BUT 2 / R3.05
PROGRAMMATION SYSTÈME

PROCESSUS

APARTÉ

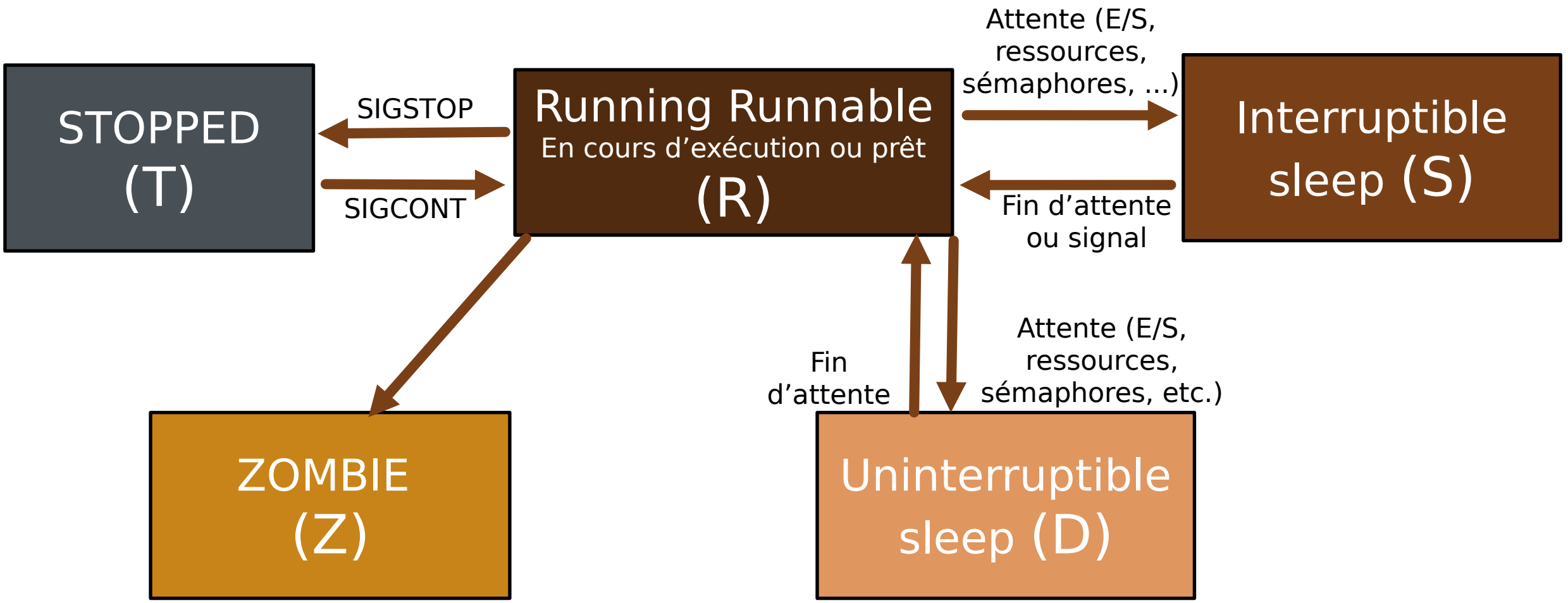
- Les appels systèmes (API système) sous Linux sont :
  - SYSTEM V (5): implémentation historique du noyau Linux
  - **ou**
  - POSIX: norme IEEE
- Sous windows
  - La plupart des fonctions POSIX sont supportées pour rendre les codes compatibles Linux/Windows
  - L’API système est la WIN-API (anciennement WIN-32)

DEFINITION

- Un processus est un programme exécutable en cours d’exécution.
- Il utilise des ressources systèmes et matériels
  - Matériel
    - Entrée / sortie
    - Processeur
    - Mémoire
  - Systèmes
    - Appels systèmes (liés aux matériels ou logiciels par ex. sémaphore)
    - Gestion des processus par le SE

PROCESSUS SOUS LINUX

ETAT D'UN PROCESSUS SOUS LINUX



INFORMATION NOYAU D'UN PROCESSUS

- Le système gère un certains nombre d'informations sur chaque processus
  - PID (Processus ID), PPID (Parent Processus ID), PGID (Parent Group ID), UID (User ID), GID (Group ID)
  - La table des descripteurs de fichiers
  - Information sur les signaux, etc.
  - Les informations sur son exécution: pile, registre, état/emplacement mémoire, etc.

FILIATION SOUS LINUX

- Un processus est créé par un autre processus
  - Les appels systèmes sont `fork()` ou `clone()`
  - fork() renvoie:
    - 0 dans le processus fils
    - Le pid du fils dans le père
- Cela créer une filiation père/fils (parent/child)

```
Prototype:
pid_t fork(void);
```

FILIATION SOUS LINUX

- La valeur retournée par le fils est à destination du processus père.
- Le père peut la récupérer avec l'appel système `wait()`ou `waitpid()`.

```
int main()
{
   pid_t retourFork; int retourFils;

   retourFork=fork();
   switch(retourFork)
   {
      case -1: perror(« Error fork() »);
               break;
      case 0: printf(« Je suis le fils\n »);
              return(24);
       default: printf(« Je suis le père\n »);
              wait(&retourFils);
              printf(« La valeur est %d\n »,WEXITSTATUS(retourFils));
              break;
   }
   return(0);
}
```

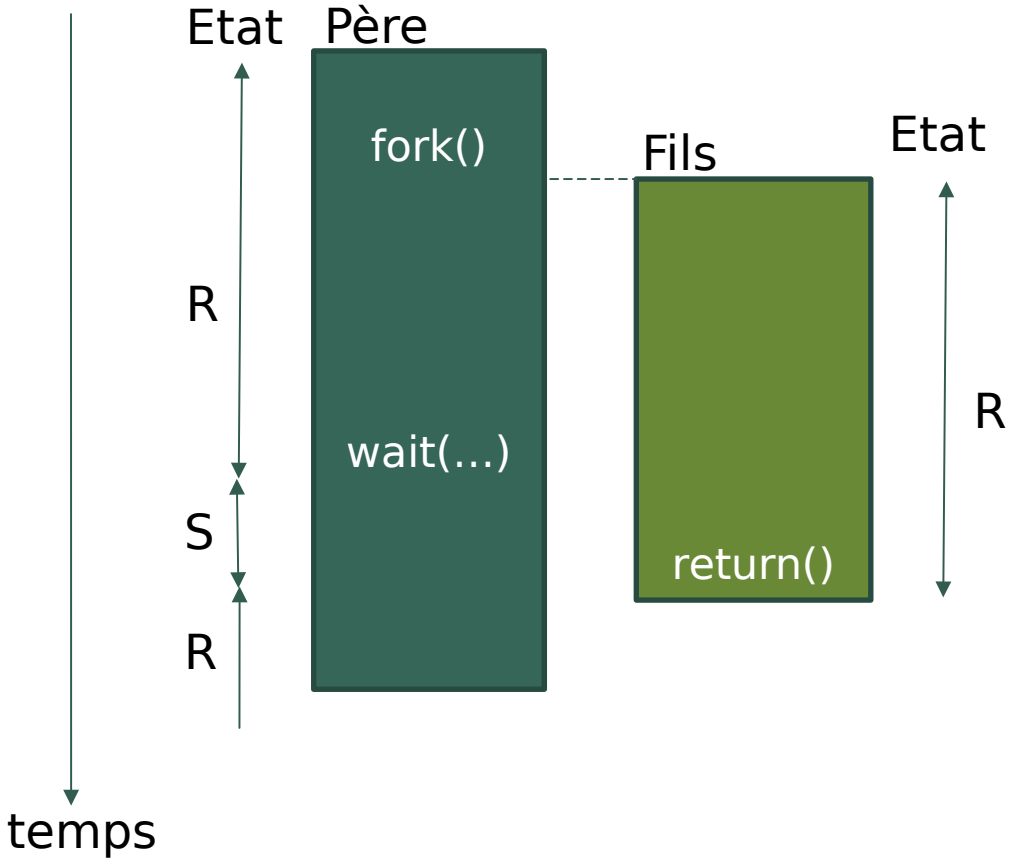
Création d'un nouveau processus. Deux processus exécutent en parallèle le code qui suit.

Le père attend jusqu'à la terminaison du fils.

Le code de retour est chiffré sur le dernier octet. Les octets de poids forts contiennent d'autres informations sur l'état du processus fils.

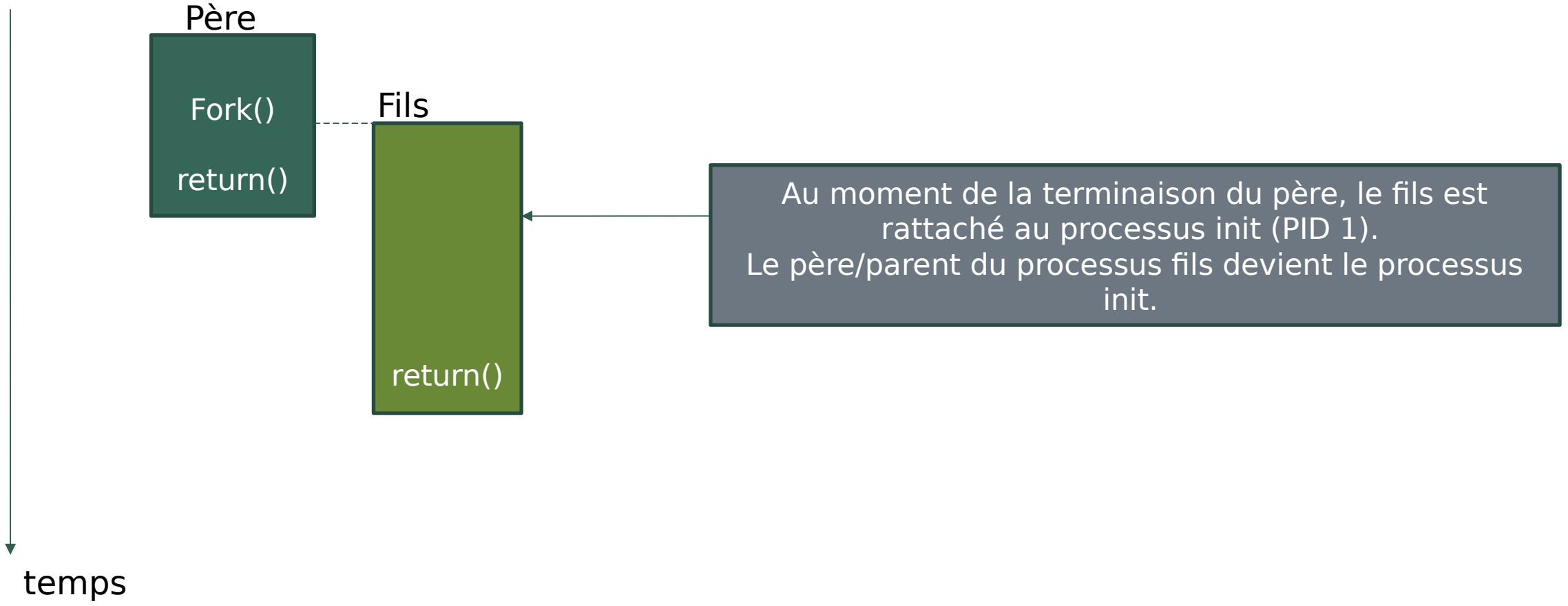
FILIATION SOUS LINUX: SCENARIO PÈRE/FILS

- Cas 1: le père effectue sont wait() avant la terminaison du fils



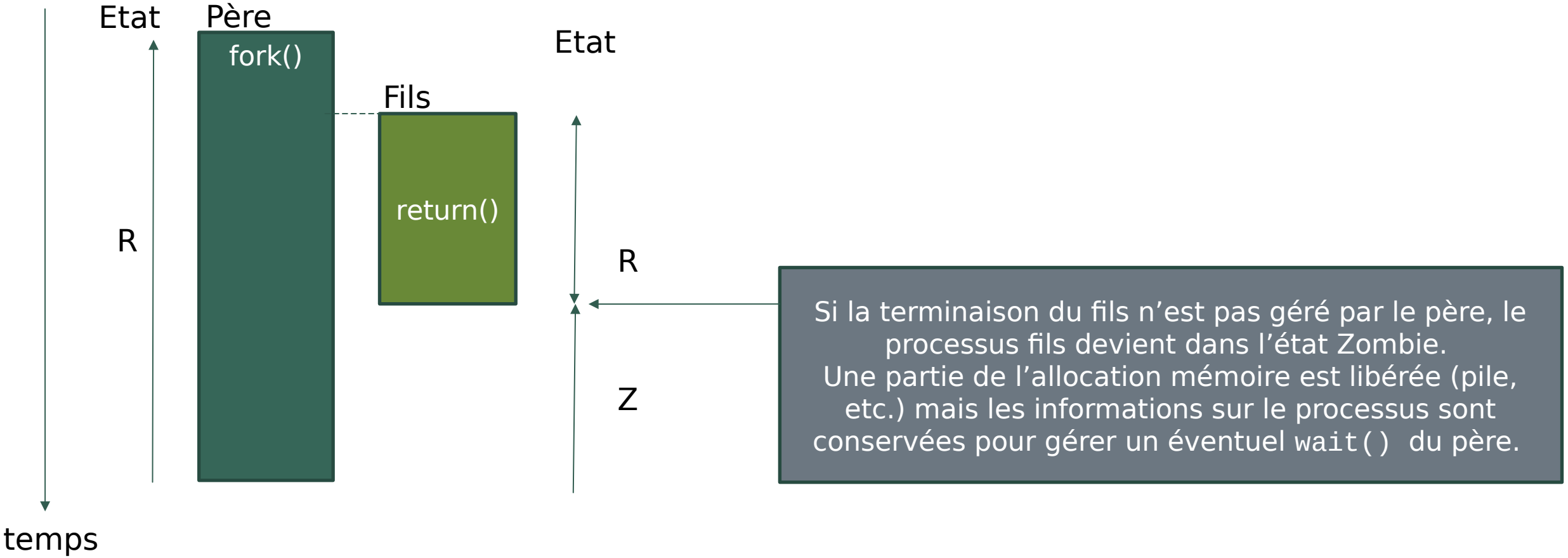
FILIATION SOUS LINUX: SCENARIO PÈRE/FILS (2)

- Cas 2: le père se termine avant son fils



FILIATION SOUS LINUX: SCENARIO PÈRE/FILS (3)

- Cas 3: le père s'exécute sans s'occuper de la terminaison du fils



QUESTIONS

- Qu’est ce que POSIX?
  - Des fonctions (codes) fournit par Linux et potentiellement windows qui fournissent des services (fonction système)
  - Une norme qui définit les prototypes des fonctions et ce qu’elles font sans fixer leur implémentation.
  - Les appels historiques d’Unix
  - Le service postal de polytechnique
- Quelle est la sortie du code ci-contre
  - Toto
  - Toto Toto
  - Toto Toto Toto
  - Toto Toto Toto Toto
  - Aucun affichage
- Y-a-t-il un zombie avec le code ci-contre?

```
int main()
{
  fork(); fork();
  printf(« Toto »);
  return(0);
}
```

```
int main()
{
  if(fork()>0) return(0);
  return(0);
}
```