Les processus

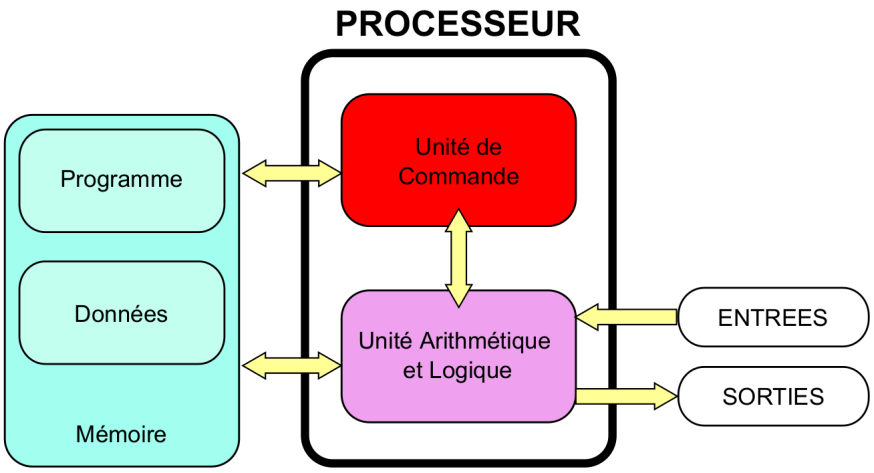
Capacités attendues

Il s'agit dans ce chapitre d'appréhender la gestion des processus et des ressources par le système d'exploitation.

- Décrire la création d'un processus, l'ordonnancement de plusieurs processus par le système ;
- Mettre en évidence le risque d'interblocage (*deadlock*) ;

1 Rappels sur l'architecture des ordinateurs : le modèle de Von Neumann

L'ordinateur moderne (de 1943 à nos jours) a très peu évolué d'un point de vue structurel. Depuis l'ENIAC (1943), premier ordinateur électronique « moderne » (on parle de « machine de Turing »), la même structure est utilisée par toute machine. Il s'agit de l'architecture de Von Neumann, nommée après son créateur, John Von Neumann, mathématicien hongrois du XXe siècle.



L'architecture de Von Neumann contient les éléments suivants :

- Un **processeur**, chargé de traiter l'information et d'organiser l'exécution des instructions.
  Celui-ci est découpé en deux parties :
  - L'**unité de commande**, contenant un ordonnanceur déterminant dans quel ordre les instructions et programmes sont exécutés ;
  - L'**unité arithmétique et logique (UAL)**, chargée de calculer à travers des opérations mathématiques de base (addition, multiplication) ainsi que l'algèbre de Boole (instructions logiques ET, OU, NON...) des résultats.
- Une **mémoire**, chargée de retenir l'information dans le temps pour stocker les résultats calculés par le processeur et les données nécessaires aux calculs. On décompose cette mémoire en deux parties :

  - La **mémoire vive (RAM)**, volatile (les données sont perdues à l'extinction de l'ordinateur), qui contient les informations pertinentes sur le court terme (données générées par l'exécution d'un programme, par exemple)
  - La **mémoire de masse**, non volatile (les données sont conservées même lorsque l'ordinateur n'a plus de courant) et qui contient des données ayant vocation à persister (fichiers, programmes...)
- Des **périphériques externes** :
  - Les **périphériques d'entrée**, qui permettent de récupérer des informations de l'utilisateur (clavier, souris, microphone...)
  - Les **périphériques de sortie**, qui permettent de transmettre des informations à l'utilisateur (écran, haut-parleurs, imprimante...)
- Un moyen de communication entre tous ces éléments, généralement sous la forme de fils appelés des **bus** (représentés sous la forme de flèches sur le schéma). En effet, effectuer un calcul est compliqué si on ne possède pas les données nécessaires à ce calcul, situées dans la mémoire. Avoir des périphériques d'entrée et de sortie n'est pas très utile s'ils ne sont pas reliés à la machine.

Un processeur est composé de deux grandes parties principales et complémentaires : l'unité arithmétique et logique (UAL), et l'ordonnanceur. Le processeur est le chef d'orchestre qui coordonne l'ensemble des composants entre eux afin d'assurer le bon fonctionnement de l'ordinateur. Le langage de programmation pouvant être compris par le processeur s'appelle l'**assembleur**.

2 Rappels sur les systèmes d'exploitation (OS pour *Operating System*)

Un système d'exploitation, souvent appelé OS (Operating System) est un programme ou un ensemble de programmes dont le but est de gérer les ressources matérielles (RAM, CPU, disques, périphériques d'entrée/sortie) et logicielles d'un ordinateur.

Aujourd'hui, les principaux systèmes d'exploitation sont :

- Windows (Microsoft)
- MacOS (Apple)
- Linux (logiciel *open source* n'appartenant pas à une marque)

> **Histoire**
> Pour plus de détails sur l'histoire des systèmes d'exploitation, voir :
> `https://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27exploitation#Histoire`

L'**interface système** est un programme permettant à l'utilisateur d'interagir avec le système d'exploitation. C'est une forme simple d'interface graphique. Cette interface peut porter différents noms selon le contexte : **shell**, **terminal**, **émulateur de terminal**, **invite de commande**, **console**, **bash**, etc.

2.1 Commandes de base sous Unix

Dans un système Unix (qui peut donc être une linux Ubuntu ou un MacOS récent), on dispose d'une **arborescence de fichiers** dont la racine est notée `/`. Une des premières nécessités d'un utilisateur est de pouvoir se déplacer dans ce système de fichiers pour accéder aux fichiers et à leurs contenus. Voici une liste des commandes qui permettent de le faire :

- `pwd` : connaître le répertoire courant dans l'arborescence de fichiers ;
- `mkdir <nom_dossier>` : créer un répertoire/dossier;

- `cd <nom_dossier>` : changer de répertoire/dossier;
- `ls <optionnel : nom_dossier>` : lister le contenu d'un répertoire/dossier dont on donne le chemin (répertoire courant par défaut);
- `cat <nom_fichier>` : visualiser le contenu d'un fichier ;
- `ps` : afficher la liste des processus en cours d'exécution.

```
raghu@raghu-Linoxide ~/Documents/linoxide $ tree
.
|-- dir1
|   |-- a
|   |   |-- file1.txt
|   |   `-- file2.txt
|   |-- b
|   |   `-- file3.txt
|   `-- c
|       `-- file4.txt
`-- dir2
    |-- a
    |   |-- file5.txt
    |   `-- file6.txt
    |-- b
    |   |-- file7.txt
    |   `-- file8.txt
    `-- c
        |-- file10.txt
        `-- file9.txt

8 directories, 10 files
raghu@raghu-Linoxide ~/Documents/linoxide $
```

Exemple d'arborescence de fichiers en mode texte

> *Un jeu en ligne pour s'entraîner* : `http://luffah.xyz/bidules/Terminus/`

> **TP**
> C'est le moment de faire le TP d'initiation à l'utilisation d'un terminal de type Unix.
> Télécharger puis à compléter : `https://holzermarie.github.io/data/term/bash-tp.pdf`

3 Notion de processus

Lors du lancement d'un programme, logiciel ou service, le système d'exploitation charge les instructions correspondantes depuis la mémoire de masse vers la mémoire vive, et crée un nouveau processus.

C'est le processeur qui s'occupe de l'exécution des processus.

> **Définition**
> Un processus représente un programme **en cours d'exécution**.
> Il est important de distinguer un processus du programme en lui-même : le programme est la donnée des instructions que le processeur doit exécuter. Le processus peut être défini quant à lui par :
> - le jeu d'instructions à exécuter (le programme) ;
> - l'état d'exécution de ce programme.

Lorsqu'on utilise un ordinateur, on peut lancer plusieurs programmes à la fois : un jeu, un navigateur, un logiciel de traitement de texte, etc. On a donc l'impression que l'ordinateur peut faire de nombreuses choses à la fois mais c'est juste une illusion. Le processeur ne peut faire qu'une chose à la fois . Seul un système multi-processeurs peut effectuer un travail réellement en parallèle, ce qui n'empêche pas que chacun des processeurs ne fait qu'une chose à la fois.

Il faut donc qu'il traite alternativement les programmes à exécuter (les processus). Cette tâche s'appelle l' ordonnancement .

3.1 L'ordonnancement

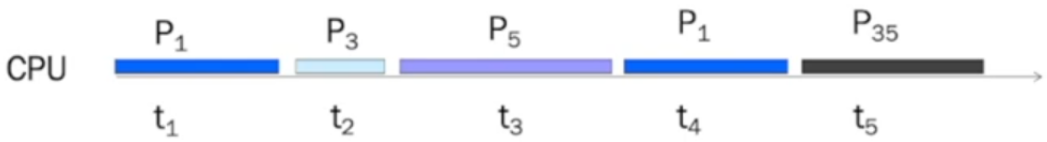
> **Définition**
> L'ordonnancement est la manière dont le système d'exploitation choisit l' ordre d'exécution des processus.

Le composant du système qui régit l'ordonnancement s'appelle l'ordonnanceur (*scheduler*) .

Parmi les **algorithmes d'ordonnancement** les plus simples, on peut citer:

- *Shortest Job First* (monotâche) : le prochain processus à exécuter est le plus court.
- *Earliest Deadline First* (monotâche) : le prochain processus à exécuter est celui qui se terminera en premier.
- *Round Robin* (multitâche) : les processus sont exécutés tour à tour durant **un *quantum* de temps dépendant de l'horloge du processeur**.
- *Round Robin with Priority Scheduling* (multitâche) : variante qui accorde une préférence aux processus dont la priorité est la plus élevée.

Dans la plupart des systèmes multitâches, l'ordonnanceur peut mettre en pause l'exécution d'un processus pour en exécuter un autre : on dit que l'ordonnanceur est **préemptif.**



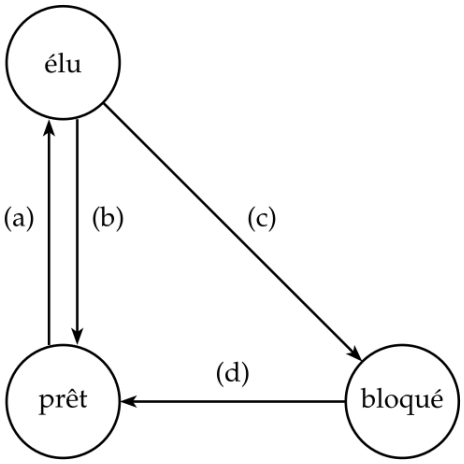
États d'un processus

Un processus peut se trouver dans l'un des états suivants :

- prêt : le processus est prêt à être exécuté (*ready*) ;
- élu : le processus est en cours d'exécution par le processeur (*running*) ;
- bloqué : le processus est en attente d'un événement ou d'une ressource (*blocked*).

À sa création, un processus se trouve dans l'état « prêt ».

Le graphe ci-dessous décrit les transitions possibles entre les différents états d'un processus.



(a) Le processus a été choisi par l'ordonnanceur.

(b) Le processus n'est pas terminé, mais un autre processus a été élu.

(c) Le processus a besoin d'une ressource pour continuer son exécution.

(d) Le processus a acquis l'accès à la ressource pour laquelle il était bloqué.

Un processus peut être bloqué pour différentes raisons :

- attente d'une action de l'utilisateur (clic sur un bouton, remplir une zone de texte, etc.) ;
- attente du réseau ;
- attente d'un autre processus ;
- etc.

> **Point Histoire**
> Au début de l'informatique, tout ceci était bien différent. Le premier ordinateur créé en 1945, l'ENIAC, ne pouvait exécuter qu'un seul processus à la fois, sans en garder d'autres en mémoire. Le temps de chargement de cet unique processus était alors plus long que son temps d'exécution.
>
> L'optimisation de l'ordonnancement par les processeurs est encore un vrai sujet de recherche car il n'existe pas de solution universelle pour résoudre cette problématique.

Plusieurs processus peuvent être chargés en mémoire en même temps donc plusieurs processus peuvent être dans l'état prêt. L'ordonnancement (ou *scheduling*) permet de répondre aux questions :

- Comment choisir quel processus exécuter ?
- Quel sera le suivant ?

Exemples illustrés

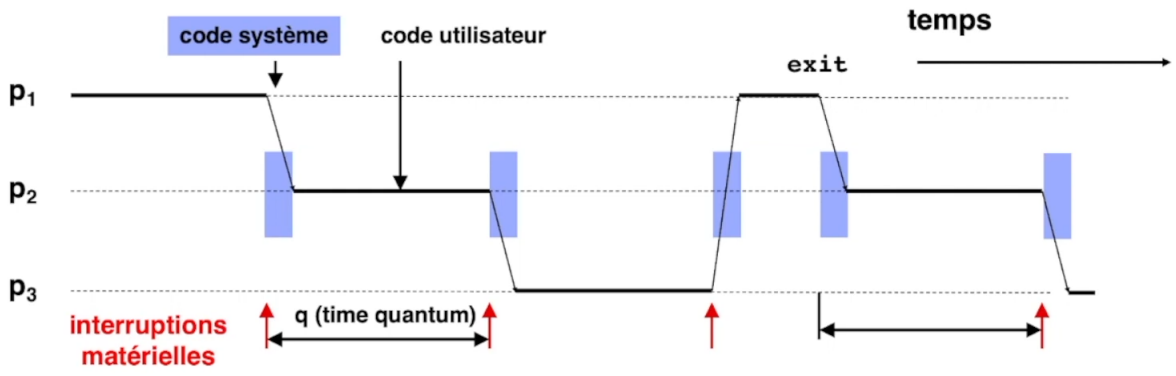


Image de R. Lachaize et S. Krakowiak

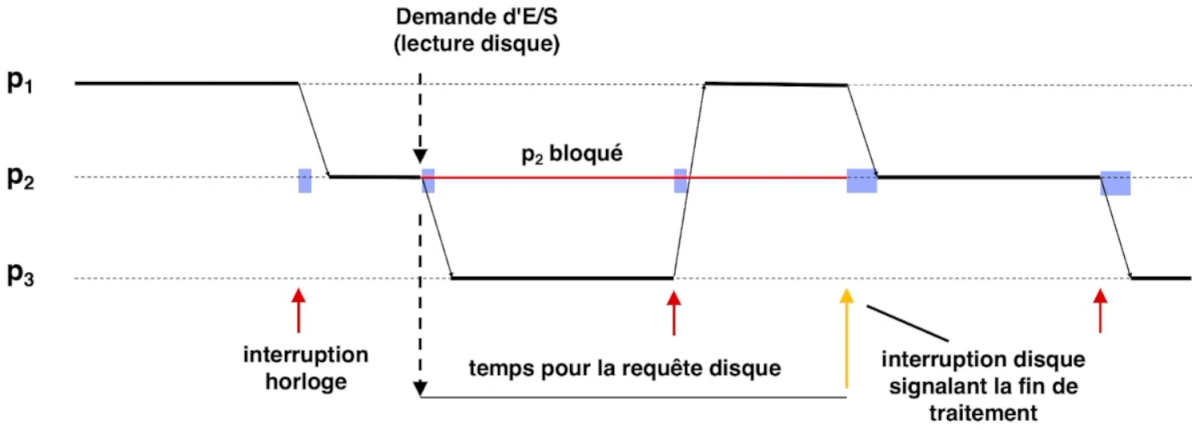


Image de R. Lachaize et S. Krakowiak

Synthèse

Les systèmes d'exploitation utilisent des algorithmes d'ordonnancement pour élire le prochain processus à exécuter parmi ceux en attente. Si le nouveau processus est différent de l'ancien, un changement de contexte est réalisé.

Pour cela, la procédure d'ordonnancement est appelée à intervalles (quanta) de temps réguliers.

La conception d'un algorithme d'ordonnancement résulte d'un compromis entre les critères suivants :

- maximiser l'utilisation du processeur pour exécuter les processus ;
- minimiser le temps d'attente entre la création d'un processus et son exécution ;
- minimiser le temps de réponse (délai avant le premier effet perceptible par l'utilisateur) ;
- garantir l'équité de traitement (*fairness*) entre les processus ;
- garantir le respect des échéances (pour les systèmes embarqués).

En pratique, les ordonnanceurs utilisent des algorithmes bien plus élaborés que ceux présentés ici.

Exercice 1

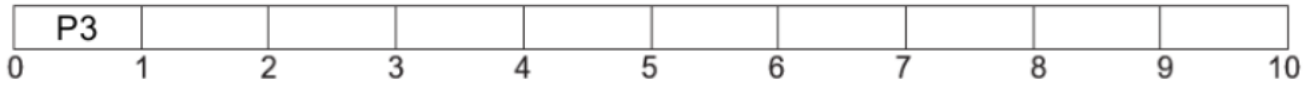
Un processeur choisit à chaque cycle d'exécution le processus qui doit être exécuté. Le tableau ci-dessous donne pour trois processus P1, P2, P3 :

- la durée d'exécution (en nombre de cycles) ;
- l'instant d'arrivée sur le processeur (exprimé en nombre de cycles à partir de 0) ;
- le numéro de priorité.

Le numéro de priorité est d'autant plus petit que la priorité est grande. On suppose qu'à chaque instant, c'est le processus qui a le plus petit numéro de priorité qui est exécuté, ce qui peut provoquer la suspension d'un autre processus, lequel reprendra lorsqu'il sera le plus prioritaire.

Processus	Durée d'exécution	Instant d'arrivée	Numéro de priorité
P1	3	3	1
P2	3	2	1
P3	4	0	3

Indiquer dans chacune des cases le processus exécuté à chaque cycle :

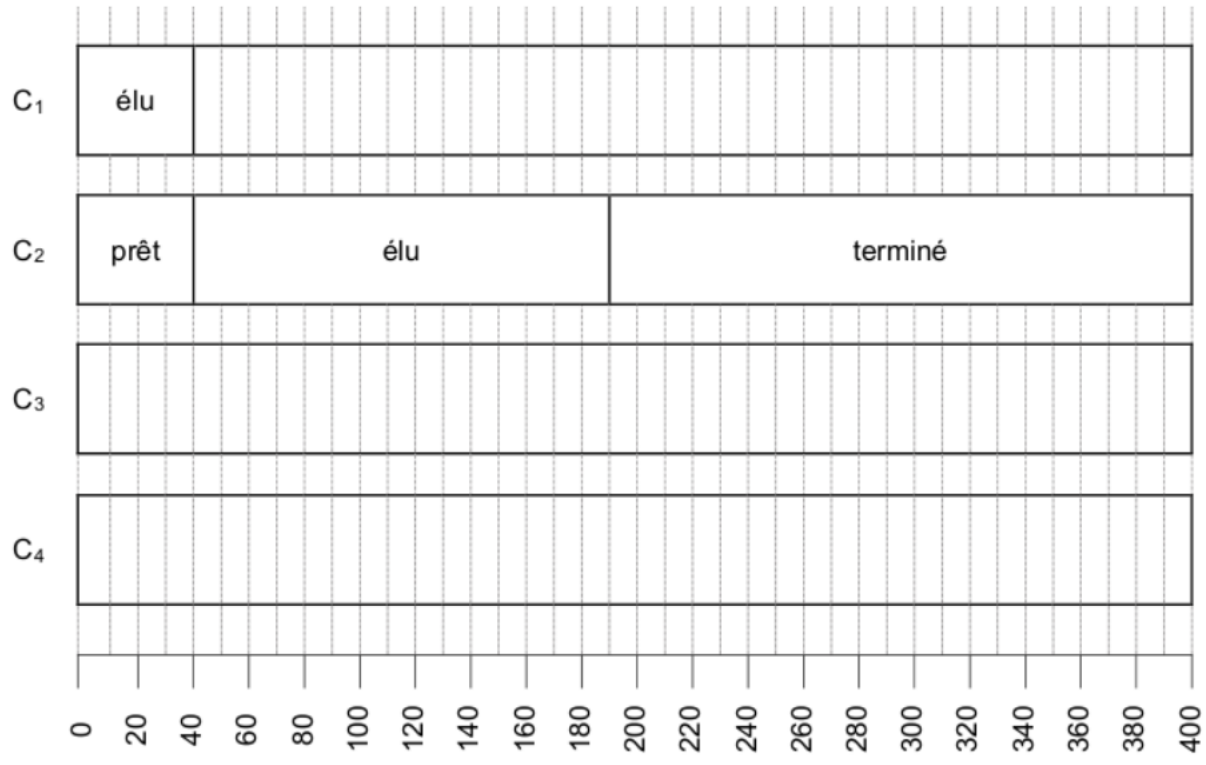


Exercice 2

On suppose que quatre processus C1, C2, C3 et C4 sont créés sur un ordinateur, et qu'aucun autre processus n'est lancé sur celui-ci, ni préalablement ni pendant l'exécution des quatre processus. L'ordonnanceur, pour exécuter les différents processus prêts, les place dans une structure de données de type file. Un processus prêt est enfilé et un processus élu est défilé.

1. Parmi les propositions suivantes, entourer celle qui décrit le fonctionnement des entrées/sorties dans une file :
   - Premier entré, dernier sorti
   - Premier entré, premier sorti
   - Dernier entré, premier sorti
2. On suppose que les quatre processus arrivent dans la file et y sont placés dans l'ordre C1, C2, C3 et C4.
   - Les temps d'exécution totaux de C1, C2, C3 et C4 sont respectivement 100 ms, 150 ms, 80 ms et 60 ms.
   - Après 40 ms d'exécution, le processus C1 demande une opération d'écriture disque, opération qui dure 200 ms. Pendant cette opération d'écriture, le processus C1 passe à l'état bloqué.
   - Après 20 ms d'exécution, le processus C3 demande une opération d'écriture disque, opération qui dure 10 ms. Pendant cette opération d'écriture, le processus C3 passe à l'état bloqué.

   Sur la frise chronologique ci-dessous, les états du processus C2 sont donnés. Compléter la frise avec les états des processus C1, C3 et C4.



3.2 L'interblocage

Définition
On désigne par ressource un **accès** à un élément physique (tel qu'un périphérique, un port réseau, la mémoire vive, la mémoire de masse, *etc.*) mais aussi à une portion de code,

qui ne peut pas être partagé entre plusieurs processus en même temps.

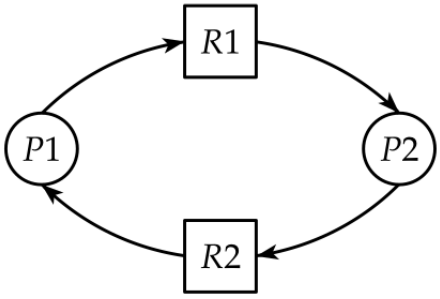
L'interblocage (*deadlock*) de plusieurs processus désigne alors la situation où

les processus sont dans l'état bloqué,

chacun dans l'attente d'une ressource détenue par un autre.

Un scénario simple d'interblocage de deux processus $P1$ et $P2$ est le suivant.

- Le processus $P1$ est élu et acquiert une ressource $R1$.
- Le processus $P2$ est élu et acquiert une ressource $R2$.
- Le processus $P2$ a besoin de la ressource $R1$ et se retrouve bloqué.
- Le processus $P1$ est élu, a besoin de la ressource $R2$ et se retrouve bloqué.



Dans cette situation, $P1$ attend la ressource $R2$ qui est détenue par $P2$ qui attend la ressource $R1$ qui est détenue par $P1$ qui attend la ressource $R2$ qui est détenue par $P2$ qui attend la ressource $R1$ qui est détenue par $P1$ qui attend la ressource $R2$ qui est détenue par $P2$ qui ...

Exemple :

Considérons 2 processus A et B, et deux ressources R et S. L'action des processus A et B est décrite ci-dessous :

Processus A	Processus B
Étape A1 : demande R	Étape B1 : demande S
Étape A2 : demande S	Étape B2 : demande R
Étape A3 : libère S	Étape B3 : libère R
Étape A4 : libère R	Étape B4 : libère S

Déroulement des processus A et B :

- A et B sont créés et passent à l'état `prêt`.
- L'ordonnanceur déclare `élu` le processus A (ou bien B, cela ne change rien).
- L'étape A1 de A est réalisée : la ressource R est donc affectée à A.
- L'ordonnanceur déclare maintenant `élu` le processus B. A est donc passé à `prêt` en attendant que son tour revienne.
- L'étape B1 de B est réalisée : la ressource S est donc affectée à B.
- L'ordonnanceur déclare à nouveau `élu` le processus A. B est donc passé à `prêt` en attendant que son tour revienne.

- L'étape A2 de A est donc enclenchée : problème, il faut pour cela pouvoir accèder à la ressource S, qui n'est pas disponible. L'ordonnanceur va donc passer A à `bloqué` et va revenir au processus B qui redevient `élu`.
- L'étape B2 de B est donc enclenchée : problème, il faut pour cela pouvoir accèder à la ressource R, qui n'est pas disponible. L'ordonnanceur va donc passer B à `bloqué`.

Les deux processus A et B sont donc dans l'état `bloqué`, chacun en attente de la libération d'une ressource bloquée par l'autre : ils se bloquent mutuellement.

Cette situation (critique) est appelée **interblocage** ou ***deadlock***.

Représentation schématique

- Les processus seront représentés par des cercles, les ressources par des carrés ;
- Si à l'étape A1 le processus A a **demandé et reçu** la ressource R, on trace une flèche de R vers A ;
- Si à l'étape A2 le processus A est **en attente** de la ressource S, on trace une flèche de A vers R.

Avec ces conventions, la situation précédente peut donc se schématiser par :

Ce type de schéma fait apparaître un **cycle d'interdépendance**, qui caractérise ici la situation d'**interblocage** (*deadlock*).

Exercice 3

Un constructeur automobile utilise des ordinateurs pour la conception de ses véhicules. Ceux-ci sont munis d'un système d'exploitation ainsi que de nombreuses applications parmi lesquelles on peut citer :

- un logiciel de traitement de texte ;
- un tableur ;
- un logiciel de Conception Assistée par Ordinateur (CAO) ;
- un système de gestion de base de données (SGBD)

Chaque ordinateur est équipé des périphériques classiques : clavier, souris, écran et imprimante réseau.

Un ingénieur travaille sur son ordinateur et utilise les quatre applications citées au début de l'énoncé. Pendant l'exécution de ces applications, des processus mobilisent des données et sont en attente d'autres données mobilisées par d'autres processus. On donne ci-dessous un tableau indiquant à un instant précis l'état des processus en cours d'exécution et dans lequel D1, D2, D3, D4 et D5 sont des données. La lettre M signifie que la donnée est mobilisée par l'application ; la lettre A signifie que l'application est en attente de cette donnée.

Lecture du tableau : le traitement de texte mobilise (M) la donnée D1 et est en attente (A) de la donnée D2.

	D1	D2	D3	D4	D5
Traitement de texte	M	A	-	-	-
Tableur	A	-	-	-	M
SGBD	-	M	A	A	-
CAO	-	-	A	M	A

Montrer que les applications s'attendent mutuellement. Comment s'appelle cette situation ?

Exercice 4

On suppose que trois processus P1, P2 et P3 s'exécutent et utilisent une ou plusieurs ressources parmi R1, R2 et R3. Parmi les scénarios suivants, lequel provoque un interblocage ? Justifier.

Scénario 1	Scénario 2	Scénario 3
P1 acquiert R1	P1 acquiert R1	P1 acquiert R1
P2 acquiert R2	P2 acquiert R3	P2 acquiert R2
P3 attend R1	P3 acquiert R2	P3 attend R2
P2 libère R2	P1 attend R2	P1 attend R2
P2 attend R1	P2 libère R3	P2 libère R2
P1 libère R1	P3 attend R1	P3 acquiert R2