Rapport Systèmes et Réseaux 1

- HAMIDOU Mohammed Nazim (Fait en mônome)

1) Description projet :

1.1) But du projet :

- Créer le jeu « The Mind », en utilisant le langage de notre choix, shell/awk et/ou C, j'ai choisi de le faire en shell/awk pour plusieurs raisons, la plus notable est que cela me permet de mieux comprendre le shell/awk que j'ai jamais utilisé avant contrairement au C.

- Le but du jeu est simple : c'est un jeu de cartes à jouer en coopératif, le jeu comporte 100 cartes numérotées de 1 à 100, on joue par rounds, X cartes est distribuée(s) à chaque joueur et à chaque round, X ici est le numéro du round dans le quel on se trouve, par exemple pendant le round 1(donc au début) seulement 1 carte est distribuée à chaque joueur, la **régle** est de jouer toutes les cartes par ordre croissant, dés que toutes les cartes ont été jouées on passe au round suivant, jusque là vous vous dites c'est assez simple, sauf que la **régle** c'est que personne doit se parler, se lancer des signes ou montrer ses cartes au joueurs avec qui on joue, voilà vous avez compris ça se joue seulement à l'instinct.

- Résumé du jeu « The Mind » (copié du sujet du projet) :

The mind est un jeu de carte simple de type coopératif inventé par Wolfgang Warsch. Voici la règle extraite du site https://jeux-cooperatifs.com: "Le jeu se joue en plusieurs manches. Lors de la première manche, chaque joueur reçoit une carte dont il prend connaissance. Ces cartes représentent les chiffres de 1 à 100. Le jeu est coopératif, il suffit de jouer les cartes par ordre croissant. De la plus petite à la plus grosse. C'est vraiment idiot. Mais voilà, le truc c'est que personne ne doit se

parler, ni se faire des signes. Il n’y a donc pas d’ordre de jeu. Celle ou celui qui pense avoir la plus petite carte la pose. La ou le suivant qui pense avoirla plus petite suivante la pose et ainsi de suite.”Si les joueurs ont réussi à poser les cartes dans l’ordre lors d’une manche, chacun recevra une carte deplus lors de la manche suivante ; sinon on retourne à la première manche.

1.2) Les problèmes posés à priori :

- Trouver une manière de communication entre les processus qui soit fiable.

- Réfléchir à une manière pour implémenter la façon dont joue les robots qui me paraît un peu compliqué à mettre en place.

2) Architecture générale :

- Mon implémentation consiste en 3 fichiers qui communiquent entre eux, GestionJeu.sh, Player.sh ainsi que Robot.sh, qui sont respectivement le script pour le main qui gère tout le jeu, le script pour jouer en tant que joueur humain et le script du joueur robot.

2.1) Communication :

- Mes 3 scripts communiquent à l’aide des flux de redirections « < » et « << », des pipes « | » ainsi et surtout grâce au signaux USR1 et USR2.

2.2) Ressources partagées :

- D’abord un fichier en partagé par les joueurs/robots qui contient le pid du processus GestionJeu.sh ce qui leurs permet d’envoyer un signal.

- On a un fichier qui contient les cartes jouées durant le round, et donc les scripts joueur et robot écrivent et lisent dessus et à chaque carte jouée le script GestionJeu lit ce qui a été écrit.

- On a aussi un fichier qui permet à GestionJeu de distribuer les cartes et donc c’est un droit en écriture, concernant les joueurs/robots ils ont un droit dessus en lecture afin qu’ils puissent récupérer leurs cartes.

2.3) GestionJeu.sh :

- Ce processus là gère tout le jeu, tout d’abord dés son lancement et aprés avoir sélectionné le nombre de joueur(s) voulu, le processus se met en attente du signal **USR1** pour l’inscription des joueurs**,** dés qu’il le reçoit il va récupérer le **nom** ainsi que le **PID** du dernier joueur inscrit dans un fichier texte spécialement conçu pour ça, une fois le nombre de joueurs voulu inscrit est atteint, le jeu peut commencer, le processus mélange les cartes aléatoirement puis commence à distribuer en fonction du round dans le quel on se trouve.

- dés que les cartes sont distribuées le signal **USR1** est envoyé à chaque processus joueur/robot pour le prévenir qu’il peut récupèrer ses cartes et commencer à jouer.

- à chaque carte posée le signal **USR2** est envoyé à chaque processus joueur/robot, pour leurs indiquer qu’une carte a été jouée et ainsi ils pourront l’afficher dans leurs terminaux respectifs.

- à chaque carte posée une vérification est passée pour vérifier si le round est terminé si c’est le cas on vérifie si c’est gagné ou perdu, et on envoie donc le signal **USR2** à chaque processus joueur/robot, pour leurs indiquer que la partie continue/est perdue.

- si la partie est perdue une proposition de continuer/s’arrêter là est déclenchée.

- à la fin de chaque partie le top 10 des parties est affiché, le classement dépend du nombre de joueurs * le nombre de rounds réussis, ainsi que la date de chaque partie.

- à la fin de chaque partie le processus « kill » tous les processus joueurs/robots grâce au signal **TERM.**

- Ce processus affiche toutes les cartes jouées durant la partie et pas seulement celles du round (c’était pas demandé dans le sujet c’est seulement un petit plus).

2.4) Player.sh :

- Ce processus là gère le fait de jouer une carte par un joueur **humain**, à son lancement il demande le nom du joueur puis émet le **USR1** au GestionJeu.sh pour lui indiquer que l'inscription s'est bien passée, aprés ça il récupère ses cartes grâce au signal **USR1** émis par GestionJeu.sh et laisse le joueur choisir la carte qu'il veut jouer, dés que la carte est choisie le signal **USR2** est émis vers GestionJeu.sh pour lui indiquer qu'une carte a été jouée, tout en affichant l'état du round (c'est à dire les cartes posées durant ce round) avec le nom du joueur qui a posé chaque carte et cela grâce au signal **USR2** émis par GestionJeu.sh vers chaque Player.sh dés qu'une carte a été posée, ce processus là est assez simple du au fait que la très grande partie de gestion du jeu se fait dans le processus main **GestionJeu.sh.**

2.5) Robot.sh :

- Ce processus là globalement se base sur la même architecture que le processus **Player.sh,** tout se passe de la même manière, simplement le choix d'une carte se fait automatiquement.

- Son intelligence est assez simple : le processus trie les cartes reçus à chaque round puis se met en **pause** (grâce à un sleep) pendant un instant avant de jouer la carte avec la plus petite valeur. Cette pause là dépend de la valeur de la carte qui doit être jouée prochainement, voici un peu plus d'explications :

- En fait c'est assez simple plus la valeure de la carte est grande plus la valeure du sleep est élevée, cela se base sur une fonction exponentiel qui donne des résultats bien plus satisfaisant qu'une fonction linéaire.

2.6) Classement des joueurs :

- Partie Optionnelle que j'ai ajouté : cela consiste simplement à sauvegarder le top 10 des parties « Les plus réussis » dans un fichier texte et les afficher à chaque fin de partie.

3) Lancement et déroulement d’une partie :

- Lancez un terminal dans le quel vous allez lancer le script GestionJeu.sh puis lancez X terminaux dans les quels vous allez lancez les scripts Player.sh et Robot.sh suivant vos choix.

X : valeure du nombre de joueurs/robots choisi.

- Pour les processus joueurs humains veuillez choisir un nom pour chaque joueur.

- Vous pouvez commencer à jouer et à suivre les instructions.

4) Conclusion :

- Concernant les langages choisis ici c'est assez simple j'ai préféré utiliser le shell et awk car c'est sûrement la première fois que j'ai eu à utiliser ces langages (ce qui n'est pas le cas du C que j'ai déjà utilisé beaucoup par le passé) j'ai appris beaucoup en faisant ce projet et j'ai spécialement aimé le awk que je trouve vraiment intéressant et sous exploité.

- Une extension possible(qui n’était pas une contrainte) qui aurait été intéressante était de rajouter une partie réseau, en C donc, pour pouvoir jouer en réseau (j’aurais vraiment aimé le faire mais faute de temps je n’ai pas pû).

- Ce que j’aurais vraiment aimé faire autrement c’est la méthode de jeu du script Robot.sh, oui le sleep que j’ai implémenté avec une fonction exponentiel fonctionne bien et est plutôt fiable, mais un algorithme meilleur aurait pû se faire.

5) Code Source :

5.1)GestionJeu.sh :

```
#!/bin/bash
Init_Fichiers()
{
>>classement.joueurs
if [ `awk 'BEGIN{a=0}{a++}END{print a}' classement.joueurs` == 0 ]; then
echo "TOP10 Joueurs/rounds" >> classement.joueurs
fi
```

```
>cartes.melangees
touch cartes.distruibuees.round
>cartes.distruibuees.round
touch inscrit.noms
>inscrit.noms #On va garder les Nom / PID par joueur
touch gestion.jeu.pid
>gestion.jeu.pid
touch round.en.cours
>round.en.cours
touch toutes.les.cartes.posees
>toutes.les.cartes.posees

}



Init_Jeu()
{
echo "Bienvenue dans the mind"
echo "Combien de joueurs voulez vous?"
Init_Fichiers
nbRound=1
echo $$ > gestion.jeu.pid
inscrit=0
read nbJoueurs
echo "En attente de l'inscription des joueurs..."
NBCartesJouees=0
trap 'arrayNames[$inscrit]=`awk -v ainscrit=$inscrit '"'"'NR==ainscrit+1{print $1}'"'"'
inscrit.noms`;
pidNames[inscrit]=`awk -v ainscrit=$inscrit '"'"'NR==ainscrit+1{print $2}'"'"'
inscrit.noms`;
echo "Joueur inscrit : ${arrayNames[$inscrit]} / PID ${pidNames[inscrit]} / TInscrit $
((inscrit+1))";
((inscrit=inscrit+1))' USR1
while [ "$inscrit" -ne "$nbJoueurs" ]
do
:
done
}

Distribuer_Cartes()
{
nb_cartes_a_distribuer=0
nb_joueur=0

while [ $nb_joueur -lt $nbJoueurs ]
do
nb_cartes_a_distribuer=0
cartes=""
#Servir les cartes aux joueurs
while [ $nb_cartes_a_distribuer -lt $nbRound ]
```

```
do
cartes="$cartes `awk -v var=$NBCartesJouees '(NR==var+1) {print}' cartes.melangees`"
((NBCartesJouees=NBCartesJouees+1))
((nb_cartes_a_distribuer=nb_cartes_a_distribuer+1))
done
echo "${arrayNames[$nb_joueur]} ${pidNames[$nb_joueur]}
$cartes">>cartes.distruibuees.round
((nb_joueur=nb_joueur+1))
done
nb_joueur=0
#Envoyer un signal aux joueurs pour leurs dire que vous pouvez recuprer vos cartes
while [ $nb_joueur -lt $nbJoueurs ]
do
kill -s USR1 `awk -v var=$nb_joueur '(NR==var+1){print $2}' inscrit.noms`
((nb_joueur=nb_joueur+1))
done
}
Random_Melange()
{
#On mélange les cartes
while [ `awk 'END{print NR}' cartes.melangees` -lt 100 ]
do
RandomCarte=$((1 + $RANDOM % 100))
#Si la carte prise a déjà été prise on reprend
while [ `awk -v var=$RandomCarte 'BEGIN{count = 0}($1==var) {count++} END { print
count }' cartes.melangees` -ne 0 ]
do
RandomCarte=$((1 + $RANDOM % 100))
done
echo $RandomCarte>>cartes.melangees
done

}
Carte_Posee()
{ #Rajouter la dernière carte posée dans la liste de toutes les cartes posees
awk 'END{print $0}' round.en.cours >> toutes.les.cartes.posees
#Afficher toutes les cartes posees par round
#clear;printf "Cartes Jouèes en tout :";awk '{printf "%s"" ",$0}END{printf "\n"}'
toutes.les.cartes.posees
clear;echo "Cartes Jouèes en tout :";awk '{print}END{printf "\n"}' toutes.les.cartes.posees
nb_joueur=0
#Envoyer un signal à tous les joueurs pour leurs indiquer qu'une carte a été posée
while [ $nb_joueur -lt $nbJoueurs ]
do
kill -s USR2 `awk -v var=$nb_joueur '(NR==var+1){print $2}' inscrit.noms`
((nb_joueur=nb_joueur+1))
done
#Fin du round si toutes les cartes du round ont été posées
#Ou Si toutes les cartes ont été posées (les 100 cartes)
#On doit faire la double vérification au cas ou à la fin un joueur
#ne recoit pas le même nombre de carte que les autres
```

```
#au cas ou il ny ai plus...
if [ `awk 'END{print NR}' round.en.cours` == $(($nbRound*$nbJoueurs)) ] || \
[ `awk 'BEGIN{a=0}/[0-9]/{a++}END{print a}' toutes.les.cartes.posees` == 100 ]
then
a=`awk 'BEGIN{a=0}/[0-9]/{a++}END{print a}' toutes.les.cartes.posees`
echo "R : $nbRound, C : $a"
Verifier_Si_Round_Gagne
fi

}
Verifier_Si_Round_Gagne()
{
i=2
#Verifier si toutes les cartes posees durant le round sont par ordre croissant
while [ ! $i -gt `awk 'END{print NR}' round.en.cours` ]
do
if [ `awk -v var=$i '(NR==var-1){print $2}' round.en.cours` -gt `awk -v var=$i '(NR==var)
{print $2}' round.en.cours` ]
then
Etablir_Classement
echo "Perdu...Renitialisation"
echo "Perdu...Renitialisation">toutes.les.cartes.posees
echo "Perdu...Renitialisation">round.en.cours
nb_joueur=0
#Envoyer un signal à tous les joueurs pour leur dire que c'est perdu
while [ $nb_joueur -lt $nbJoueurs ]
do
kill -s USR2 `awk -v var=$nb_joueur '(NR==var+1){print $2}' inscrit.noms`
((nb_joueur=nb_joueur+1))
done
#Rejouer ou pas?
echo "Voulez vous rejouer ? (O/N)"
read decision
while [ $decision != "N" ] && [ $decision != "O" ]
do
echo "Voulez vous rejouer ? (O/N)"
read decision
done
#Si on rejoue pas
if [ $decision == "N" ]
then
En_JEU=false
#Si on rejoue
else
nbRound=1
NBCartesJouees=0
>round.en.cours
>toutes.les.cartes.posees
>cartes.distruibuees.round
>cartes.melangees
Random_Melange
```

```
Distribuer_Cartes
fi


return 0
fi
((i=i+1))
done
#Si toutes les cartes ont pas été posées(on reste en jeu)
if [ $NBCartesJouees -lt 100 ]
then
((nbRound=nbRound+1))
>round.en.cours
>cartes.distruibuees.round
echo "//">>toutes.les.cartes.posees
Distribuer_Cartes
#Sinon gagné
else
echo "Gagné"
echo "Gagné">toutes.les.cartes.posees
echo "Gagné">round.en.cours
nb_joueur=0
#Envoyer un signal à tous les joueurs pour leur dire que c'est gagné
while [ $nb_joueur -lt $nbJoueurs ]
do
kill -s USR2 `awk -v var=$nb_joueur '(NR==var+1){print $2}' inscrit.noms`
((nb_joueur=nb_joueur+1))
done
En_JEU=false
Etablir_Classement
fi

}
Etablir_Classement()
{
RatioPartie=`echo "scale=2;$((nbRound-1))*$nbJoueurs" | bc`
ligne=2
while [ $ligne -le `awk 'END{print NR}' classement.joueurs` ]
do
Text[$((ligne-2))]=`awk -v var=$ligne '(NR==var){for (i=2; i<NF; i++) printf $i " "; print
$NF}' classement.joueurs`
Ratio[$((ligne-2))]=`awk -v var=$ligne '(NR==var){print $15}' classement.joueurs`
((ligne=ligne+1))
done
printf "\t\tTOP10 Joueurs/rounds:\n" > classement.joueurs
Ratio[${#Ratio[@]}]=$RatioPartie
i=0
RatioTries=($(for l in ${Ratio[@]}; do echo $l; done | sort -r -n));unset Ratio

while [ $i -lt ${#RatioTries[@]} ] && [ $i -lt 10 ]
do
```

```
pos=0
while [ $pos -lt ${#Text[@]} ] && \
[ "${RatioTries[$i]}" != "$(echo ${Text[$pos]} | awk '{print $14}')" ]
do
((pos=pos+1))
done
if [ $pos == ${#Text[@]} ]
then
echo "$((i+1))- Nombre Joueurs : $nbJoueurs // Rounds consécutifs réussies : $((nbRound-
1)) // Ratio : $RatioPartie // Date : $(date "+%d-%m-%Y %H:%M:%S")" >>
classement.joueurs
else
echo "$((i+1))- ${Text[$pos]}" >> classement.joueurs
Text[$pos]="terminé"
fi
((i=i+1))
done
unset Text
unset RatioTries
}
main()
{
En_JEU=true
Init_Jeu #Initialiser le jeu
Random_Melange #Melanger les cartes
Distribuer_Cartes #Destribuer les cartes
while [ $En_JEU = true ]
do
trap 'Carte_Posee' USR2
done
nb_joueur=0
awk '{print}' classement.joueurs
#Fin du jeu on termine tous les processus joueurs
while [ $nb_joueur -lt $nbJoueurs ]
do
kill -s TERM `awk -v var=$nb_joueur '(NR==var+1){print $2}' inscrit.noms`
((nb_joueur=nb_joueur+1))
done


}

main
```

5.2)Player.sh :

```
#!/bin/bash
Init_Joueur()
{
```

```
echo "Veuillez entrer votre nom s'il vous plait"
read NomJoueur
echo "$NomJoueur $$">>inscrit.noms
pidGestionJeu=`awk {print} gestion.jeu.pid`
kill -s USR1 $pidGestionJeu

}
Recuperation_Carte()
{
nbCartes=`awk '(NR==1) {print NF-2}' cartes.distruibuees.round`
prendreCartes=0
while [ $prendreCartes -lt $nbCartes ]
do
MesCartes[$prendreCartes]=`awk -v var=$prendreCartes -v var2=$$ '($2==var2){print $
(var+3)}' cartes.distruibuees.round`
((prendreCartes=prendreCartes+1))
done
Jouer
}
Jouer()
{
carteJouees=0
choixCarte=-1
while [ $carteJouees -lt $nbCartes ]
do
clear;
echo "Veuillez choisir une carte entre ${MesCartes[*]}"
read choixCarte
#Verifier qu'il veut jouer une carte qu'il posséde
while [[ ! " ${MesCartes[*]} " =~ " ${choixCarte} " ]]
do
echo "Vous ne possédez pas cette carte veuillez rechoisir une carte entre : $
{MesCartes[*]}"
read choixCarte
done
((carteJouees=carteJouees+1))
echo "$NomJoueur $choixCarte" >> round.en.cours
kill -s USR2 $pidGestionJeu
pos=0
if [ ${#MesCartes[@]} -ne 0 ]; then
while [ "${MesCartes[$pos]}" != "$choixCarte" ]
do
((pos=pos+1))
done
unset 'MesCartes[$pos]' #Enlever la carte jouée
fi
done


}
```

```
Affichage()
{
clear;
if [ "`awk '(NR==1){print}' round.en.cours`" != "Gagné" ] && [ "`awk '(NR==1){print}'
round.en.cours`" != "Perdu...Renitialisation" ];then
echo "Cartes Jouées round :";awk '{print}' round.en.cours
if [ ${#MesCartes[@]} -eq 0 ];then
echo "Vous avez joué toutes vos cartes, Veuillez attendre la fin du round"
else
echo "Veuillez choisir une carte entre ${MesCartes[*]}"
fi

else
awk '{print}' round.en.cours
fi
}

main()
{
trap 'Affichage' USR2
Init_Joueur
trap 'Recuperation_Carte' USR1
while true; do
:
done
}

main
```

5.3)Robot.sh :

```
#!/bin/bash
Init_Robot()
{


NameRobot=`awk 'BEGIN{count = 0} (/Robot/) {count++} END { print count }'
inscrit.noms`
((NameRobot=NameRobot+1))
NameRobot="Robot"$NameRobot
echo "$NameRobot $$">>inscrit.noms
pidGestionJeu=`awk {print} gestion.jeu.pid`
kill -s USR1 $pidGestionJeu

}
Recuperation_Carte()
{
nbCartes=`awk '(NR==1) {print NF-2}' cartes.distruibuees.round`
prendreCartes=0
```

```
while [ $prendreCartes -lt $nbCartes ]
do
MesCartes[$prendreCartes]=`awk -v var=$prendreCartes -v var2=$$ '($2==var2){print $
(var+3)}' cartes.distruibuees.round`
((prendreCartes=prendreCartes+1))
done
Jouer
}
Jouer()
{
carteJouees=0
while [ $carteJouees -lt $nbCartes ]
do
echo "Veuillez choisir une carte entre ${MesCartes[*]}"
MesCartesTriees=($(for l in ${MesCartes[@]}; do echo $l; done | sort -n))
choixCarte=${MesCartesTriees[0]}

((carteJouees=carteJouees+1))
a=`echo "e($choixCarte/24)" | bc -l`
sleep `echo "scale=5;$a/1.5" | bc`
echo "$NameRobot $choixCarte" >> round.en.cours
kill -s USR2 $pidGestionJeu
pos=0
if [ ${#MesCartes[@]} -ne 0 ]; then
while [ "${MesCartes[$pos]}" != "$choixCarte" ]
do
((pos=pos+1))
done
unset 'MesCartes[$pos]' #Enlever la carte jouée
fi
done
echo "Vous avez joué toutes vos cartes, Veuillez attendre la fin du round"

}

main()
{
trap 'echo "Cartes Jouées round :";awk '"'"'{print}'"'"' round.en.cours' USR2
Init_Robot
trap 'Recuperation_Carte' USR1
while true; do
:
done
}

main
```