Pour le moment, la **libelf** est composée de 3 couches :

- la couche *elf* qui charge des objets elf (applications/shared libraries) en mémoire. Le descripteur d'un objet elf est : `struct elf_object_s;`
- la couche *rtld* qui est capable de charger une chaine d'objets elf (applications et ses dépendances) et de les reloger en mémoire.
- la couche *tls* qui supporte que les applications dispose d'une zone mémoire perso (alors que le code/data des apps/libs sont partagées et tout le monde est en mémoire partagée).

On veut, par exemple pour la couche *elf*, que la lecture des objets elf et l'allocation de la mémoire associée soient génériques. On peut alors imaginer un système de callbacks, dans lequel on a autant de callbacks possible que d'opérations qu'on veut rendre générique. Par exemple, pour la couche *elf*, on définit :

- un prototype de fonction générique pour du read : `error_t (read*)(void *stream, void *buffer, size_t size)`
- un prototype de fonction générique pour du seek : `error_t (seek*)(void *stream, size_t offset)`
- un prototype de fonction générique pour de l'allocation, pour une ouverture de flux, etc

Ensuite, on a un enum qui permet de choisir un des callbacks : `enum libelf_callback_e {libelf_open, libelf_read, libelf_seek, etc};` et enfin, une fonction qui permet d'associer chaque callback (pour un objet elf donné) : `error_t libelf_setopts(struct elf_object_s *elfobj, enum libelf_callback_e, void *callback);`

Par défaut, la couche *elf* fournit des fonctions qui utiliserait le vfs pour la manipulation des objets et malloc/free pour l'allocation mémoire.