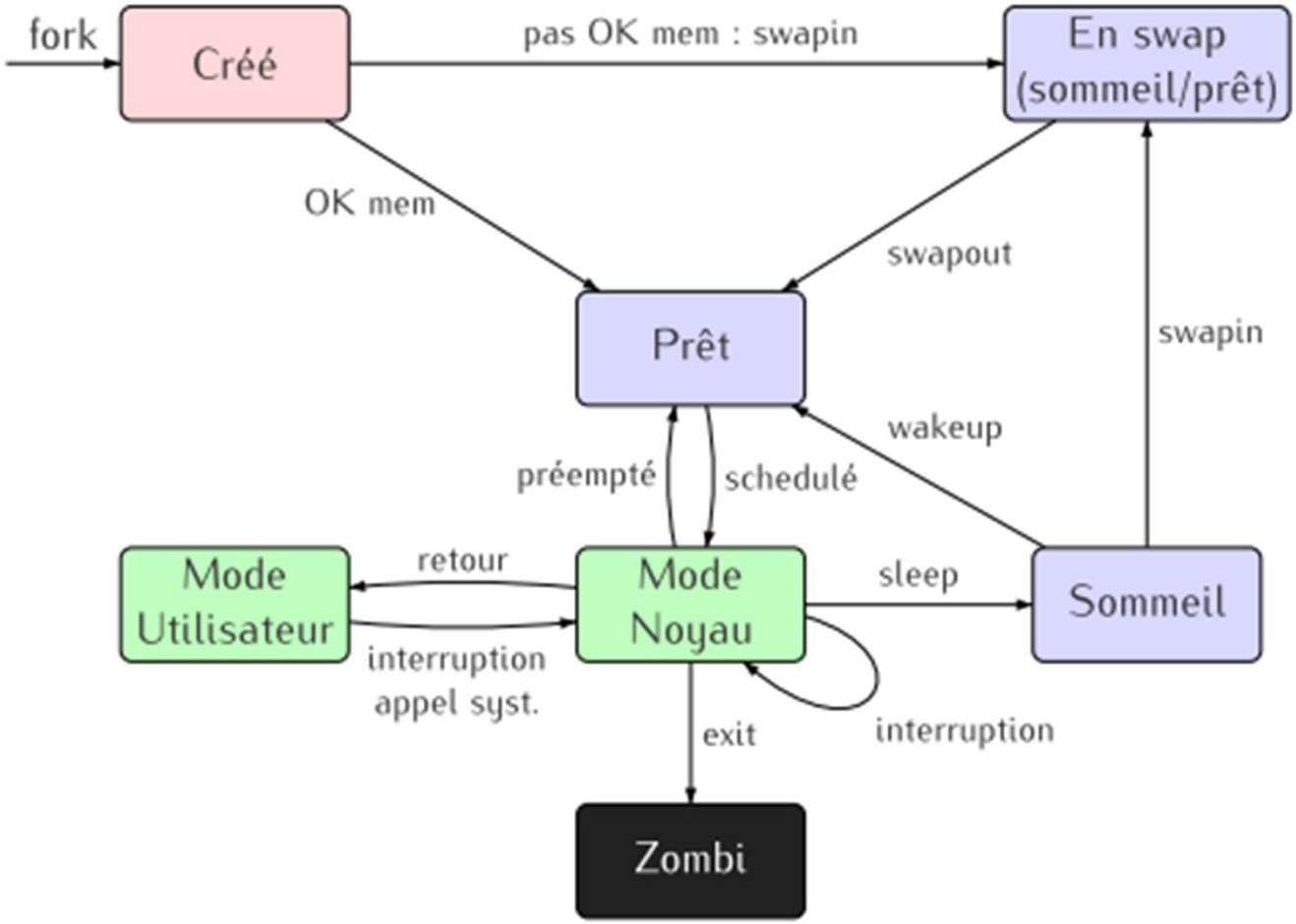
Processus

Caractéristiques des processus

- Un processus est un programme en cours d'exécution. Chaque processus est caractérisé par :
  - son identité (pid),
  - son état (prêt, mode kernel/user, endormi, zombi, créé, swappé),
  - ses propriétaires (réel, effectif),
  - son répertoire courant, etc.
  - les ressources qui lui ont été attribuées, par exemple les fichiers ouverts ;
  - ses demandes non encore satisfaites.

Etats de processus



Identification

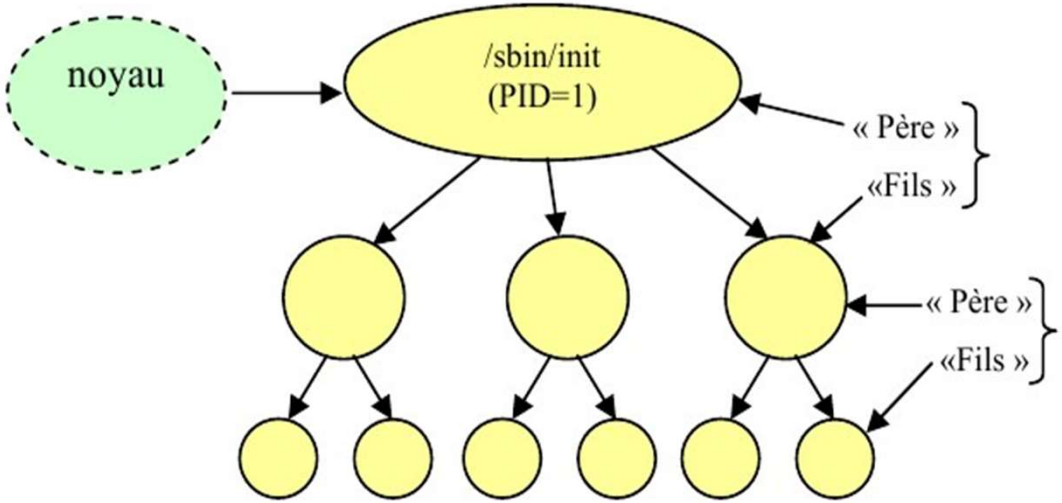
- Chaque processus est identifié par un numéro entier unique, son **pid**.
- Deux processus ne peuvent pas avoir le même numéro.
- Chaque processus a un processus père : celui qui a demandé sa création.
- Tout processus orphelin est adopté par le processus **init**, de pid 1.
- Un processus récupère son identificateur et celui de son père par :

  pid_t **getpid** (void)

  pid_t **getppid**(void)

Création

- Création des processus avec l'appel système **fork()** (unistd.h).
  - le processus créé reprend son exécution au même point que le programme appelant ;
  - retourne **-1** en cas d'erreur ;
  - retourne **0** dans le processus fils ;
  - retourne le **PID** du fils dans le processus père.



Recouvrement

- Un processus exécute un programme exécutable par l'une des 6 fonctions

exec(l/v)(-/p/e).

- Le processus « repart à 0 » avec le texte d'un autre programme.
- Il garde son pid, ppid, son propriétaire et groupe réel, son répertoire courant, son umask, ses signaux pendants, …
- Les fonctions execl* spécifient les arguments comme une liste.
- Les fonctions execv* spécifient les arguments comme un vecteur.
- Dans les 2 cas, on spécifie :
  - 1. le fichier binaire à exécuter,
  - 2. les arguments, sous forme liste ou vecteur.

Terminaison

- Lorsqu'un processus se termine, il passe à l'état « zombi ».
  - Si son père est 1, il est retiré de la liste des processus.
  - Sinon, il reste zombi (et consomme des ressources internes) jusqu'à ce que son père le libère.
- Le père le libère en récupérant des informations sur la terminaison par
  - pid_t **wait**(int *status);
  - pid_t **waitpid**(pid_t pid , int *status, int option)
- L'appel wait
  - retourne immédiatement si le processus n'a pas de fils, ou a déjà un fils zombi, ;
  - Sinon, l'appel est bloquant ;
  - Au retour de l'appel wait, l'entier pointé par status contient des informations sur la terminaison du fils.

Terminaison

- Un processus se termine normalement en appelant exit, _exit ou return dans le 1er cadre de la fonction main.
- La fonction exit()
  - 1. appelle les fonctions enregistrées par atexit,
  - 2. vide les buffers de la bibliothèque standard.
  - 3. continue comme _exit().
- La fonction _exit()
  - 1. ferme les descripteurs,
  - 2. termine le processus.
- Un processus peut se terminer anormalement, sur réception d'un signal.