders;

public:
  ReaderWriterEqual() :
    mutex(1),
    fifo(1),
    writer(1),
    nbReaders(0) {}

  void lockReading() {
    fifo.acquire();
    mutex.acquire();
    nbReaders++;
    if (nbReaders==1) {
      writer.acquire();
    }
    mutex.release();
    fifo.release();
  }

  void unlockReading() {
    mutex.acquire();
    nbReaders -= 1;
    if (nbReaders==0) {
      writer.release();
    }
    mutex.release();
  }

  void lockWriting() {
    fifo.acquire();
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
    fifo.release();
  }
};
```

Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

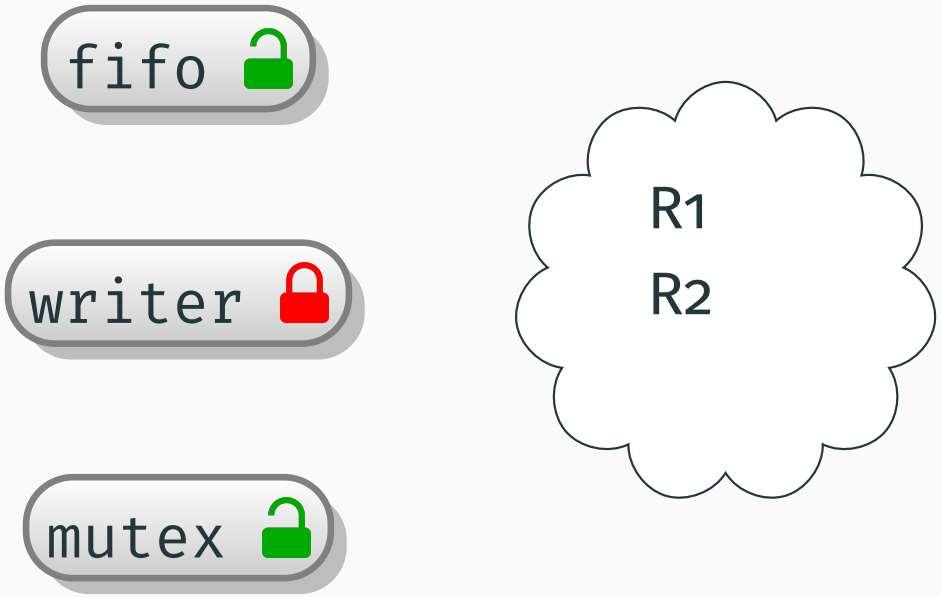
Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



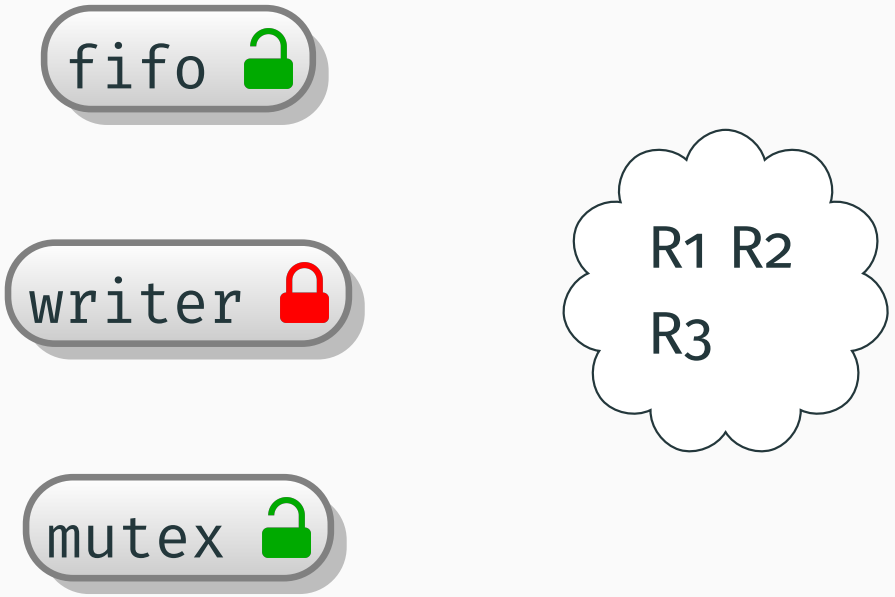
Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité égale - Exemple

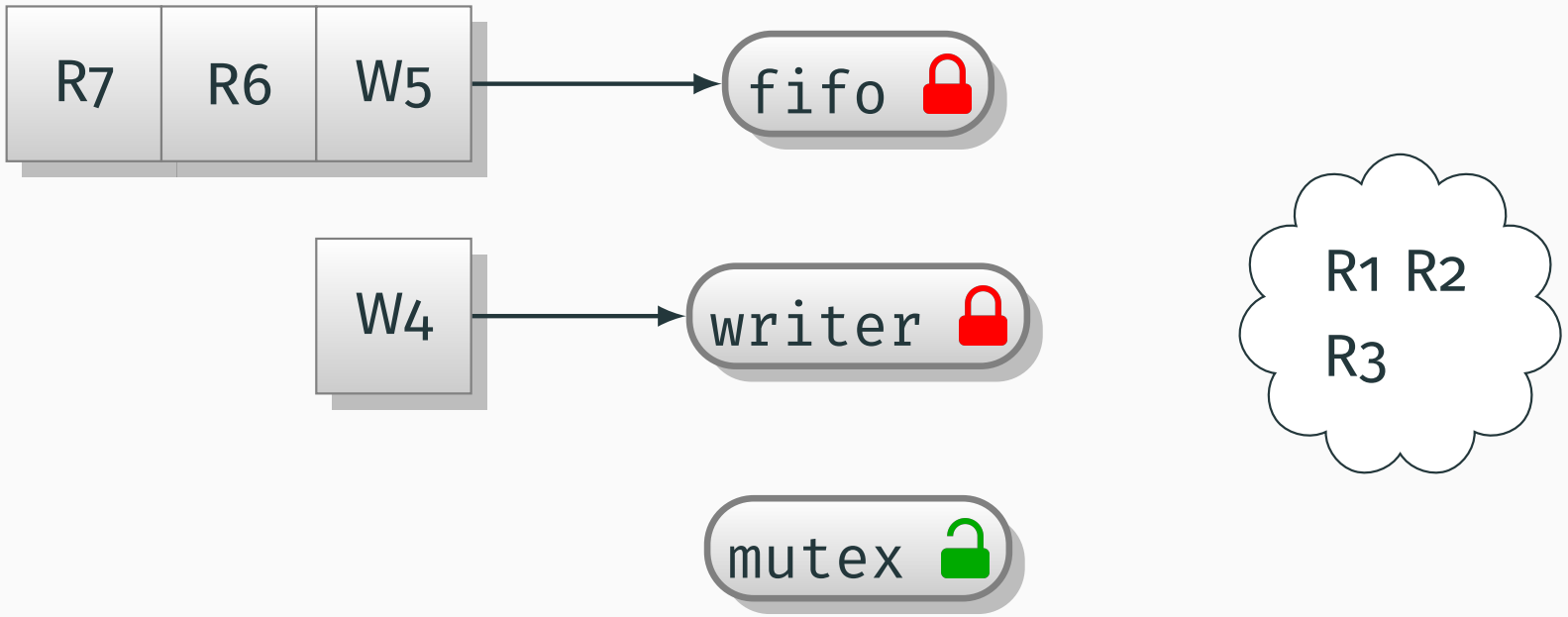
- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

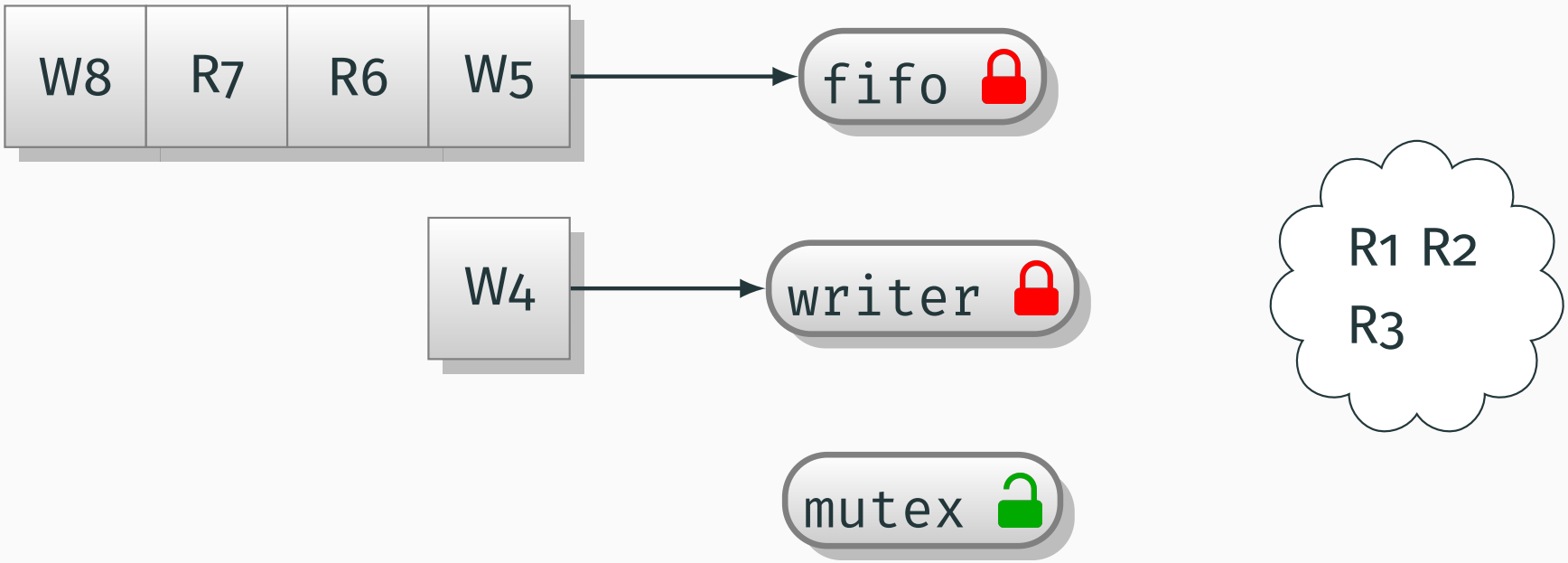
Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



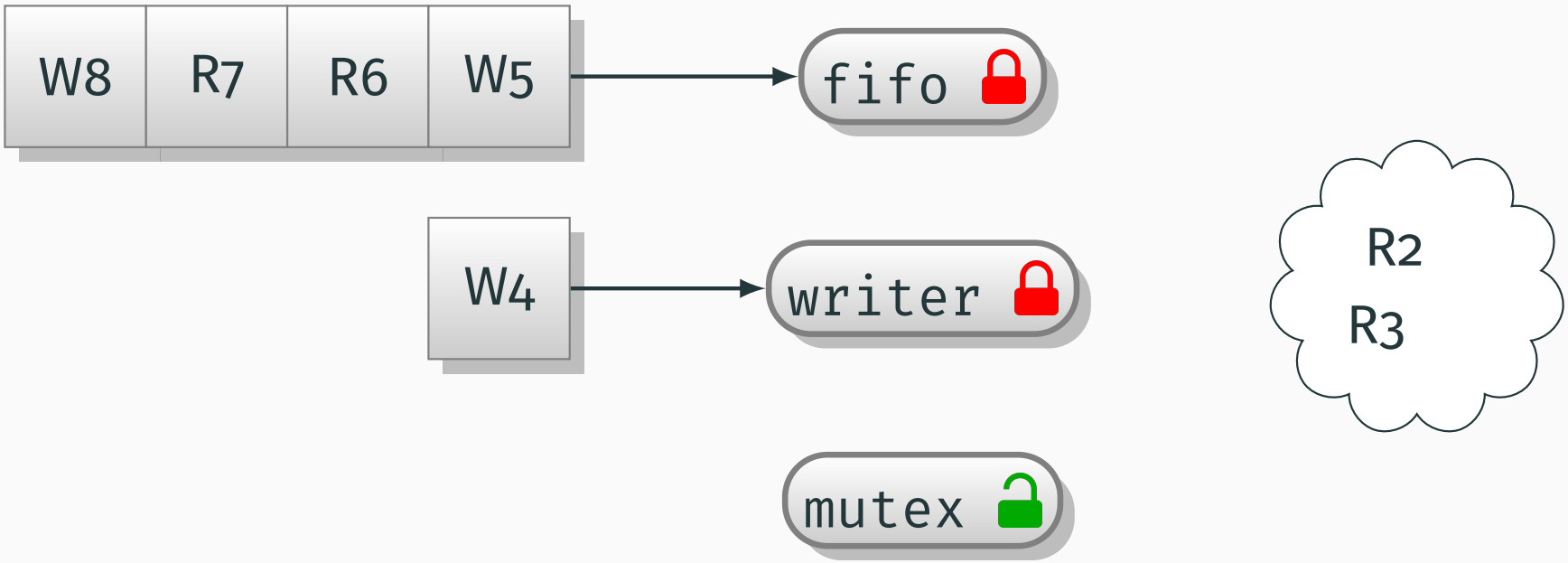
Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



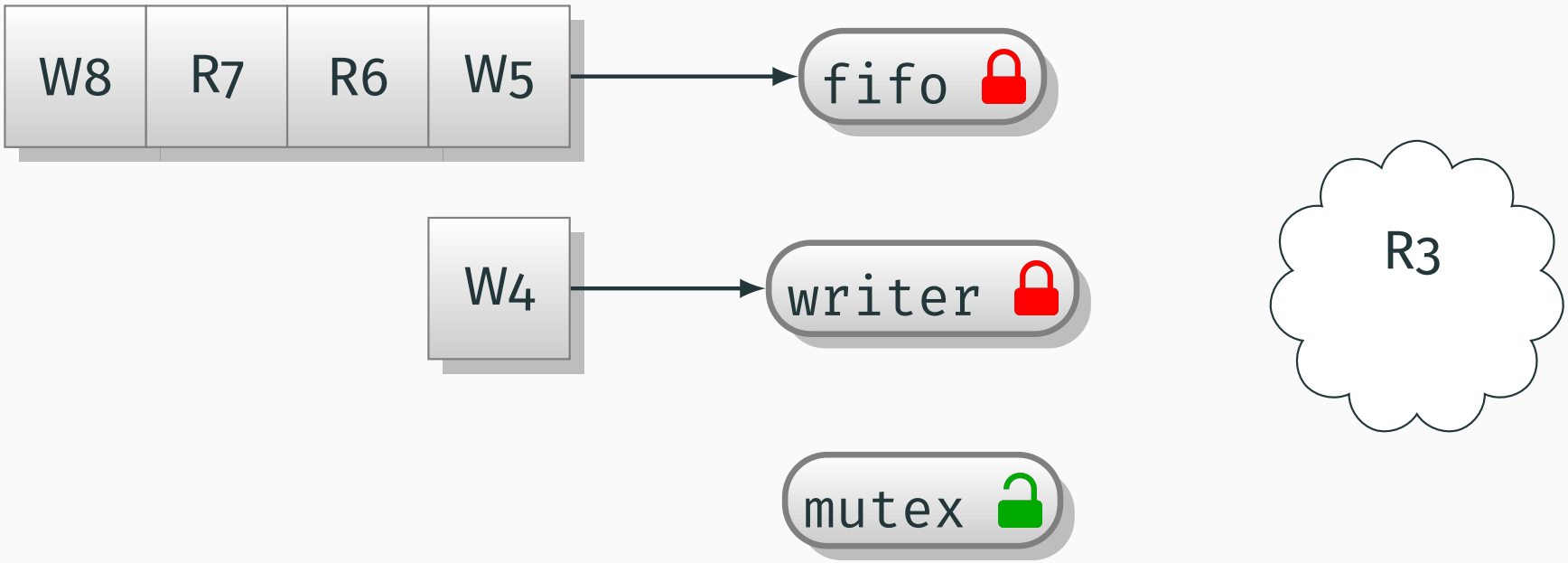
Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



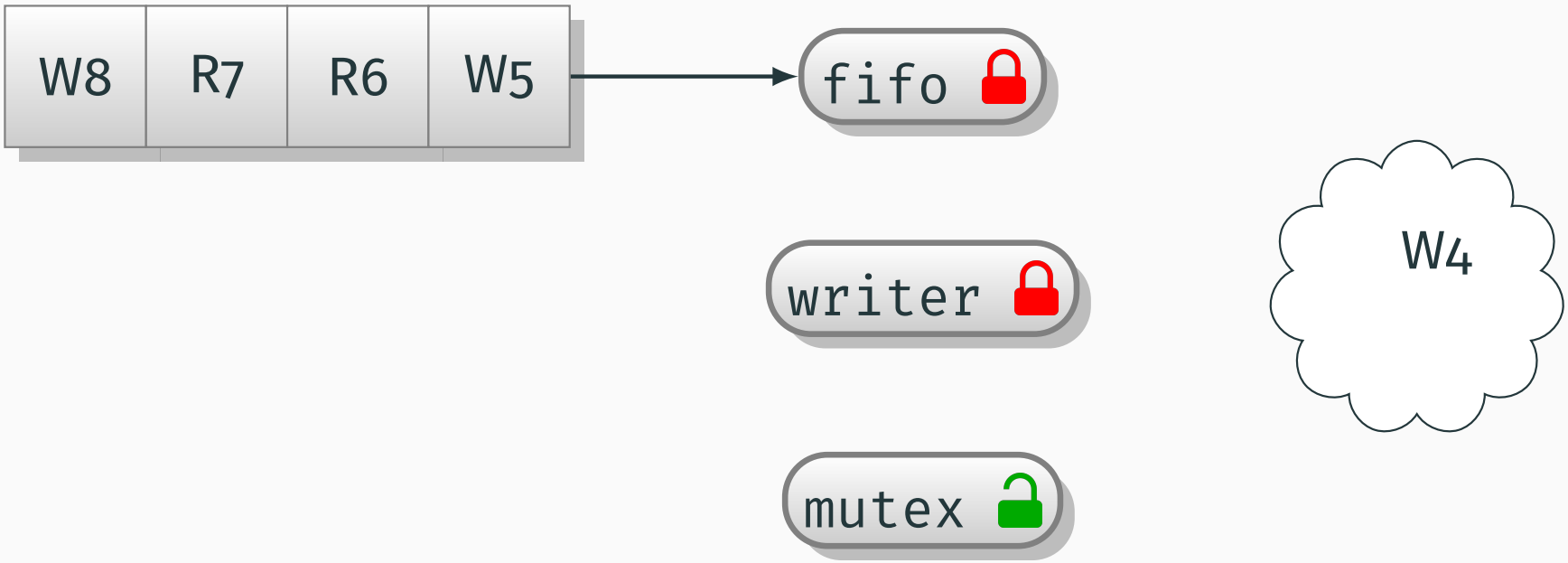
Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



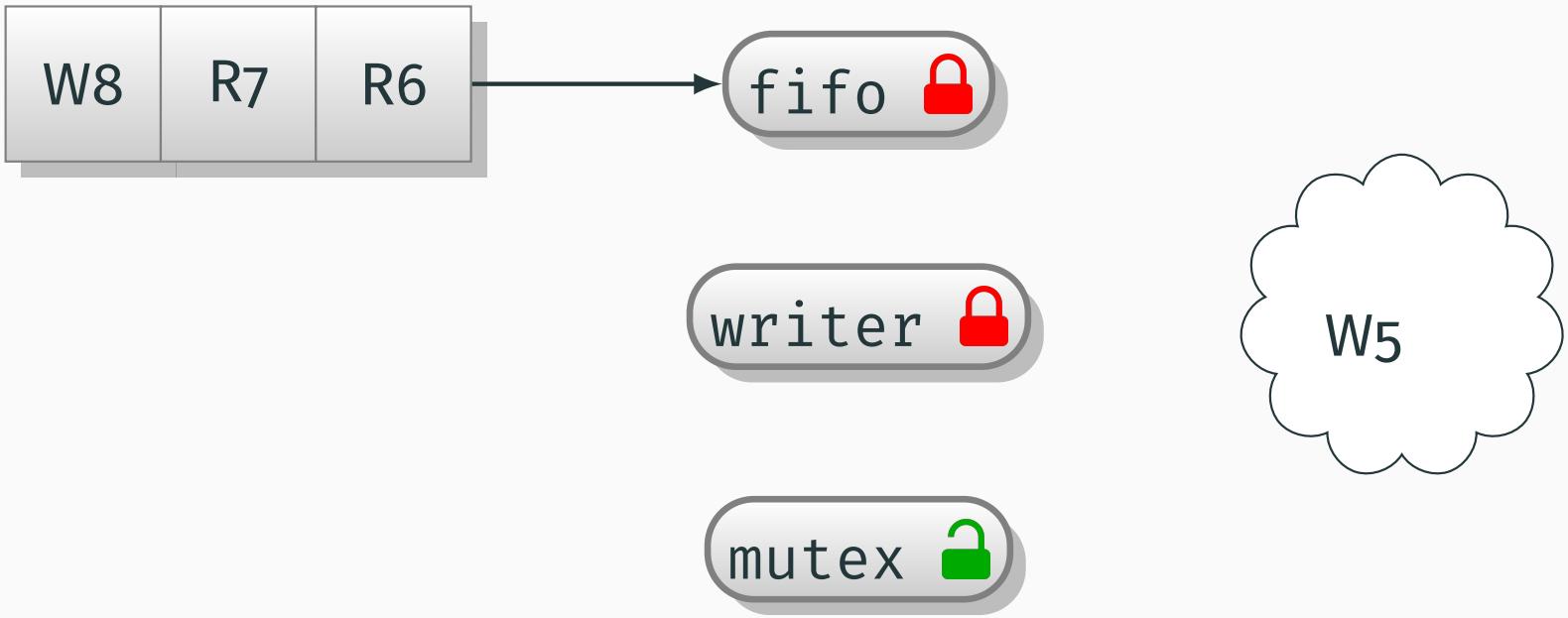
Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



Priorité égale - Exemple

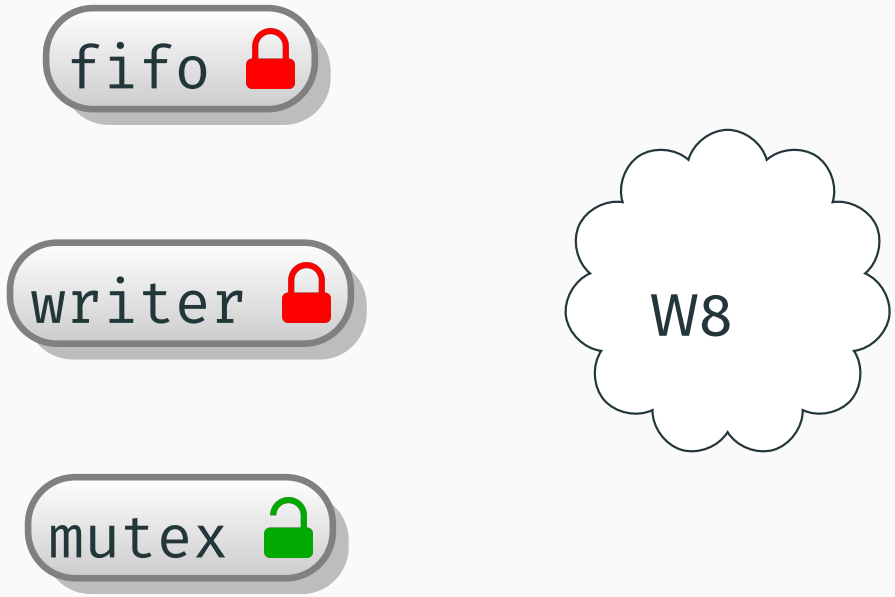
- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



Priorité égale - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité aux rédacteurs

Priorité aux rédacteurs - Règles

- Un lecteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0
  - ET le nombre de rédacteurs en attente d'écriture vaut 0

Priorité aux rédacteurs - Règles

- Un lecteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0
  - ET le nombre de rédacteurs en attente d'écriture vaut 0
- Un rédacteur peut accéder à la ressource si:
  - Le nombre de rédacteurs en cours d'écriture vaut 0
  - ET le nombre de lecteurs en cours de lecture vaut 0

Variables et sémaphores nécessaires

- Une variable `nbReaders`

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutex`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- Une variable `nbWriters`

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutex`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- Une variable `nbWriters`
- `mutexWriters`, qui est en charge de protéger l'accès à la variable `nbWriters`.

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutex`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- Une variable `nbWriters`
- `mutexWriters`, qui est en charge de protéger l'accès à la variable `nbWriters`.
- `writer`, qui permet au premier lecteur qui accède la ressource de bloquer les potentiels rédacteurs.

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutex`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- Une variable `nbWriters`
- `mutexWriters`, qui est en charge de protéger l'accès à la variable `nbWriters`.
- `writer`, qui permet au premier lecteur qui accède la ressource de bloquer les potentiels rédacteurs.
- `reader`, qui permet au premier rédacteur arrivé de bloquer les potentiels futurs lecteurs.

Variables et sémaphores nécessaires

- Une variable `nbReaders`
- `mutex`, qui est en charge de protéger l'accès à la variable `nbReaders`.
- Une variable `nbWriters`
- `mutexWriters`, qui est en charge de protéger l'accès à la variable `nbWriters`.
- `writer`, qui permet au premier lecteur qui accède la ressource de bloquer les potentiels rédacteurs.
- `reader`, qui permet au premier rédacteur arrivé de bloquer les potentiels futurs lecteurs.
- $\Rightarrow$ `mutexReaders`, qui permet de favoriser les réacteurs selon le même principe que la priorité au lecteurs en empêchant plusieurs lecteurs de s'ajouter à `reader`

Priorité aux rédacteurs

```
class ReaderWriterPrioWriter :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders, mutexWriters;
  PcoSemaphore writer, reader;
  PcoSemaphore mutex;
  int nbReaders, nbWriters;

public:
  ReaderWriterPrioWriter() :
    mutexReaders(1), mutexWriters(1),
    writer(1), reader(1),
    mutex(1),
    nbReaders(0), nbWriters(0) {}

  void lockReading() {









  }
```

```
  void unlockReading() {




  }

  void lockWriting() {




  }

  void unlockWriting() {




  }
};
```

Priorité aux rédacteurs

```
class ReaderWriterPrioWriter :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders, mutexWriters;
  PcoSemaphore writer, reader;
  PcoSemaphore mutex;
  int nbReaders, nbWriters;

public:
  ReaderWriterPrioWriter() :
    mutexReaders(1), mutexWriters(1),
    writer(1), reader(1),
    mutex(1),
    nbReaders(0), nbWriters(0) {}

  void lockReading() {









  }
```

```
  void unlockReading() {




  }

  void lockWriting() {




    writer.acquire();
  }

  void unlockWriting() {
    writer.release();




  }
};
```

Priorité aux rédacteurs

```
class ReaderWriterPrioWriter :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders, mutexWriters;
  PcoSemaphore writer, reader;
  PcoSemaphore mutex;
  int nbReaders, nbWriters;

public:
  ReaderWriterPrioWriter() :
    mutexReaders(1), mutexWriters(1),
    writer(1), reader(1),
    mutex(1),
    nbReaders(0), nbWriters(0) {}

  void lockReading() {









  }
```

```
  void unlockReading() {









  }

  void lockWriting() {

    nbWriters++;
    if (nbWriters==1)
      reader.acquire();

    writer.acquire();
  }

  void unlockWriting() {
    writer.release();

    nbWriters -= 1;
    if (nbWriters==0)
      reader.release();

  }
};
```

Priorité aux rédacteurs

```
class ReaderWriterPrioWriter :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders, mutexWriters;
  PcoSemaphore writer, reader;
  PcoSemaphore mutex;
  int nbReaders, nbWriters;

public:
  ReaderWriterPrioWriter() :
    mutexReaders(1), mutexWriters(1),
    writer(1), reader(1),
    mutex(1),
    nbReaders(0), nbWriters(0) {}

  void lockReading() {









  }

  void unlockReading() {









  }

  void lockWriting() {
    mutexWriters.acquire();
    nbWriters++;
    if (nbWriters==1)
      reader.acquire();
    mutexWriters.release();
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
    mutexWriters.acquire();
    nbWriters -= 1;
    if (nbWriters==0)
      reader.release();
    mutexWriters.release();
  }
};
```

Priorité aux rédacteurs

```
class ReaderWriterPrioWriter :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders, mutexWriters;
  PcoSemaphore writer, reader;
  PcoSemaphore mutex;
  int nbReaders, nbWriters;

public:
  ReaderWriterPrioWriter() :
    mutexReaders(1), mutexWriters(1),
    writer(1), reader(1),
    mutex(1),
    nbReaders(0), nbWriters(0) {}

  void lockReading() {
    mutexReaders.acquire();









    mutexReaders.release();
  }
```

```
  void unlockReading() {









  }

  void lockWriting() {
    mutexWriters.acquire();
    nbWriters++;
    if (nbWriters==1)
      reader.acquire();
    mutexWriters.release();
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
    mutexWriters.acquire();
    nbWriters -= 1;
    if (nbWriters==0)
      reader.release();
    mutexWriters.release();
  }
};
```

Priorité aux rédacteurs

```
class ReaderWriterPrioWriter :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders, mutexWriters;
  PcoSemaphore writer, reader;
  PcoSemaphore mutex;
  int nbReaders, nbWriters;

public:
  ReaderWriterPrioWriter() :
    mutexReaders(1), mutexWriters(1),
    writer(1), reader(1),
    mutex(1),
    nbReaders(0), nbWriters(0) {}

  void lockReading() {
    mutexReaders.acquire();
    reader.acquire();




    reader.release();
    mutexReaders.release();
  }

  void unlockReading() {




  }

  void lockWriting() {
    mutexWriters.acquire();
    nbWriters++;
    if (nbWriters==1)
      reader.acquire();
    mutexWriters.release();
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
    mutexWriters.acquire();
    nbWriters -= 1;
    if (nbWriters==0)
      reader.release();
    mutexWriters.release();
  }
};
```

Priorité aux rédacteurs

```
class ReaderWriterPrioWriter :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders, mutexWriters;
  PcoSemaphore writer, reader;
  PcoSemaphore mutex;
  int nbReaders, nbWriters;

public:
  ReaderWriterPrioWriter() :
    mutexReaders(1), mutexWriters(1),
    writer(1), reader(1),
    mutex(1),
    nbReaders(0), nbWriters(0) {}

  void lockReading() {
    mutexReaders.acquire();
    reader.acquire();

    nbReaders++;
    if (nbReaders==1)
      writer.acquire();

    reader.release();
    mutexReaders.release();
  }

  void unlockReading() {

    nbReaders -= 1;
    if (nbReaders==0)
      writer.release();

  }

  void lockWriting() {
    mutexWriters.acquire();
    nbWriters++;
    if (nbWriters==1)
      reader.acquire();
    mutexWriters.release();
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
    mutexWriters.acquire();
    nbWriters -= 1;
    if (nbWriters==0)
      reader.release();
    mutexWriters.release();
  }
};
```

Priorité aux rédacteurs

```
class ReaderWriterPrioWriter :
  public AbstractReaderWriter {
protected:
  PcoSemaphore mutexReaders, mutexWriters;
  PcoSemaphore writer, reader;
  PcoSemaphore mutex;
  int nbReaders, nbWriters;

public:
  ReaderWriterPrioWriter() :
    mutexReaders(1), mutexWriters(1),
    writer(1), reader(1),
    mutex(1),
    nbReaders(0), nbWriters(0) {}

  void lockReading() {
    mutexReaders.acquire();
    reader.acquire();
    mutex.acquire();
    nbReaders++;
    if (nbReaders==1)
      writer.acquire();
    mutex.release();
    reader.release();
    mutexReaders.release();
  }

  void unlockReading() {
    mutex.acquire();
    nbReaders -= 1;
    if (nbReaders==0)
      writer.release();
    mutex.release();
  }

  void lockWriting() {
    mutexWriters.acquire();
    nbWriters++;
    if (nbWriters==1)
      reader.acquire();
    mutexWriters.release();
    writer.acquire();
  }

  void unlockWriting() {
    writer.release();
    mutexWriters.acquire();
    nbWriters -= 1;
    if (nbWriters==0)
      reader.release();
    mutexWriters.release();
  }
};
```

Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité aux rédacteurs - Exemple

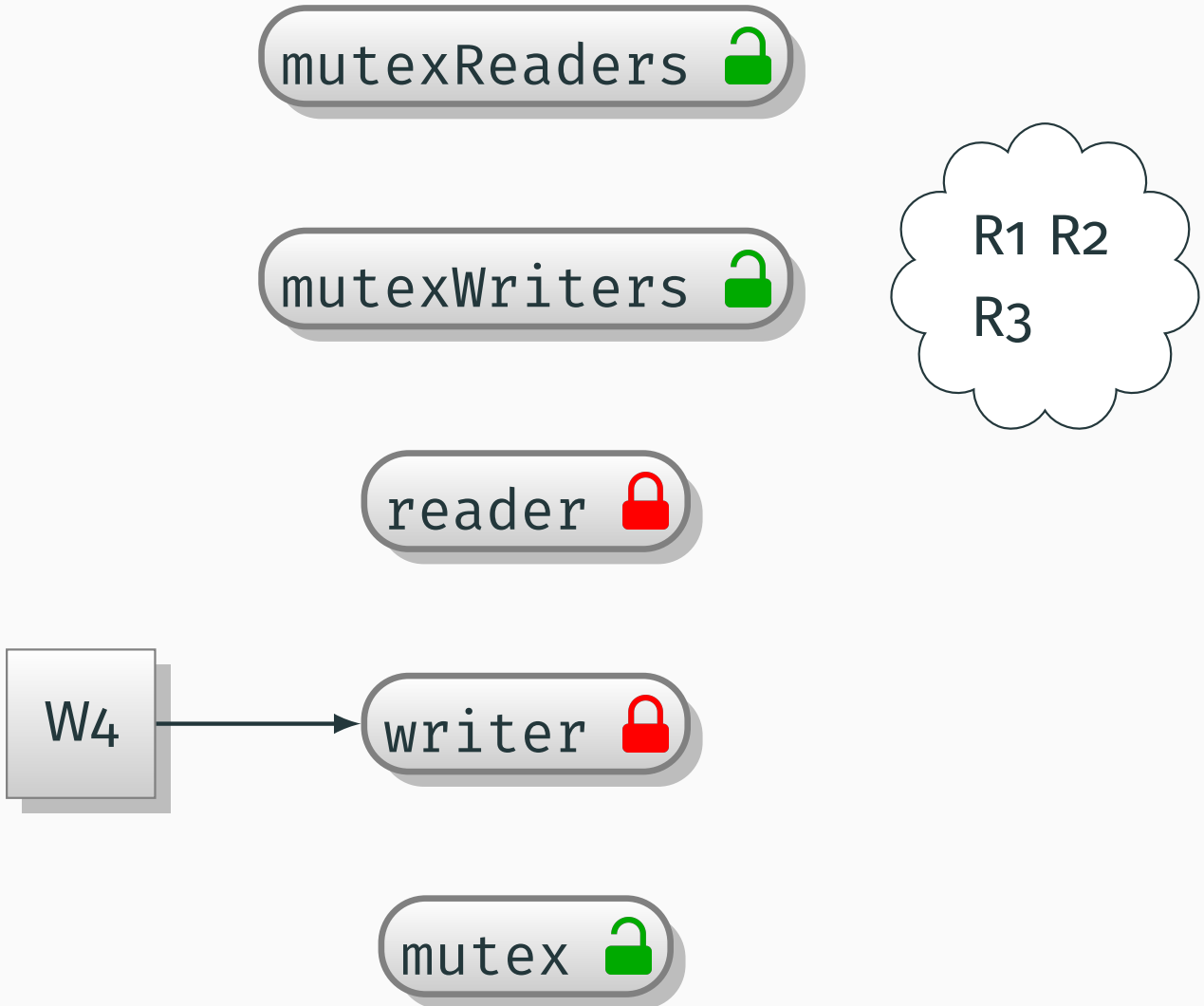
- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

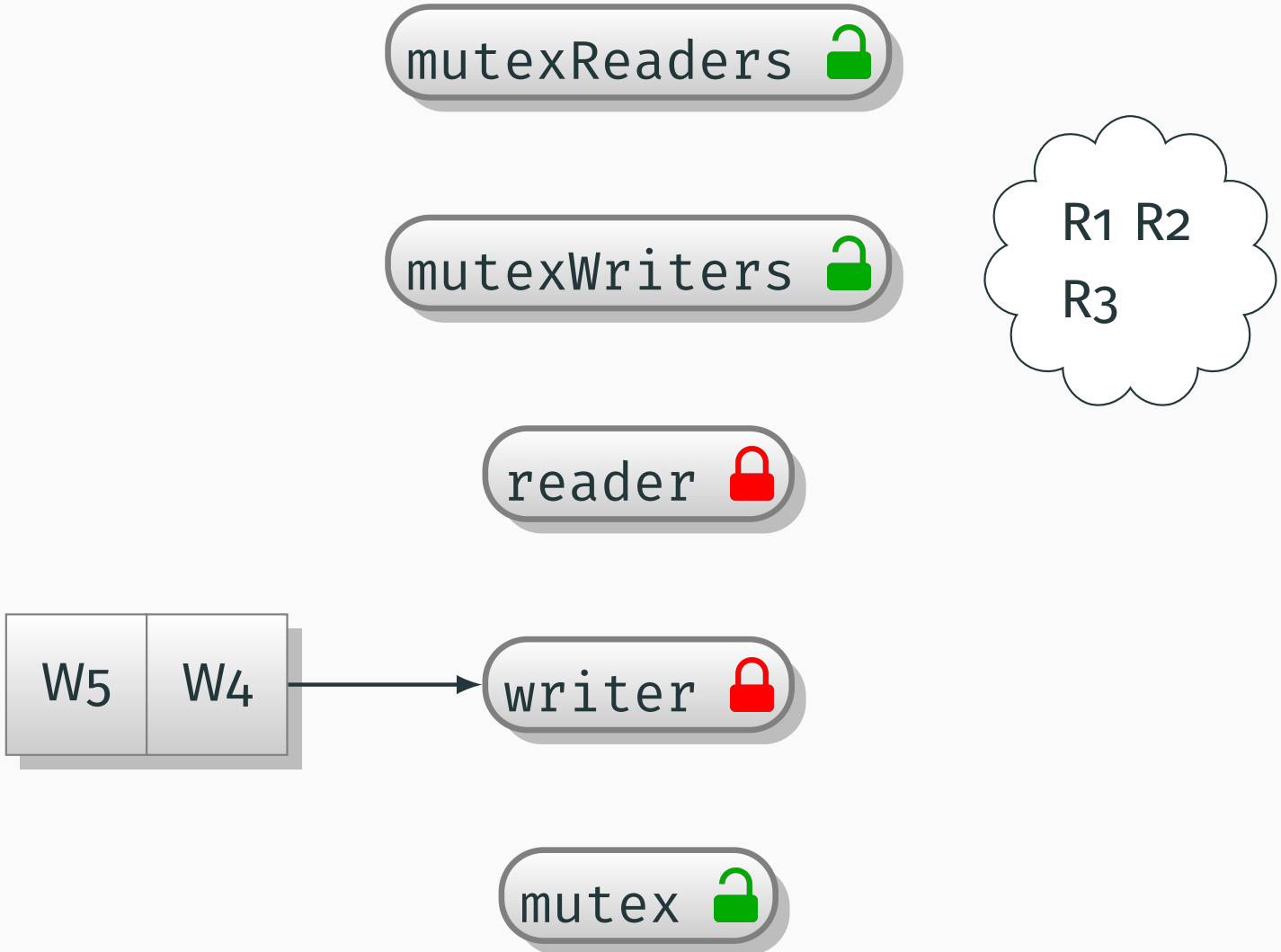
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



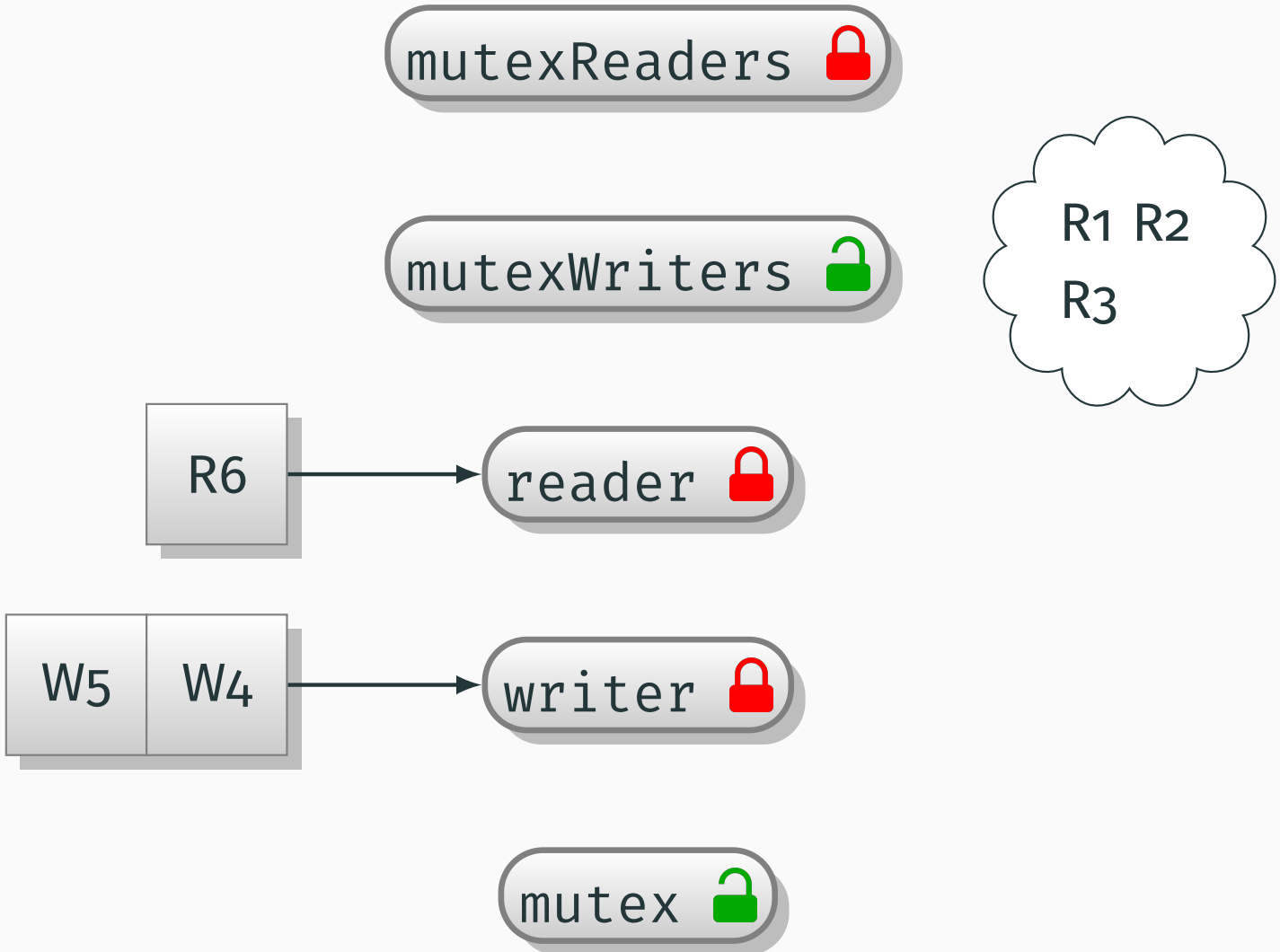
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



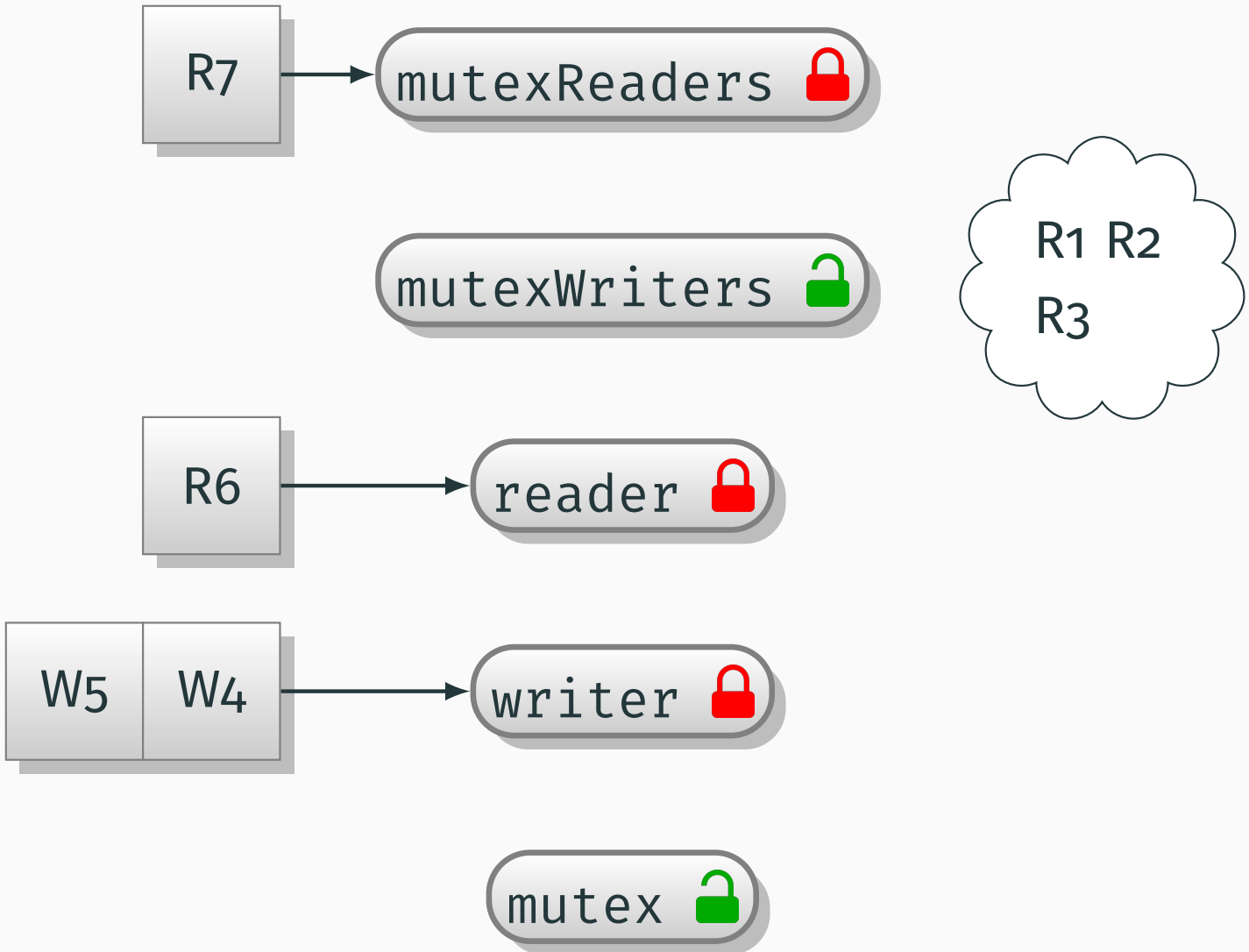
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$



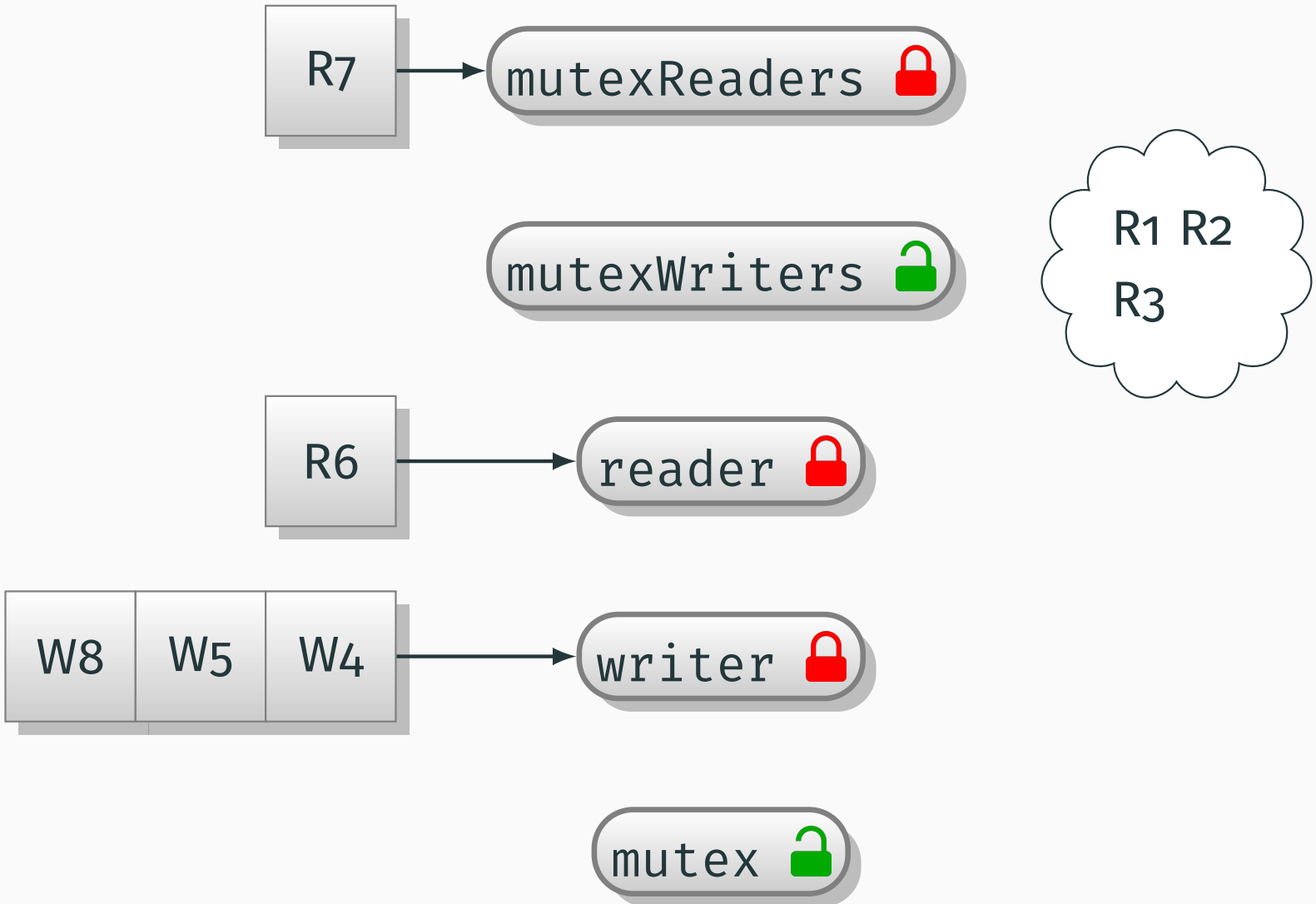
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*



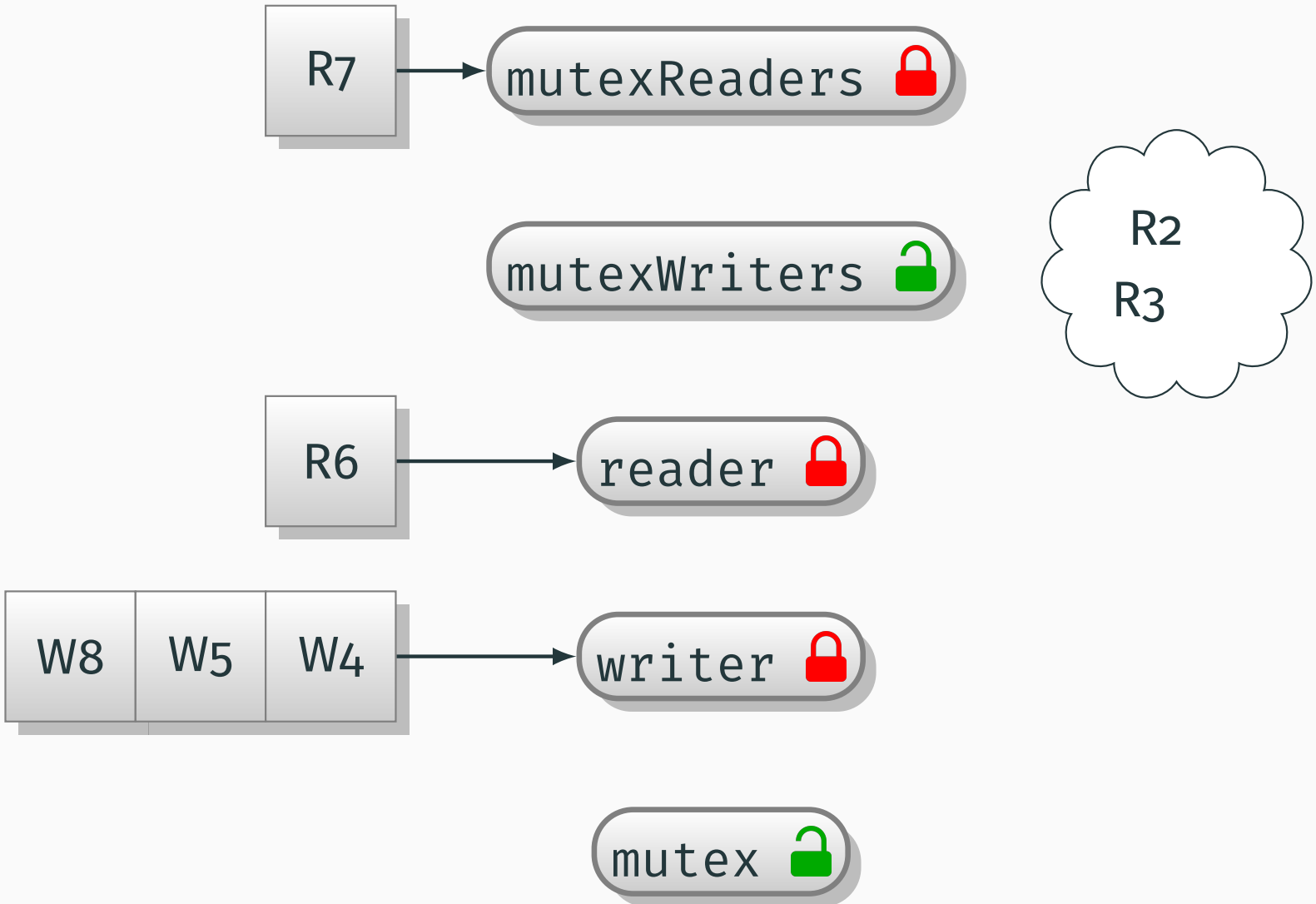
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*



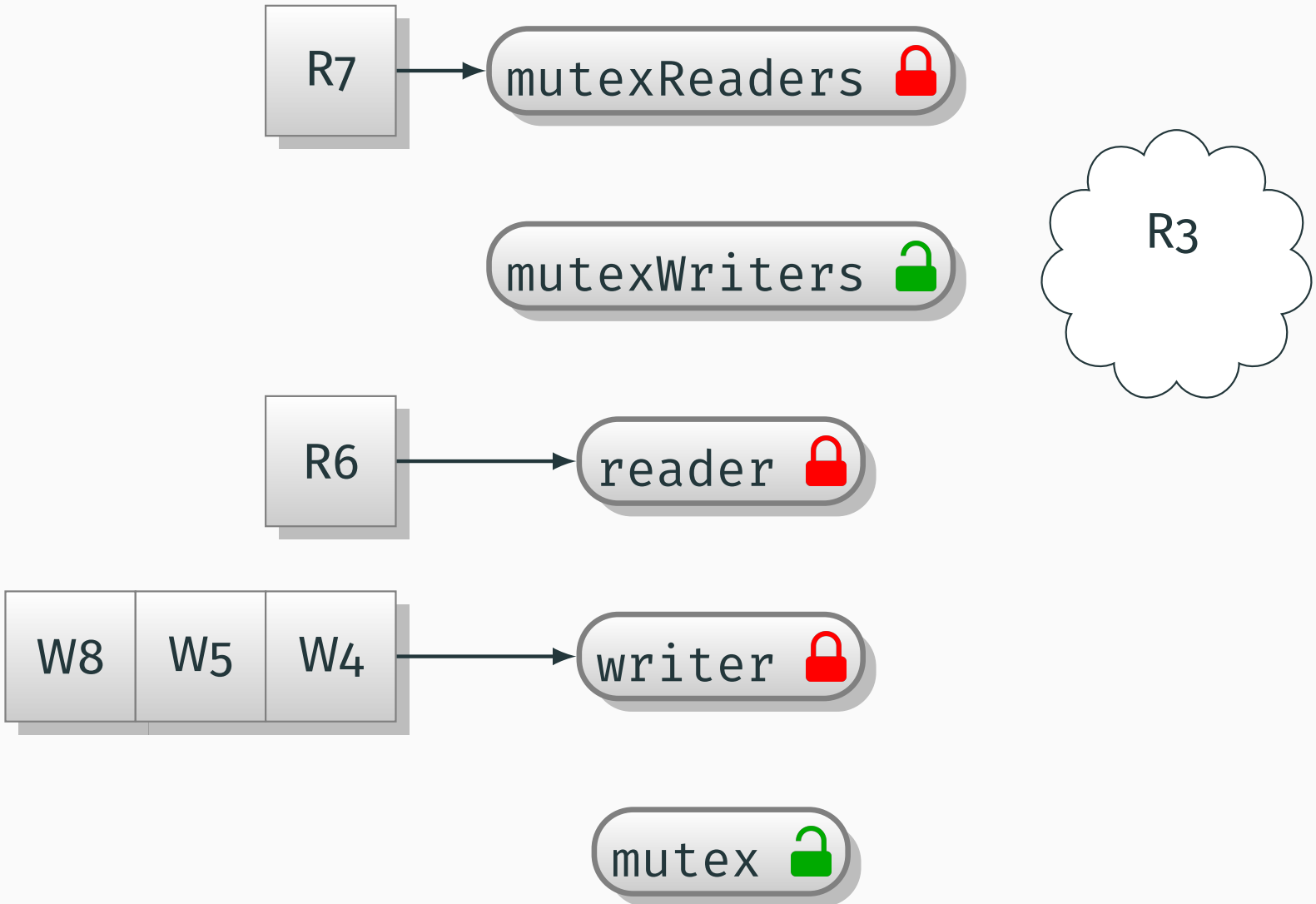
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$



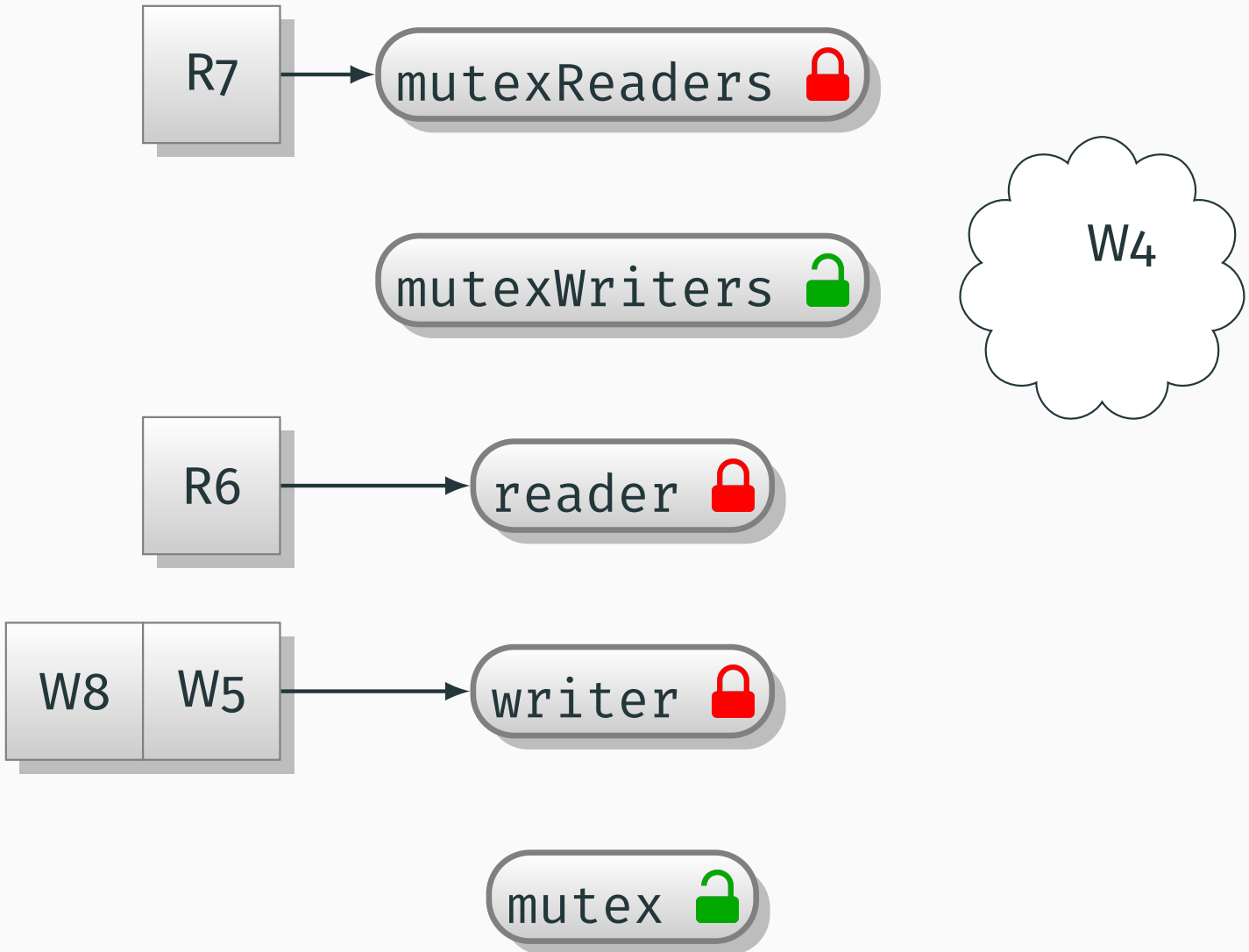
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$



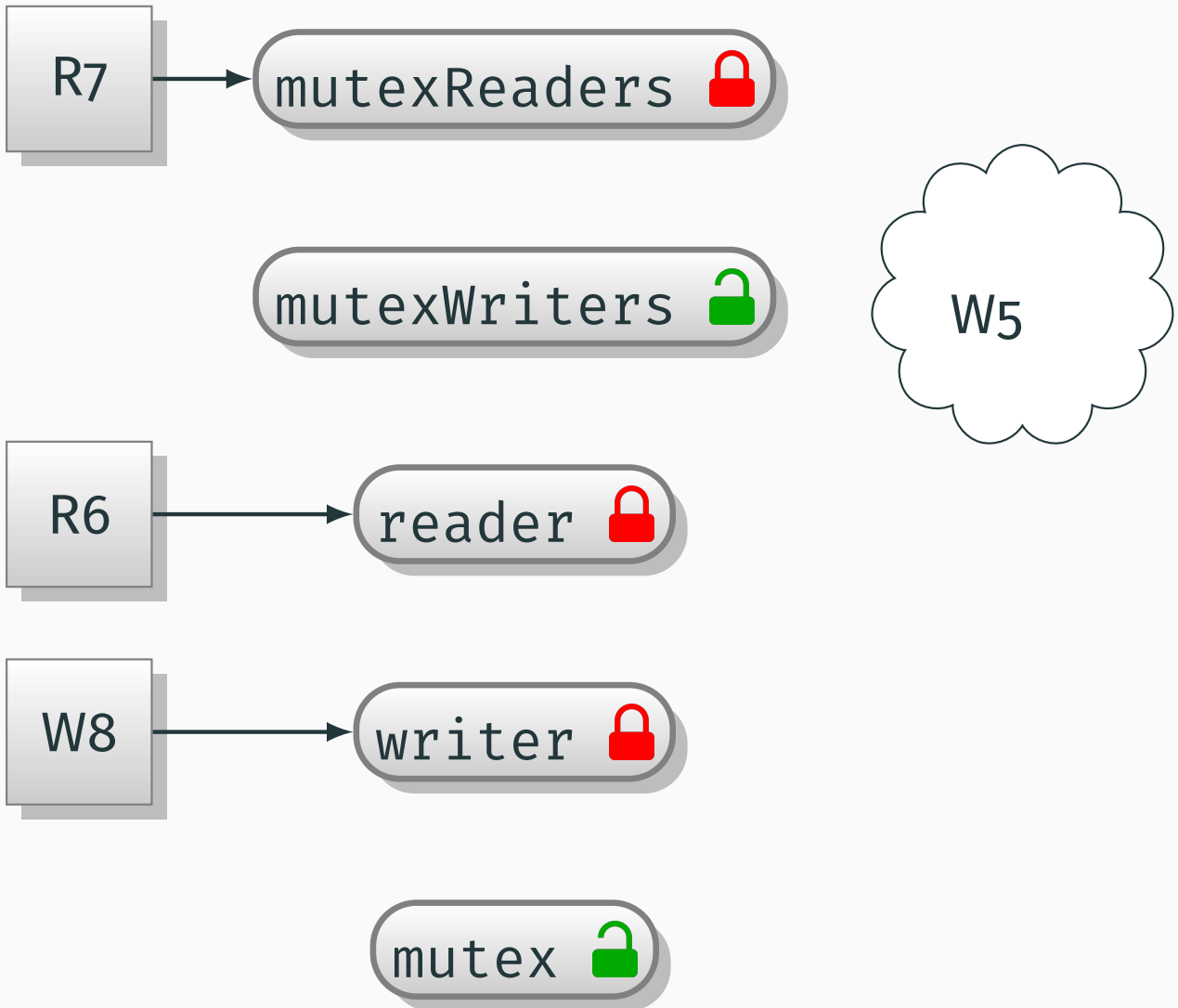
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de *R1*, *R2*, *R3*, *W4*, *W5*, *R6*, *R7*, *W8*



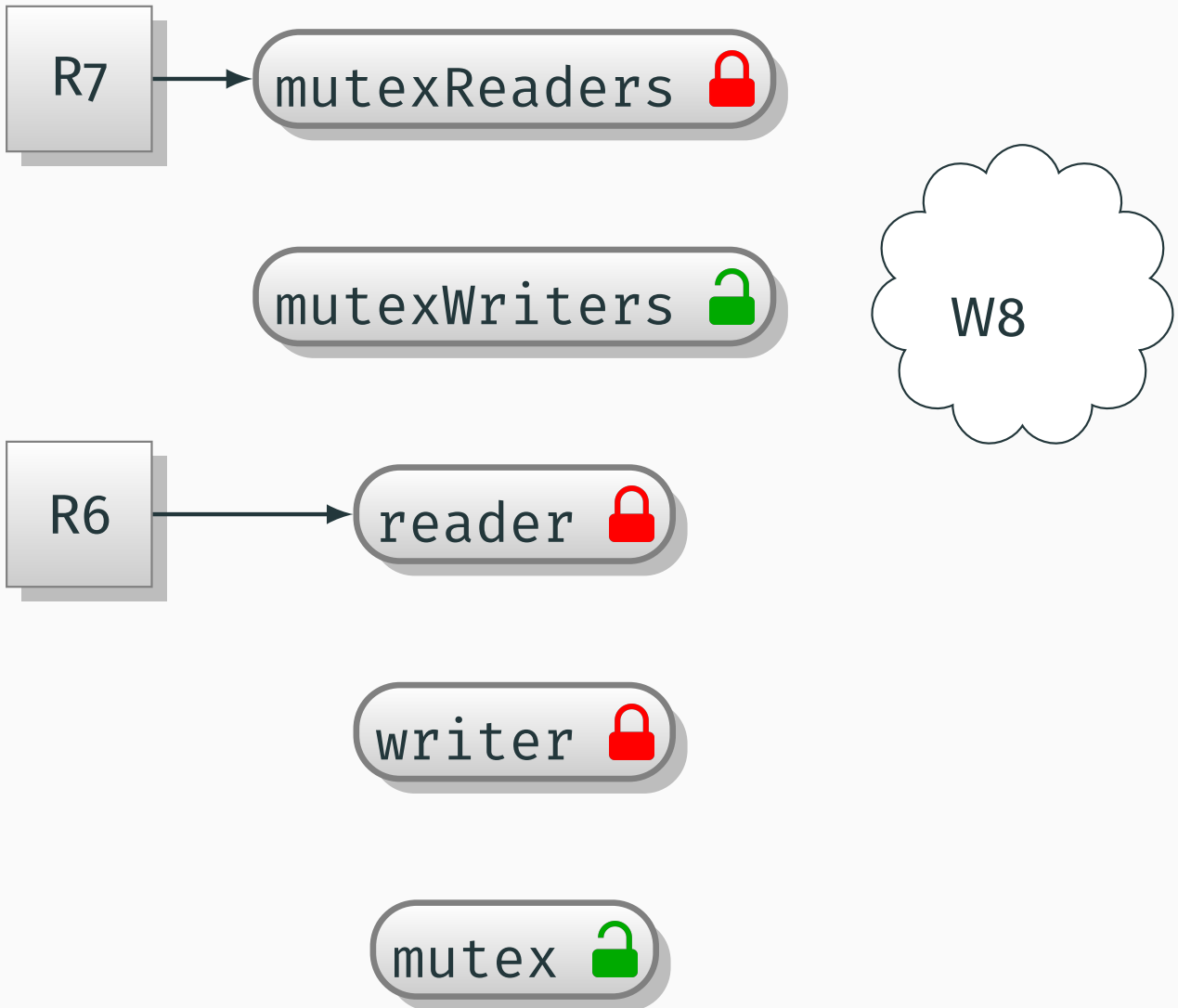
Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1$, $R2$, $R3$, $W4$, $W5$, $R6$, $R7$, $W8$



Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$



Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Priorité aux rédacteurs - Exemple

- Scénario:
  - Arrivée de $R1, R2, R3, W4, W5, R6, R7, W8$

Exercices

Exercices

- Reprendre l'algorithme général avec priorité aux lecteurs, et l'adapter pour la priorité aux rédacteurs.

Exercices

- Une application est composée de threads de deux classes, A et B. Une ressource est partagée entre tous les threads, selon les contraintes suivantes:
  1. Les threads de classe A peuvent accéder concurrement à la ressource.
  2. Les threads de classe B peuvent accéder concurrement à la ressource.
  3. Les threads de différente classe ne peuvent accéder à la ressource au même instant.

  Proposer un algorithme permettant de gérer l'accès à la ressource, en s'inspirant des solutions du chapitre. Considérez une solution où la coalition est possible entre threads d'une même classe.

Code source

```
run:../code/6-lectred/lectred1
```