Introduction

Thomas Lavergne
lavergne@lisn.fr

Qu'est-ce que l'informatique ?

Qu'est-ce que l'informatique ?

La science du traitement automatique de l'information.

Qu'est-ce que l'informatique ?

La science du traitement automatique de l'information.

Qu'est-ce qu'un ordinateur ?

Qu'est-ce que l'informatique ?

La science du traitement automatique de l'information.

Qu'est-ce qu'un ordinateur ?

Une machine pour traiter de l'information :

Qu'est-ce que l'informatique ?

La science du traitement automatique de l'information.

Qu'est-ce qu'un ordinateur ?

Une machine pour traiter de l'information :

- On lui donne des instructions

Qu'est-ce que l'informatique ?

La science du traitement automatique de l'information.

Qu'est-ce qu'un ordinateur ?

Une machine pour traiter de l'information :

- On lui donne des instructions
- On lui donne des données

Qu'est-ce que l'informatique ?

La science du traitement automatique de l'information.

Qu'est-ce qu'un ordinateur ?

Une machine pour traiter de l'information :

- On lui donne des instructions
- On lui donne des données
- Il transforme les données

Qu'est-ce que l'informatique ?

La science du traitement automatique de l'information.

Qu'est-ce qu'un ordinateur ?

Une machine pour traiter de l'information :

- On lui donne des instructions
- On lui donne des données
- Il transforme les données

Une calculatrice est-elle un ordinateur ?

Qu'est-ce que l'informatique ?

La science du traitement automatique de l'information.

Qu'est-ce qu'un ordinateur ?

Une machine pour traiter de l'information :

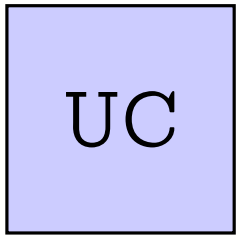
- On lui donne des instructions
- On lui donne des données
- Il transforme les données

Une calculatrice est-elle un ordinateur ?

Non: notion de programme !

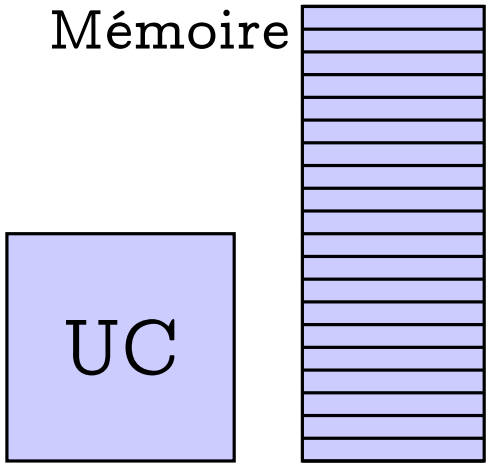
Composition:

Un processeur



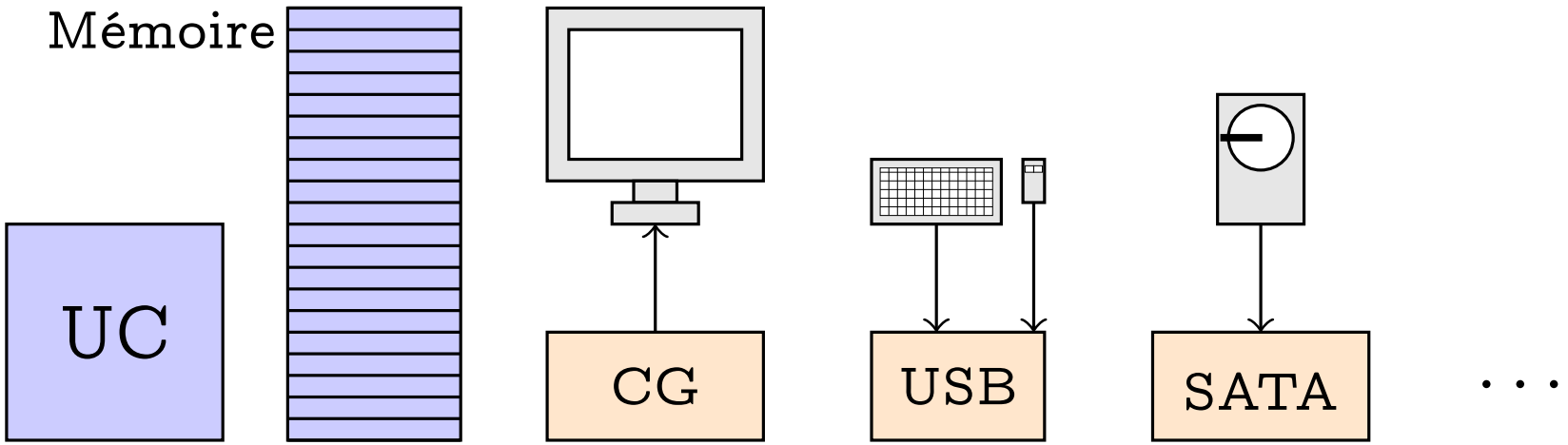
Composition:

Un processeur, de la mémoire



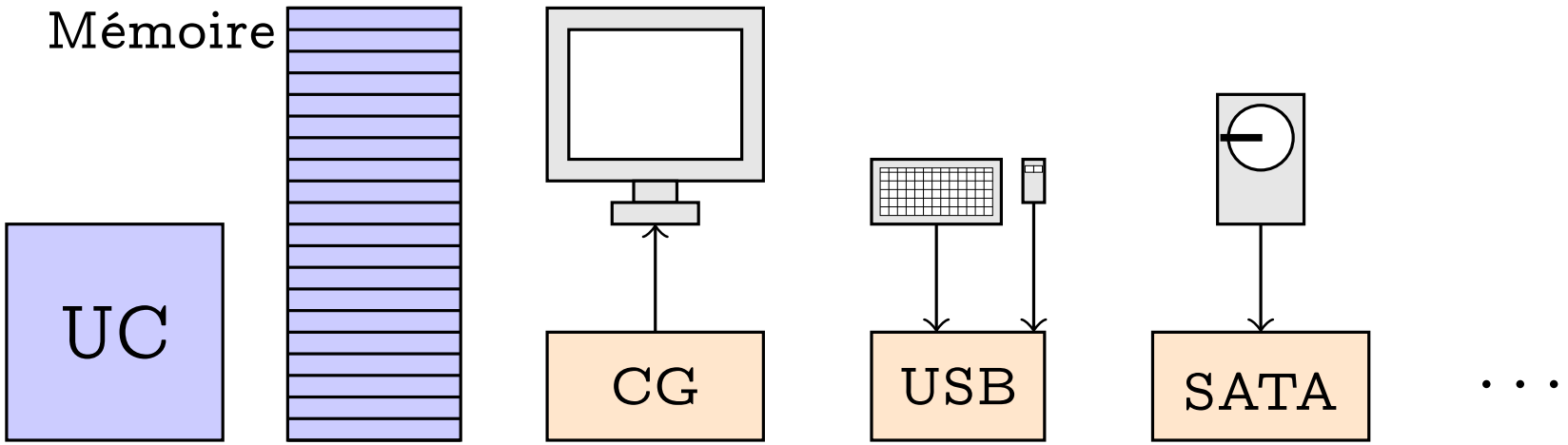
Composition:

Un processeur, de la mémoire, des périphériques



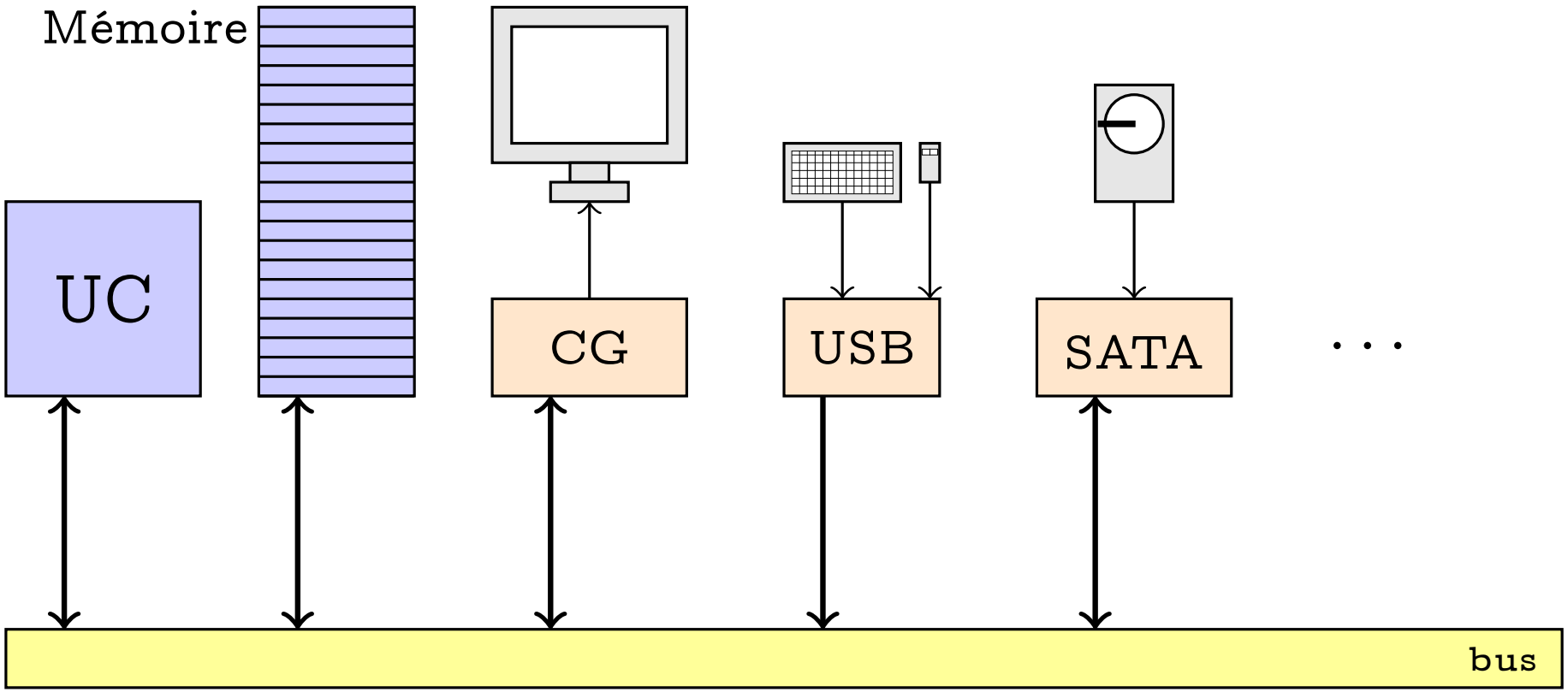
Composition:

Un processeur, de la mémoire, des périphériques et un bus



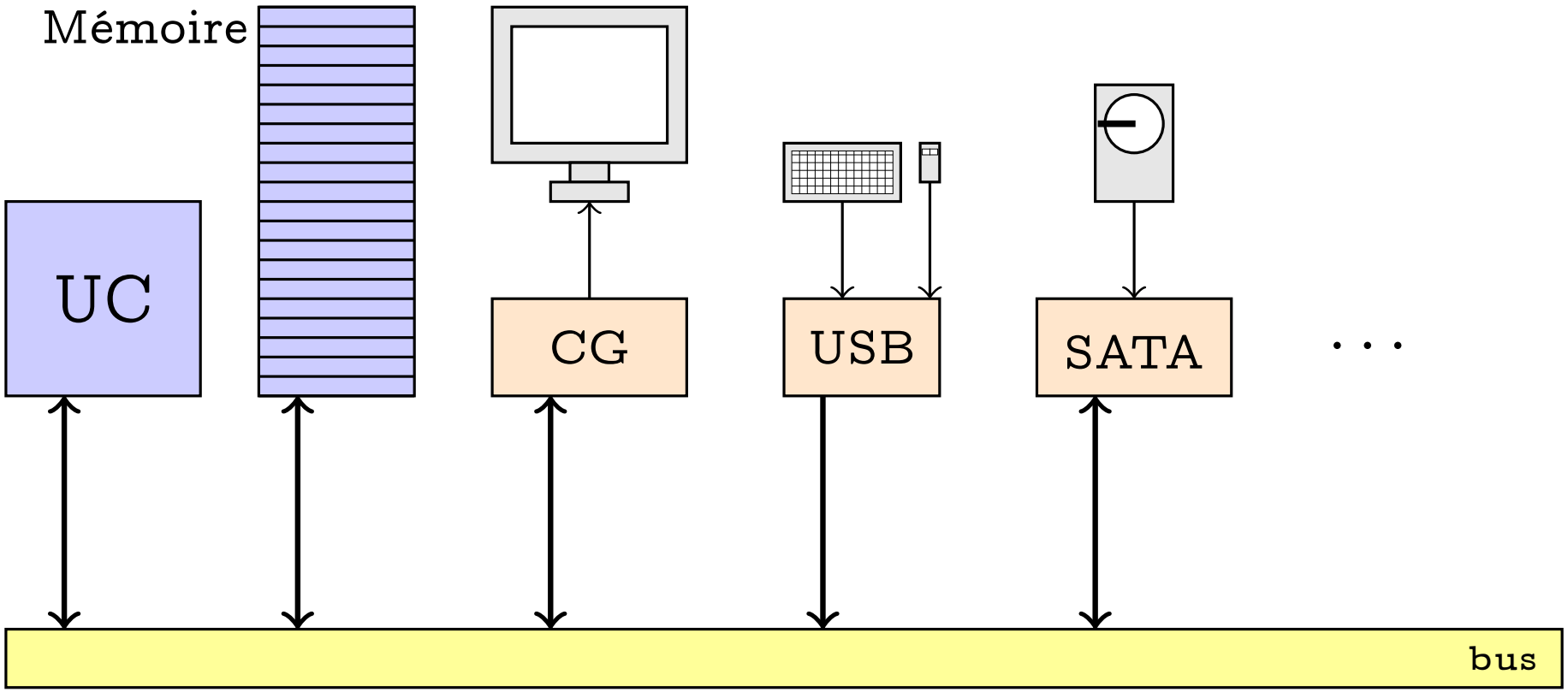
Composition:

Un processeur, de la mémoire, des périphériques et un bus pour tout connecter.



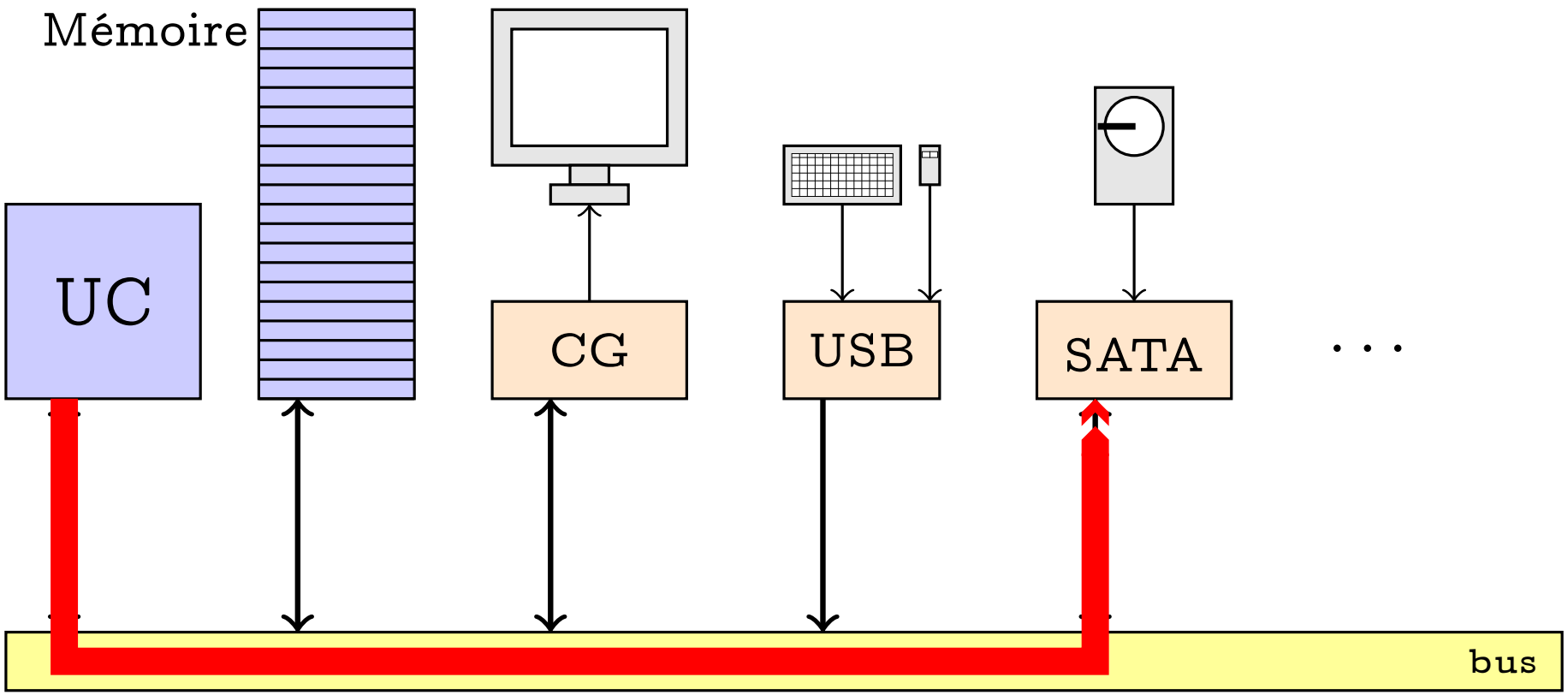
Fonctionnement d'un ordinateur

Juste des fils qu'on allume et éteint...



Fonctionnement d'un ordinateur

Juste des fils qu'on allume et éteint...

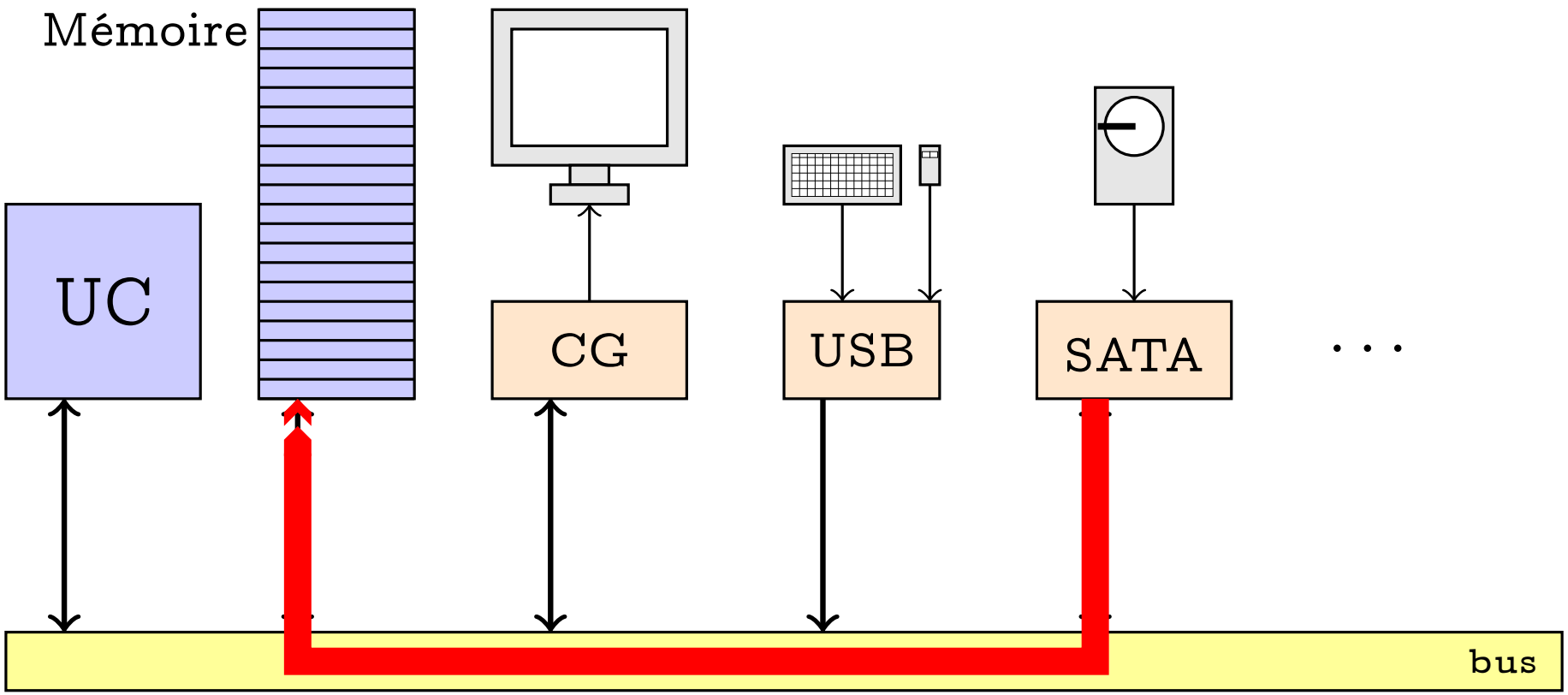


Charger des données depuis le disque :

Envoie d'une commande de chargement au disque

Fonctionnement d'un ordinateur

Juste des fils qu'on allume et éteint...

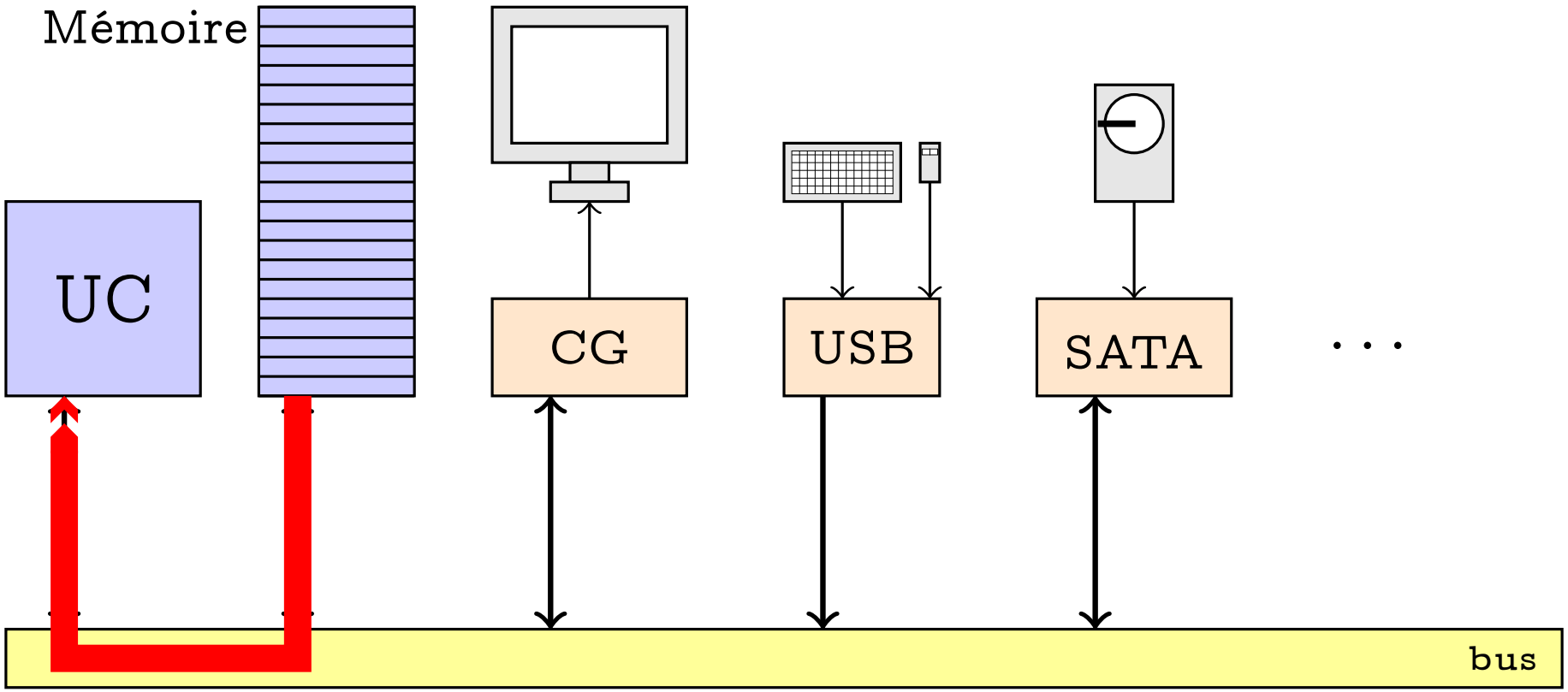


Charger des données depuis le disque :

Transfert du disque vers la mémoire

Fonctionnement d'un ordinateur

Juste des fils qu'on allume et éteint...



Charger des données depuis le disque :

Utilisation des données par le processeur

Juste des fils qu'on allume et éteint...

Il y a bien longtemps...

Un opérateur déplaçait des interrupteurs pour ouvrir et fermer les accès au bus.

Juste des fils qu'on allume et éteint...

Il y a bien longtemps...

Un opérateur déplaçait des interrupteurs pour ouvrir et fermer les accès au bus.

Peut-on automatiser ces opérations ?

Juste des fils qu'on allume et éteint...

Il y a bien longtemps...

Un opérateur déplaçait des interrupteurs pour ouvrir et fermer les accès au bus.

Peut-on automatiser ces opérations ?

Système d'exploitation

Un programme qui :

- Décide qui fait quoi à quel moment
- Fait le lien entre applications et materiel

Un peu d’histoire

1950 : Traîtement par lots

Problème

- Données et programmes $\rightarrow$ Cartes perforées
- Beaucoup de manipulations par un opérateur

1950 : Traîtement par lots

Problème

- Données et programmes $\rightarrow$ Cartes perforées
- Beaucoup de manipulations par un opérateur

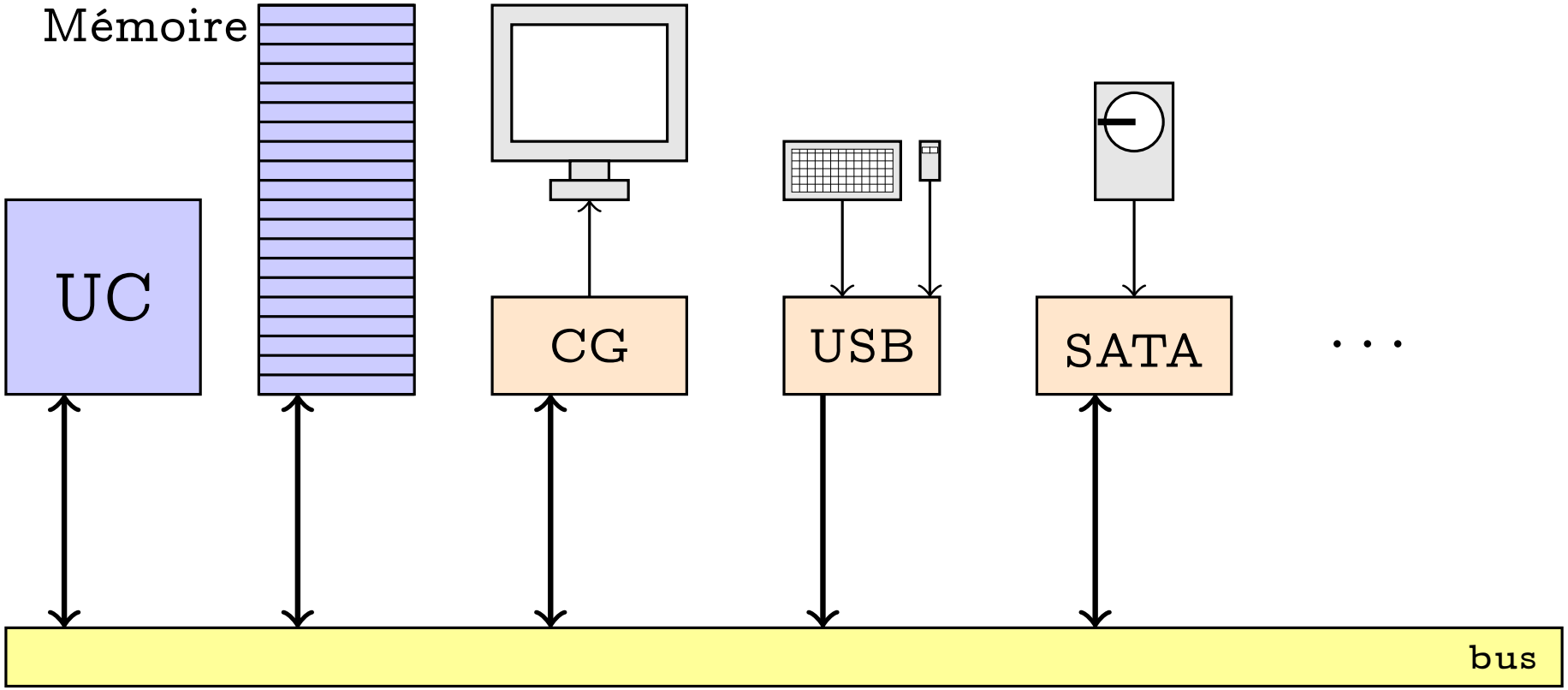
Carte de traîtement

- Ajout de cartes remplacement les manipulations
  - Lectures ou écritures
  - Connection au bus
  - ...

1960 : Multiprogrammation

Problème

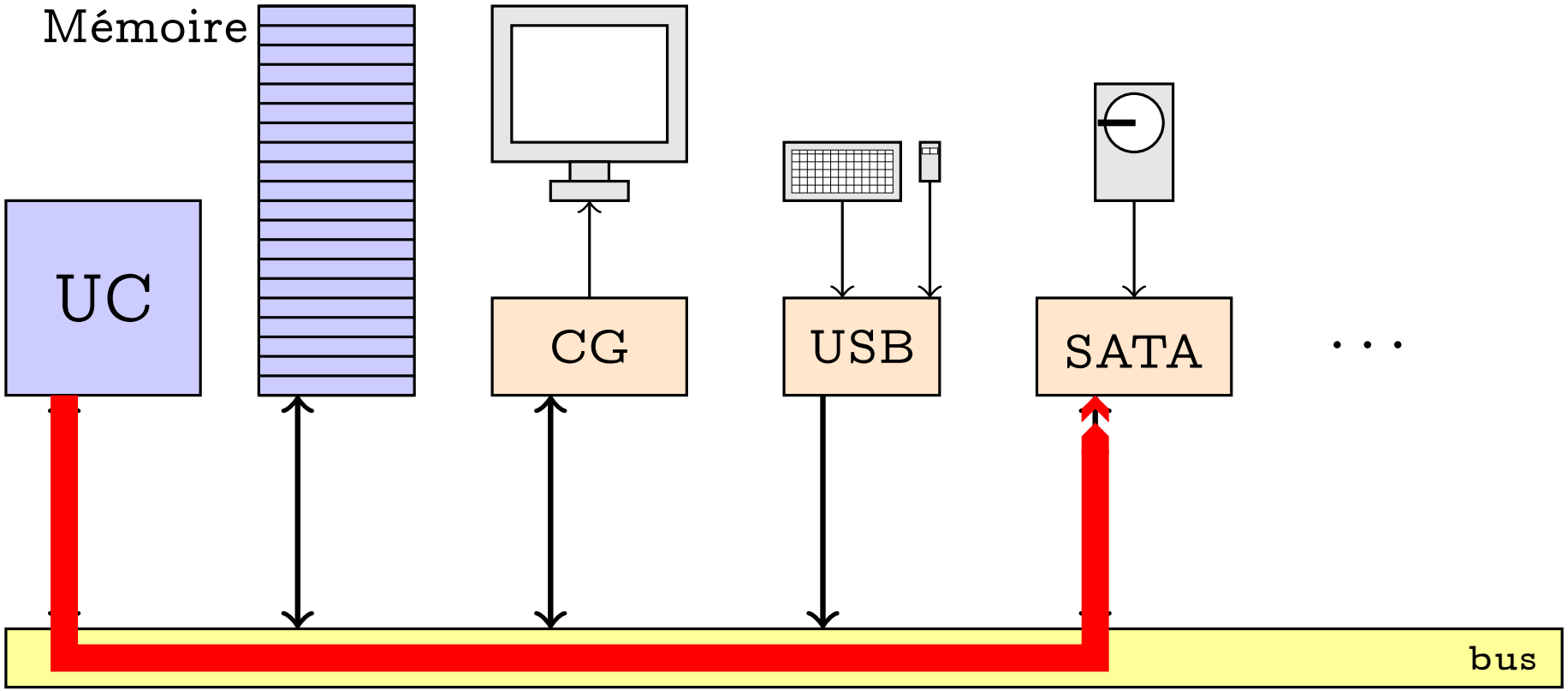
- Un seul composant utilisé à la fois
- Comment mieux exploiter les ressources ?



1960 : Multiprogrammation

Problème

- Un seul composant utilisé à la fois
- Comment mieux exploiter les ressources ?

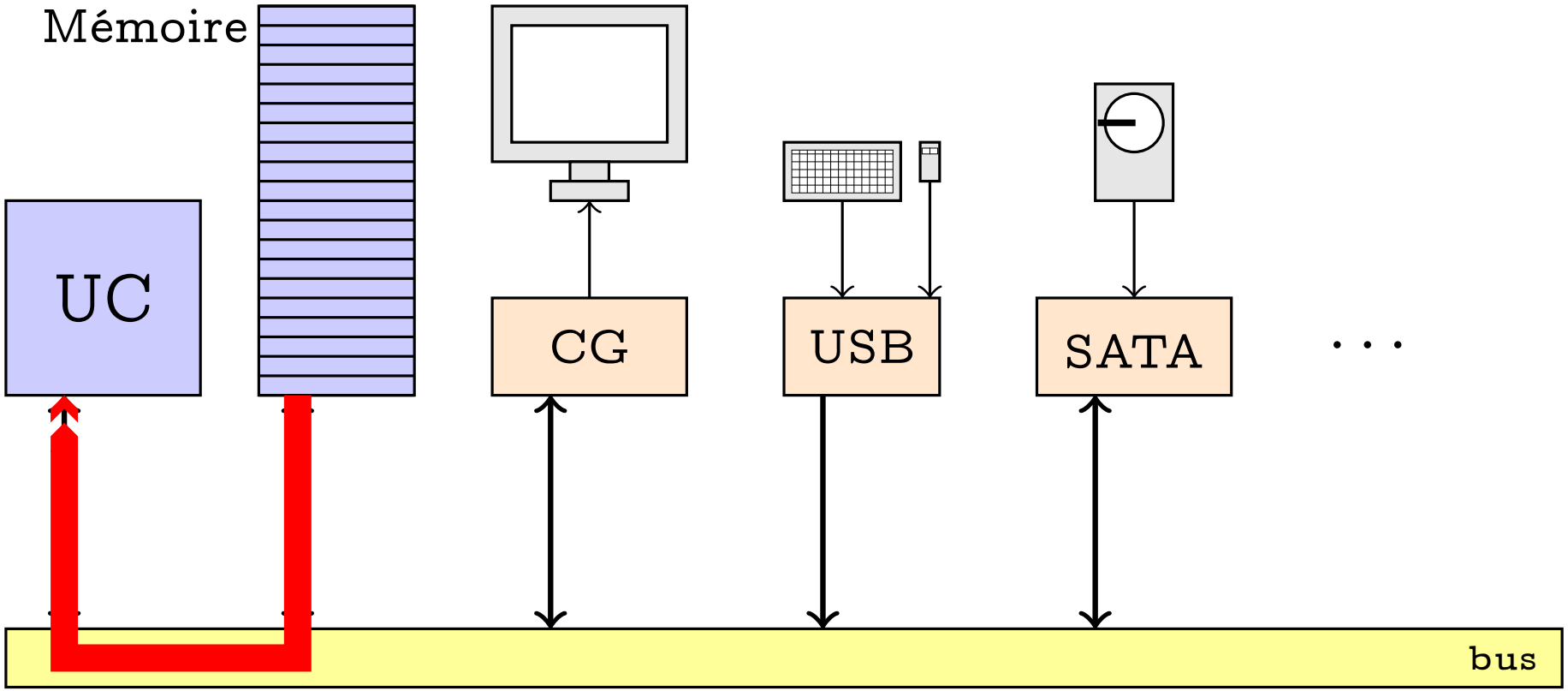


On demande le chargement de données du disque

1960 : Multiprogrammation

Problème

- Un seul composant utilisé à la fois
- Comment mieux exploiter les ressources ?

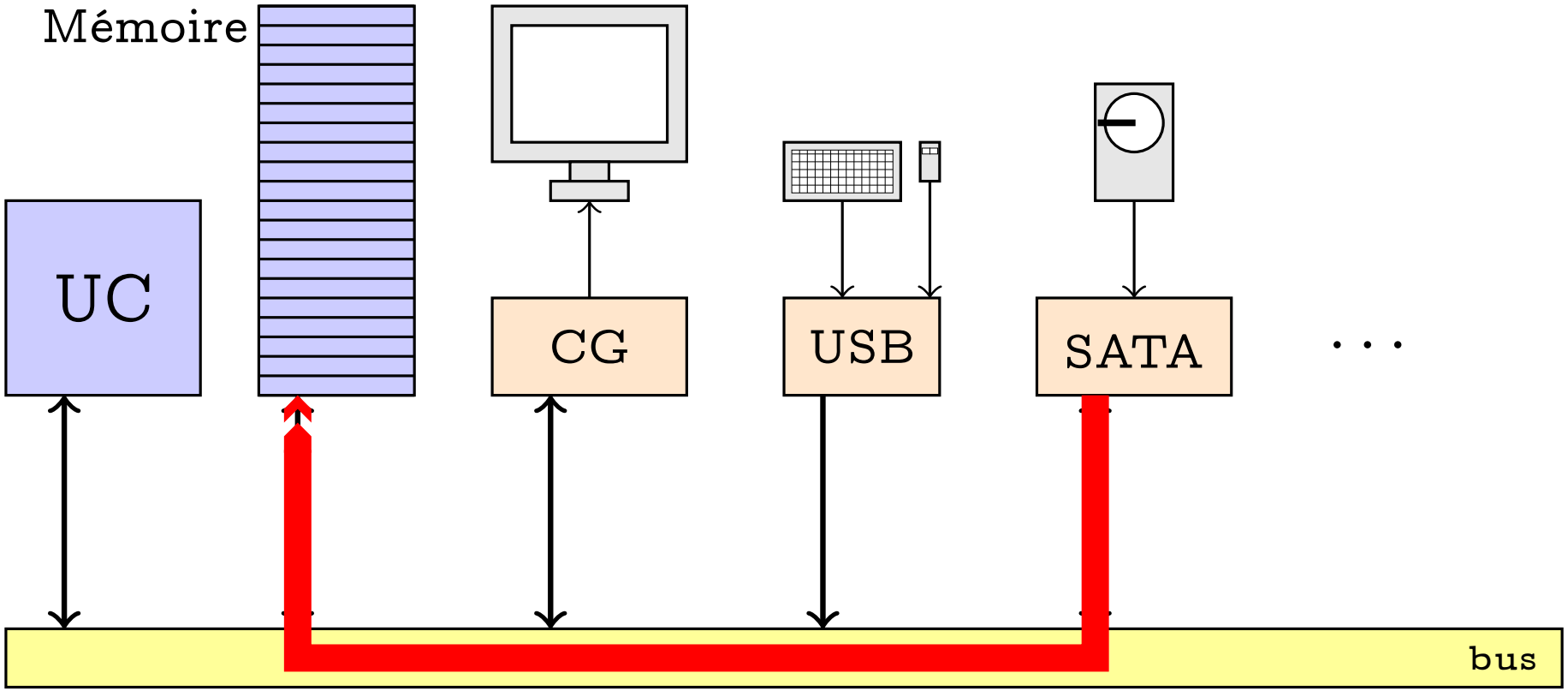


Pendant la préparation on lit depuis la RAM

1960 : Multiprogrammation

Problème

- Un seul composant utilisé à la fois
- Comment mieux exploiter les ressources ?

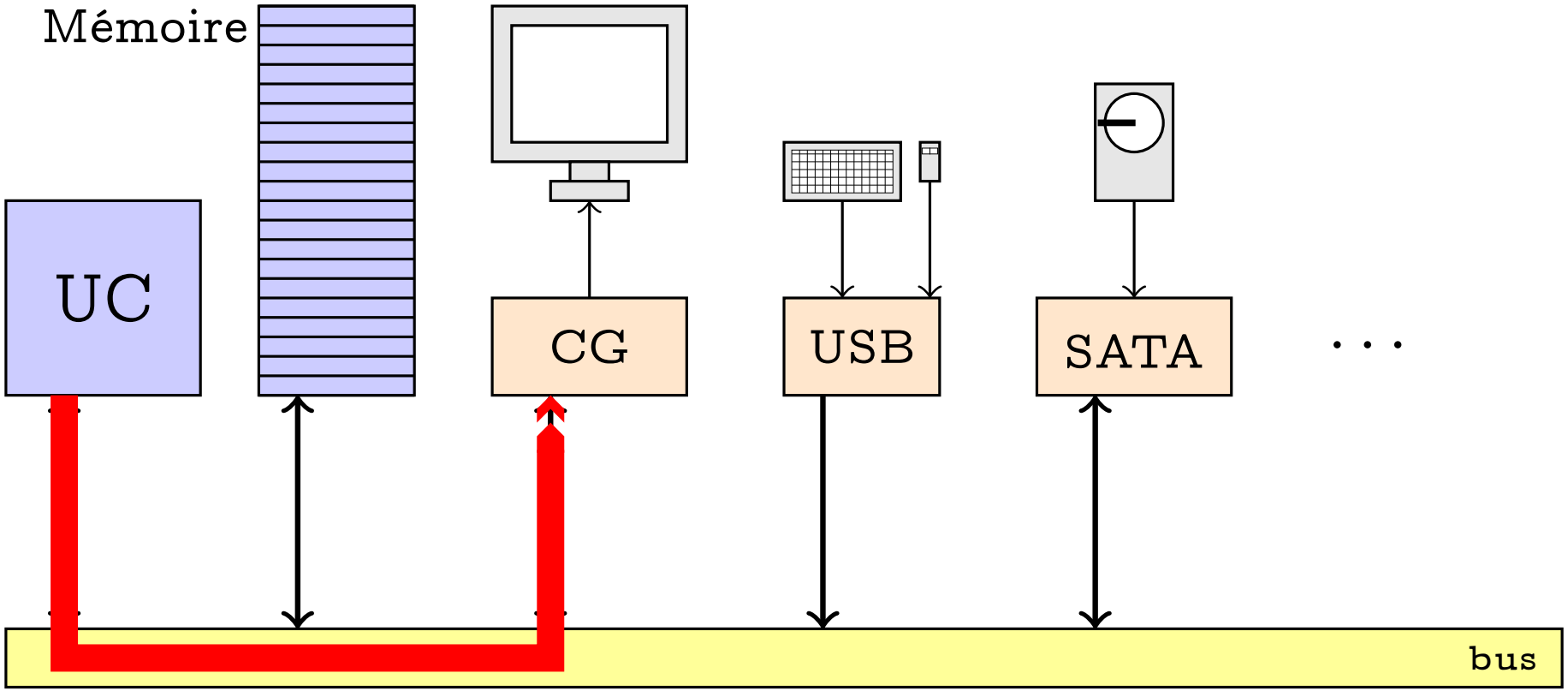


Pendant la lecture l'UC calcule

1960 : Multiprogrammation

Problème

- Un seul composant utilisé à la fois
- Comment mieux exploiter les ressources ?

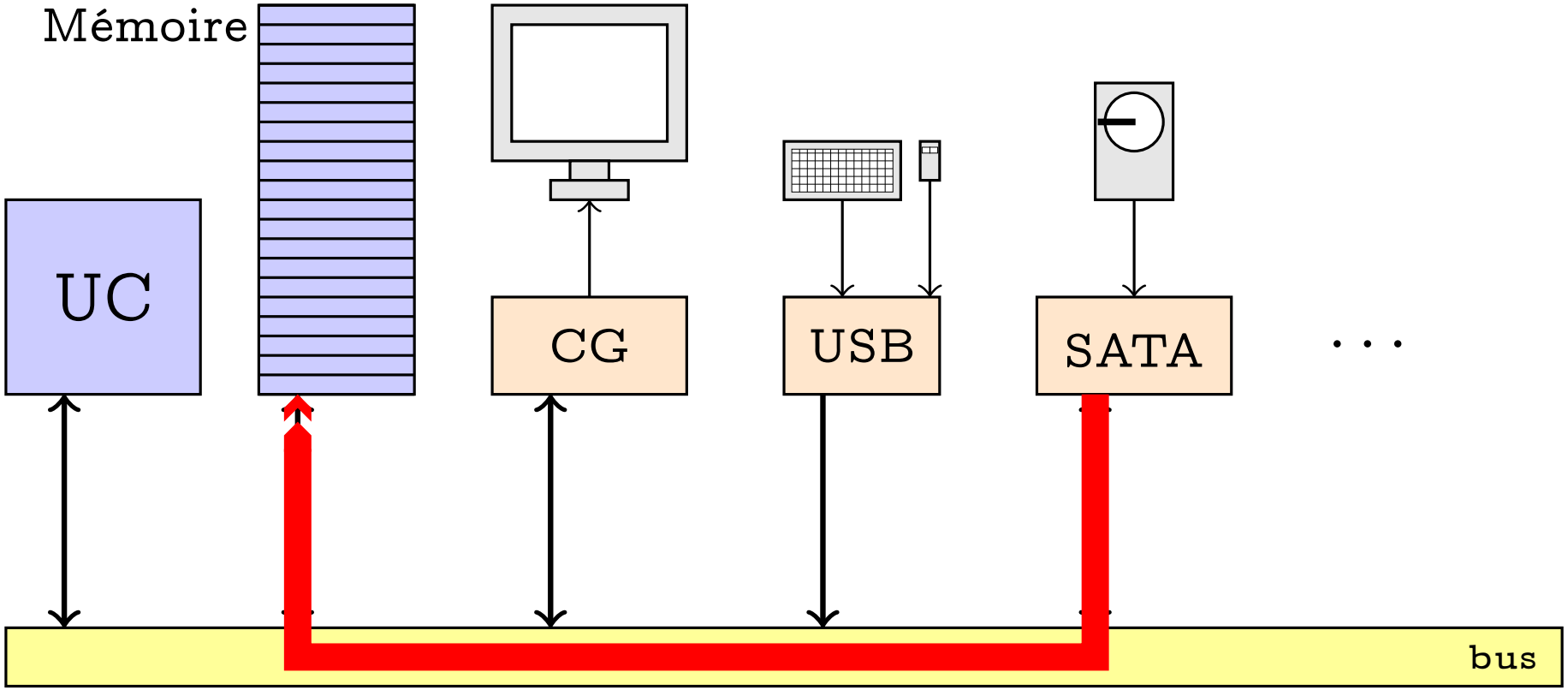


À la fin du calcul, on met en pause pour l'affichage

1960 : Multiprogrammation

Problème

- Un seul composant utilisé à la fois
- Comment mieux exploiter les ressources ?

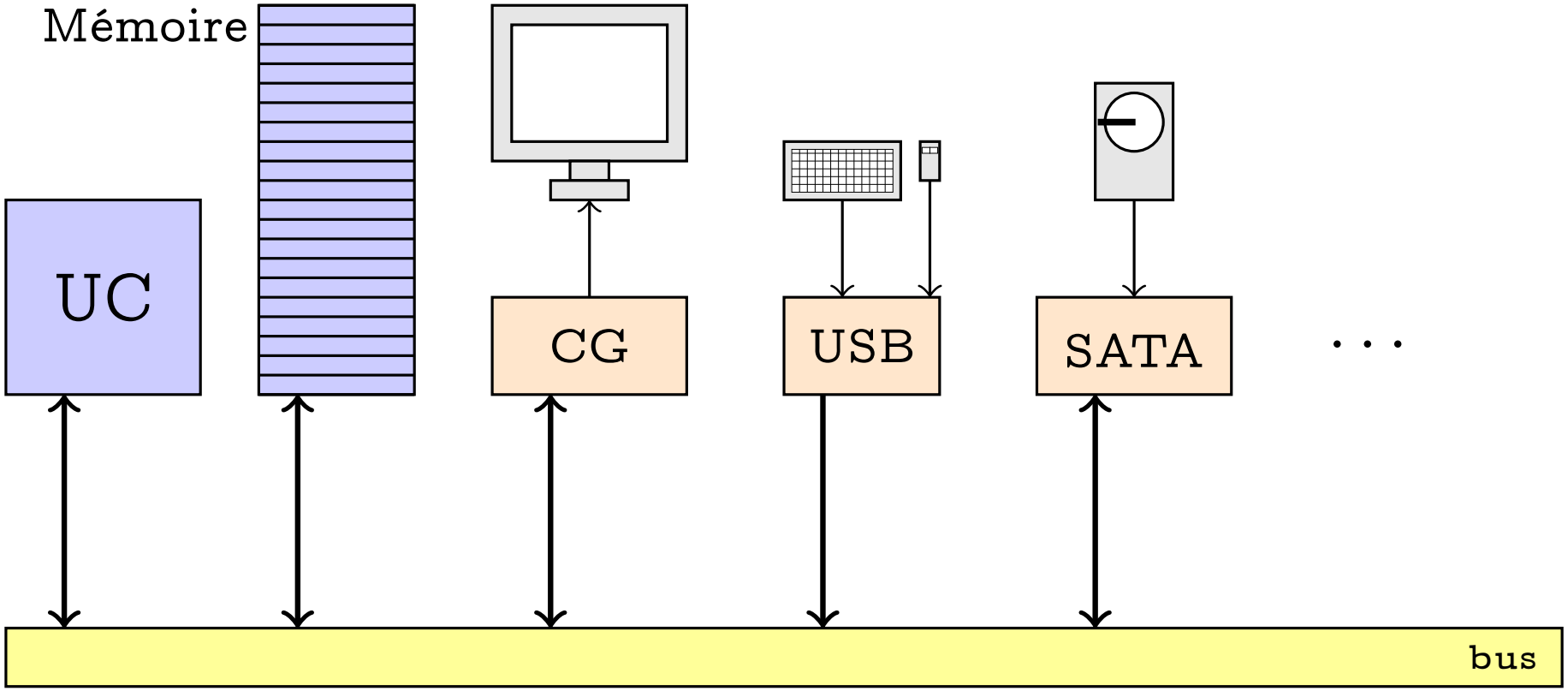


Puis on reprend

1970 : Temps partagé

Problème

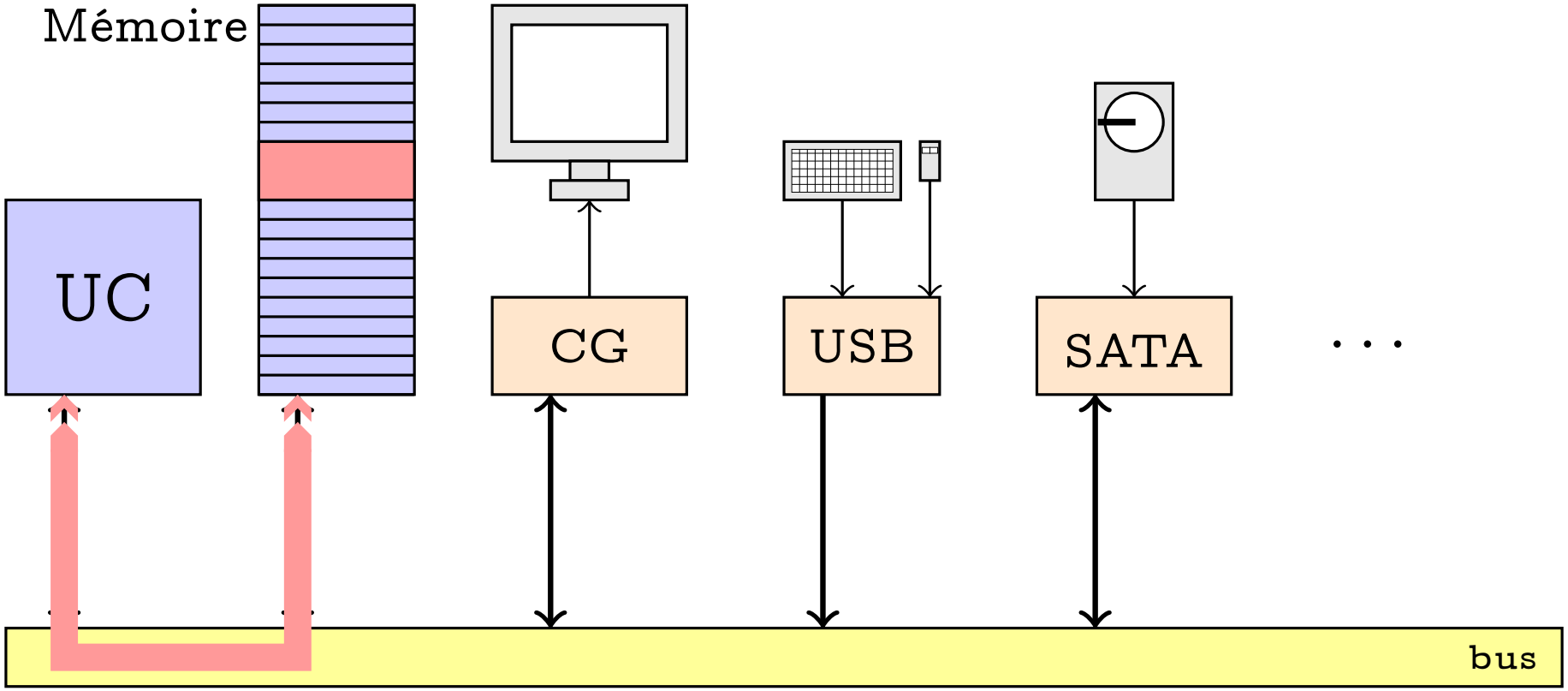
- Certains processus processus attendent...
- Certains processus sont simplement très long...



1970 : Temps partagé

Problème

- Certains processus processus attendent...
- Certains processus sont simplement très long...

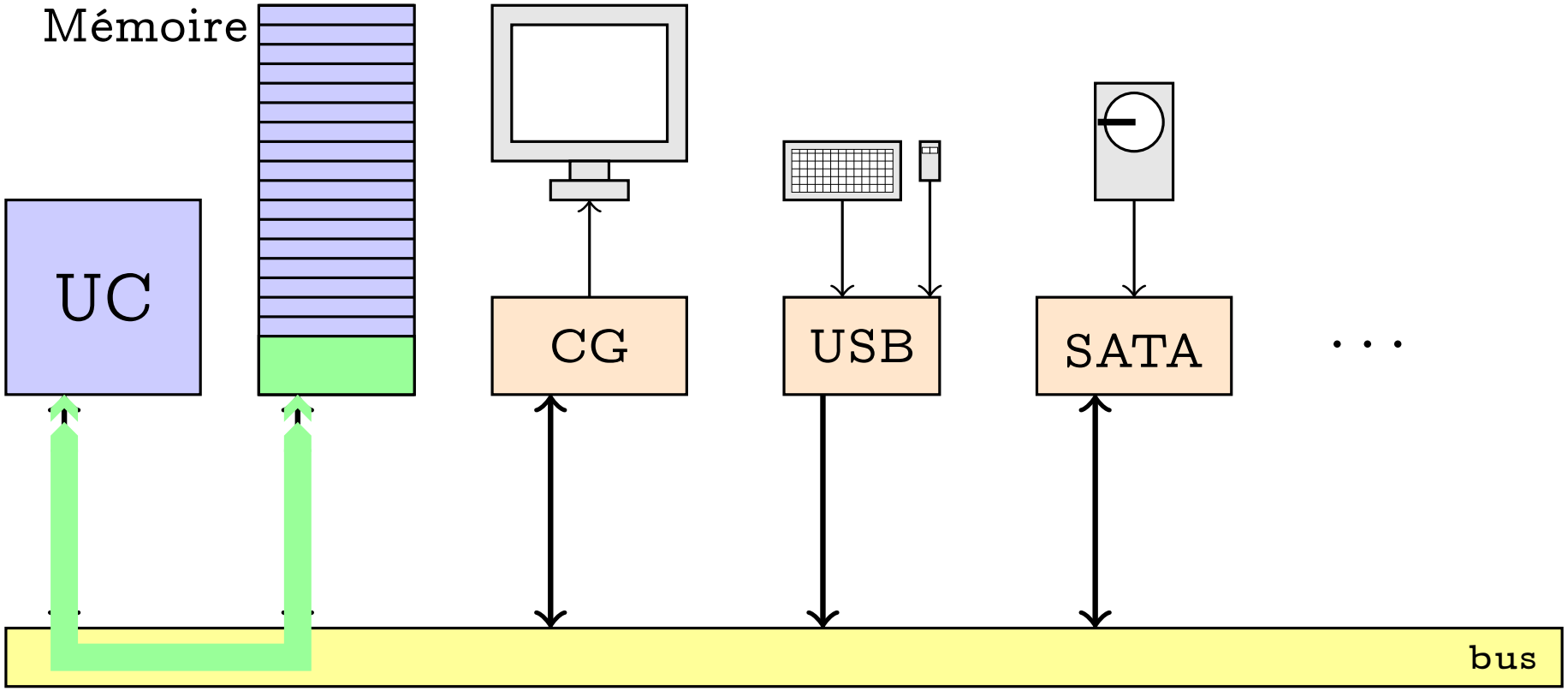


Le processus 1 travaille

1970 : Temps partagé

Problème

- Certains processus processus attendent...
- Certains processus sont simplement très long...

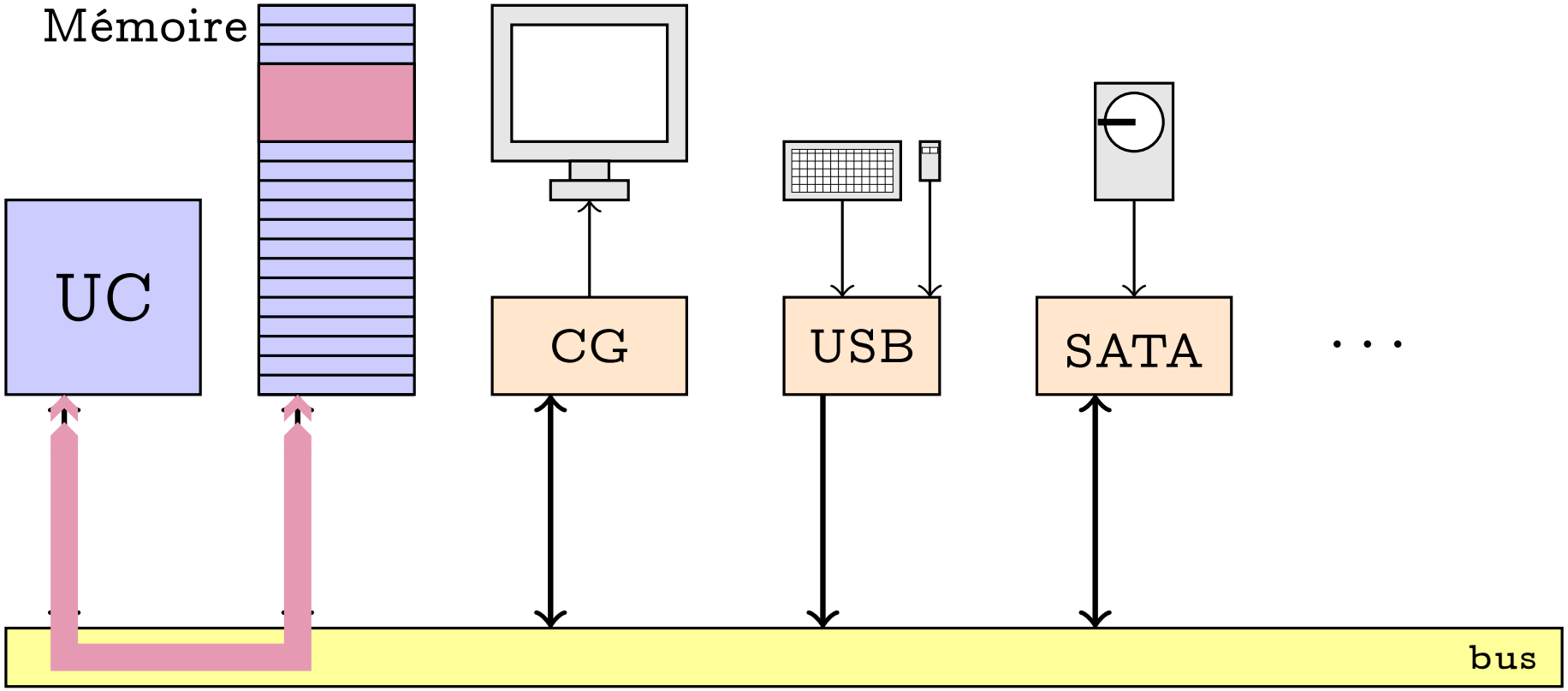


L'OS reprend la main

1970 : Temps partagé

Problème

- Certains processus processus attendent...
- Certains processus sont simplement très long...



Pour la donner au processus 2

Rôle de l'OS

Décide qui fait quoi à quel moment

Répartit les ressources de manière équitable entre les processus.

Rôle de l'OS

Décide qui fait quoi à quel moment

Répartit les ressources de manière équitable entre les processus.

Fait le lien entre application et materiel

Permet aux processus d'accéder aux péripheriques et à ceux-ci d'informer les processus.

Rôle de l'OS

Décide qui fait quoi à quel moment

Répartit les ressources de manière équitable entre les processus.

Fait le lien entre application et materiel

Permet aux processus d'accéder aux péripheriques et à ceux-ci d'informer les processus.

Isole les processus les un des autres

S'assure qu'un processus ne peut accéder aux données des autres processus qui s'exécutent en même temps.

Isolation des processus

Comment isoler les processus ?

Les processus sont exécutés avec des droits réduits dans un mode particulier du processeur.

Isolation des processus

Comment isoler les processus ?

Les processus sont exécutés avec des droits réduits dans un mode particulier du processeur.

L'OS s'exécute en mode superviseur.

Comment isoler les processus ?

Les processus sont exécutés avec des droits réduits dans un mode particulier du processeur.

L'OS s'exécute en mode superviseur.

Mode superviseur

Mode du processeur où les opérations ne sont pas contrôlées :

- accès direct au bus système ;
- accès complet à la mémoire ;
- ...

Communication materiel / OS

Communication OS / logiciel

Communication materiel / OS

- Comment indiquer au materiel ce qu'il faut faire ?

Communication OS / logiciel

Communication materiel / OS

- Comment indiquer au materiel ce qu'il faut faire ?
- Comment le materiel prévient-il l'OS lorsqu'il est prêt ?

Communication OS / logiciel

Communication materiel / OS

- Comment indiquer au materiel ce qu'il faut faire ?
- Comment le materiel prévient-il l'OS lorsqu'il est prêt ?

Communication OS / logiciel

- Comment indiquer à un processus qu'un évennement ce produit ?

Communication materiel / OS

- Comment indiquer au materiel ce qu'il faut faire ?
- Comment le materiel prévient-il l'OS lorsqu'il est prêt ?

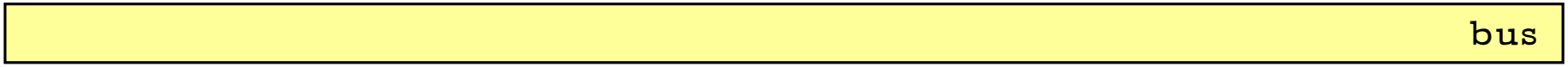
Communication OS / logiciel

- Comment indiquer à un processus qu'un évennement ce produit ?
- Comment soumettre une requête à l'OS ?

Bus système

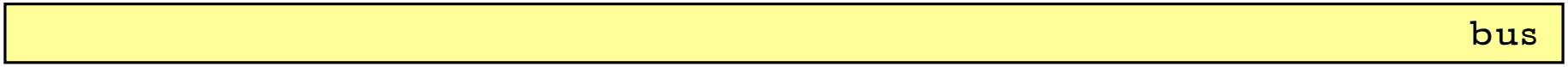
Comunication OS $\leftrightarrow$ materiel

Le bus permet la comunication avec le système



Comunication OS $\leftrightarrow$ materiel

Le bus permet la comunication avec le système

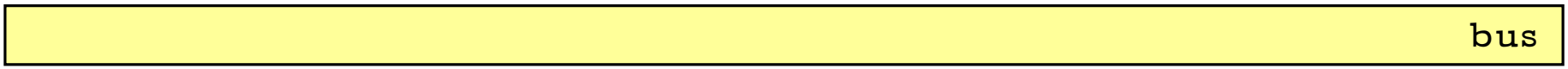


Comment communiquer sans problèmes

- À qui sont adressés les messages ?
- Le destinataire est-il prêt à écouter ?

Comunication OS $\leftrightarrow$ materiel

Le bus permet la comunication avec le système



Comment communiquer sans problèmes

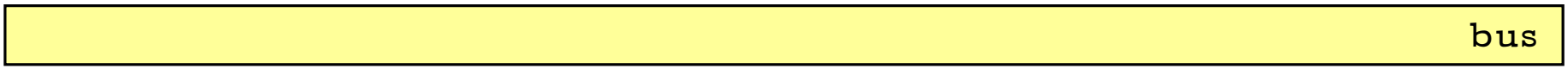
- À qui sont adressés les messages ?

  $\rightarrow$ Fils d'adresses

- Le destinataire est-il prêt à écouter ?

Comunication OS $\leftrightarrow$ materiel

Le bus permet la comunication avec le système



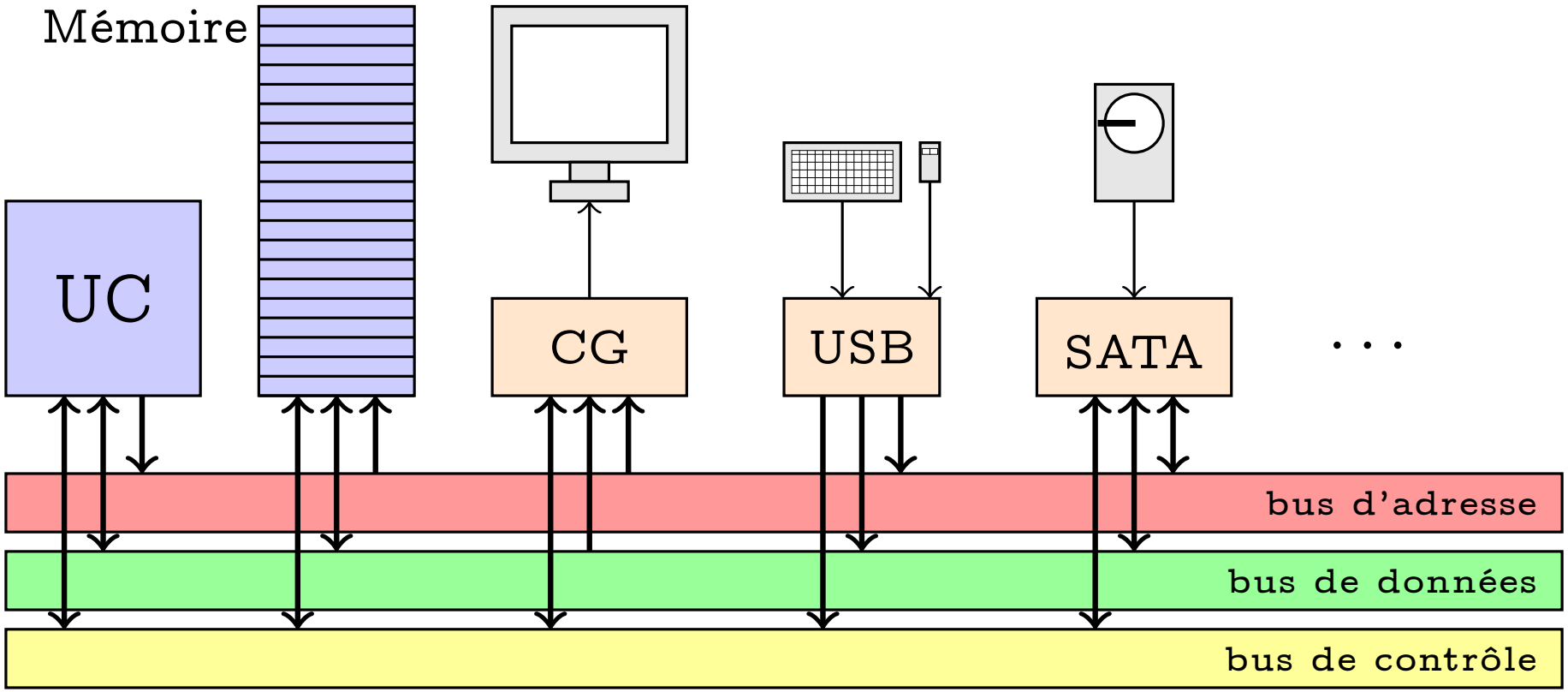
Comment communiquer sans problèmes

- À qui sont adressés les messages ?

  $\rightarrow$ Fils d'adresses
- Le destinataire est-il prêt à écouter ?

  $\rightarrow$ Protocole poignée de main

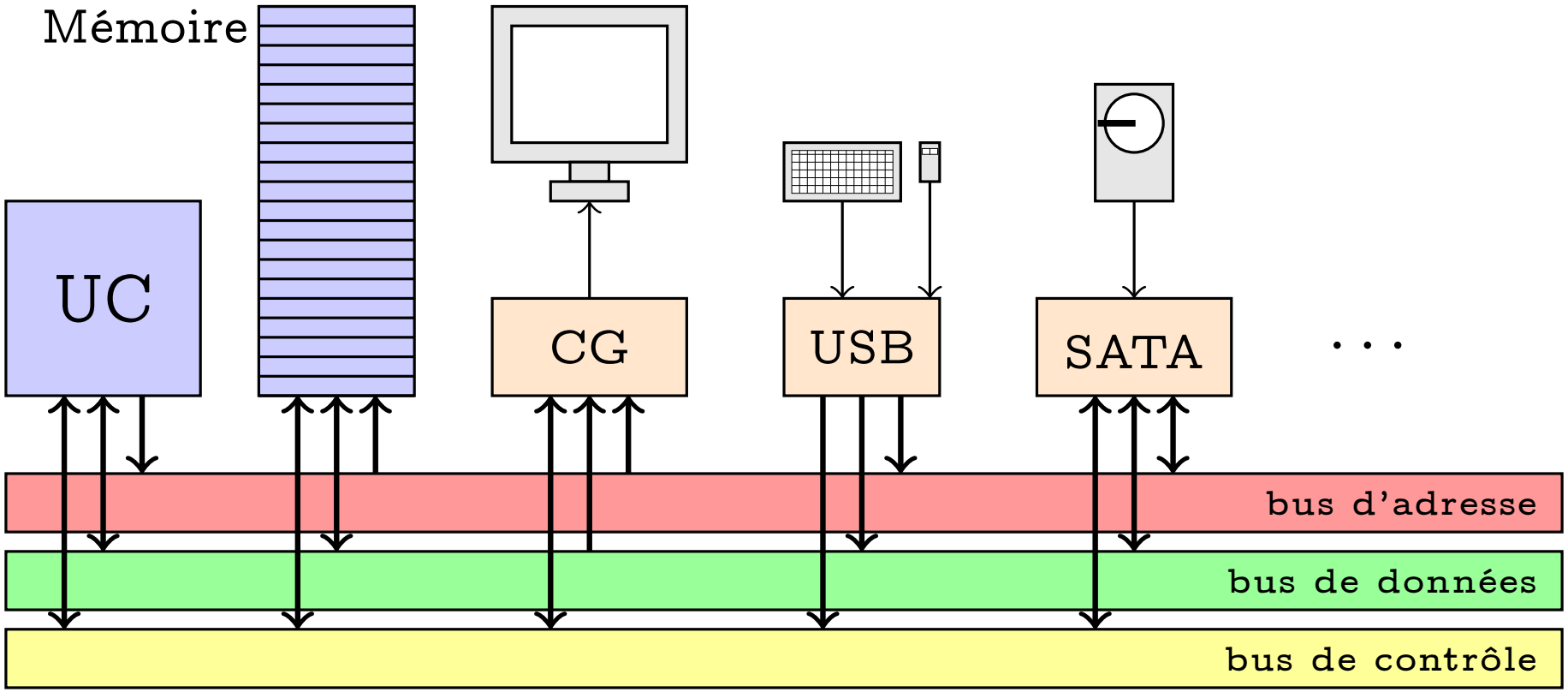
Composition du bus



Bus d'adresse

Indique avec qui l'on souhaite communiquer.

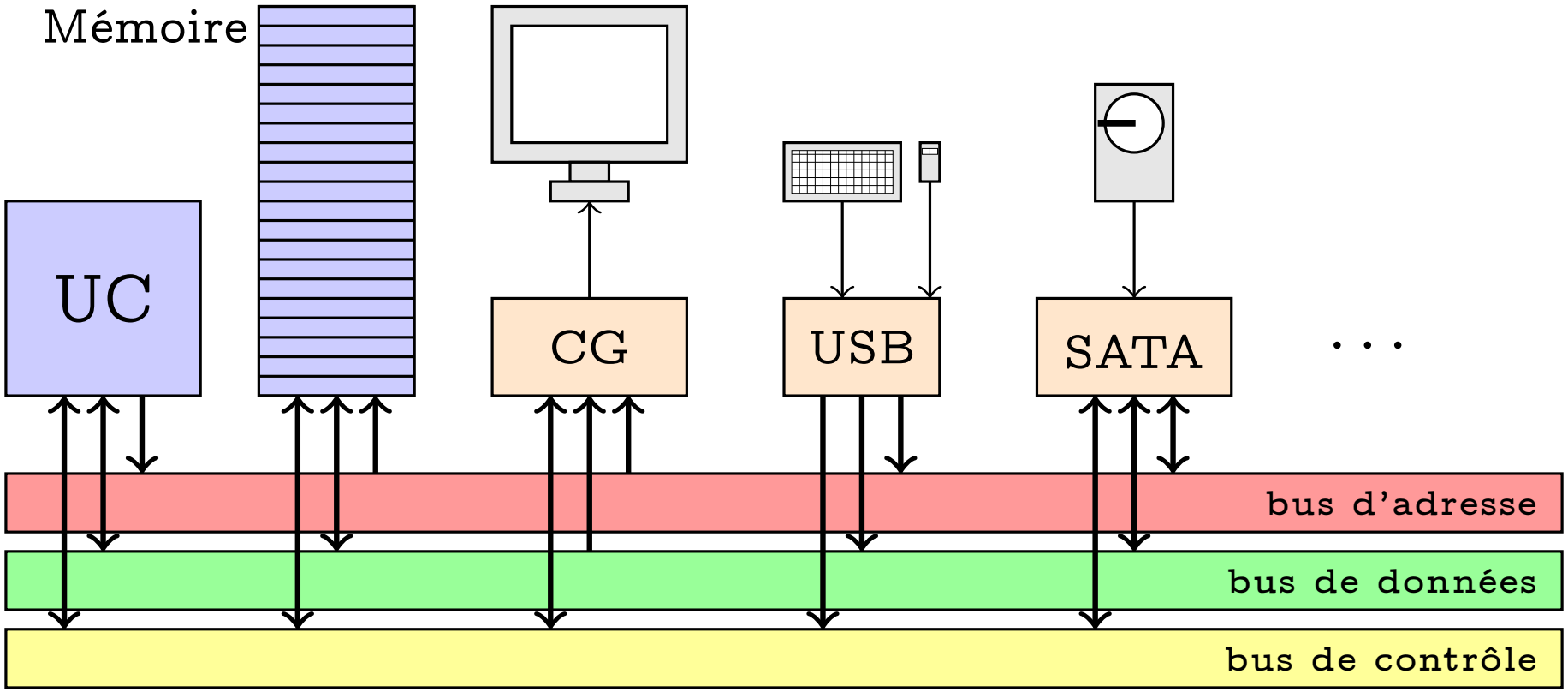
Composition du bus



Bus de contrôle

Coordonne la communication.

Composition du bus



Bus de données

Permet le transfert des informations.

Protocole poignée de main

Côté périphérique

- Communication gérée par un contrôleur
- Deux bits de contrôle : occupé et commande

Coté OS :

Côté périphérique

- Communication gérée par un contrôleur
- Deux bits de contrôle : occupé et commande

Coté OS :

tant que `occupé == 1`

attendre

Protocole poignée de main

Côté périphérique

- Communication gérée par un contrôleur
- Deux bits de contrôle : occupé et commande

Coté OS :

tant que `occupé == 1`

attendre

`commande = 1`

Côté périphérique

- Communication gérée par un contrôleur
- Deux bits de contrôle : occupé et commande

Coté OS :

```
tant que occupé == 1
    attendre
commande = 1
tant que occupé == 0
    attendre
```

Côté périphérique

- Communication gérée par un contrôleur
- Deux bits de contrôle : occupé et commande

Coté OS :

```
tant que occupé == 1
    attendre
commande = 1
tant que occupé == 0
    attendre
transfert
```

Comment prévenir l'OS que l'on a besoin de lui ?

Problème

L'OS à rarement la main, l'UC exécute les processus le plus souvent.

Comment prévenir l'OS que l'on a besoin de lui ?

Problème

L'OS à rarement la main, l'UC exécute les processus le plus souvent.

Notion d'interruption (IRQ)

Comment prévenir l'OS que l'on a besoin de lui ?

Problème

L'OS à rarement la main, l'UC exécute les processus le plus souvent.

Notion d'interruption (IRQ)

Principe

- Code envoyé sur le bus pour prévenir l'UC
- L'UC interrompt le processus courant
- Puis donne la main à l'OS

Au démarrage :

Configuration par l'OS des routines d'interruptions.

Au démarrage :

Configuration par l'OS des routines d'interruptions.

Traitement par l'UC

- Réception d'un code d'interruption
- Mise en pause du processus courant
  - Sauvegarde du pointeur de code
  - Sauvegarde des registres
- Passage en mode superviseur
- Appel de la routine d'interruption

Au démarrage :

Configuration par l'OS des routines d'interruptions.

Traitement par l'UC

- Réception d'un code d'interruption
- Mise en pause du processus courant
  - Sauvegarde du pointeur de code
  - Sauvegarde des registres
- Passage en mode superviseur
- Appel de la routine d'interruption

Exemples :

Fin d'E/S, touches clavier, paquet réseau...

Interface logicielle

Comment prévenir l'OS que l'on a besoin de lui ?

Problème

Les applications sont isolées de l'OS par sécuritée.

Comment prévenir l'OS que l'on a besoin de lui ?

Problème

Les applications sont isolées de l'OS par sécuritée.

Notion d'exception

Comment prévenir l'OS que l'on a besoin de lui ?

Problème

Les applications sont isolées de l'OS par sécuritée.

Notion d'exception

Principe

- Interruption déclenchée par le logiciel
- Passage de commande et de paramètres
- Similaire à un appel de fonction

Définition

Ensemble des commandes offerte par le système aux applications.

Définition

Ensemble des commandes offerte par le système aux applications.

Exemples

- Contrôle de processus *Création et terminaison...*
- Gestion de la mémoire *Allocation , libération...*
- Gestion des fichiers *Création, accès, droits...*
- Informations système *Date, heure...*
- ...

Comment l'OS peut-il informer les applications ?

Problème

L'OS peut avoir besoin d'une réponse rapide à n'importe quel moment.

Comment l'OS peut-il informer les applications ?

Problème

L'OS peut avoir besoin d'une réponse rapide à n'importe quel moment.

Notion de signal

Comment l'OS peut-il informer les applications ?

Problème

L'OS peut avoir besoin d'une réponse rapide à n'importe quel moment.

Notion de signal

Principe

- Définition de fonctions de réaction
- Interruption du processus et appel des fonctions

Comment l'OS peut-il informer les applications ?

Problème

L'OS peut avoir besoin d'une réponse rapide à n'importe quel moment.

Notion de signal

Principe

- Définition de fonctions de réaction
- Interruption du processus et appel des fonctions
- Très limité en pratique

Gestionnaire de signaux :

```
void fonction(int sig);
```

Gestionnaire de signaux :

```
void fonction(int sig);
```

Enregistrement :

```
signal(SIGINT, fonction);
```

Gestionnaire de signaux :

```
void fonction(int sig);
```

Enregistrement :

```
signal(SIGINT, fonction);
```

Exemples

- `SIGINT` : Arrêt demandé via Ctrl-C
- `SIGFPE` : Erreur de calcul (division par zéro...)
- `SIGSEGV` : Accès mémoire invalide
- ...

Processus

Programme

Un programme est une suite d'instructions

Programme

Un programme est une suite d'instructions

Processeur

Un processeur est un automate de traitement

- il peut exécuter un programme
- il modifie sont état en fonction des instructions

Programme

Un programme est une suite d'instructions

Processeur

Un processeur est un automate de traitement

- il peut exécuter un programme
- il modifie sont état en fonction des instructions

Processus

Un processus est un programme exécuté par un processeur

Constitution d'un processus

Instructions

Copie du code en mémoire (partie exécutable du programme)

Instructions

Copie du code en mémoire (partie exécutable du programme)

Environnement

- Variables d'environnement héritées : `PATH`...
- Descripteurs de fichiers (cf. cours 8)
- ...

Constitution d'un processus

Instructions

Copie du code en mémoire (partie exécutable du programme)

Environnement

- Variables d'environnement héritées : `PATH`...
- Descripteurs de fichiers (cf. cours 8)
- ...

Mémoire

- Pile : variable locales, contexte...
- Tas : allocation dynamique...

Création et suppression de processus

Programme $\longrightarrow$ Processus (Munir le programme des informations nécessaires pour son exécution)

Création et suppression de processus

Programme $\longrightarrow$ Processus (Munir le programme des informations nécessaires pour son exécution)

Suspension et reprise

Multiprogrammation et temps partagé

$\longrightarrow$ Interrompre et reprendre les processus

Création et suppression de processus

Programme $\longrightarrow$ Processus (Munir le programme des informations nécessaires pour son exécution)

Suspension et reprise

Multiprogrammation et temps partagé

$\longrightarrow$ Interrompre et reprendre les processus

Communication et synchronisation (cf. cours 4)

- Partage de données entre plusieurs processus
- Consistence de la mémoire

Représentation d'un processus

PCB Process Control Block

Structure de données contenant les informations relative à un processus utilisée par l'OS pour le gérer

PCB Process Control Block

Structure de données contenant les informations relative à un processus utilisée par l'OS pour le gérer

Contenu :

PCB Process Control Block

Structure de données contenant les informations relative à un processus utilisée par l'OS pour le gérer

Contenu :

- Identification (`pid` et `ppid`)
- Statut et privilèges

PCB Process Control Block

Structure de données contenant les informations relative à un processus utilisée par l'OS pour le gérer

Contenu :

- Identification (`pid` et `ppid`)
- Statut et privilèges
- Informations sur la mémoire (cf. cours 5 à 7)
- Informations d'ordonnancement (cf. cours 3)
- Réservations de périphériques

PCB Process Control Block

Structure de données contenant les informations relative à un processus utilisée par l'OS pour le gérer

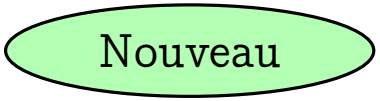
Contenu :

- Identification (`pid` et `ppid`)
- Statut et privilèges
- Informations sur la mémoire (cf. cours 5 à 7)
- Informations d'ordonnancement (cf. cours 3)
- Réservations de périphériques

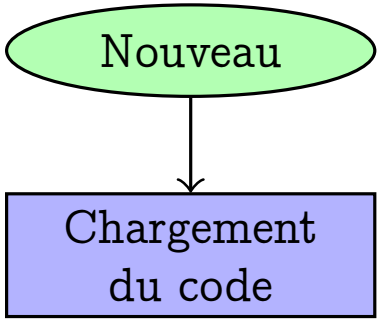
Inaccessible au processus !

Cycle de vie

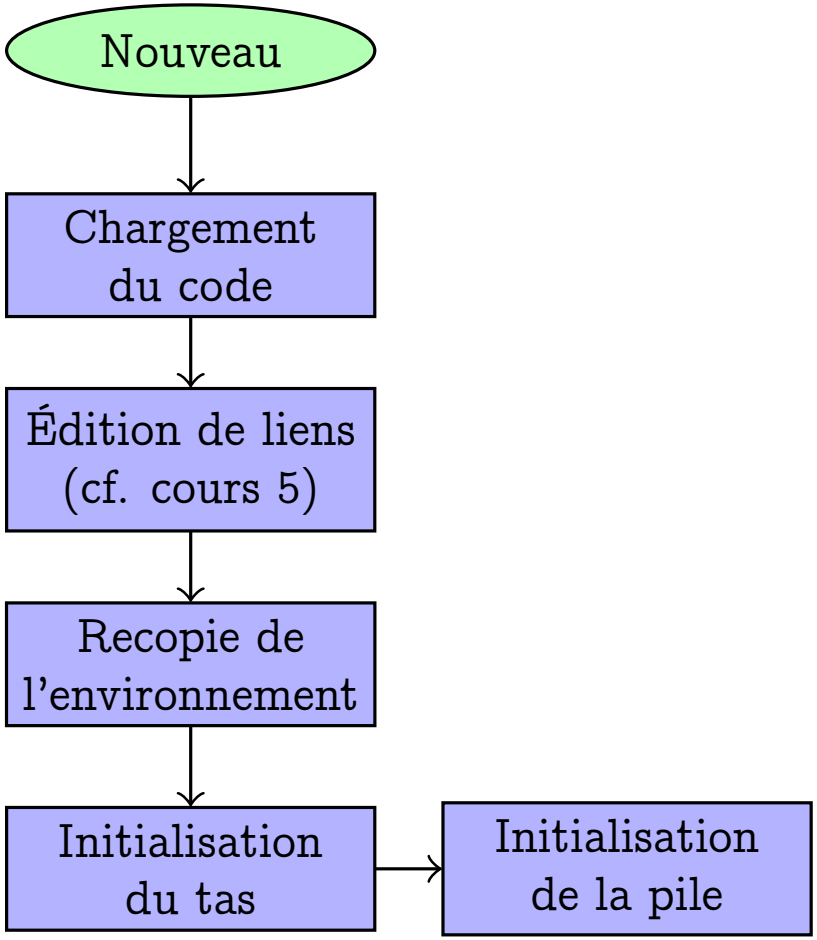
Cycle de vie du processus



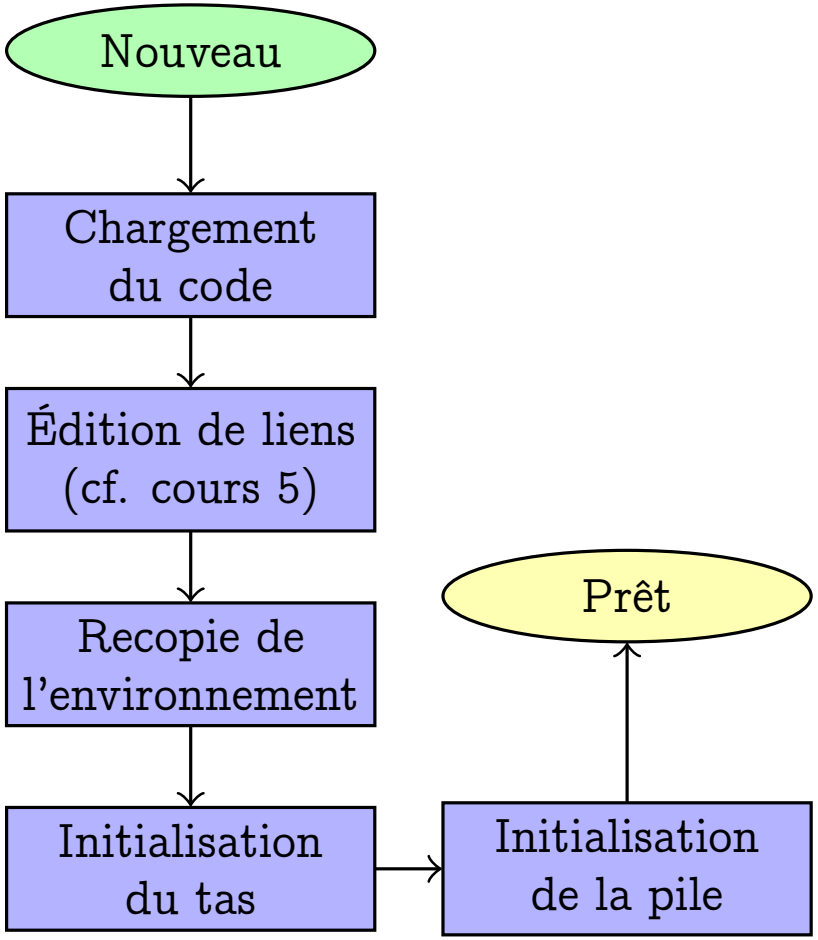
Cycle de vie du processus



Cycle de vie du processus

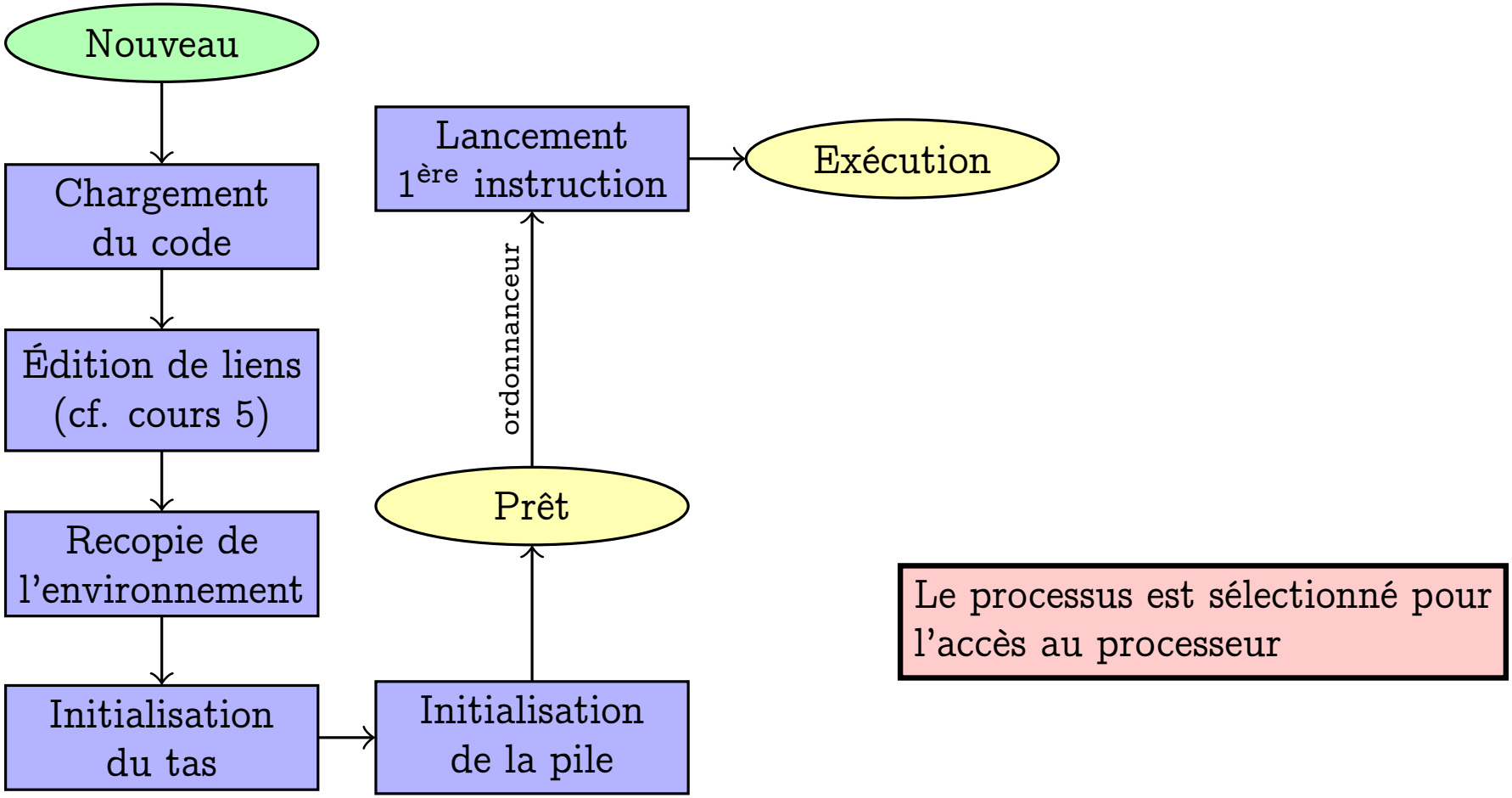


Cycle de vie du processus

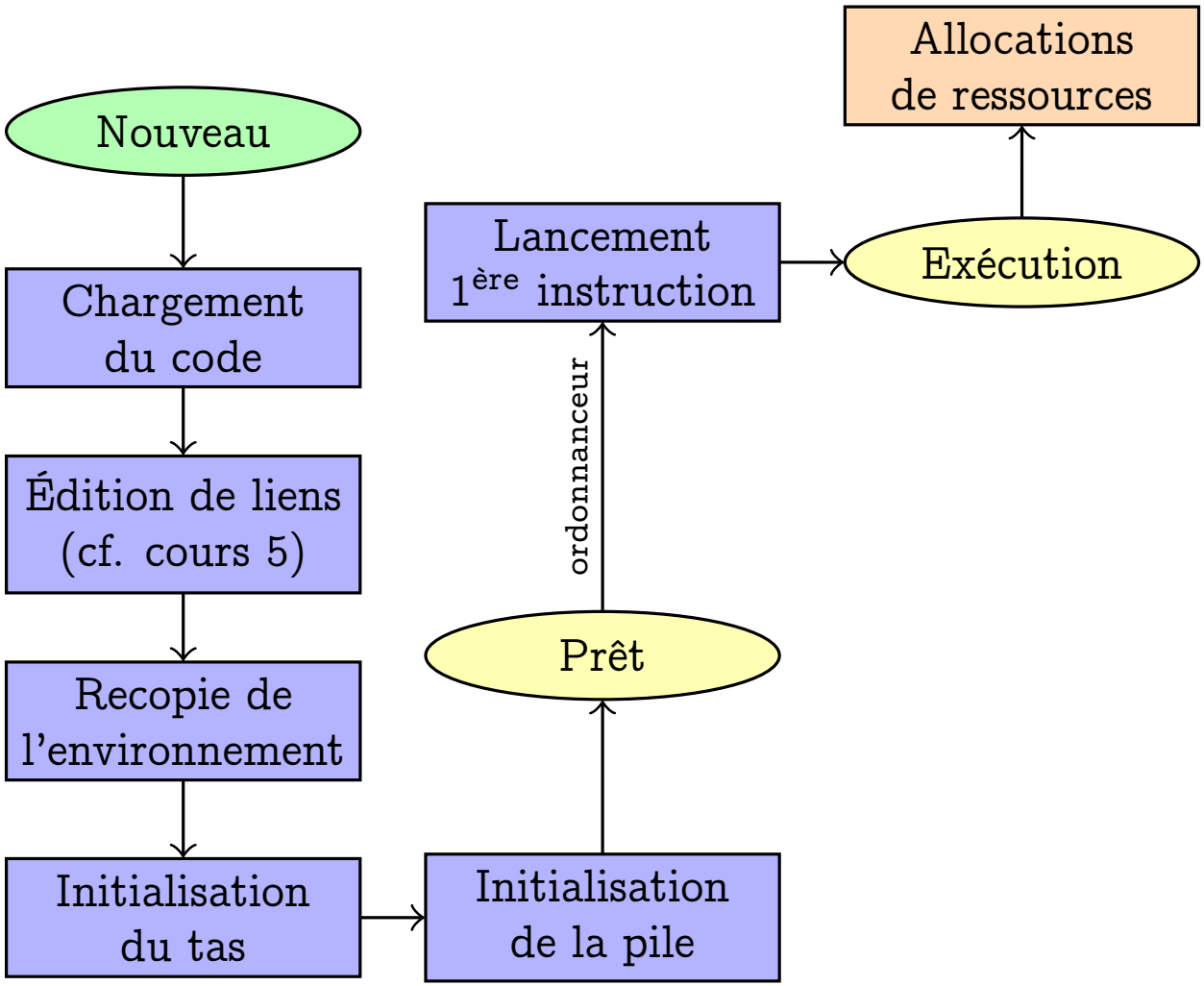


Le processus prêt est mis dans la file d'attente

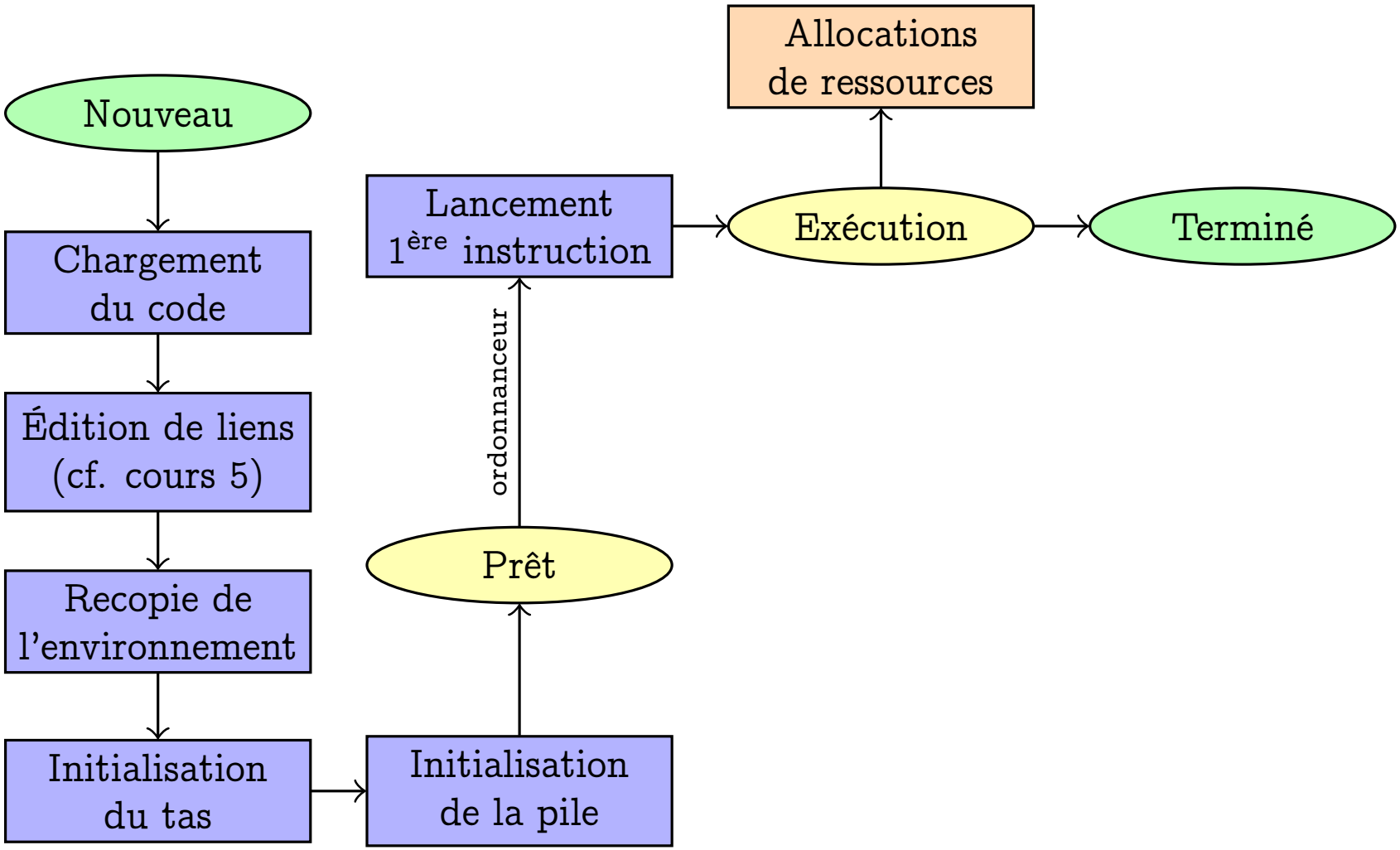
Cycle de vie du processus



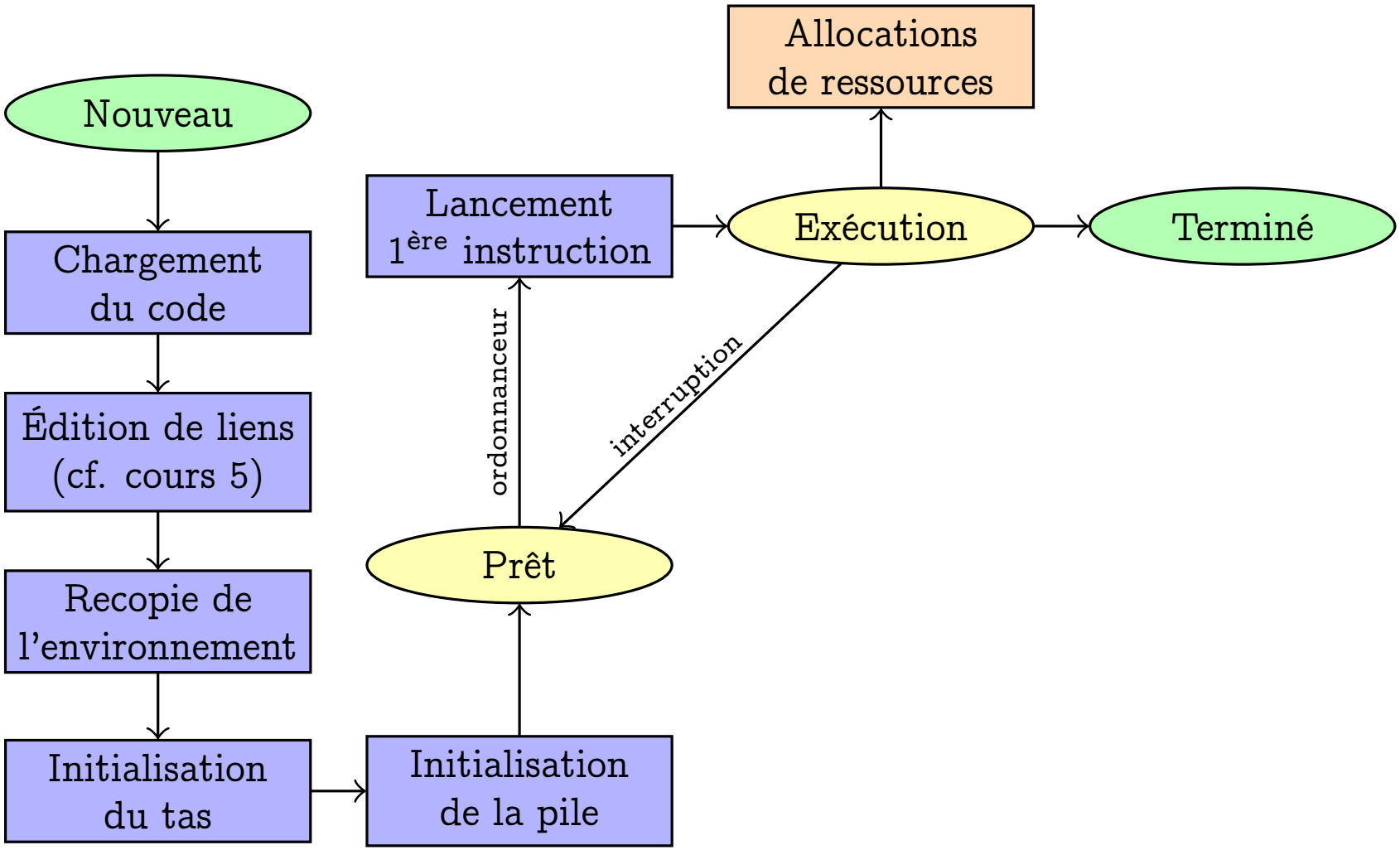
Cycle de vie du processus



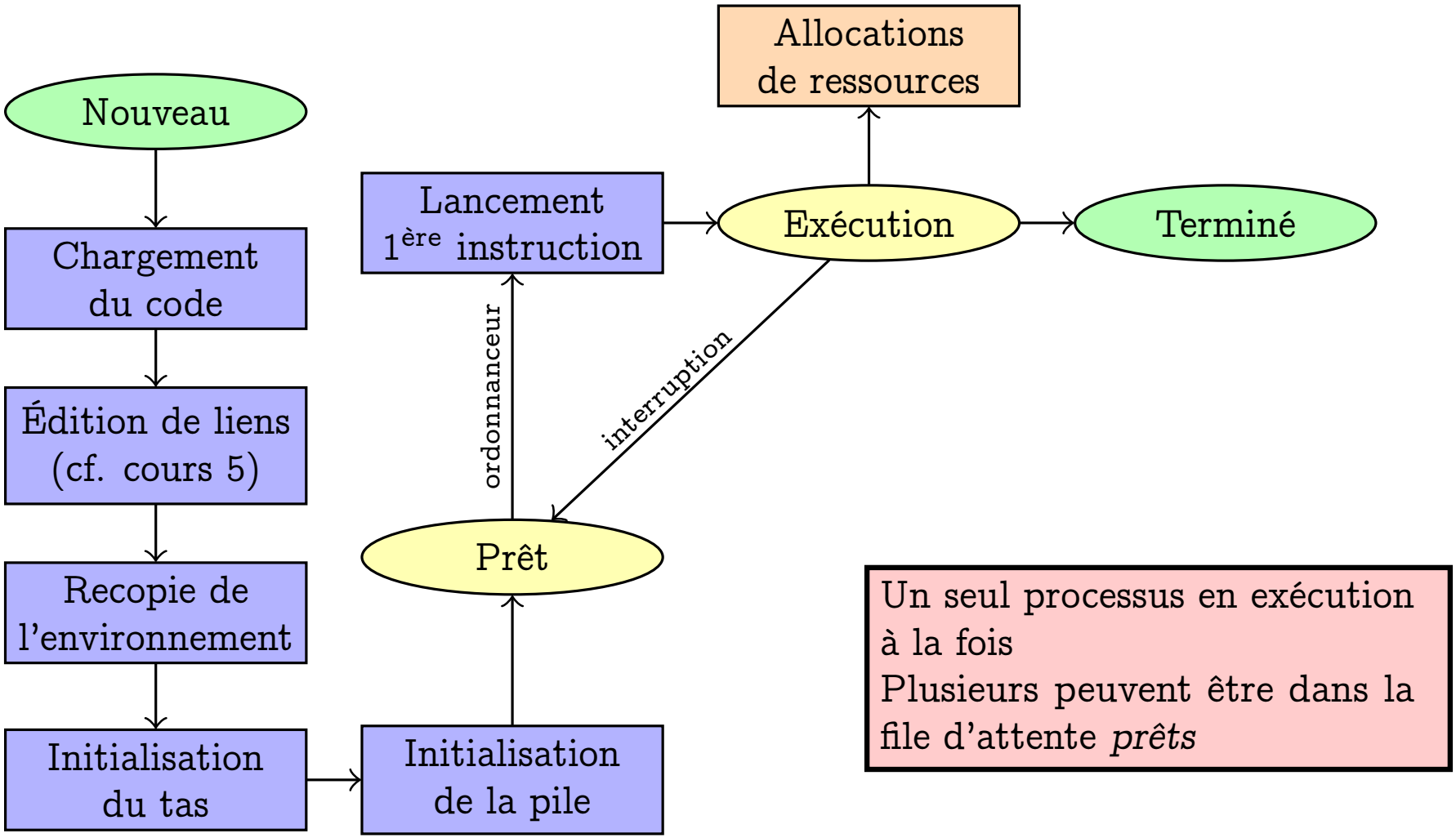
Cycle de vie du processus



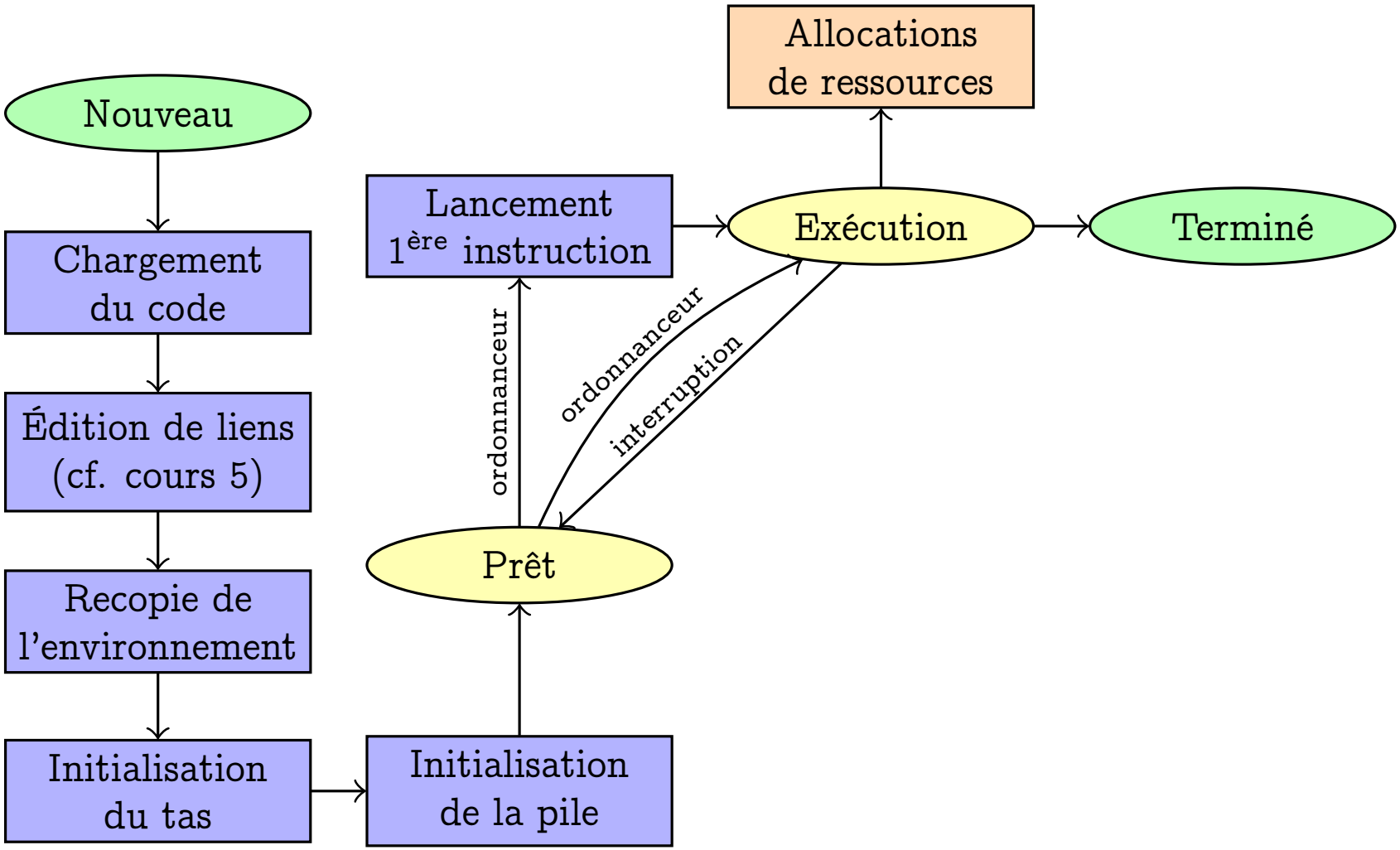
Cycle de vie du processus



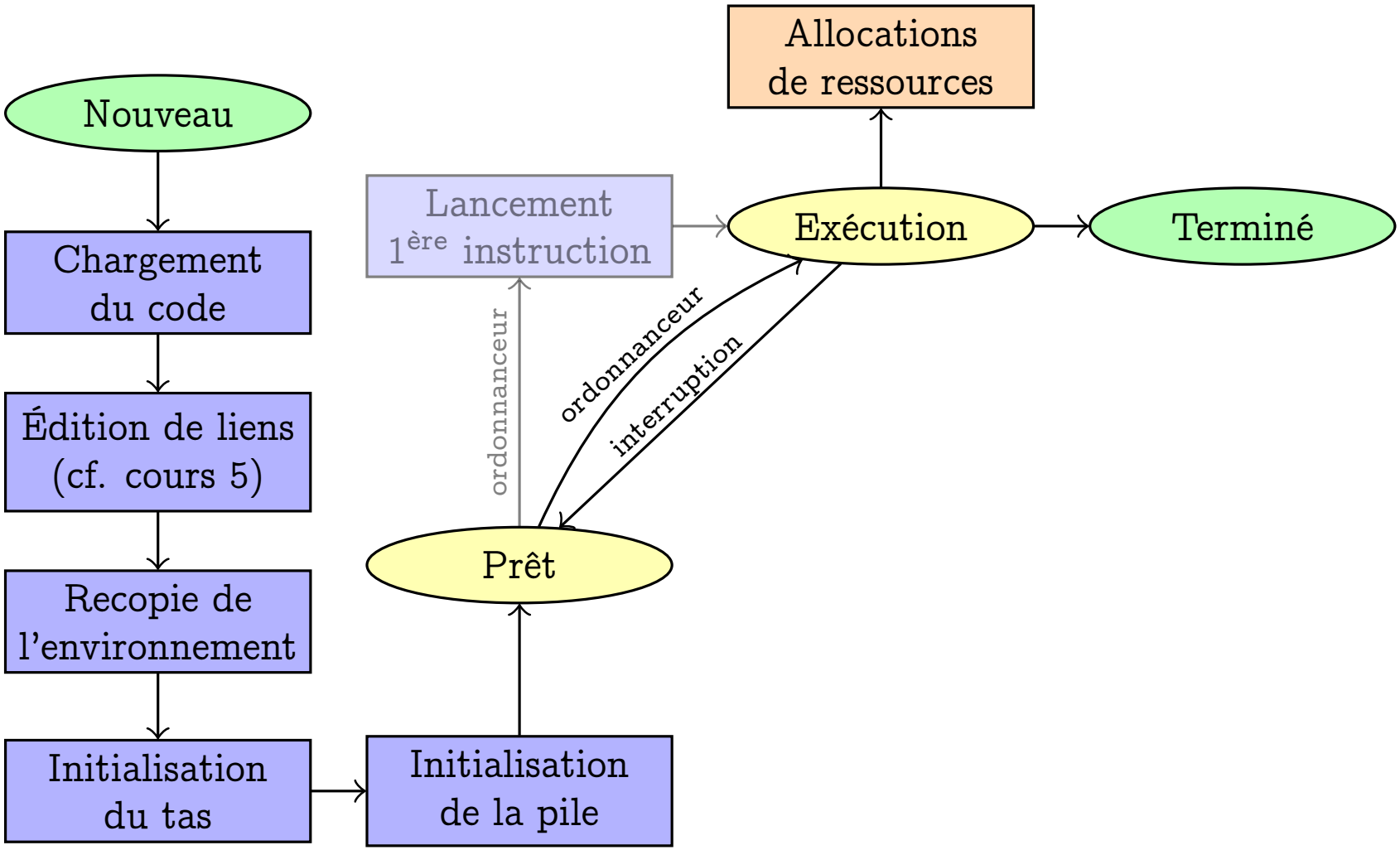
Cycle de vie du processus



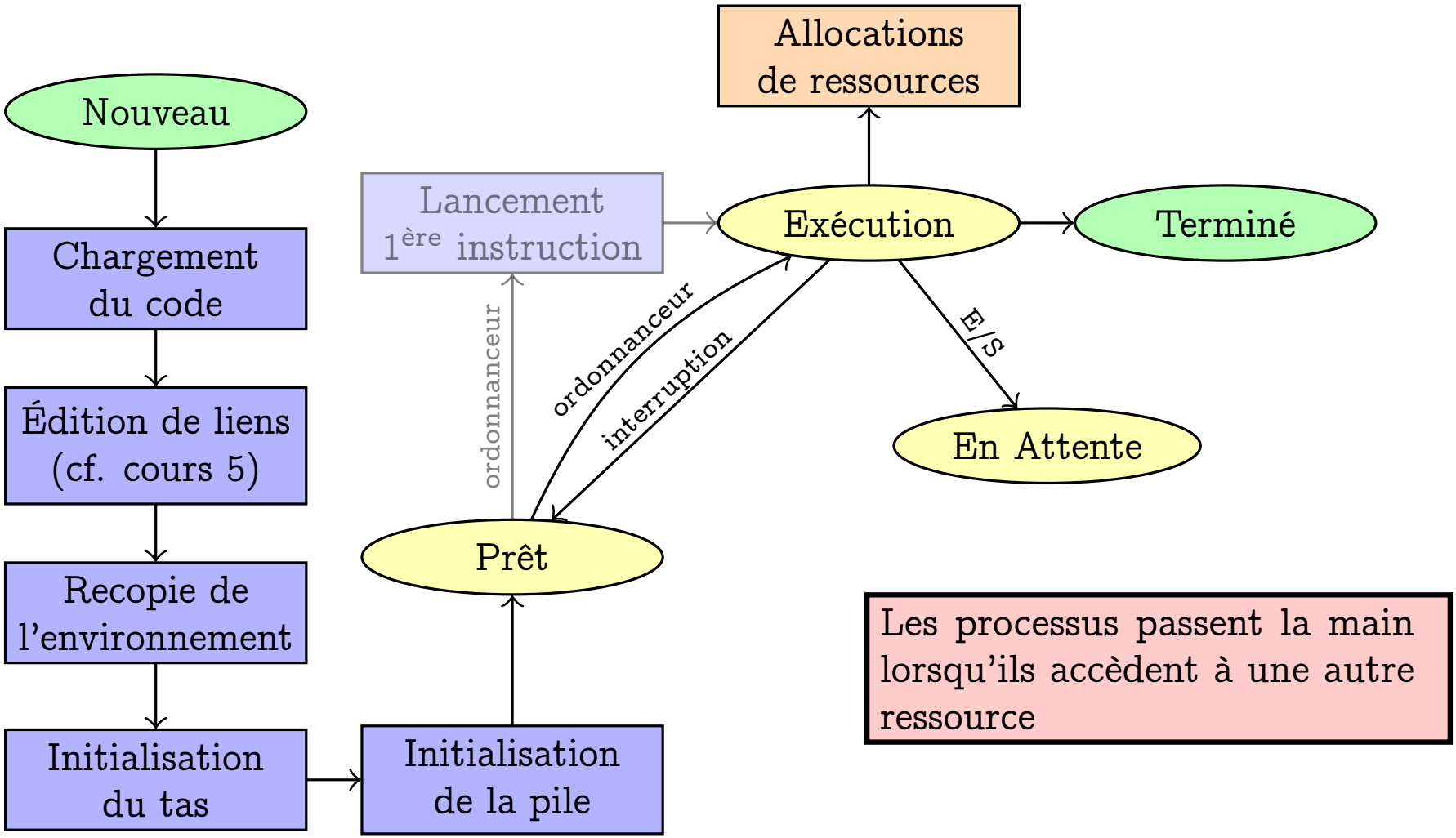
Cycle de vie du processus



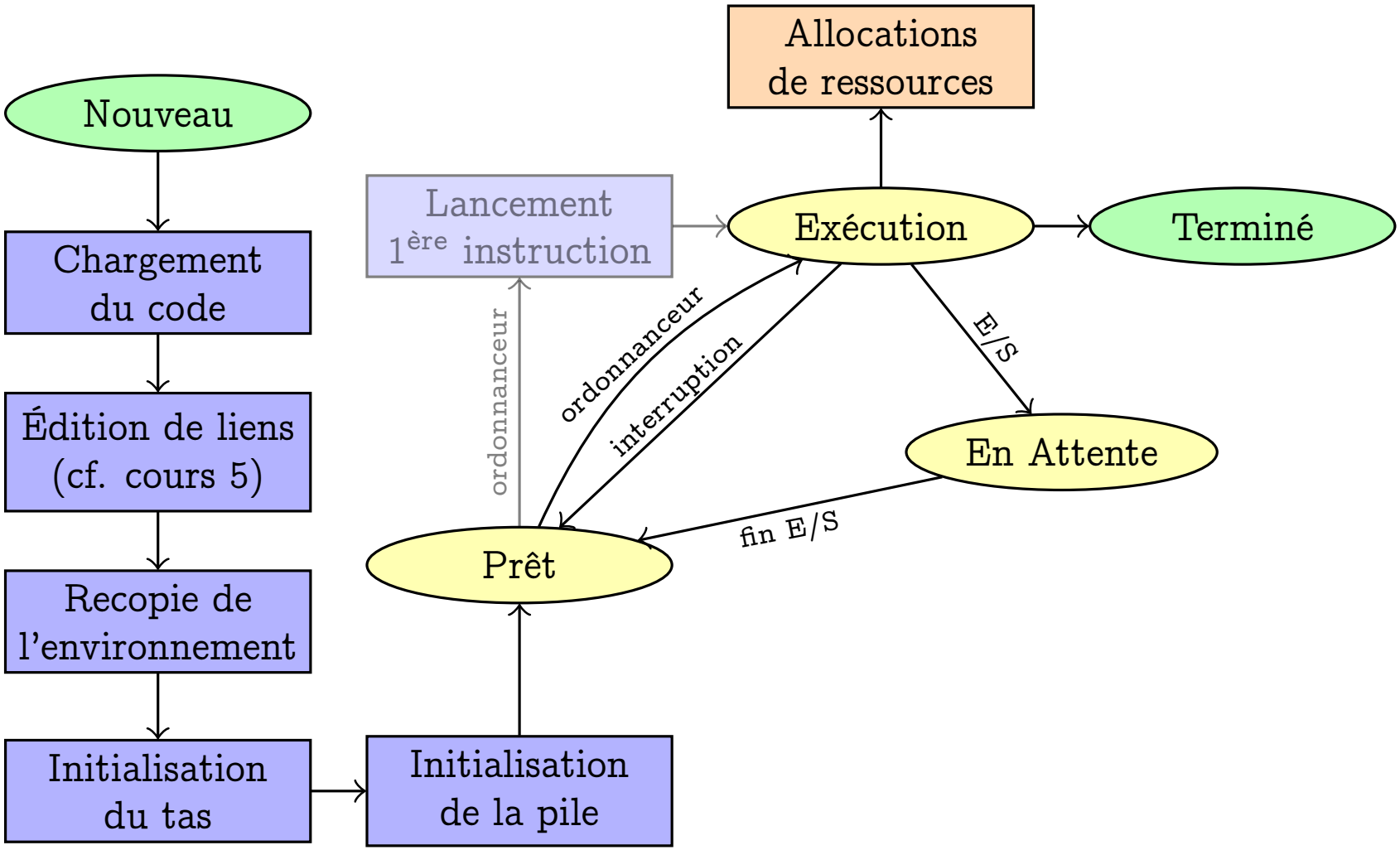
Cycle de vie du processus



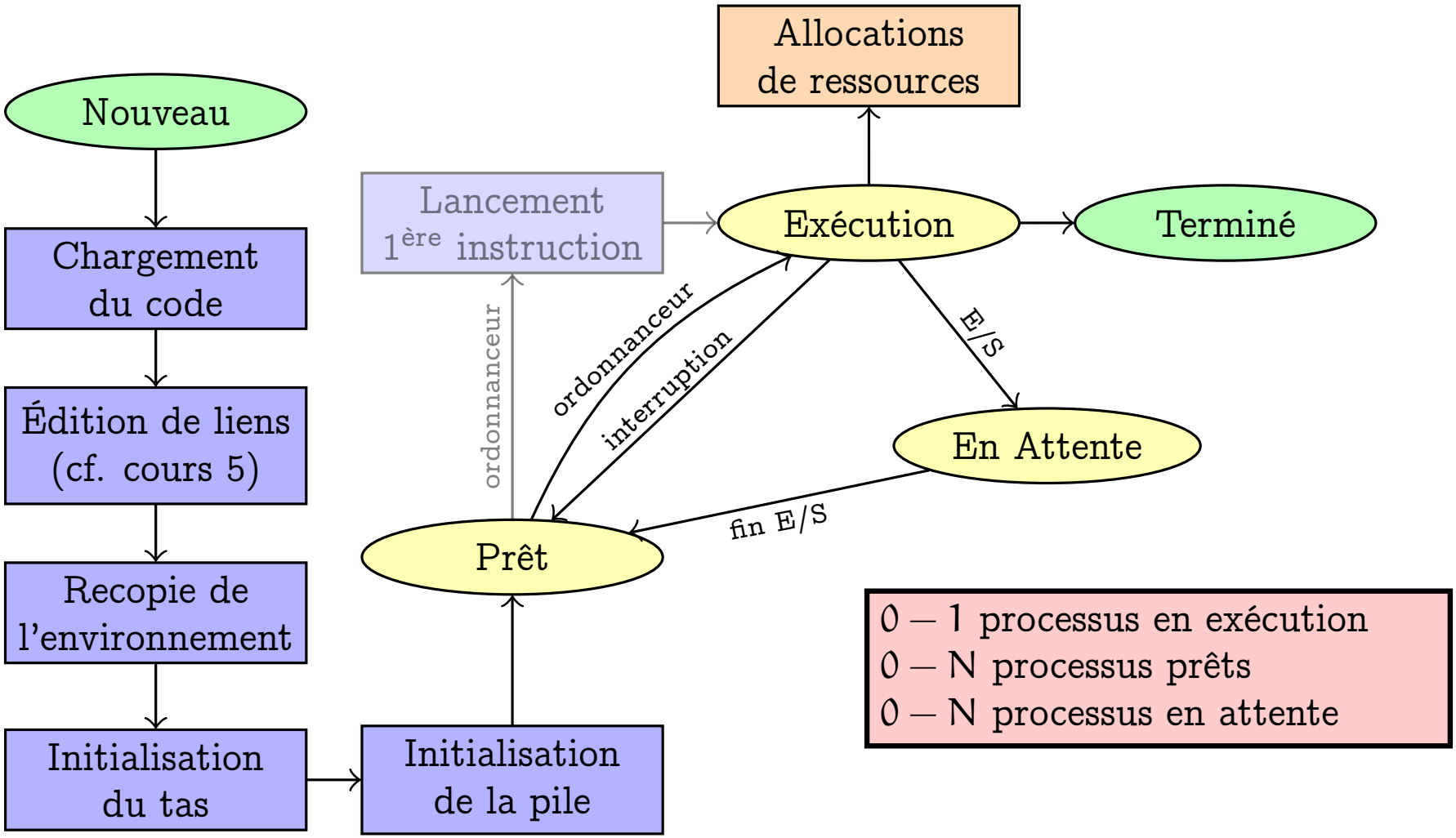
Cycle de vie du processus



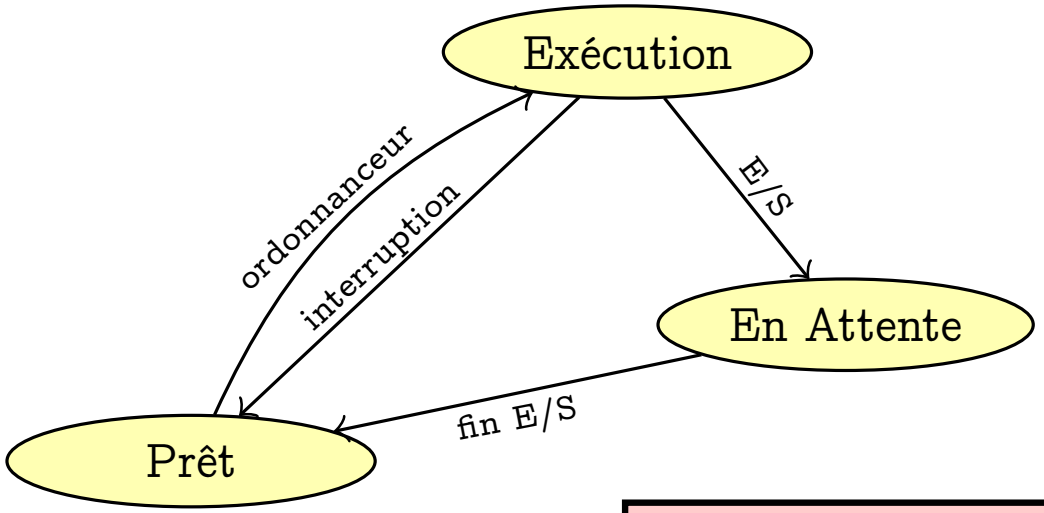
Cycle de vie du processus



Cycle de vie du processus

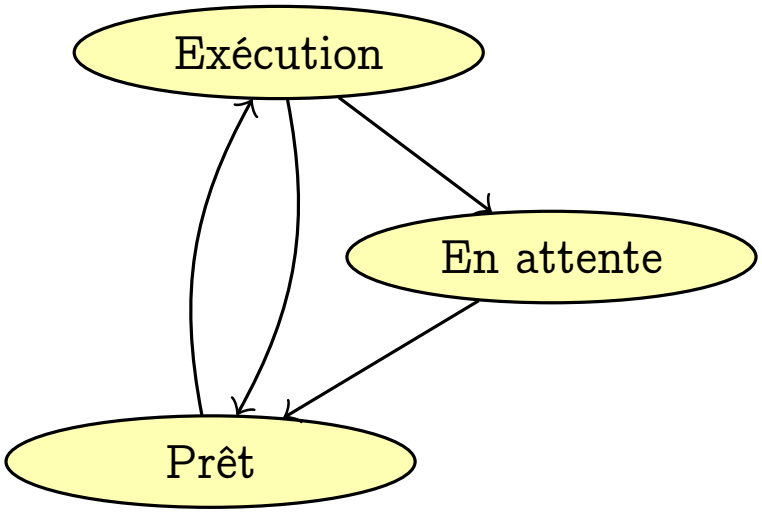


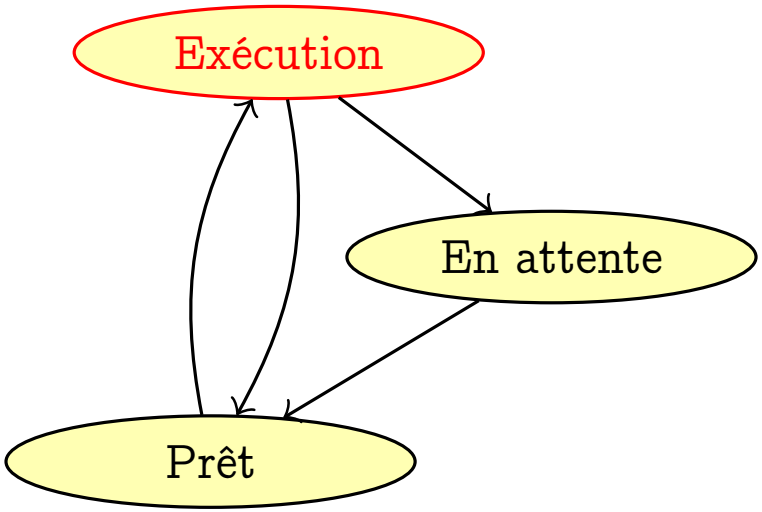
Cycle de vie du processus



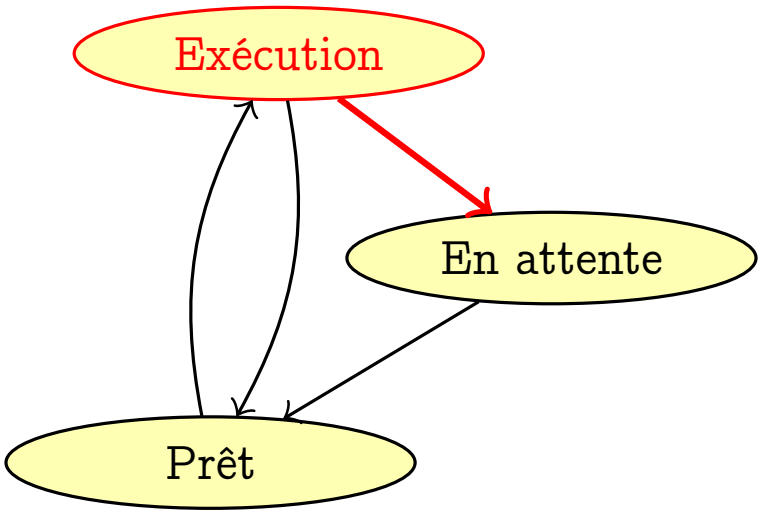
0 – 1 processus en exécution
0 – N processus prêts
0 – N processus en attente

Cycle de vie du processus



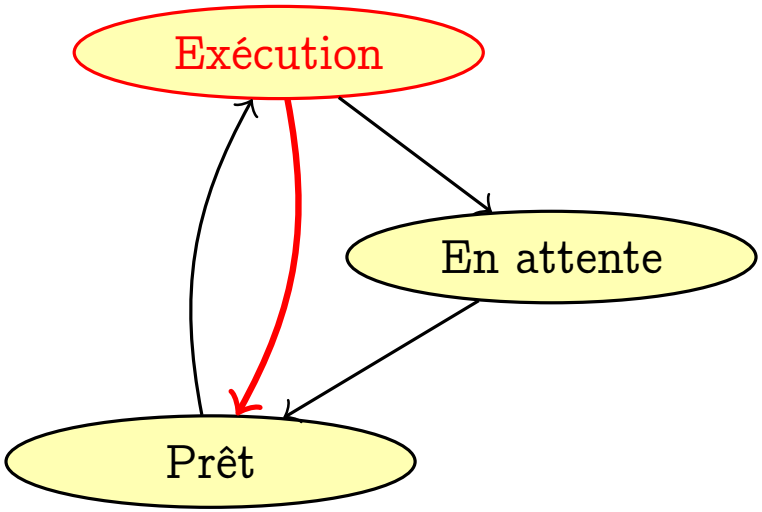


Suspension de l'exécution



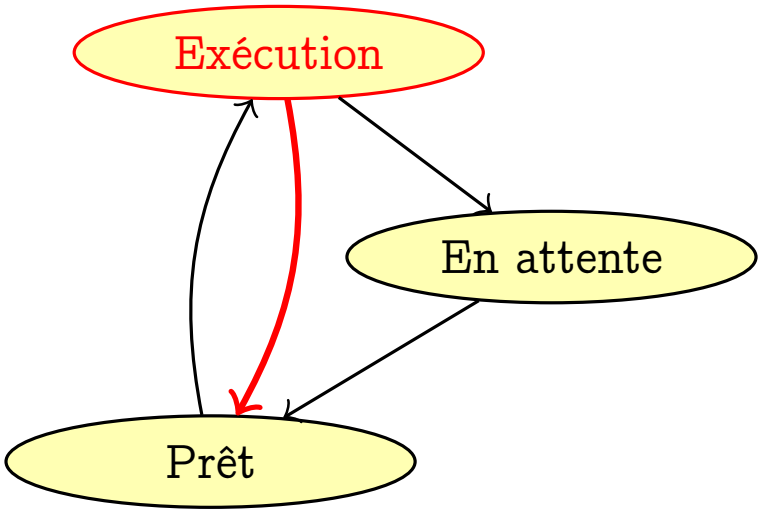
Suspension de l'exécution

- Demande d'E/S
  $\rightarrow$ File en attente



Suspension de l'exécution

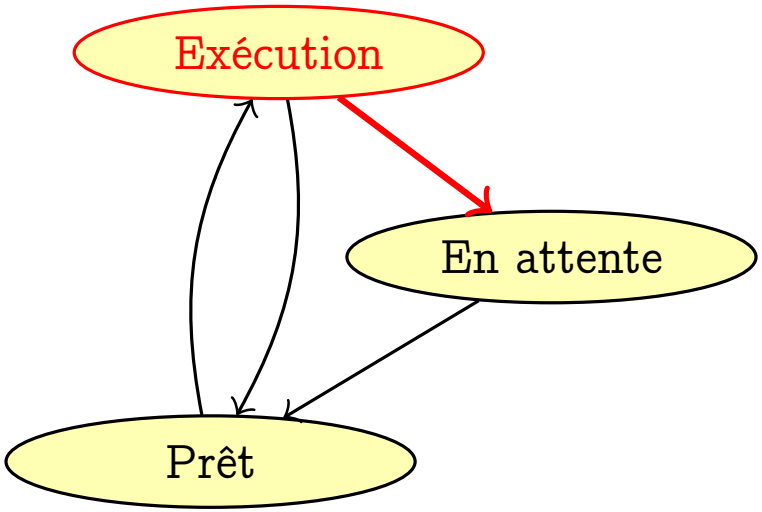
- Demande d'E/S
  $\rightarrow$ File en attente
- Temps écoulé
  $\rightarrow$ File prêt



Suspension de l'exécution

- Demande d'E/S
  $\rightarrow$ File en attente
- Temps écoulé
  $\rightarrow$ File prêt
- Crée un processus
  $\rightarrow$ File prêt

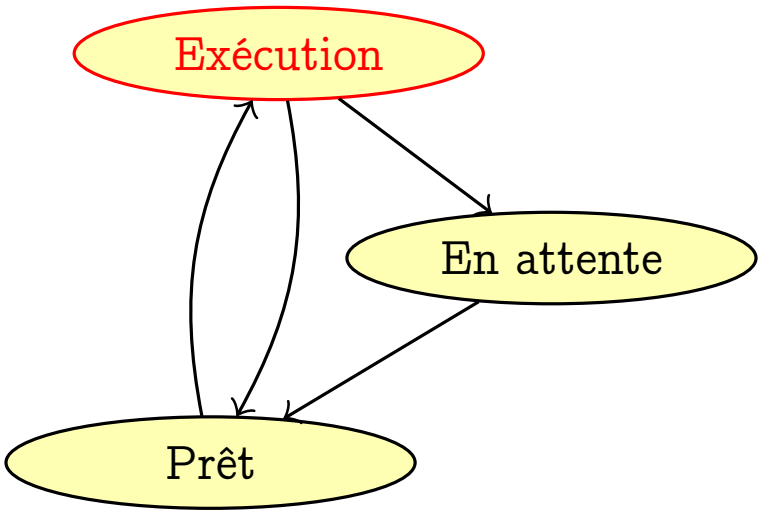
Cycle de vie du processus



Suspension de l'exécution

- Demande d'E/S
  $\rightarrow$ File en attente
- Temps écoulé
  $\rightarrow$ File prêt
- Crée un processus
  $\rightarrow$ File prêt
- Appel système *wait*
  $\rightarrow$ File en attente

Cycle de vie du processus



Suspension de l'exécution

- Demande d'E/S
  $\rightarrow$ File en attente
- Temps écoulé
  $\rightarrow$ File prêt
- Crée un processus
  $\rightarrow$ File prêt
- Appel système *wait*
  $\rightarrow$ File en attente

Interface

Création par clonage

Un processus est toujours créé par un autre processus

Création par clonage

Un processus est toujours créé par un autre processus

Appel système `fork`

Le processus est dupliqué

- Code, mémoire et environnement sont identiques dans le père et le fils
- Seule la valeur de retour, le pid et le ppid diffèrent

Création par clonage

Un processus est toujours créé par un autre processus

Appel système `fork`

Le processus est dupliqué

- Code, mémoire et environnement sont identiques dans le père et le fils
- Seule la valeur de retour, le pid et le ppid diffèrent

(En pratique)

Une technique de copie à l'écriture (COW) est utilisée par soucis d'efficacité.

Création d'un processus

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main() {
    printf("Démarrage...\n");
    return 0;
}
```

Création d'un processus

```
#include ...
int main() {
    printf("Démarrage...\n");
    return 0;
}
```

Création d'un processus

```
#include ...
int main() {
    printf("Démarrage...\n");
    pid_t proc = fork();
    return 0;
}
```

Création d'un processus

```
#include ...
int main() {
    printf("Démarrage...\n");
    pid_t proc = fork();
    if (proc == -1) {
        fprintf(stderr, "fork failed\n");
    }
    return 0;
}
```

Création d'un processus

```
#include ...
int main() {
    printf("Démarrage...\n");
    pid_t proc = fork();
    if (proc == -1) {
        fprintf(stderr, "fork failed\n");
    } else if (proc != 0) {
        /* c'est le père */
    } else {
        /* c'est le fils */
    }
    return 0;
}
```

Création d'un processus

```
#include ...
int main() {
    printf("Démarrage...\n");
    ...
    } else {
        /* c'est le fils */
        printf("%d:fils de %d \n",
                getpid(), getppid());
        sleep(5);
        printf("%d:termine \n",getpid());
        exit(5);
    }
    ...
```

Création d'un processus

```
#include ...
int main() {
    printf("Démarrage...\n");
    ...
    } else if (proc != 0) {
        /* c'est le père */
        printf("%d:père de %d \n",
                getpid(), proc);
        int r;
        waitpid(proc, &r, 0); /* attente */
        printf("%d:fils %d sort (code %d)\n",
                getpid(), proc, r);
    } else{
    ...
```

À l'exécution

À l'exécution

```
Démarrage...
```

À l'exécution

```
Démarrage...
29227:père de 29228
29228:fils de 29227
```

À l'exécution

```
Démarrage...
29227:père de 29228
29228:fils de 29227
29228:termine
```

À l'exécution

```
Démarrage...
29227:père de 29228
29228:fils de 29227
29228:termine
29227:fils 29228 sort (code 1280)
```

À l'exécution

```
Démarrage...
29227:père de 29228
29228:fils de 29227
29228:termine
29227:fils 29228 sort (code 1280)
```

1280 != 5

Utilisation des fonctions `WIFEXITED`, `WEXITSTATUS`... pour décoder la valeur de retour.

Lancement d'un programme

L'exécution d'un programme se fait par recouvrement d'un processus

Appel système `exec`

Le processus est remplacé

- Le nouveau code est chargé à la place de l'ancien
- La mémoire (pile et tas) est réinitialisée
- L'environnement est conservé

```c
#include <stdio.h>
#include <unistd.h>
int main() {
    char *arg[] = {"uname", "-a", NULL};
    execv("/bin/uname", argu);
    printf("Jamais exécuté !\n");
    return 0;
}
```

Remarques :

- Différentes variantes de `exec*` existent
- Le 1er argment est le nom du programme

Problème

Comment lancer un autre programme depuis un autre processus sans disparaitre ?

Solution : `fork` puis `exec`

```
pid_t fils = fork();
if (fils == -1)
    perror("fork failed !");
else if (fils == 0)
    execlp("uname", "uname", "-a", NULL);
/* suite du programme père */
```

Conclusion

Processus

- Un processus est un programme exécuté par un processeur
- Un processus est composé de code et de données : variables d'environnements, pile et tas
- Cycle de vie : prêt, en exécution et en attente

Conclusion

Processus

- Un processus est un programme exécuté par un processeur
- Un processus est composé de code et de données : variables d'environnements, pile et tas
- Cycle de vie : prêt, en exécution et en attente

Rôle de l'OS

- L'OS stocke les informations sur les processus dans des PCB
- L'OS assure la consistence de la mémoire
- Création de processus via `fork` et `exec`