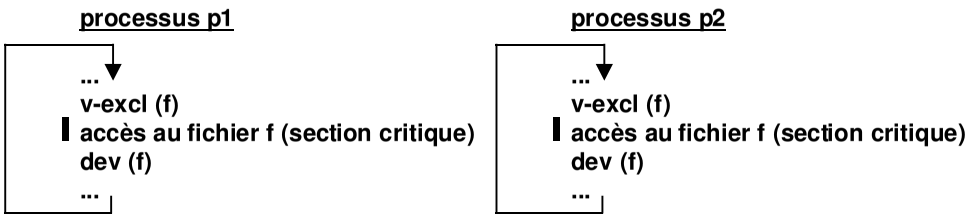
Opérations de verrouillage

- Les opérations de verrouillage sont une manière de réaliser l'exclusion mutuelle, pour une opération particulière (l'accès à un fichier)
- Deux opérations
  - v-excl (f) : verrouille le fichier f avec accès exclusif
  - dev (f) : déverrouille le fichier f
  - Remarque : ces noms sont symboliques, la réalisation en Unix est donnée plus loin
- Propriétés garanties
  - les opérations v-excl et dev sont atomiques (réalisées par appel système)
  - un fichier f verrouillé en accès exclusif par un processus ne peut pas être verrouillé par un autre processus
  - un processus qui tente de verrouiller un fichier déjà verrouillé en accès exclusif est bloqué (mis en attente du verrou)
  - l'opération de déverrouillage réveille un processus en attente du verrou (et un seul)

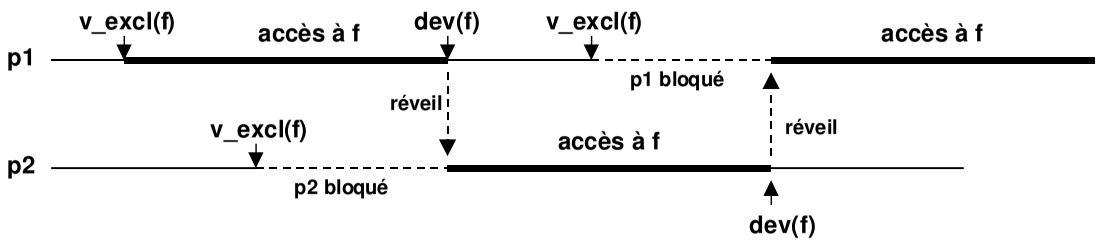
```
...                                    ...
v-excl (f)                             v-excl (f)
accès au fichier f (section critique)  accès au fichier f (section critique)
dev (f)                                dev (f)
...                                    ...
```


Verrouillage des fichiers (suite)

- Rappel des opérations de verrouillage
  - les processus p1 et p2 partagent un fichier f, dans lequel ils écrivent
  - chaque séquence d'accès à f est une section critique (l'accès à f est exclusif)



- Fonctionnement



Problèmes de réalisation du verrouillage

- **Les opérations v-excl et dev sont réalisées comme des opérations atomiques**
  - en particulier, si on exécute :

  - alors l'effet de l'exécution parallèle de A1 et A2 sera globalement : soit A1 ; A2, soit A2 ; A1, à l'exclusion de tout autre.
- **Cela est nécessaire au bon fonctionnement du mécanisme.**
  - Exercice : essayer de réaliser les opérations v-excl et dev, en utilisant un indicateur d'occupation
    - occ = 0 : l'accès est libre ; occ = 1 : le fichier est occupé, accès impossible ; initialement, occ = 0
    - Quels sont les problèmes si on n'assure pas l'exécution atomique ?
      - nous avons déja rencontré une situation analogue
- **L'atomicité de v-excl et dev est assurée par le système d'exploitation**
  - mécanisme interne aux "appels systèmes"

Verrouillage partagé

- **Le verrouillage exclusif n'est pas toujours nécessaire**
  - Supposons que p1 et p2 lisent le fichier f (sans le modifier), et que p3 écrive dans le fichier.
  - Alors on peut permettre l'accès simultané à f de p1 et p2 (mais non p1 et p3, ou p2 et p3)
- **Un nouveau mode de verrouillage : le verrouillage partagé : v-part**
  - un fichier f verrouillé en accès exclusif par un processus ne peut pas être verrouillé par un autre processus (en mode exclusif ou partagé)
  - un fichier f verrouillé en accès partagé par un processus ne peut pas être verrouillé par un autre processus en mode exclusif, mais peut être verrouillé en mode partagé
  - une opération de verrouillage bloque le processus qui l'exécute si l'une des règles ci-dessus s'applique
  - l'opération de déverrouillage réveille un processus en attente du verrou (et un seul)
- **Exercice (difficile !)**
  - En utilisant v-excl, v-part et dev, programmer l'accès d'un ensemble de processus à un fichier, sachant que certains processus ("lecteurs") ne font que lire le fichier, alors que d'autres ("rédacteurs") peuvent lire et modifier le fichier. Écrire le programme des lecteurs et celui des rédacteurs.
  - Indications : utiliser des compteurs pour compter le nombre de lecteurs actifs et le nombre de lecteurs en attente

Verrouillage de fichiers dans Unix

- **Opérations disponibles**
  - Deux primitives (appels système) sont utilisables
    - fcntl : fonction très générale, permet tout type de verrouillage
    - lockf : fonction plus restreinte, verrouillage exclusif
  - Voir man ; le verrouillage sera aussi expérimenté en TP
- **Caractéristiques**
  - On peut verrouiller un fichier entier ou seulement une partie (permet d'augmenter le parallélisme)
  - On peut aussi tester si le fichier est déjà verrouillé (opération non bloquante)

Utilisation simultanée de plusieurs fichiers (1)

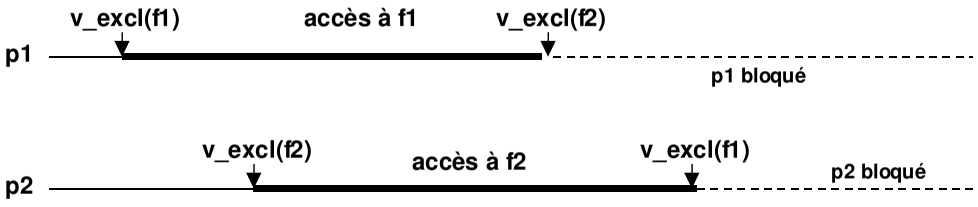
- **Situation**
  - Les processus p1 et p2 partagent deux fichiers f1 et f2, et les utilisent en accès exclusif

processus p1	processus p2
...	...
(1a) v-excl (f1)	(2a) v-excl (f2)
accès à f1	accès à f2
...	...
(1b) v-excl (f2)	(2b) v-excl (f1)
accès à f1 et f2	accès à f1 et f2
dev (f2)	dev (f1)
dev (f1)	dev (f2)

- **Déroulement**
  - p1 et p2 s'exécutent en parallèle (ou pseudo-parallèle)
  - Pour une exécution particulière, la séquence est : 1a ; 1b ; 2a ; 2b ; ...
  - Pour une autre exécution, la séquence est : 1a ; 2a ; 1b ; 2b ; ...
  - Que se passe-t-il dans chaque cas ?

Utilisation simultanée de plusieurs fichiers (2)

- **Exécution dans l'ordre 1a ; 2a ; 1b ; 2b ; ...**



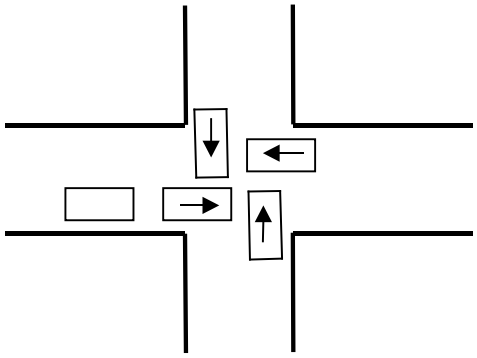
- **SItuation après le deuxième v_excl(f1) : les deux processus p1 et p2 sont bloqués, et le resteront indéfiniment :**
  - **pour réveiller p1, il faut faire dev(f2) qui ne peut être fait que par p2, bloqué**
  - **pour réveiller p2, il faut faire dev(f1) qui ne peut être fait que par p1, bloqué**
- **Cette situation s'appelle interblocage (en anglais *deadlock*)**

Interblocage : caractérisation

- **Définition de l'interblocage**
  - **Situation dans laquelle plusieurs processus (au moins 2) sont bloqués, chacun d'eux ne pouvant être réveillé que par une action de l'un des autres**
  - **On en peut pas sortir d'une situation d'interblocage sans intervention extérieure**
- **Conditions d'apparition**
  - **L'interblocage se produit lorsque plusieurs processus sont en compétition pour utiliser simultanément plusieurs ressources, et que les demandes sont mal coordonnées (attente circulaire)**
  - **un exemple hors informatique : carrefour**

Questions :
que sont ici les ressources ?

comment sortir de l'interblocage ?



Comment prévenir l'interblocage ?

- **Solution 1 : réservation globale**
  - Chaque processus demande globalement (en bloc) toutes les ressources dont il a besoin
  - Inconvénient : réduit les possibilités de parallélisme
  - Exemple : c'est la solution adoptée pour les carrefours (le parallélisme n'est pas l'exigence première)
- **Solution 2 : requêtes ordonnées**
  - L'interblocage est impossible si tous les processus demandent dans le même ordre les ressources dont ils ont besoin
  - Exemple

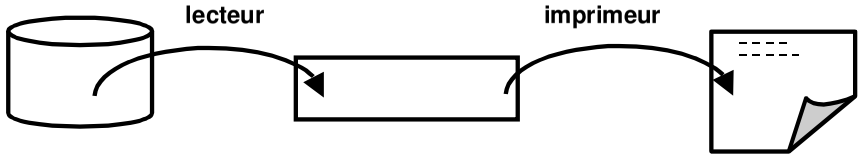
processus p1	processus p2
...	...
v-excl (f1)	v-excl (f1)
accès à f1	accès à f1
...	...
v-excl (f2)	v-excl (f2)
accès à f1 et f2	accès à f1 et f2
dev (f2)	dev (f1)
...	...
dev (f1)	dev (f2)

Lutter contre l'interblocage

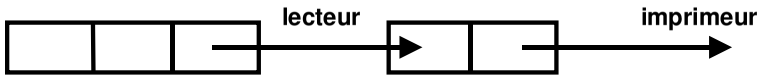
- **On ne peut pas sortir d'une situation d'interblocage sans perdre quelque chose**
- **Possibilités de "guérison"**
  - Faire revenir un ou plusieurs processus en arrière, dans un état antérieur (on perd le travail réalisé depuis cet état)
    - nécessite d'avoir conservé l'état antérieur
  - Tuer un ou plusieurs processus pour libérer les ressources
- **Conclusion**
  - Pour choisir entre prévention et guérison, il faut apprécier les coûts relatifs de l'une et de l'autre dans la situation particulière
    - coût de la prévention : application d'une politique systématique de réservation de ressources
    - coût de la guérison : détection de l'interblocage + pertes résultant du retour en arrière

Introduction à la coopération entre processus

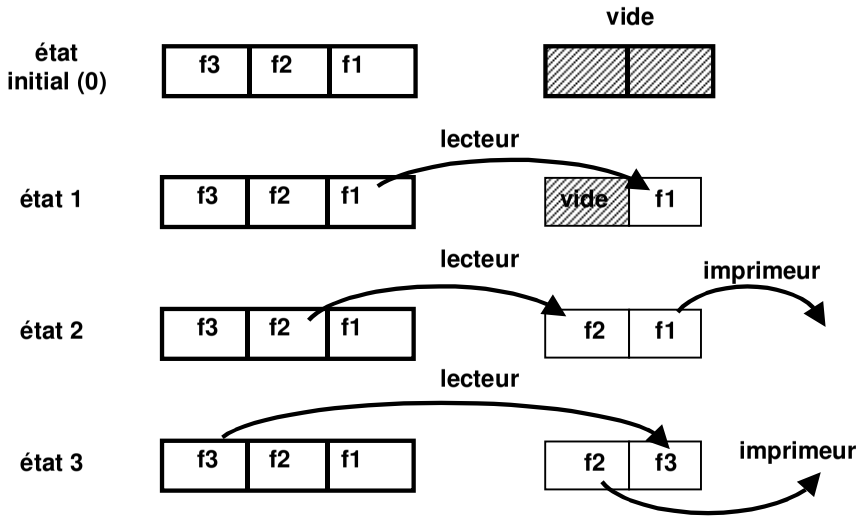
- **Jusqu'ici, on n'a vu que des situations de compétition entre processus**
  - pour l'utilisation du processeur
  - pour l'accès à un fichier
- **Un exemple de situation de coopération**
  - le processus *lecteur* lit un fichier depuis le disque vers la mémoire centrale
  - un processus *imprimeur* récupère le fichier dans la mémoire centrale et l'envoie à l'imprimante

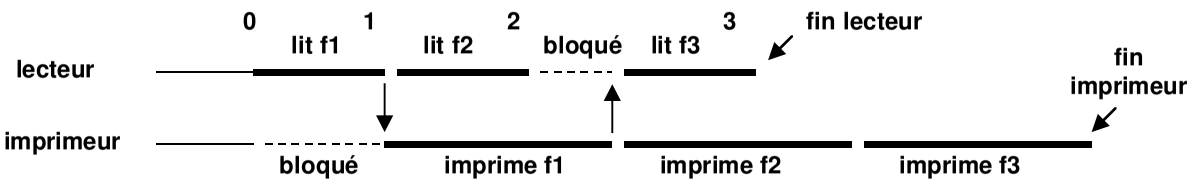


- **Contraintes**
  - la place en mémoire est restreinte …
  - … mais on veut exécuter lecteur et imprimeur en parallèle
  - solution : un "tampon" (ensemble de n "cases") ; ici n = 2, et le fichier a une taille de 3 cases



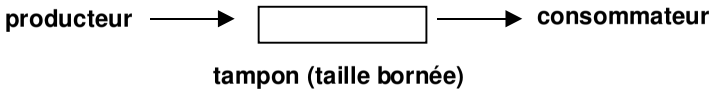
Coopération entre processus





Le schéma producteur - consommateur

- Le schéma lecteur -> tampon -> imprimeur est un exemple du schéma général de coopération dit “producteur - consommateur”



- **Utilité**
  - On cherche à augmenter le parallélisme entre producteur et consommateur, malgré une différence possible de leurs vitesses d’exécution
  - La taille du tampon est d’autant plus grande que la différence de vitesses est grande
- **Fonctionnement**
  - condition de blocage du consommateur : le tampon est vide
  - condition de blocage du producteur : le tampon est plein (on ne peut pas y déposer de l’information sans “écraser” de l’information encore non consommée)
  - en fait, le fonctionnement est symétrique (le consommateur est un producteur de cases vides)
- **Applications**
  - entrées-sorties tamponnées (*spool*)
  - traitements en pipe-line (cf plus loin les tubes d‘Unix (*pipes*)

Résumé de la séance 3

- **Réalisation de l’exclusion mutuelle par verrouillage**
  - accès à des informations en mode exclusif
  - accès à des informations en mode exclusif ou partagé
- **Les problèmes de l’exclusion mutuelle**
  - Problèmes de réalisation (atomicité)
  - Problèmes d’utilisation (interblocage)
- **Introduction à la coopération entre processus**
  - schéma producteur-consommateur