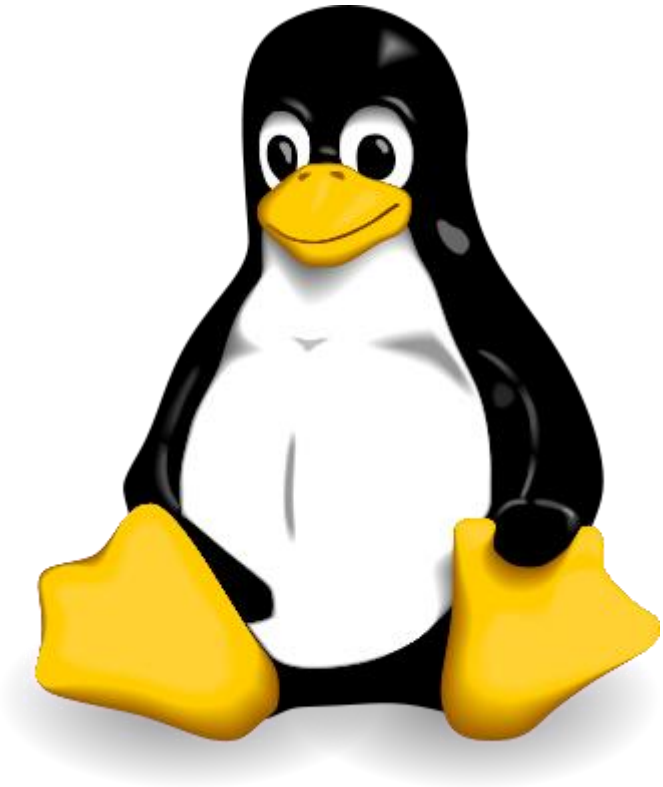
Rapport du projet Unix

7 Février 2012



LEPOT Florian – FABRE Maxime
Groupe D

Sommaire

I. Introduction 3
II. Répartition du travail 4
A. Projet 4
B. Rapport 4
III. Présentation générale 5
A. Objectifs 5
B. Architecture du projet 5
IV. Mise en œuvre du projet 7
A. Génération de la base du jeu 7
1. Le labyrinthe 7
2. Le personnage 7
B. Déroulement des actions 8
1. Armes 8
2. Potions 8
3. Combats 9
4. Enigmes 9
5. Déplacement du personnage 9
C. Fin du jeu 10
1. Gagner 10
2. Perdre 10
D. Utilitaires 10
1. Sauvegarde / Chargement 11
2. Menu accessible à tout moment 11
V. Améliorations 12
A. Mise en page 12
1. L'affichage 12
2. Les couleurs 12
VI. Conclusion 13
VII. Annexes 14
1. Fichier de base 14
2. Sources 15

I. Introduction

Pour ce premier projet d'Unix, nous devions développer un mini-jeu en respectant divers critères fournis dans le cahier des charges. Ce programme ne nécessitait aucune intégration graphique obligatoire, nous l'avons donc développé en console. Le programme consistait à faire déplacer un personnage dans le but de trouver la sortie, d'un labyrinthe généré, sans mourir.

Après avoir programmé sous windows tout au long de la première année à l'EFREI. Ce projet avait pour but de nous faire acquérir les connaissances de bases du bash Unix. Nous avons ainsi pu voir les possibilités et la puissance de ce langage. De plus nous avons beaucoup appris sur la manière à réfléchir et visualiser un programme avant de se lancer dans le code, ceci dans le but d'éviter d'obtenir un programme trop lourd, non optimisé, et donc, lent.

II. Répartition du travail

A. Projet

	Maxime	Florian
Générer le labyrinthe et le personnage	X	X
Déplacement du personnage		X
Déroulement des actions dans une salle	X	
Script de combat	X	
Gagner ou perdre une partie		X
Sauvegarde et chargement de partie		X
Menu accessible à tout moment		X
Mise en page	X	

B. Rapport

	Maxime	Florian
Introduction	X	
Présentation générale	X	
Mise en œuvre du projet	X	X
Améliorations	X	X
Conclusion		X
Mise en page		X
Annexe		X

III. Présentation générale

A. Objectifs

Ce projet d'Unix a pour but de réaliser un mini-jeu de type RPG, dans lequel l'utilisateur incarne un héros se déplaçant dans un labyrinthe, lui réservant bien des surprises. Le personnage doit pouvoir se déplacer facilement, pouvoir affronter différents ennemis, récupérer des armes et potions. La partie pourra être sauvegardée et chargée à tout moment. La partie se termine lorsque le héros meurt ou trouve la sortie.

B. Architecture du projet

Avant de nous lancer dans ce projet assez conséquent et dans son algorithme, nous avons dû prendre une décision quant à l'architecture du projet. En effet, prévoyant un script assez long, nous ne voulions pas tout mettre dans un seul fichier, pour des raisons de clarté et de simplicité. Le projet devient ainsi beaucoup plus simple à débuggé et à modifier.

La racine du Projet englobe deux dossiers :

- **Sources** : Contient tous les fichiers de générations et d'actions des différents éléments du Jeu.
    - Fichiers de génération :
        - genererLabyrinthe.sh
        - genererPersonnage.sh
    - Fichiers bibliothèques
        - biblioArmes.txt
        - biblioDescriptions.txt
        - biblioEnigmes.txt
        - biblioMonstres.txt
        - biblioPotions.txt
    - Fichiers d'action dans une salle
        - actionArmes.sh
        - actionEnigmes.sh
        - actionPassages.sh
        - actionPotions.sh
        - charger.sh
        - fonctions.sh
        - jeu.sh
        - sauvegarder.sh
        - scriptCombat.sh

- **Labyrinthe** : Contient le labyrinthe, ainsi que les différents fichiers temporaires, lié au héros, qui seront supprimé après chaque fin de partie.
    - Personnage : fiche du personnage (accessible à n'importe quel moment)
    - Armes : arme présente dans la salle
    - Enigme : énigme de la salle
    - Ennemis : monstre à combattre dans la salle
    - Potions : potion présente dans la salle
    - Description : description de la salle

IV. Mise en œuvre du projet

A. Génération de la base du jeu

Pour ce projet, plusieurs éléments de base nous étaient imposés, notamment pour la génération du labyrinthe.

1. Le labyrinthe

La structure du labyrinthe devait être fixe, nous avons donc crée un labyrinthe relativement complexe dans le fichier **generationLabyrinthe.sh**, en respectant les conditions qui nous étaient imposées, à savoir le nombre de pièces minimale qui est définie à 20 ainsi que la profondeur d'arbre qui devait être de minimum 5.

Une fois la structure du labyrinthe établie, il nous fallait placer la sortie, de façon aléatoire. Pour cela, nous nous déplaçons aléatoirement dans le labyrinthe, en utilisant la variable **$RANDOM** couplée à un modulo pour choisir un répertoire au hasard. Et nous faisons cela jusqu'à ce que nous soyons placés dans un dossier sans sous répertoire, et nous y plaçons alors la sortie.

Nous devions ensuite remplir le labyrinthe avec différents fichiers : **armes**, **description**, **potions**, **enigme** et **ennemis**. Pour cela, nous utilisons également la variable aléatoire pour sélectionner une ligne dans la bibliothèque du fichier correspondant.

2. Le personnage

Dans ce labyrinthe, nous avons un fichier personnage, dans lequel sont placées toutes les informations concernant le personnage :

- Son nom
- Sa description
- Ses points de vie actuels et maximum
- Son arme
- Sa localisation

Pour son nom et sa description, c'est l'utilisateur qui les saisit lors du lancement de la partie, pour les points de vie, ceux-ci ont une base fixe à laquelle on rajoute une partie aléatoire, comprise entre 0 et 20 HP.

Pour l'arme, on propose un choix au joueur parmi 3 possibilités, et on place l'arme correspondante dans le fichier **personnage**.

Pour la localisation, on place le chemin absolu de l'endroit où se trouve le personnage dans ce même fichier, puisque celui-ci ne se déplace pas.

B. Déroulement des actions

Le joueur se déplace de salle en salle, et à chacune d'elle différentes actions s'offre à lui.

1. Armes

Plusieurs armes sont générées un peu partout dans le labyrinthe. Il y a précisément 45% de chance de trouver une arme dans une pièce. Notre jeu comporte une majorité d'ennemis bien plus forts que le héros au début de la partie.

Pour espérer survivre il faudra donc trouver le plus rapidement de meilleurs armes. Ainsi lorsque le joueur trouvera une arme, automatiquement celle-ci viendra remplacer l'ancienne. A condition qu'elle est des points d'attaque (PA) plus élevés que l'ancienne. Dans le cas contraire l'arme ne sera pas prise par le joueur. C'est un système voulu en relation avec le type de difficultés choisi.

Le Jeu comporte plusieurs armes, toutes contenues dans **biblioArmes**. Les armes seront donc placées aléatoirement dans le labyrinthe. Voici la liste des armes :

- Cure dent : 5 PA
- Bout de bois : 5 PA
- Cuillère : 5 PA
- Pelle en plastique : 10
- Batte en caoutchouc : 15 PA
- Dague rouillée en fer : 25 PA
- Arc souple : 30 PA
- Epee Spartiate : 200 PA

Quand le joueur arrive dans la salle, un script préalable va vérifier si le fichier arme crée dans le répertoire ou le personnage se trouve contient une arme. Si c'est le cas, soit :

- Arme plus forte ou égale que la précédente : le joueur prend l'arme et supprime le contenu du fichier, afin d'éviter de reprendre l'arme si l'on repart en arrière, et de signifier que l'arme à été prise.

Arme moins forte que la précédente : le joueur laisse l'arme et le fichier arme n'est pas vidé

2. Potions

En ce qui concerne les potions, on donne 35% de chance d'en trouver dans une pièce. Chaque potion régénère un nombre de points de vie au héros déterminé à l'avance. Evidemment un contrôle est fait à chaque régénération afin d'éviter que la vie du personnage ne dépasse la capacité maximal.

Toutes les potions sont contenues dans **biblioPotions.** Liste des potions :

- Potion seum : 1 PV
- Potion simple : 15 PV
- Potion rouge : 25 PV
- Potion blanc à pois rouge : 50 PV
- Potion verte : 150 PV

Comme pour les armes, si la potion est prise le fichier **potion** sera vidé de son contenu. Sinon non.

3. Combats

Le script de combat est une des parties importantes du projet. Il n'y a que deux options pour que le joueur perde. Soit il perd un combat face à un ennemi, soit il répond mauvais à une énigme. Lors d'un combat les coups de chaque joueur sont donnés tour après tour. Le héros commence obligatoirement. Le script lit le fichier ennemis ou le héros se trouve actuellement, détecte le monstre et débute le combat.

Le personnage a 50 % de chance que sont attaque touche l'ennemi. Pour réaliser ceci, on calculera a chaque passage de la boucle (tant que le joueur ou l'ennemi n'est pas mort) la probabilité qu'il touche son adversaire avec la fonction **$RANDOM** modulé par 2. Ce qui nous donne que deux résultats possible 0 ou 1. On fait la même chose pour l'ennemi, mais cette fois-ci avec un modulo de 3. Ce qui lui donne 33% de chance de vous toucher. Ainsi si la variable aléatoire du joueur comme du monstre est égal à 1, l'attaque sera portée. Toutes ces règles étaient demandées dans le cahier des charges.

Avec les points d'attaque de votre arme ou de l'ennemi, souvent les points de vie sont ramenés en dessous de zéro. Une condition surveillera donc ce paramètre afin d'avoir au minimum une vie du héros ou du monstre égale à zéro. Ceci permet par la suite de mieux gérer le projet.

Par volonté de clarté, les combat ne sont pas détaillés, mais une pause de 2 secondes est faite afin de pouvoir lire l'énoncé de l'attaque avant de connaitre l'issus de celui-ci.

Le script n'est appelé que si le fichier ennemi n'est pas vide. Une fois l'ennemi battu le fichier devient vide pour les mêmes raisons que les potions et armes.

4. Enigmes

Durant son périple le héros devra se confronter à des énigmes. Il y a une probabilité de 30 % de tomber sur une énigme. Chaque énigme est sous forme de QCM. Si la réponse est mauvaise, le héros mourra, sinon il aura la possibilité de passer à la pièce suivante. Le choix de la réponse est sécurisées comme toutes les fois ou le joueur est appelé à écrire du texte ou choisir une option.

De même que précédemment le contenu du fichier **énigme** sera effacé si la réponse donnée par le l'utilisateur est bonne. Toutes les énigmes sont présentes dans **biblioEnigmes**. Elles sont placées aléatoirement dans le labyrinthe.

5. Déplacement du personnage

Une fois toutes les actions précédentes vérifiées et exécutées, on passe au déplacement du personnage.

Pour cela, on liste les différents sous-répertoires et on propose au joueur de se déplacer dans celui qu'il veut au moyen d'un **switch**. On lui permet également de revenir à la salle précédente lorsqu'il n'est pas dans l'entrée, et pour cela, on vérifie que le nom du dossier actuel n'est pas **entree**, au moyen de la variable **${PWD##*/}**.

Pour finir, on met à jour l'information **Localisation** dans le fichier **personnage** grâce à la fonction **majPerso** qui permet de le faire facilement.

C. Fin du jeu

Il arrive un moment où la partie est terminée, et nous les avons définis, il n'y a que 2 options possibles :

- Le joueur est arrivé à la sortie du labyrinthe,
- Le joueur est mort dans celui-ci.

1. Gagner

Dans le premier cas, celui où le joueur a réussi à traverser le labyrinthe et à trouver la sortie sans mourir, la partie est gagnée.

Pour vérifier cela, on repère le nom du dossier dans lequel est le personnage, si le nom est égal à sortie, alors on quitte la boucle qui entraine toutes les actions dans la salle et on initialise une variable **win** à 1.

Après la boucle, on vérifie donc la valeur de **win**, et si c'est 1, on affiche un message pour signifier au joueur qu'il a gagné et on lui affiche le menu principal pour qu'il puisse rejouer ou quitter.

2. Perdre

Dans le jeu, on peut perdre de 2 façons différentes :

- On meurt tué par un monstre : Dans ce cas-là, on initialise la variable **win** à 0 et une variable **mort** à 1. Ceci dans le but de pouvoir savoir dans quel cas on est à la fin du jeu, en sortant de la boucle,
- On échoue lors de la résolution d'une énigme : Dans ce cas-ci, on initialise la variable **win** à 0, **mort** à 0 et on quitte la boucle.

A la sortie de la boucle, on vérifie donc la valeur de la variable **mort** et on affiche le bon message suivant le cas.

D. Utilitaires

Il nous était demandé, pour ce projet, de rajouter un système de sauvegarde et de chargement de partie, mais également de permettre à l'utilisateur d'accéder au menu principal à tout moment.

1. Sauvegarde / Chargement

La première chose que nous avons dû faire a été de permettre au joueur de sauvegarder la partie en cours. Pour permettre ceci, nous archivons simplement le contenu du dossier **labyrinthe**, qui contient aussi bien les informations du personnage (et donc sa localisation actuelle), et le contenu du labyrinthe.

Toutes les informations étant sauvegardées, nous n'avons pas besoin de plus.

La deuxième étape a été de prévoir le chargement d'une partie enregistrée. Pour cela, nous proposons au joueur, à nouveau via le menu principal, de charger la partie. On commence alors par vérifier qu'il existe une sauvegarde. Si c'est le cas, on supprime le dossier labyrinthe et on désarchive la sauvegarde. On peut enfin exécuter **jeu.sh** qui va lancer le jeu là où il s'était arrêté, en se basant sur la localisation du joueur indiquée dans le fichier **personnage**.

2. Menu accessible à tout moment

Le deuxième utilitaire imposé était le menu principal accessible à tout moment. Pour permettre cela, nous avons créé une fonction **afficherMenu** qui effectue cette action.

Cela nous permet de l'appeler lorsque nous en avons besoin, et nous permet de gérer la saisie sécurisée plus facilement.

Ce menu est différent selon l'étape dans laquelle on est dans le jeu.

Lorsque l'on a pas encore crée de partie, le menu est le suivant :

```
1.  Lancer une nouvelle partie
2.  Charger une partie existante
3.  Quitter
```

Et lorsqu'une partie est en cours et qu'on appelle le menu, ce sont ces choix qui s'affichent :

```
1.  Reprendre la partie
2.  Sauvegarder la partie
3.  Lancer une nouvelle partie
4.  Voir la fiche du personnage
5.  Quitter
```

Pour permettre la différentiation des 2 cas, on passe tout simplement un argument à la fonction. S'il n'y a pas d'argument passé, on affiche le premier menu, sinon le deuxième.

V. Améliorations

A. Mise en page

1. L'affichage

L'option du choix graphique n'était pas obligatoire. Nous n'avions pas prévu de faire d'affichage graphique 2D en ASCII par soucis de temps et de rendus. Le jeu se déroule dans le terminal, il est donc impératif que l'utilisateur se retrouve facilement dans le jeu, puisse accéder aux menus rapidement. Pour cela, on a essayé de simplifier et de synthétiser au maximum les informations pour un affichage clair et lisible.

Dans chaque salle les étapes se passent à la suite :

- Récupération d'une arme (si le joueur en trouve une)
- Récupération d'une potion (si le joueur en trouve une)
- Combat contre un ennemi présent dans la salle
- Enigme posé au joueur
- Demande la direction au joueur

Toutes ces étapes se déroulent rapidement, et on se retrouve très vite avec un affichage rempli, et une quantité d'information à lire importante. Pour remédier à cela, nous avons mis un temps d'attente de 2 secondes entre chaque étape, afin de laisser le temps à l'utilisateur de lire au fur et à mesure ce qui s'inscrit à l'écran.

2. Les couleurs

Une fois l'information synthétisée, la lecture peut se faire plus rapidement grâce à l'incrustation de couleurs dans le texte. Chaque couleur représentera quelque chose de précis

- **Rouge** : nom des ennemis, action échoué (combat perdu, énigme non-trouvée) et points d'attaque du joueur et du monstre
- **Orange** : demande de saisir du texte à l'utilisateur
- **Vert** : action réussi (combat gagné, énigme trouvée), HP du joueur et de l'ennemi
- **Bleu** : armes trouvée ou actuellement en main
- **Magenta** : titre des étapes (combat, arme, énigme …)

Ainsi en un seul coup d'œil, l'utilisateur repère les informations importantes sans devoir chercher pendant des heures, dans le texte présent à l'écran. De plus à chaque salle un récapitulatif du joueur (HP restant et arme en main) est donné.

VI. Conclusion

Pour finir sur ce projet, nous avons aussi beaucoup appris quant au scripting shell bash, et cela nous a montré que ce langage avait une syntaxe assez difficile à manipuler la première fois, vraiment stricte, mais qu'on peut toutefois créer des scripts ludiques.

Nous retenons de ce travail les problèmes qu'engendrent le fait de travailler en groupe, et notamment celui de l'organisation, la répartition du travail et l'adaptation du code de chacun pour qu'il puisse fonctionner avec l'ensemble du projet, d'autant plus sur un projet de scripting d'un langage non maitrisé et dont nos connaissances sont minimes.

VII. Annexes

1. Fichier de base

Labyrinthe.sh

```
#!/bin/bash

rm labyrinthe/
clear
Base=`pwd`
source sources/fonctions.sh
new=1


echo "-----------------------------------"
echo -e "\033[31m      Projet Unix\033[00m "
echo -e "     Maxime \033[31m&\033[00m Florian"
echo "-----------------------------------"
echo ""
echo "Ce jeu est optmise pour une console en plein ecran"
echo ""

afficherMenu
```

2. Sources

genererLabyrinthe.sh

```
rm -rf labyrinthe/entree

#########################
# On génère le labyrinthe
#########################

echo "Génération du labyrinthe en cours.."

mkdir -p labyrinthe/entree/salle1/salle11/salle111/salle1111/salle11111
mkdir -p labyrinthe/entree/salle1/salle11/salle111/salle1111/salle11112
mkdir -p labyrinthe/entree/salle1/salle11/salle111/salle1112/salle11121
mkdir -p labyrinthe/entree/salle1/salle11/salle111/salle1112/salle11122
mkdir -p labyrinthe/entree/salle1/salle11/salle111/salle1112/sortie
mkdir -p labyrinthe/entree/salle1/salle11/salle112/salle1121/salle11211
mkdir -p labyrinthe/entree/salle1/salle11/salle112/salle1121/salle11212
mkdir -p labyrinthe/entree/salle1/salle11/salle113/salle1131/salle11311
mkdir -p labyrinthe/entree/salle1/salle11/salle113/salle1131/salle11312
mkdir -p labyrinthe/entree/salle1/salle11/salle113/salle1132/salle11321
mkdir -p labyrinthe/entree/salle1/salle12/salle121

mkdir -p labyrinthe/entree/salle2/salle21/salle211
mkdir -p labyrinthe/entree/salle2/salle21/salle212/salle2121
mkdir -p labyrinthe/entree/salle2/salle21/salle212/salle2122
mkdir -p labyrinthe/entree/salle2/salle22/salle221/salle2211/salle22111
mkdir -p labyrinthe/entree/salle2/salle22/salle221/salle2212
mkdir -p labyrinthe/entree/salle2/salle22/salle222
mkdir -p labyrinthe/entree/salle2/salle22/salle223/salle2231
mkdir -p labyrinthe/entree/salle2/salle22/salle223/salle2232/salle22321
mkdir -p labyrinthe/entree/salle2/salle22/salle223/salle2232/salle22322
mkdir -p labyrinthe/entree/salle2/salle22/salle223/salle2232/salle22323
mkdir -p labyrinthe/entree/salle2/salle23/salle231

ln -s $Base/labyrinthe/entree/salle1/salle11/salle113/salle1131/salle11312
labyrinthe/entree/salle1/salle11/secret
ln -s $Base/labyrinthe/entree/salle2/salle23/salle231
labyrinthe/entree/salle1/salle12/salle121/secret
ln -s $Base/labyrinthe/entree/salle1
labyrinthe/entree/salle2/salle22/salle223/salle2231/secret
ln -s $Base/labyrinthe/entree/
labyrinthe/entree/salle1/salle11/salle113/salle1131/salle11311/secret

#########################
# On place la sortie
#########################

base=`pwd`
nb=1
cd labyrinthe/entree/

while [ $nb != 0 ]
do
        nb=`ls | wc -l`
        nbsecret=`ls | grep secret | wc -l`
        let "nb -= $nbsecret"
        if [ $nb != 0 ]
        then
                let "rep = 1 + $RANDOM % $nb"
                cd `ls | grep salle*$rep`

        fi
done

mkdir sortie
echo -e "\033[32m -> Sortie placée \033[00m "
```

```
cd $base

################################
# On place le contenu des salles
################################

racine=`pwd`
cd labyrinthe/entree/
base=`pwd`

echo "Placement des fichiers du labyrinthe.."
for i in `find -type d`
do
        cd $i
        touch armes description potions enigme ennemis

        # On place les description

        let "desc = 1 + $RANDOM % 4"
        Desc=`sed -n $desc"p" $racine/sources/biblioDescriptions.txt`
        echo $Desc > description

        # On place les armes

        let "rand = $RANDOM % 101"
        let "arme = 3 + $RANDOM % 6"
        if [ $rand -le 45 ]; then
                Arme=`cat $racine/sources/biblioArmes.txt | grep $arme: | cut -d ':' -
f 2,3,4`
                echo $Arme > armes
        fi

        # On place les potions

        let "rand = $RANDOM % 101"
        let "potion = 1 + $RANDOM % 5"
        if [ $rand -le 35 ]; then
                Potion=`sed -n $potion"p" $racine/sources/biblioPotions.txt`
                echo $Potion > potions
        fi

        # On place les enigmes

        let "rand = $RANDOM % 101"
        let "enigme = 1 + $RANDOM % 8"
        if [ $rand -le 30 ]; then
                Enigme=`sed -n $enigme"p" $racine/sources/biblioEnigmes.txt`
                echo $Enigme > enigme
        fi

        # On place les monstres

        let "rand = $RANDOM % 101"
        let "monstre = 1 + $RANDOM % 9"
        if [ $rand -le 70 ]; then
                Monstre=`sed -n $monstre"p" $racine/sources/biblioMonstres.txt`
                echo $Monstre > ennemis
        fi

        cd $base
done
echo -e "\033[32m -> Fichiers placés\033[00m "
cd ../..
```

genererPersonnage.sh

```
echo ""
echo -n -e "\033[33mChoisissez le nom de votre personnage : \033[00m "
read nom

touch labyrinthe/entree/personnage

let "nbPdvMax = 100 + $RANDOM % 20"
echo "nom:$nom" > labyrinthe/entree/personnage
echo "nbPdvMax:$nbPdvMax" >> labyrinthe/entree/personnage
echo "nbPdvActuel:$nbPdvMax" >> labyrinthe/entree/personnage

echo ""
for i in `seq 1 3`;
do
      echo -e "\033[33m$i.\033[00m  `sed -n $i"p" sources/biblioArmes.txt | cut -d
':' -f2` (`sed -n $i"p" sources/biblioArmes.txt | cut -d ':' -f4` PA)"
done

echo ""
echo -n -e "\033[33mChoisissez une arme : \033[00m "
read arme

ARMEV=`cat sources/biblioArmes.txt | grep $arme: | cut -d ':' -f 2,3,4`
echo "Arme:$ARMEV" >> labyrinthe/entree/personnage

echo "Localisation:`pwd`/labyrinthe/entree" >> labyrinthe/entree/personnage
```

jeu.sh

```
win=0
Base=`pwd`
cd labyrinthe/entree
clear
if [ $new = 1 ]; then
	echo "-----------------------------------------------"
	echo -e "-		\033[35mLe jeu demarre !\033[00m		-"
	echo "-----------------------------------------------"
	echo ""
	echo "Bienvenue jeune guerrier ! "
	echo "Ton aventure debute en ce labyrinthe truffe de pieges, de monstres et
de bien d'autres surprises !"
else
	echo "-----------------------------------------------"
	echo -e "-		\033[35mPartie chargee ! \033[00m		-"
	echo "-----------------------------------------------"
	echo ""
	echo "Ta partie precedemment sauvegardee a bien ete chargee !"
	echo "Nous te souhaitons bon retour guerrier !"
fi

REP=${PWD##*/}

while [ $REP != 'sortie' ]; do

	RepFull=`grep Localisation $Base/labyrinthe/entree/personnage | cut -d : -f
2`
	cd $RepFull
	echo ""
	cat description
	echo ""
	REP=${PWD##*/}

	if [ $REP = "sortie" ]; then
		win=1
		break
	fi

	echo "-------------------------------------------------------------------
--"
	echo -e "|	Nom du héro : \033[33m`grep nom
$Base/labyrinthe/entree/personnage  | cut -d : -f 2` \033[00m"
	echo -e "|	HP : \033[32m`grep nbPdvActuel
$Base/labyrinthe/entree/personnage  | cut -d : -f 2` / `grep nbPdvMax
$Base/labyrinthe/entree/personnage | cut -d : -f 2` \033[00m | Arme : \033[34m`grep
Arme $Base/labyrinthe/entree/personnage  | cut -d : -f 2` \033[00m(\033[31m`grep
Arme $Base/labyrinthe/entree/personnage  | cut -d : -f 4` \033[00mPA)"
	echo "-------------------------------------------------------------------
--"

	if [ "$passage" = "1" ]; then
		echo "Aucun passage secret n'a ete trouve"
		echo ""
		passage=0
	fi

	#########################################################################
	# Gestion des armes
	#########################################################################

	let "nb = `cat armes | grep : | wc -l`"

	if [ $nb = "1" ]; then
		source $Base/sources/actionArmes.sh
		sleep 2
	fi
```

```
        ######################################################################
        # Gestion des potions
        ######################################################################

        let "nb = `cat potions | grep : | wc -l`"

        if [ $nb = "1" ]; then
                source $Base/sources/actionPotions.sh
                sleep 2
        fi

        ######################################################################
        # Gestion des combats
        ######################################################################

        let "nb = `cat ennemis | grep : | wc -l`"
        if [ $nb = "1" ]; then
                source $Base/sources/scriptCombat.sh
                sleep 2
                pdvActuel=`grep nbPdvActuel $Base/labyrinthe/entree/personnage  | cut
-d : -f 2`

                if [ $pdvActuel = 0 ]; then
                        mort=1
                        break
                else
                        echo -e "\033[32mCombat gagné !\033[00m"
                        echo -e "Il vous reste \033[32m$pdvActuel\033[00m /
\033[32m`grep nbPdvMax $Base/labyrinthe/entree/personnage  | cut -d : -f 2`\033[00m
HP !"
                fi

        fi

        ######################################################################
        # Gestion des énigmes
        ######################################################################

        let "nb = `cat enigme | grep : | wc -l`"
        if [ $nb = "1" ]; then
                enigme=1
                source $Base/sources/actionEnigmes.sh
                sleep 2
                if [ "$enigme" = "0" ]; then
                        win=0
                        mort=0
                        break
                else
                        echo "Vous pouvez dès à présent choisir la salle suivante.."
                fi

        fi



        ######################################################################
        # Déplacement du personnage
        ######################################################################

        if [ $REP = 'entree' ]; then
                gauche='salle1'; droite='salle2'
        else
                gauche=$REP"1"; droite=$REP"2"; face=$REP"3";
        fi

        let "nbRep = `find -maxdepth 1 -type d | wc -l` - 1"
        let "sortie = `find -maxdepth 1 -type d | grep sortie | wc -l`"
```

```
        if [ $sortie = '1' ]; then
                gauche="sortie"
        fi

        echo ""
        echo -e "--------- \033[35mDirection\033[00m---------"

        case $nbRep in
                0) echo "Vous etes dans un cul de sac !";;
                1) echo "La piece ne dispose que d'une porte sur votre gauche"; echo -
e "\033[33m1.\033[00m Prendre la porte de gauche";;
                2) echo "La salle dispose de $nbRep sorties :"; echo -e
"\033[33m1.\033[00m Prendre la porte de gauche"; echo -e "\033[33m2.\033[00m
Prendre la porte de droite";;
                3) echo "La salle dispose de $nbRep sorties :"; echo -e
"\033[33m1.\033[00m Prendre la porte de gauche"; echo -e "\033[33m2.\033[00m
Prendre la porte de droite"; echo -e "\033[33m3.\033[00m Continuer tout droit";;
        esac
        if [ $REP != 'entree' ]; then
                echo -e "\033[33m9.\033[00m Chercher les passages secrets"
                echo -e "\033[33m0.\033[00m Revenir a la salle precedente"
        fi
        echo -e "\033[33mM.\033[00m Revenir au menu principal"

        echo ""
        echo -n -e "\033[33mOu voulez-vous aller ? \033[00m"
        read direction

        case $direction in
                1) cd $gauche;;
                2) cd $droite;;
                3) cd $face;;
                9) chercherPassage; passage=1;;
                0) cd ..;;
                m | M) afficherMenu 1;;
                *) ;;
        esac


        localisation=`pwd`
        majPerso Localisation $localisation

        clear
done

clear
echo "-----------------------------------------------"
echo -e "-                \033[35mJeu termine !\033[00m                -"
echo "-----------------------------------------------"
echo ""

if [ $win = 1 ]; then
        echo "Felicitations jeune guerrier !"
        echo "Tu auras finalement reussi a traverser le labyrinthe et a en sortir
vivant !"
        echo "Tu peux maintenant decider de retenter l'experience ou de quitter le
jeu.."
elif [ $win = 0 ] && [ $mort = 1 ]; then
        echo -e "Un monstre (\033[31m`cat ennemis | cut -d : -f1`\033[00m
[\033[32m`cat ennemis | cut -d : -f3` \033[00mHP | \033[31m`cat ennemis | cut -d :
-f4`\033[00m PA])  aura finalement eu raison de ta peau"
        echo "Tu peux retenter ta chance ou abandonner definitivement et rester sur
cet echec cuisant.."
else
        echo "N'ayant pas réussi à résoudre l'énigme, une horde d'ennemis s'est
abbatue sur toi et t'as tué"
        echo "Tu peux retenter ta chance ou abandonner definitivement et rester sur
cet echec cuisant.."
```

```
fi

afficherMenu
```

actionArmes.sh

```
#!/bin/bash

echo ""
echo -e "--------- \033[35mArme trouvée\033[00m---------"
lieuBase=$Base/labyrinthe/entree
cd $lieuBase

lieuPerso=`grep "Localisation" personnage | cut -d : -f 2`

yourPvArme=`grep Arme personnage | cut -d : -f 4`
cd $lieuPerso

nomArme=`cut -d : -f 1 armes`
describArme=`cut -d : -f 2 armes`
pvArme=`cut -d : -f 3 armes`
echo -e "Vous trouvez une arme : \033[34m$nomArme\033[00m ($describArme) -
\033[31m$pvArme\033[00m PA"

if [ $pvArme -le $yourPvArme ]
then
	echo "L'arme trouvée n'est pas plus puissante que la votre, vous ne la
prenez donc pas !"
else
	echo -e "Votre nouvelle arme actuel est donc : \033[34m$nomArme\033[00m "
	majPerso Arme "$nomArme:$describArme:$pvArme"
fi

echo "" > armes
```

actionEnigmes.sh

```
#!/bin/bash

verif () {
	echo ""
	if [ "$1" = "$2" ]
	 then
		echo -e "\033[32mBonne réponse !\033[00m"
		echo "" > enigme
	else
		echo -e "\033[31mMauvaise réponse !\033[00m"
		enigme=0
	fi
}

echo ""
echo -e "--------- \033[35mEnigme ?\033[00m ---------"

lieuBase=$Base/labyrinthe/entree
cd $lieuBase

lieuPerso=`grep "Localisation" personnage | cut -d : -f 2`
cd $lieuPerso

enigme=`cut -d : -f 1 enigme`

r1=`cut -d : -f 2 enigme`
r2=`cut -d : -f 3 enigme`
r3=`cut -d : -f 4 enigme`
reponse=`cut -d : -f 5 enigme`

echo "$enigme"
echo ""
echo "Veuillez sélectionner votre réponse : "

echo -e "\033[33m1.\033[00m $r1"
echo -e "\033[33m2.\033[00m $r2"
echo -e "\033[33m3.\033[00m $r3"

echo ""
echo -n -e "\033[33mVotre choix? : \033[00m"
read choice

case $choice in
1) verif "$r1" "$reponse";;
2) verif "$r2" "$reponse";;
3) verif "$r3" "$reponse";;
*) source $Base/sources/actionEnigmes.sh;;
esac
```

actionPassages.sh

```
echo ""
echo "Vous avez trouve un passage secret.."
echo -e "\033[33m1.\033[00m L'emprunter"
echo -e "\033[33m2.\033[00m Continuer sans y faire attention"
echo ""

echo -n -e "\033[33mQue voulez-vous faire ? \033[00m"
read choice

if [ $choice = 1 ]; then
        cd secret/
        echo "Vous avez pris le passage secret et arrivez dans une nouvelle salle !"
fi
```

actionPotions.sh

```
#!/bin/bash

echo ""
echo -e "--------- \033[35mPotion trouvée\033[00m ---------"
lieuBase=$Base/labyrinthe/entree
cd $lieuBase

lieuPerso=`grep "Localisation" personnage | cut -d : -f 2`
pvHero=`grep "nbPdvActuel" personnage | cut -d : -f 2`
pvHeroMax=`grep "nbPdvMax" personnage | cut -d : -f 2`

cd "$lieuPerso"

nomPotion=`cut -d : -f 1 potions`
describPotion=`cut -d : -f 2 potions`
pvPotion=`cut -d : -f 3 potions`
echo -e "Vous trouvez une \033[33m$nomPotion\033[00m ($describPotion) - Vous donne
\033[32m$pvPotion\033[00m HP !"

let "vie = $pvHero + $pvPotion"
if (($vie >= $pvHeroMax))
then
        echo -e "Vos points de vie sont maintenant au maximum :
\033[32m$pvHeroMax\033[00m"
        majPerso "nbPdvActuel" $pvHeroMax
else
        echo -e "Vous avez maintenant \033[32m$vie\033[00m points de vie"
        majPerso nbPdvActuel $vie
fi
echo "" > potions
```

scriptCombat.sh

```
#!/bin/bash

echo ""
echo -e "--------- \033[35mCombat !\033[00m ---------"

lieuBase=$Base/labyrinthe/entree
cd $lieuBase

lieuPerso=`grep "Localisation" personnage | cut -d : -f 2`

yourAtta=`grep "Arme" personnage | cut -d : -f 4`
pvHero=`grep "nbPdvActuel" personnage | cut -d : -f 2`

cd $lieuPerso

nomEnnemis=`cut -d : -f 1 ennemis`
describEnnemis=`cut -d : -f 2 ennemis`
echo -e "\033[31m$nomEnnemis\033[00m ($describEnnemis) vous attaque !"

pvEnnemis=`cut -d : -f 3 ennemis`
attaEnnemis=`cut -d : -f 4 ennemis`
echo -e "Caracteristiques : \033[32m$pvEnnemis\033[00m HP |
\033[31m$attaEnnemis\033[00m PA"

echo ""

while [ $pvHero -gt 0 ] && [ $pvEnnemis -gt 0 ]
do
	let "yourRand = $RANDOM % 2"
	let "ennemisRand = $RANDOM % 3"

	if [ $yourRand = 1 ]
	then
		let "pvEnnemis = $pvEnnemis - $yourAtta"
		if [ $pvEnnemis -le 0 ]
		then
			pvEnnemis=0
			echo "" > ennemis
			break
		fi
	fi

	if [ $ennemisRand = 1 ]
	then
		let "pvHero = $pvHero - $attaEnnemis"
		if [ $pvHero -le 0 ]
		then
			pvHero=0
		fi
	fi
done

majPerso nbPdvActuel $pvHero
```

fonctions.sh

```
lancerJeu () {
	cd $Base
	echo ""
	source $Base/sources/genererLabyrinthe.sh
	source $Base/sources/genererPersonnage.sh
	source sources/jeu.sh
}

sauverPartie () {
	source $Base/sources/sauvegarder.sh
}

chargerPartie () {
	source $Base/sources/charger.sh
}

afficherPerso () {
	clear
	echo "-------------------------------------------"
	echo -e "-          \033[35mFiche personnage\033[00m             -"
	echo "-------------------------------------------"
	echo ""

echo "   / \     "
echo -e "  _\_/_          Nom : \033[33m`grep nom
$Base/labyrinthe/entree/personnage  | cut -d : -f 2`\033[00m"
echo " /     \    "
echo " ||    ||    "
echo -e " \\\---//         Points de vie : \033[32m`grep nbPdvActuel
$Base/labyrinthe/entree/personnage  | cut -d : -f 2`\033[00m / \033[32m`grep
nbPdvMax $Base/labyrinthe/entree/personnage  | cut -d : -f 2`\033[00m HP"
echo "   | _ |     "
echo "   || ||     "
echo -e "   || ||          Arme : \033[34m`grep Arme
$Base/labyrinthe/entree/personnage  | cut -d : -f 2`\033[00m (\033[31m`grep Arme
$Base/labyrinthe/entree/personnage  | cut -d : -f 4`\033[00m PA)"
echo "   (/ \)     "
echo ""
	afficherMenu 0
}

chercherPassage () {

	let "nb = `ls | grep secret | wc -l`"
	if [ $nb -eq "1" ]; then
		source $Base/sources/actionPassages.sh
		passage=0
	fi
}

majPerso () {

	KEY=$1
	VALUE=$2

	if [ $KEY = "Arme" ]; then
		BASEVALUE=`grep $KEY $Base/labyrinthe/entree/personnage  | cut -d : -f
2,3,4`
	else
			BASEVALUE=`grep $KEY $Base/labyrinthe/entree/personnage  | cut
-d : -f 2`
	fi

	sed -i -e "s|$KEY:$BASEVALUE|$KEY:$VALUE|g"
$Base/labyrinthe/entree/personnage
}
```

```
afficherMenu () {
	if [ $# != 0 ] && [ "$1" = "1" ]; then
		clear
	fi

	echo ""
	echo -e "\033[35mMenu Principal\033[00m"
	echo ""

	if [ $# = 0 ]; then
		echo -e "\033[33m1.\033[00m  Lancer une nouvelle partie"
		echo -e "\033[33m2.\033[00m  Charger une partie existante"
		echo -e "\033[33m3.\033[00m  Quitter"
	else
		echo -e "\033[33m1.\033[00m  Reprendre la partie"
		echo -e "\033[33m2.\033[00m  Sauvegarder la partie"
		echo -e "\033[33m3.\033[00m  Lancer une nouvelle partie"
		echo -e "\033[33m4.\033[00m  Voir la fiche du personnage"
		echo -e "\033[33m5.\033[00m  Quitter"
	fi

	echo ""
	echo -n -e "\033[33mVotre choix? : \033[00m "
	read choice


	if [ $# = 0 ]; then
		case $choice in
		1) lancerJeu ;;
		2) chargerPartie;;
		3) exit;;
		*) afficherMenu;;
		esac
	else
		case $choice in
		1)  ;;
		2) sauverPartie ;;
		3) lancerJeu ;;
		4) afficherPerso ;;
		5) exit ;;
		*) afficherMenu 1;;
		esac
	fi
}
```

sauvegarder.sh

```
clear
echo "-------------------------------------------"
echo -e "-                  \033[35mSauvegarde\033[00m                  -"
echo "-------------------------------------------"
echo ""
ref=`pwd`

cd $Base
tar -zcf save.tar.gz labyrinthe > /dev/null 2>&1
cd $ref

echo -e "\033[32mPartie sauvegardee !\033[00m"
afficherMenu 0
```

charger.sh

```
let "verif = `ls | grep save.tar.gz | wc -l`"

if [ $verif = 1 ]; then
        echo "Chargement de la partie en cours.."
        rm -rf labyrinthe/
        tar xzf save.tar.gz
        new=0
        source sources/jeu.sh
else
        echo -e "\033[31mIl n'y a aucune sauvegarde\033[00m"
        afficherMenu
fi
```