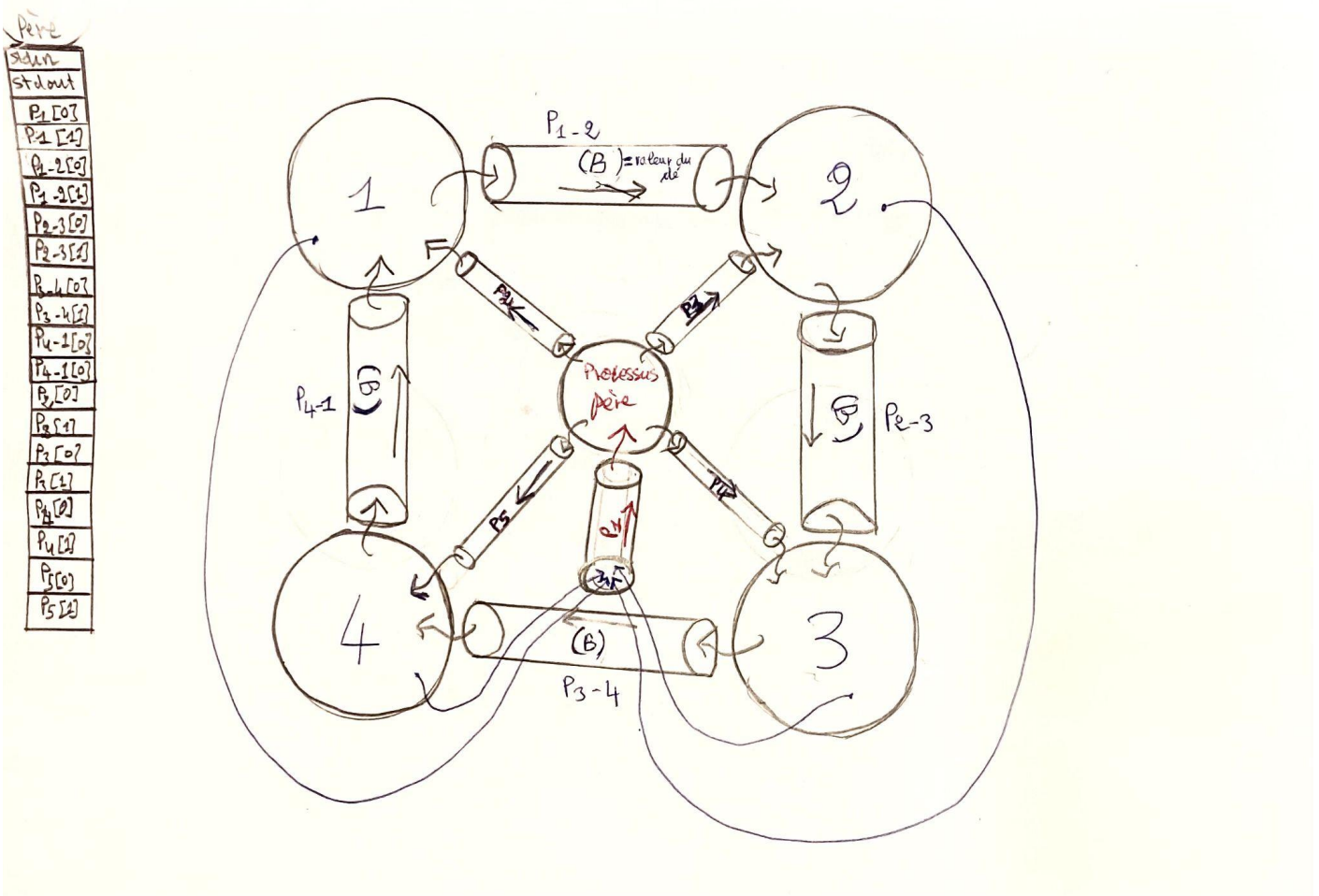
**DEMBELE Oumou** **DIAGNE Aminata**

**L3-MIAGE**

# RAPPORT FINAL

## Analyse:

### Conception:



Nous avons fait la conception pour 4 joueurs(processus fils).
1,2,3 et 4 sont respectivement les processus fils joueur 1,joueur 2,joueur 3 et joueur 4.
Le processus père envoie simultanément le numéro du joueur qui doit jouer à ses processus fils(joueurs) en écrivant dans les pipes P2,P3,P4 et P5 et qui seront ensuite lu par les joueurs 1,2,3,4 respectivement dans les pipes 1,2,3,4.
Chaque joueur envoie le numéro qu'il doit jouer à son frère(processus suivant)qui lui aussi l'envoie à son frère(processus suivant) ainsi de suite .Par exemple si le joueur actuel est le joueur 1(processus fils 1):
le processus fils 1 enverra le numéro qu'il doit jouer au processus fils 2 par le pipe P1_2,
le processus fils 2 l'enverra au processus fils 3 par le pipe P2_3,
le processus fils 3 l'enverra au processus fils 4 par le pipe P3_4,
et le processus fils 4 l'enverra au processus fils 1 par le pipe P4_1.
Le joueur courant(processus fils qui joue) envoie sa valeur de retour au père par le pipe P1.

## Modélisation:

Soit:

- **P** le processus père ,
- **P1** le pipe entre le processus père et le joueur courant
- **P1_2** , **P2_3** , **P3_4** et **P4_1** les pipes respectifs entre les processus fils **f1** ,**f2** ,**f3** et **f4** (les joueurs du petit cheval).
- **P2,P3,P4,P5** sont les pipes respectifs entre le processus père et ses fils f1,f2,f3 et f4.
- Une variable **A comprise entre 1 et N inclus** qui contiendra au départ le numéro du premier joueur qui débutera le jeu, donc initialisé par le processus père.Il sera mise à jour par le joueur actuel avec comme valeur son numéro (le numéro du joueur actuel ) ou **-2** si ce dernier a gagné.
- Une variable **B comprise entre 1 et 6**(valeur **obtenue aléatoirement**) indiquant la **valeur du tirage du dé** du joueur actuel et qui sera ensuite communiqué au joueur suivant(par exemple si le joueur actuel est f1 c'est à f2 qu'on envoie le numéro joué par f1).
- Une constante N à définir

## projet_0:

Pour la création des N processus fils respectifs, nous avons d'abord créé un processus fils et ensuite nous l'avons inséré dans une boucle for allant de 1(inclu) à N+1(exclu) pour obtenir au total N processus fils créés.
Pour tester notre programme, on a défini la valeur de N à 4(mais N est modifiable depuis le programme principale main).

### Difficultés rencontrées:

Pour un début, nous avions créer un seul processus fils et dans le père(default) nous avions voulu créer N-1 processus fils à l'aide d'une boucle for allant de 2(inclut) à N+1(exclu).Soit le code suivant:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#define N 4

int main(){

        switch(fork()){
              case -1:
                  fprintf(stderr,"erreur de fork\n");
                  exit(-1);
              case 0:  // création des n-1 fils restants
                  printf("creation du fils numero %d, dont le pid du pere est %d\n", i,getppid());
                  exit(0);
              default:
                  printf("le fils %d est mort \n", wait(NULL));
                  for (int i=1;i<N+1;i++){
                      switch(fork()){
                          case -1:
                              fprintf(stderr,"erreur de fork\n");
                              exit(-1);
                          case 0:  // création des n-1 fils restants
                              printf("creation du fils numero %d, dont le pid du pere est %d\n", i,getppid());
                              exit(0);
               }
          }
    return 0;

}
```

qui nous a donné le résultat suivant:

```
[15:40:24]diagneam$ ./brouilon
creation du fils numero 1, dont le pid du pere est 13613
creation du fils numero 1, dont le pid du pere est 13613
le fils 13614 est mort
le fils 13615 est mort
creation du fils numero 2, dont le pid du pere est 13613
le fils 13616 est mort
le fils -1 est mort
creation du fils numero 3, dont le pid du pere est 13613
le fils 13617 est mort
le fils -1 est mort
creation du fils numero 4, dont le pid du pere est 13613
le fils 13618 est mort
le fils -1 est mort
```

Nous avons eu des processus fils  créés mais pas comme on l'attendait.
De plus au début notre programme ne compilait pas car nous n'avions pas les bibliothèques nécessaires pour l'utilisation de certaines commandes(#include <sys/types.h> et #include <unistd.h> pour fork() et #include <sys/wait.h> pour wait()).

*conclusion:*

Il faut toujours penser à  inclure les bibliothèques nécessaires à l'utilisation des commandes pour permettre la compilation.

# projet_1:

- **Création de pipes:**

Nous avons d'abord créé un pipe p1 qui permettra le transfert d'information du processus fils qui joue, vers le processus père.

Ensuite nous avons créé deux tableaux de pipe de taille N contenant N pipes:

Un tableau tab_pipes contenant à l'indice 1,.....,N respectivement les pipes servant à communiquer entre les processus fils (f1 et f2),(f2 et f3),....,(fN-1 et N) .

Un autre tableau tab_pipes_pere_fils contenant à l'indice 1,.....,N respectivement les pipes permettant le transfert d'information du processus père simultanément à tous ses processus fils (f1,....fN).

**Difficultés rencontrées:**

Nous avions au début essayé de créer les pipes dans la boucle **for** de la création des processus fils mais nous avions anticipé de potentiels soucis sur la gestion de la communication.

Nous avions aussi rencontré un problème avec la communication (car nous avions au départ créé qu'un seul pipe **p2** qui transférait l'information du père directement à tous les fils).

Nous avons eu après exécution que seul le premier processus fils arrivait à écrire aléatoirement(souvent) dans le pipe **p2** et à lire l'information dans le pipe du père,c'est à dire que la valeur contenue dans le pipe est prise par le premier fils.Raison pour laquelle nous avions créé un pipe pour chaque fils stocké dans un tableau pour qu'ils puissent tous recevoir la valeur transféré par leur père.

*conclusion:*

Le processus père ne peut pas envoyer une information simultanément à tous ses processus fils via un seul pipe.Du coup il faudra créer autant de pipe qu'il y'a de fils.

Par contre tous les processus fils peuvent successivement envoyer une information au père à l'aide d'un seul pipe.

- **Communications:**

Nous avons créé deux fonctions que nous appelons dans le programme main selon l'utilité :

Une fonction ercriturePereFils qui nous a permis de gérer le transfert d'information du père à tous ses fils simultanément.

Une fonction enfantParleAuPere qui nous a permis de transférer les informations du fils qui joue à son père mais également à son frère(processus fils suivant).

Nous nous sommes rassurées de la bonne fonctionnalité de la gestion de la communication en faisant plusieurs tests:

Nous avons fait circuler des valeurs entre le père et ses processus fils, de même entre un processus fils et son frère(processus suivant).Ainsi que de transférer une valeur d'un des processus fils vers le processus père via le pipe p1.

**Difficultés rencontrées:**

Nous avions au début voulu gérer la communication directement dans la création des processus fils.Seul le premier fils arrivait à communiquer parfaitement avec le processus père (lecture et écriture).

On a rencontré des soucis de lecture(à l'affichage de la valeur lu, nous avions eu des valeurs ressemblant à des adresses) chose qu'on a pu finalement résoudre en utilisant une valeur intermédiaire dans lequel on a stocké le résultat de la fonction read après plusieurs recherches.

Mais nous aimerions bien avoir une explication plus claire sur l'utilisation de la fonction read qui reste un peu flou.

*conclusion:*

Nous pensons que l'utilisation de fonctions intermédiaires définissant des actions répétitives( ici le transfert d'information du père à tous ses fils simultanément et le transfert des informations du fils qui joue à son père mais également à son frère(processus fils suivant) facilite la compréhension et permet  l'optimisation du code.

- **Assemblage des fonctionnalités (gestion des règles du jeux):**

  **Règles du jeu:**

  Nous avons choisi N joueurs avec la possibilité de changer la valeur de N.
  Chaque joueur aura aussi N pions.

  N*N le nombre de case à parcourir par un pion à sa sortie jusqu'à la case gagnante.
  La valeur du dé( **B** ) est comprise entre 1 et 6.
  Chaque joueur pour faire sortir un pion doit forcément avoir un 6 comme valeur du dé.
  Tant que la valeur du dé lancé par un joueur est 6, il rejouera.

  **Début du jeu:**

  Au lancement du jeu, le processus père renvoie la valeur 1 à tous ses processus fils simultanément à l'aide de la fonction ercriturePereFils,car nous avons décidé que ça soit le premier joueur(processus fils) qui commence le jeu.Ensuite tous les fils font successivement la lecture de cette valeur en la comparant à leur indice et dès que ces deux valeurs (la valeur envoyé par le père et l'indice d'un des fils) coïncident alors c'est ce fils qui jouera.

  Pour jouer, le joueur va d'abord lancé le dé:
  si la valeur du dé lancé est inférieure à 6, le joueur fera avancer un de ces pions autant de fois que la valeur du dé obtenue et enverra la valeur du dé à son frère(joueur

suivant) à l'aide de la fonction enfantParleAuPere, si seulement si il avait déjà un ou des pions sortis et son frère à son tour enverra la valeur à son suivant et ainsi de suite.Si le joueur n'avait pas de pion(s) déjà sorti(s) il ne jouera pas.Mais il enverra dans tous les cas son numéro ou bien la valeur **-2** au processus père si il a **gagné(c'est à dire dès qu'un de ses chevaux aura atteint la case gagnante**"centrale").

Le processus père à la réception de cette valeur teste si la valeur est différente de **-2:**

Dans le cas où la valeur envoyée par le joueur(processus fils) est différente de **-2** alors il l'incrémentera de 1 si cette valeur est différente du numéro du dernier joueur sinon il la change en 1 pour que le joueur suivant soit le joueur 1 avant de le renvoyer à tous ses fils simultanement.Et ces actions vont se boucler jusqu'à la fin du jeu.

Dans le cas où la valeur envoyé par le joueur(processus fils) est égale à **-2**, le père fera savoir le gagnant du jeu avec un affichage du genre **"Le joueur numéro i a gagné" i** étant l'indice du joueur venant de finir de jouer et il mettra fin au jeu en tuant tous ses processus fils.

**Difficultés rencontrées:**

Nous n'avions malheureusement pas eu suffisamment de temps pour terminer le codage de toutes ses fonctionnalités que nous trouvons d'ailleurs super intéressantes.

*conclusion:*

La gestion du temps,la rapidité de réflexion et de codage sont des compétences cruciales que nous devons développer pour la réussite de nos projets futurs.

- **EXÉCUTION DU CODE:**

**Le Makefile:**

Nous avons écrit un petit Makefile permettant de compiler les fichiers:

projet_0.c (contenant le programme pour la création des N processus fils) et projet_cheval.c (contenant le programme pour la création des N processus fils , la création des tubes utilisés et des fonctions intermédiaires pour la gestion de la communication et l'assemblage des fonctionnalités du jeu).

Pour exécuter notre programme , nous faisons:
un make pour compiler les fichiers ensuite ./nomExecutableDuFichier
Si vous souhaitez exécuter le fichier projet_0.c le nomExecutableDuFichier sera *projet_0*
et si vous souhaitez exécuter le fichier projet_cheval.c le nomExecutableDuFichier sera *projet_cheval*