Salma AOUIBATE
Robin BONNIN

INSA Lyon

Travaux Pratiques

Table des matières

Introduction

Le but de ce TP et de nous familiariser avec la programmation multi-processus et savoir comment manipuler les IPC et ce, en transmettant un message codé à travers un réseau (poste émetteur+poste récepteur = notre poste) ; le poste émetteur devra envoyer au poste récepteur le code et la phrase codée, le poste récepteur va en boucle recevoir les codes et les phrases codées et les décrypter pour les mettre dans l'ordre dans le fichier « secret défense »

Conception

Dans ce TP, 5 processus communiqueront entre eux. On détaille ici le rôle de chaque processus :

- ## Saisie

Processus de récupération des phrases saisies par l'utilisateur et transmission à « Code » via un tube nommé :

```
Rediriger SIGINT vers la fonction quittertube()
Créer tube
        TANT QUE : pas fin de phrase
                Scanner caractères
                Ecrire les lettres à Code par tube
        FIN TANT QUE
Fermer tube
```

- Code

Ce processus récupère les lettres via le tube nommé et se synchronise à « Choix » par des signaux. Celui-ci envoie alors la lettre aléatoire de cryptage au travers d'un sémaphore.

La lettre ainsi transmise permet ensuite de coder la phrase par le décalage de chacun de ses signes de la valeur ASCII de la clé. Ce codage prend 1 seconde (de pause).

La phrase codée est transmise à la Boîte aux lettres.

```
Créer letter box
Rediriger le signal SIGINT vers quitter()
Rediriger le signal SIGUSR2 vers coder()
Sémaphore Sentence
S'attacher à letter box
S'attacher au Sémaphore
GetPID(choix)
Se détacher du Sémaphore
TANT QUE : pas fin de phrase
        Récupérer lettre_phrase de tube
        S'attacher au Sémaphore
        Sentence=numéro phrase+1
        Récupérer lettre
        coder()
        Se détacher du Sémaphore
        Envoyer le signal SIGUSR2 au processus Choix
        sleep(1)
        Transmettre lettre_code à letter box
FIN TANT QUE
```

- Choix

Choisit une lettre aléatoirement pour la transmettre à code, ce processus transmet ensuite la lettre et le numéro

de la phrase à la boite aux lettres, comme on l'a dit précédemment il communique avec « Code » à l'aide de signaux.

```
    Rediriger SIGINT vers la fonction quitter ()
    Rediriger SIGUSR2 vers writefile()
    S'attacher à letter box
    S'attacher au Sémaphore
Choisir une lettre aléatoirement
Down(Sentence)
numéro_phrase=Sentence
 Transférer lettre à Code
 Transférer numéro_phrase && lettre à letter box
```

- Réception

Récupère les phrases codées en s'attachant à la Boîte aux lettres, ce processus crée ensuite une mémoire partagée et écrit les phrases codées dans un buffer

```
Rediriger le signal SIGINT vers quitterreception()
S'attacher à letter box
Créer mémoire partagée
S'attacher à la mémoire partagée en écriture
Récupérer numéro_phrase
verif=numéro_phrase
        TANT QUE : pas fin de phrase
                Récupérer lettre_code
                Mettre lettre_code dans buffer
        FIN TANT QUE
```

- Décode

Ce processus récupère les phrases codées depuis la mémoire partagée de « Réception », ainsi que la lettre de décryptage dans la boîte aux lettres. Le programme peut alors décoder les phrases et les écrire dans le fichier texte dans l'ordre d'arrivée.

```
S'attacher à letter box
S'attacher à la mémoire partagée en lecture
Récupérer numéro phrase et lettre
TANT QUE : pas fin de phrase && numéro de phrase==vérif
        Accéder à la mémoire partagée
        lancer decoder()
        Mettre lettre_décode dans fichier "SecretDefense"
FIN TANT QUE
Se détacher de tous les IPC
```

Plan d'implémentation :

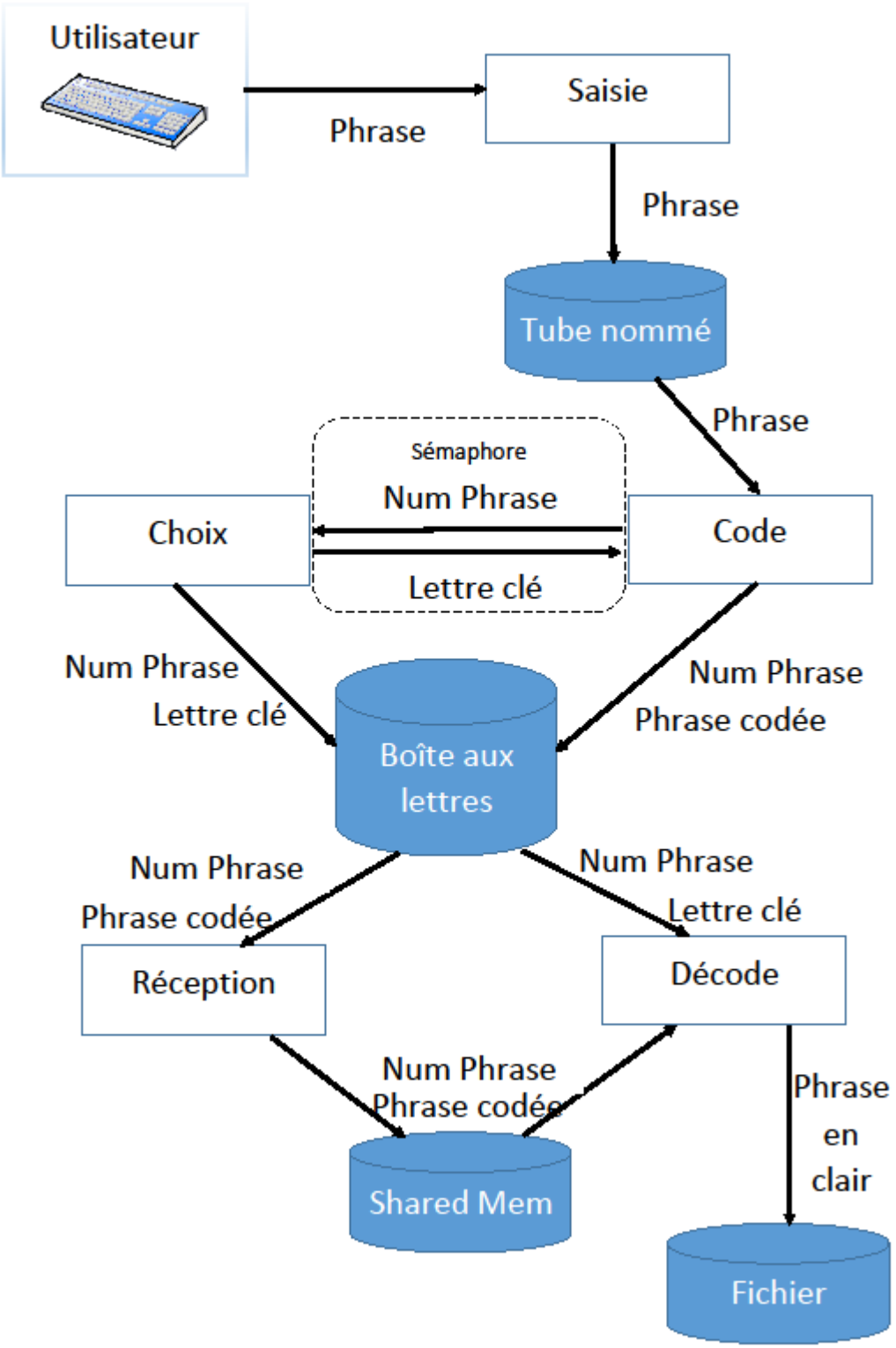
- ## Méthode

Comme il est mentionné dans le sujet nous avons pensé qu'il était plus approprié de coder par blocs, de telle sorte que l'implémentation se fera petit à petit et construire notre chaîne de processus ; nous serons donc contraints de tester chaque sous ensemble avant de passer au suivant.

Une vision plus segmentée du sujet et nous permettra de savoir de quel bloc vient une erreur et nous saurons donc la corriger plus rapidement.

Dans un premier temps nous avons pensé à créer le processus saisie qui est indépendant d'une IPC créée par un autre processus. Nous procéderons alors dans cet ordre : saisie, code, choix, réception et décode.

- ## Diagramme



- Fin des processus

En ce qui concerne la fermeture des processus il faut en premier lieu fermer ceux qui n'ont pas créé d'IPC à savoir

« Choix » et « Décode » et donc quand chaque processus se terminera, il détruira l'IPC qu'il a créé et libèrera ainsi l'espace réservé.

Contraintes :

Nous avons estimé qu'il n'était pas nécessaire de faire intervenir de relation père-fils car les processus communiquent à l'aide des sémaphores, et d'autre part dans le sujet il est mentionné qu'on ne devait utiliser les relations père-fils que si c'est nécessaire.

D'autre part les processus choix et code communiquent à l'aide de signaux donc il a était judicieux que code va écrire son PID dans la boite aux lettres et choix va le récupérer pour pouvoir lui envoyer des signaux

Scénario :

1. Saisir au clavier la chaine de " "
2. Saisie transmet la chaine à code via le tube nommé
3. Code reçoit la chaine et la chiffre avec la lettre
4. Résultat est afficher la chaine
5. Réception écrit les lettres dans un buffer et crée la mémoire partagée
6. Décode récupère les phrases saisies par l'utilisateur

Les phrases sont dans l’ordre dans lequel elles ont étés décodés dans le fichier texte