

---

# Systemes d'exploitation

## Mécanismes d'exécution et de communication

Sergiu Ivanov

`sergiu.ivanov@univ-evry.fr`

`https://www.ibisc.univ-evry.fr/~sivanov/fr/os-ueve.html`

# Objet du chapitre

Étude des mécanismes permettant

- ▶ le **partage des ressources** entre des tâches multiples
- ▶ **l'interaction** avec le monde extérieur

# Outline

## 1. Exécution séquentielle

Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

Exemples d'utilisation des interruptions

## 3. Programmation des entrées-sorties

Organisation générale

Modes de pilotage des périphériques

Entrées-sorties tamponnées en mémoire

Entrées-sorties tamponnées sur disque

# Outline

## 1. Exécution séquentielle

### Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

Exemples d'utilisation des interruptions

## 3. Programmation des entrées-sorties

Organisation générale

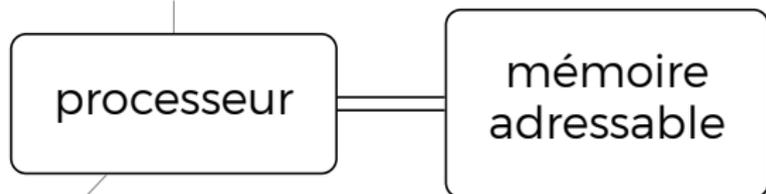
Modes de pilotage des périphériques

Entrées-sorties tamponnées en mémoire

Entrées-sorties tamponnées sur disque

# Machine abstraite

possède de la mémoire  
interne (registres)



interprète les  
instructions

contient les instruc-  
tions et les données

segment



n'est pas modifié au  
cours de l'exécution

# Procédure, activité, contexte

**Programme** = ensemble de procédures

Procédure  $\frac{n}{1}$  segment de procédure

L'exécution de la procédure peut être **interrompue**.

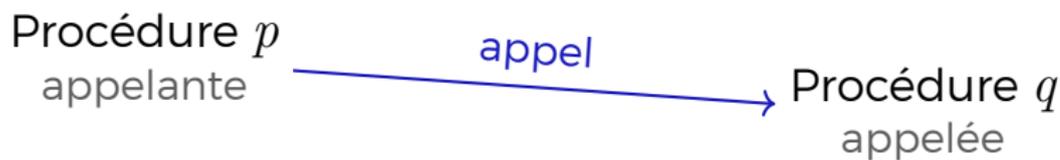
- ▶ multiprogrammation
- 

**Activité** = exécution interrompue de la procédure.

**Contexte** d'une activité = l'ensemble des **informations accessibles** au processeur

- ▶ processeur + mémoire

# Appel et retour de procédure



- ▶ préparation des paramètres transmis
- ▶ sauvegarde du contexte de  $p$ 
  - ▶ il sera retrouvé au retour
- ▶ remplacement du contexte de  $p$  par le contexte de  $q$



Appel normal : quasi-symétrique, le contexte de  $q$  se perd

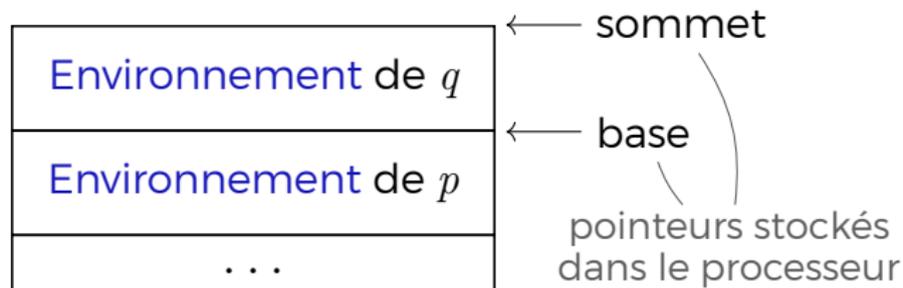
Appel en coroutines : séquence symétrique

# Pile d'exécution

**Pile** = structure de données utilisée pour les appels

---

Quand la procédure  $p$  appelle la procédure  $q$  :

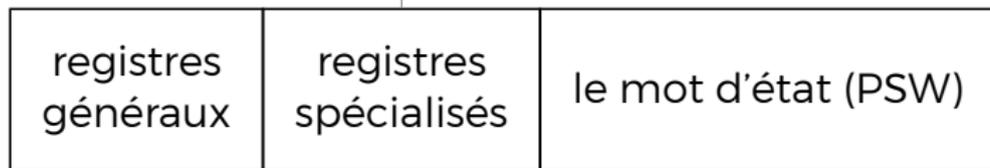


L'**environnement** de  $q$  contient :

- ▶ les liens de retour
  - ▶ base de  $p$ , adresse de retour, adresse pour le résultat
- ▶ les paramètres
- ▶ les variables locales de  $q$

# L'état du processeur

base, sommet, etc.



- ▶ l'état du processeur : actif/attente
- ▶ mode : maître/esclave
  - ▶ mode maître : instructions privilégiées autorisées
  - ▶ les entrées-sorties exigent le mode maître
- ▶ informations sur le contexte en mémoire
- ▶ informations sur le déroulement de l'activité
  - ▶ résultat de la dernière comparaison (code de condition)

# Outline

## 1. Exécution séquentielle

Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

Exemples d'utilisation des interruptions

## 3. Programmation des entrées-sorties

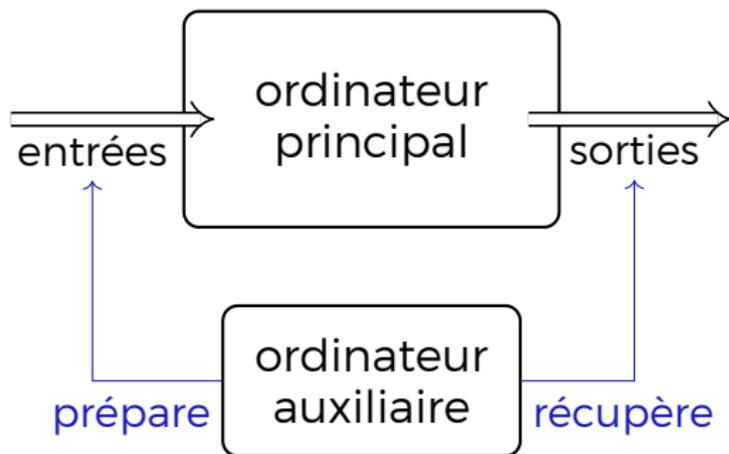
Organisation générale

Modes de pilotage des périphériques

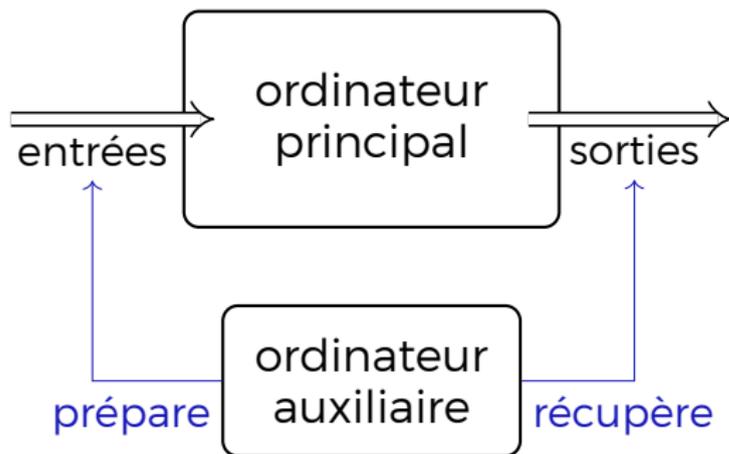
Entrées-sorties tamponnées en mémoire

Entrées-sorties tamponnées sur disque

Comment connaître la fin d'une entrée/sortie ?



Comment connaître la fin d'une entrée/sortie ?



Attente active ?

# Action monde processeur

- ▶ Notification d'une **entrée/sortie**
    - ▶ disponibilité des données
    - ▶ fin de sortie
  - ▶ Arrêt du programme à l'**expiration d'un temps** donné
  - ▶ Interruption de **sécurité**
- 

## Attente active : **coûteuse** et **imprécise**

- ▶ consultation périodique de variables spéciales

# Commutation de contexte

Exécuter de façon indivisible :

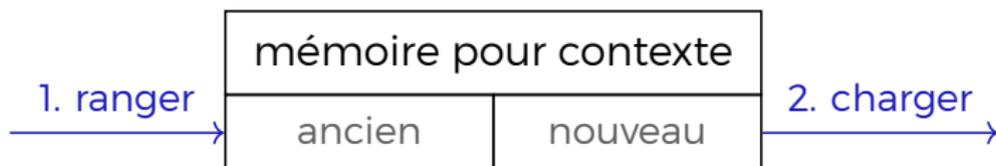
1. ranger le mot d'état  
dans un emplacement mémoire spécifique
2. charger un autre mot d'état  
à partir d'un emplacement mémoire spécifique

Possible quand le processeur est

- ▶ en attente
- ▶ en un point observable (entre deux instructions)
  - ▶ point interruptible = point observable

# Schémas de commutation de contexte

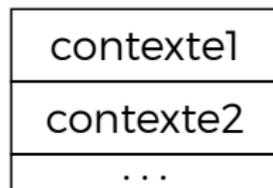
- ▶ Sauvegarde dans des emplacement fixes



- ▶ Sauvegarde sur une pile

1. ranger = empiler

2. charger = dépiler



# Mécanismes de commutation de contexte

## Interruption

Cause :       extérieure

Utilisation : réaction à un évènement **asynchrone externe**

## Déroutement

Cause :       liée à l'exécution de l'instruction en cours

Utilisation : traitement d'une **erreur/situation exceptionnelle**

## Appel au superviseur

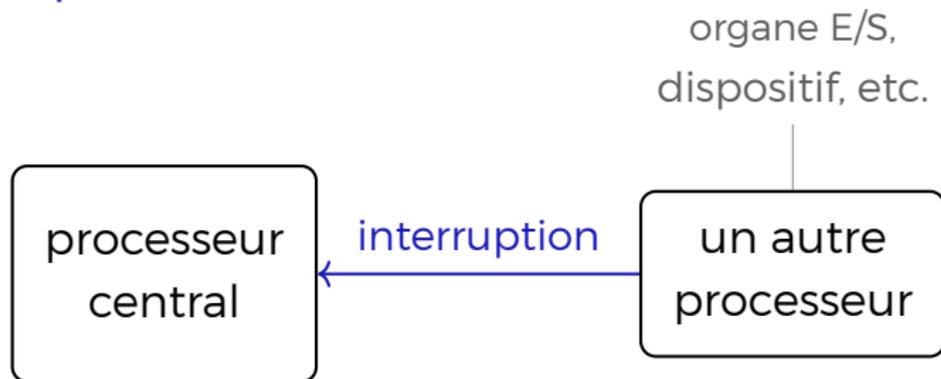
Cause :       **commutation explicite**

Utilisation : appel d'une fonction du **système d'exploitation**

---

Souvent le terme « interruption » est utilisé pour désigner tous ces mécanismes.

# Interruptions



- ▶ suspension du programme en cours
- ▶ exécution du **traitant d'interruption**
  - ▶ programme de traitement

Le **traitant** s'exécute dans un **contexte différent** de celui du programme interrompu.

# Niveaux et masquage d'interruptions

Niveau d'interruption = cause de l'interruption

Chaque niveau  $\rightarrow$  traitant d'interruption

Les niveaux peuvent être ordonnés par priorité.

---

Il est possible de masquer les interruptions d'après leur niveau/priorité pour

- ▶ protéger l'exécution d'un programme
  - ▶ exemple : le traitant lui-même
- ▶ retarder la commutation de contexte

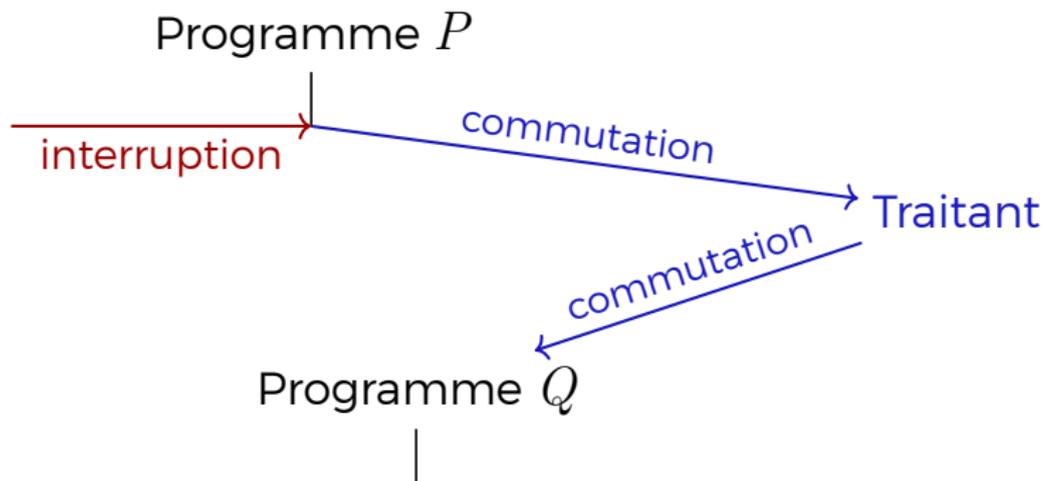
# (Dés)armement des interruptions

**Désarmement** = suppression complète d'un niveau

- ▶ comme si la cause était supprimée
- ▶ plus fort que le masquage

**(Ré)armement** = remise en service du niveau

# Schéma du traitement d'interruption



$Q$  n'est pas nécessairement le même que  $P$ .

# Déroutements

Réactions à des **anomalies** dans l'exécution.

- ▶ données incorrectes, tentative d'opération interdite, instruction non exécutable

Traitement **organisé par cause** du déroutement.

Un déroutement peut être **supprimé**, **mais non masqué**.

- ▶ aucun sens à retarder la cause d'un déroutement

# Appels au superviseur



La commutation de contexte explicite permet de

- ▶ vérifier les droits de l'utilisateur
- ▶ isoler les données du système d'exploitation
  - ▶ le programme est isolé du traitant

Exemples : Les entrées-sorties, les accès aux fichiers se font au travers des appels au superviseur.

# Outline

## 1. Exécution séquentielle

Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

Exemples d'utilisation des interruptions

## 3. Programmation des entrées-sorties

Organisation générale

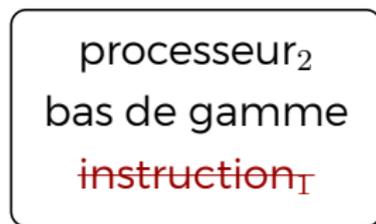
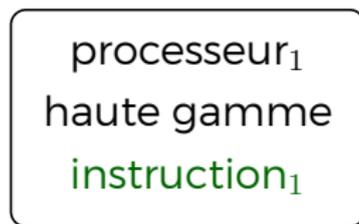
Modes de pilotage des périphériques

Entrées-sorties tamponnées en mémoire

Entrées-sorties tamponnées sur disque

# Simulation d'instructions manquantes

Situation:



J'appelle **instruction<sub>1</sub>** sur processeur<sub>2</sub>.

**Déroutement** : Le traitant simule l'effet de l'**instruction<sub>1</sub>**.

- ▶ exécuté en **mode esclave**
- ▶ permet de traiter les **erreurs**

---

Exemples : instructions d'arithmétique en virgule flottante, instructions de traitement massif de données, etc.

# Mesure de la taille d'une mémoire

Comment mesurer la taille de la mémoire vive ?

# Mesure de la taille d'une mémoire

Comment mesurer la taille de la mémoire vive ?

# Mesure de la taille d'une mémoire

Comment mesurer la taille de la mémoire vive ?

```
i := 0
cycle
  lire la case i
  i := i + 1
fcycle
```

À la fin du cycle,  $i$  = taille de la mémoire

# Outline

## 1. Exécution séquentielle

Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

**Exemples d'utilisation des interruptions**

## 3. Programmation des entrées-sorties

Organisation générale

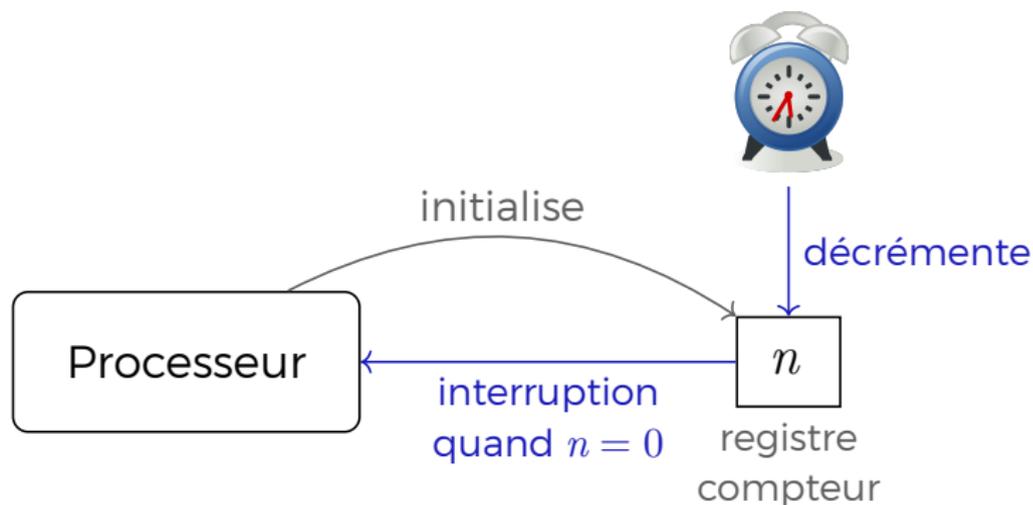
Modes de pilotage des périphériques

Entrées-sorties tamponnées en mémoire

Entrées-sorties tamponnées sur disque

# Mesure du temps

Mécanisme du **réveil** : initialiser le registre compteur au temps d'attente souhaité.



## Délai de garde      timeout

**Arrêter** l'exécution d'une procédure si elle n'a pas terminé après un **délai** fixé préalablement.

## Prélèvement de mesures

Schéma du traitement d'interruption d'horloge :

1. **sauver** le contexte actuel  $C$
2. **prélever** les mesures
3. **réinitialiser** l'horloge au temps d'attente souhaité
4. **restaurer** le contexte  $C$

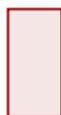
# Gestion de travaux en temps partagé

Nombre de tâches  $>$  nombre de processeurs



- ▶ laisser tourner tâche<sub>*i*</sub> le temps *t*
- ▶ suspendre tâche<sub>*i*</sub>
- ▶ passer à tâche<sub>*i+1*</sub>

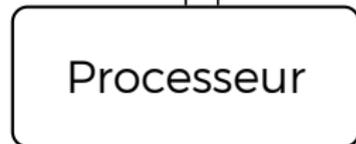
tâche<sub>1</sub>



tâche<sub>2</sub>



tâche<sub>3</sub>



# Outline

## 1. Exécution séquentielle

Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

Exemples d'utilisation des interruptions

## 3. Programmation des entrées-sorties

Organisation générale

Modes de pilotage des périphériques

Entrées-sorties tamponnées en mémoire

Entrées-sorties tamponnées sur disque

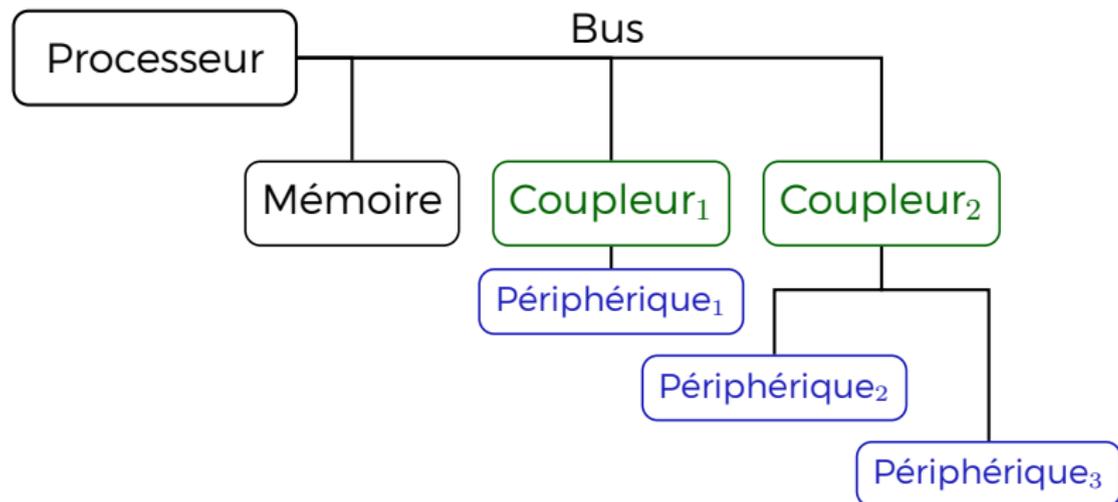
Qu'est-ce qu'un organe d'E/S ?

Qu'est-ce qu'un organe d'E/S ?

## Qu'est-ce qu'un organe d'E/S ?

Organe d'E/S = un dispositif capable de transférer l'information entre le processeur ou la mémoire et un support d'information externe

# Organisation générale



**Coupleur/contrôleur** = dispositif de commande adapté à un certain type de périphérique

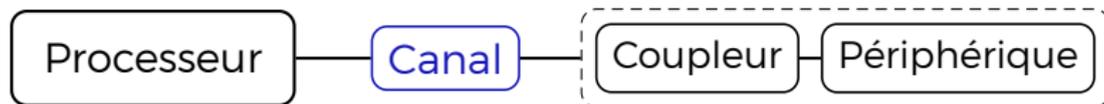
Le **coupleur** **décharge** le processeur central.

**Coupleur** — **Périphérique** souvent vus comme une seule unité

# Canaux/ADM

Canal = processeur spécialisé dans les opérations d'entrée-sortie

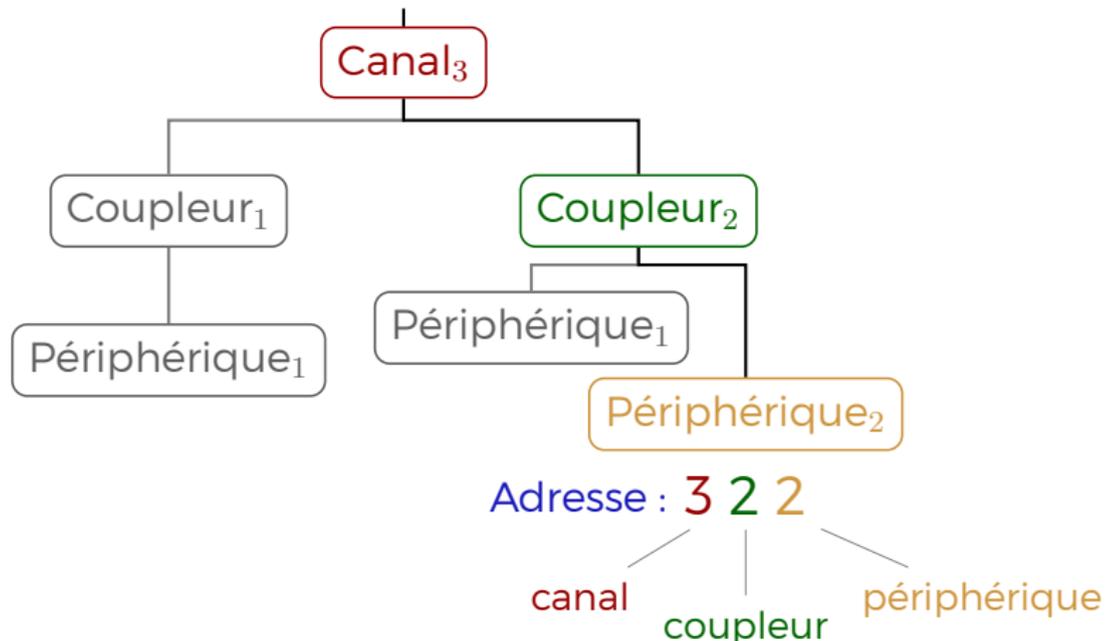
- ▶ lancé par le processeur central
- ▶ n'a pas d'interruptions
- ▶ peut interrompre le processeur central



Unité ADM = canal simplifié

- ▶ unité d'accès direct à la mémoire

# Adressage des périphériques



périphérique  $\frac{1}{n}$  voies d'accès  
⇓  
périphérique  $\frac{1}{n}$  adresses

# Outline

## 1. Exécution séquentielle

Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

Exemples d'utilisation des interruptions

## 3. Programmation des entrées-sorties

Organisation générale

Modes de pilotage des périphériques

Entrées-sorties tamponnées en mémoire

Entrées-sorties tamponnées sur disque

# Pilotes

Pilote = programme qui commande le fonctionnement élémentaire d'un périphérique :

- ▶ communique avec le coupleur
- ▶ traite les interruptions
- ▶ traite les erreurs

Les utilisateurs accèdent aux pilotes à travers les appels au superviseur.

# Entrées-sorties synchrones

Le processeur central est mobilisé pendant le transfert.

- ▶ **attente active** de la fin

pour  $i$  jusqu'à  $taille - 1$

sortir  $données[i]$

tant que  $\neg(\text{transfer fini})$  faire

ftq

fpour

L'E/S synchrones sont utilisées quand :

- ▶ le processeur est très simple
- ▶ le processeur ne peut pas être utilement employé
  - ▶ chargement initial du système
  - ▶ enregistrement de mesures

# Entrées-sorties asynchrones

Le processeur central exécute une **activité en parallèle** pendant le transfert.

- ▶ fin du transfert signalée par **une interruption**

[une autre activité]

## Traitant d'interruption

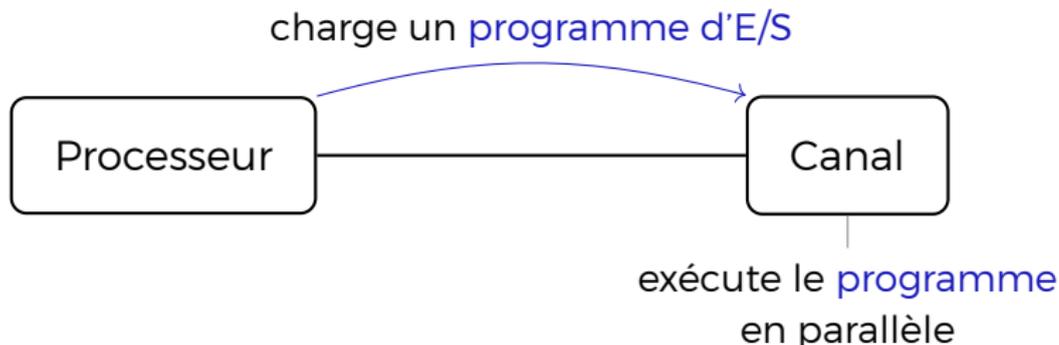
- ▶ **i** initialisé au **0** au début

```
si i < taille alors  
    sortir données[i]  
fsi  
i := i + 1
```

Logique du traitant :

- ▶ **sortir** lance le transfert
- ▶ **interruption** à la fin du transfert

# Entrées-sorties par canal/ADM



Programme d'E/S = suite de commandes d'E/S

Commande d'E/S =

nature de l'opération	adresse mémoire	taille données
--------------------------	--------------------	-------------------

---

**ADM** : Le programme contient une seule commande.

# Outline

## 1. Exécution séquentielle

Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

Exemples d'utilisation des interruptions

## 3. Programmation des entrées-sorties

Organisation générale

Modes de pilotage des périphériques

**Entrées-sorties tamponnées en mémoire**

Entrées-sorties tamponnées sur disque

# Entrées-sorties tamponées

Comment palier les différences de vitesse ?

Processeur

vitesse élevée

??

Périphérique

vitesse réduite

# Entrées-sorties tamponées

Comment palier les différences de vitesse ?

Processeur

vitesse élevée

??

Périphérique

vitesse réduite

# Entrées-sorties tamponées

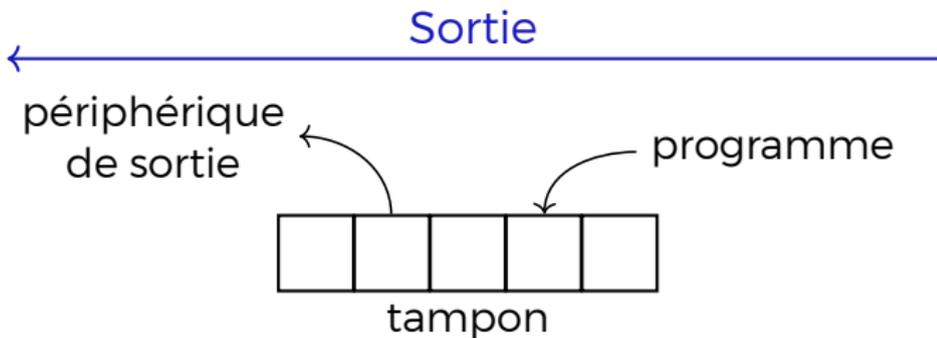
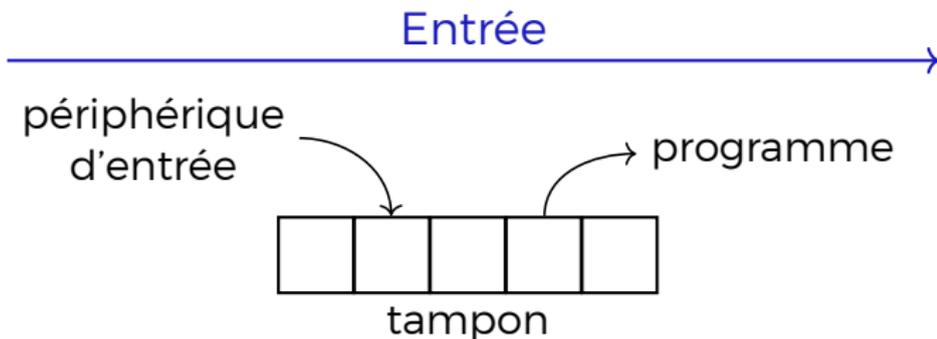
Comment palier les différences de **vitesse** ?



**Tampon** = zone mémoire réservée aux E/S

- ▶  $N$  enregistrements de taille fixe
- ▶ l'ordre de lecture/écriture préservé
- ▶ aucun enregistrement perdu
- ▶ réduction de l'attente inutile

# Tampons

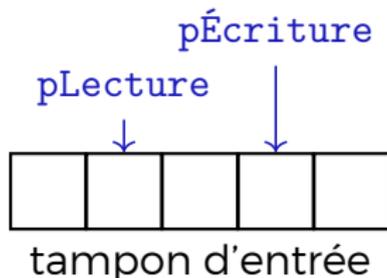


# Tampons : remarques pour la réalisation

## File circulaire

Le programme lit et avance le pointeur `pLecture`.

Le périphérique écrit et avance le pointeur `pÉcriture`.



---

## Interruptions

Le gestionnaire du tampon **doit masquer** les interruptions.

- ▶ sinon corruption des données

# Outline

## 1. Exécution séquentielle

Notions de base

Interruptions, déroutements, appels au superviseur

## 2. Mise en œuvre de la commutation

Utilisation des déroutements et appels au superviseur

Exemples d'utilisation des interruptions

## 3. Programmation des entrées-sorties

Organisation générale

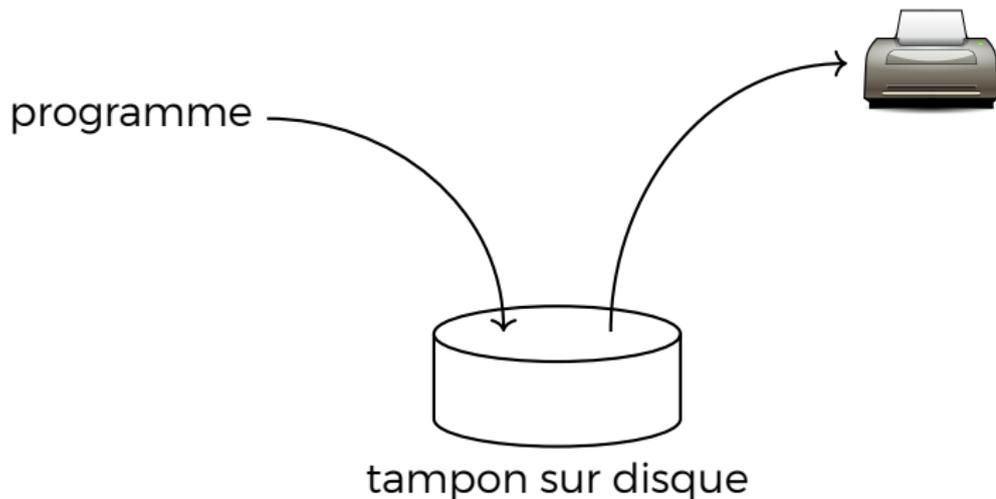
Modes de pilotage des périphériques

Entrées-sorties tamponnées en mémoire

Entrées-sorties tamponnées sur disque

Pour les E/S lentes, mettre le tampon sur le disque dur.

- ▶ imprimante, scanner, etc.



Pour les E/S lentes, mettre le tampon sur le disque dur.

- ▶ imprimante, scanner, etc.

