Tout processus est issu du clonage d'un autre processus grâce à l'appel système **fork().**

Le processus créé (dit **fils**) est en tout point similaire au processus ayant fait l'appel à fork() (dit **père**) mis à part quelques différences dont :
- le **PID** : un nouveau PID est affecté au fils parmi les valeurs ne correspondant pas un processus vivant
- le **PPID** : celui du fils devient le PID du père
- la **valeur de retour** de l'appel système fork() : 0 pour le fils et PID du fils pour le père

L'appel à **fork()** ne permet pas de lancer un autre exécutable :
- le fils exécute le même code que le père
- puisque le registre RIP (pointeur d'instruction) est dupliqué, le fils "démarre" à l'instruction de retour du fork()

Les appels système de la famille des **execX()** (execv(), execl(), ...) permettent de changer le code qui s'exécute :
- les données du kernel sur le processus sont conservées (PID, PPID, UID, GID, ...)
- les variables d'environnement sont préservées
- tout le reste de la mémoire est réinitialisé avec les variables du nouveau programme
- la pile correspond au `main()` du nouveau programme avec comme `argv[]` les paramètres d'appel passé dans l'appel système `exec()`
- le registre RIP (pointeur d'instruction) est placé sur la première instruction du nouveau programme

Attention : l'appel système ne créé pas de nouveau processus et comme tout est perdu impossible de revenir dans le code qui contenait le `exec()`