# Analyse de l’application

## Infos administratives

Groupe numéro : 10

Etudiant 1 : D’haeyere Corentin

Etudiant 2 : Vandeputte Francois

Etudiant 3 : Lapinski Damien

## Découpe de l’application

L’application sera découpée de la manière suivante :

controller.h - controller.c

zombie.h - zombie.c

labo.h - labo.c

common.h - common.c

utils\_v2.h - utils\_v2.c

makefile

### Modules

Donnez ici le nom des modules et une brève description. Donnez les signatures des fonctions et une brève description de ces fonctions.

* **controller.h** : contient une liste de ports ainsi que la documentation des fonctions permettant le contrôle de programmes zombie
* **zombie.h** : documentation du programme zombie
* **labo**.**h** : documentation du programme labo
* **utils\_v2.c** : librairie de commandes utilisée en cours
* **utils\_v2.h** : documentation de utils\_v2.c
* **common.c** : librairie de fonctions partagées par les fichiers ci-dessus
* **common.h** : documentation de common.c

### Types Utilisateurs

Donnez ici les types utilisateurs que vous comptez créer et une brève description.

enum ports { 10000, 10001, 10002, 10003, 10004, 10005, 10006, 10007, 10008, 10009 } ;

### Fonctions controller.c zombie.c labo.c

Donnez ici les signatures des fonctions qui seront présentes dans controller.c, zombie.c et labo.c. et Donnez une brève description de ces fonctions.

**controller.c** :

**-** bool **is\_process\_running\_on\_port(int port, const char\* process\_name)** :

/\*\*

Vérifie si un processus avec le nom donné est actif sur le port spécifié.

PRE : port, le port à vérifier

process\_name, nom du processus recherché

POST : Exécute la commande ss pour récupérer les processus et leur port.

Vérifie que le processus recherché est en cours d’exécution sur le port correspondant.

RES : true si le processus est actif sur le port donné, false sinon

\*/

- int\* get\_zombie\_ports(int \*nb)

/\*\*

\* Recupere les ports ou le programme zombie tourne

\* PRE : nb le tableau de port ou tourne le programme

\* POST  : on recherche via la méthode is\_process\_running\_on\_port les ports qui font tourner le processus zombie

\* RES : un tableau d’int représentant les ports ou le processus tourne.

\*/

**zombie.c** :

- void **create\_bash(int pipefd)** :

/\*\*

Créé un fichier .sh executable via le pipe

PRE  : pipefd le pipe permettant de recuperer la commande a mettre dans le fichier

POST : Création du .sh avec shebang et commande récupérée du pipe

RES  : Void

\*/

- void **child(void \*pipe, void \*socket)** :

/\*\*

Méthode permettant de lancer le shell bash et modifier le flux standard avec celui du parent

PRE  : pipe le pipe pour communiquer avec le parent

Socket : le socket pour envoyer l’output

POST : le fichier bash est exécuté et le flux standard est modifié pour être celui du parent

RES : Void

\*/

- int **create\_socket(unsigned short PORT)**:

/\*\*

Crée un socket sur le port spécifié

PRE  : PORT, port sur lequel créé

* POST : Création du socket sur le port donné

RES  : file descriptor du socket

\*/

**labo.c** :

- void **create\_zombie()**:

/\*\*

Crée un processus zombie

POST:Fork sur un port choisi au hasard dans l’enum ports et affiche le PID de celui-ci.

\*/