

2012

Damos L@bs.

*Aurélien David & Adrien
Ramos*



[RAPPORT DE PROJET UNIX]

Table des Matières

Introduction.....	1
Choix Préliminaires.....	2
Implémentation du programme.....	3
Conclusion	4
Annexe:	5
Bash.....	5
Github.....	5
Sources	5
labyrinthe.sh	6
core.sh	7
init.sh.....	8
game.sh.....	10
actions.sh	12

Introduction

Avant même d'étudier à l'EFREI, créer un RPG est souvent notre objectif lorsque l'on apprend à programmer. Ce projet fût la chance pour d'allier apprentissage du Bash et plaisir.

Il nous a fallu manier un environnement UNIX, fort heureusement nous en avions déjà l'habitude ce qui nous a facilité la tâche. Pour concevoir l'univers du jeu nous avons été grandement inspirés par l'univers de Donjon et Dragon, surtout pour ce qui est des monstres.

Choix Préliminaires

Le projet a été réalisé en Bash, langage de prédilection d'UNIX sans utiliser d'IHM (interface homme-machine). Ces choix nous ont obligé à réaliser une interface utilisateur textuelle qui devait pourtant rester complète et agréable. Nous avons cherché à reproduire un modèle MVC afin de faciliter l'organisation du code, cependant celui-ci est assez limité par le langage qui n'est pas orienté objet. Nous avons également réfléchi à la création d'une interface ASCII mais nous avons préféré ne pas suivre cette idée jusqu'au bout afin de privilégier le fonctionnement du projet à l'interface graphique.

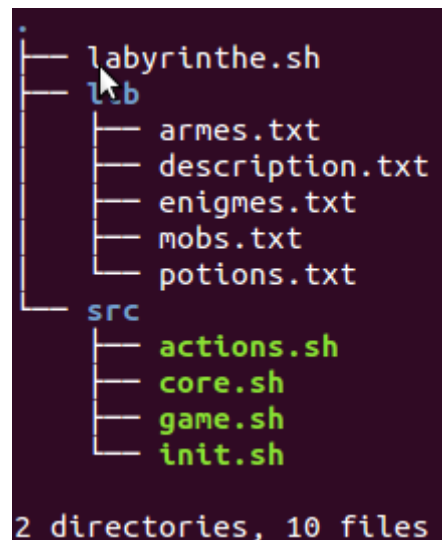
Afin de concevoir ce projet facilement nous avons utilisé le gestionnaire de version Git. Le système de commit nous permettant de savoir sur quelle partie du code chacun des membres du binôme avait travaillé et de comprendre les modifications. De plus l'utilisation de différentes branches nous a permis d'effectuer des tests pour chercher une meilleure performance sans nuire à la branche principale du programme qui se devait d'être à 100% fonctionnelle.

Le code source a été rédigé en anglais, sauf pour les commentaires qui adoptent eux la convention utilisée en TP de Structure De Données, semblable à Javadoc.

Pour réaliser notre projet nous avons commencé par adopter une bonne syntaxe, afin de faciliter le développement en Bash qui diffère de nombreux langages. Il a également été nécessaire d'étudier à nouveau plusieurs commandes souvent déjà vu en TP.

Implémentation du programme

Profitant du modèle MVC, nous avons organisé notre programme en différents fichiers, afin d'adopter une structure claire:



- **Labyrinthe.sh**, contient le lanceur du jeu ainsi que le menu.
- Le dossier **lib** contient la bibliothèque du jeu:

Armes, description des pièces, énigmes, monstres et potions.

- Le dossier **src** contient les sources nécessaires au fonctionnement du programme:
 1. **Core.sh**, contient des éléments essentiels au fonctionnement du projet comme la création du personnage, la sauvegarde.
 2. **Game.sh**, contient le déroulement de chacun des tours de la partie.
 3. **Actions.sh**, regroupe toutes les actions possibles de l'utilisateur, attaquer, boire une potion...
 4. **Init.sh**, permet la création de la carte du jeu ainsi que les différents éléments qui la compose.

Conclusion

Ce projet nous aura permis de développer nos compétences en bash, qui est le langage de prédilection des scripts mais également de découvrir qu'il était possible de faire un jeu très sympa avec 300 lignes de code. Nous avons hâte d'utiliser ces nouvelles connaissances afin, par exemple d'administrer un serveur Linux.

Annexe:

Bash

" Bash, acronyme de Bourne-again shell, est le shell du projet GNU. Son nom est un jeu de mots sur le nom du shell historique d'Unix, le Bourne shell, Bash lui apporte de nombreuses améliorations, provenant notamment du Korn shell et du C shell. Il est l'interprète par défaut sur de nombreux Unix libres, notamment sur les systèmes GNU/Linux."

Github

L'intégralité du gestionnaire de version du projet, historique, etc est disponible de façon open source à l'adresse suivante:

<https://github.com/spyl94/Projet-Unix>

Sources

- http://www.arachnoid.com/linux/shell_programming.html
- <http://tldp.org/LDP/abs/html/>

labyrinthe.sh

```
#!/bin/bash
# Projet: Unix
# Authors: Spyl & Alexian: Damos l@bs.
# Github: https://github.com/spyl94/Projet-Unix
echo A Maze Ing by Damos l@bs
DEF_PATH=`pwd`

source src/core.sh

function menu {
while read -p "Menu:
1) Nouvelle partie
2) Charger une partie
3) Sauvegarder la partie en cours
4) Voir la feuille de personnage
q) Quitter
" choice ; do
    case "$choice" in
        q|Q) exit;;
        1) newchar;;
        2) load;;
        3) save;;
        4) perso;;
        *) echo "Choix non valide..."
    esac
done
}

menu
```

core.sh

```
#Ecrit dans char.txt
function char {
    rm -rf "$DEF_PATH/char.txt"
    for i; do echo $i >> "$DEF_PATH/char.txt"; done
}

#Victoire
function win {
    echo -e "Vous avez réussi à sortir du labyrinthe vivant...\nOsez-vous retenter l'expérience?"; menu
}

#Confirmation de sauvegarde avant de quitter
function quitter {
    while read -p "Voulez vous sauvegarder la partie en cours? (o/n)
    " choice ; do
        case "$choice" in
            n|N|q|Q) exit;;
            o|O) save;;
            *) echo "Choix non valide..."
        esac
    done
}

#Sauvegarde
function save {
    read -p "Veuillez indiquer le nom de votre sauvegarde [save]:
    " save;
    if [ ! $save ]; then save="save"; fi
    cd $DEF_PATH
    tar cvf $save.tar.gz entree/ char.txt 1>/dev/null
    rm -rf entree "char.txt"
    clear; echo "Partie Sauvegardée avec succès."; menu
}

#Charge une partie sauvegardée
function load {
    declare -a save perso
    cd $DEF_PATH; clear; nb=0;
    for i in $(ls *.tar.gz 2>/dev/null); do save[$nb]=$i; let
    "nb+=1";done
    nb=0; for i in "${save[@]}"; do echo $nb) Charger : $i"; let
    "n+=1";done
    echo "M) Menu";
    echo "Q) Quitter";
    read choice
        case "$choice" in
            q|Q) exit;;
            m|M) menu;;
            0|1|2|3|4|5) clear; echo "Chargement en cours...";tar -
    xvzf ${save[$choice]} 1>/dev/null
            nb=0; while read ligne ; do perso[$nb]=$ligne; let
    "nb+=1";

            done < "$DEF_PATH"/char.txt
            cd "${perso[5]}"; clear; game "Partie chargée avec
    succès\n";;
            *) echo "Choix non valide..."
        esac
    clear; menu
}
```

```
#Crée une nouvelle partie
function newchar {
    cd $DEF_PATH; clear;

    read -p "Quel est ton nom jeune aventurier? " nom ; clear
    read -p "Bienvenue, $nom, raconte moi ton histoire..." description ;
clear
    pvmax=$((10 + ($RANDOM % 10)))
    pv=$pvmax
    echo -e "Je vois que tu as $pvmax PV.\n\nPour commencer ton aventure,
choisi une arme dans la liste suivante:"
    echo -e Arme "\E[37;44m1\E[0m:"; cut -d: -f1,2 ./lib/armes.txt | sed
-n '1p'
    echo -e Arme "\E[37;44m2\E[0m:"; cut -d: -f1,2 ./lib/armes.txt | sed
-n '2p'
    echo -e Arme "\E[37;44m3\E[0m:"; cut -d: -f1,2 ./lib/armes.txt | sed
-n '3p'
    echo -e "\033[0m"
    read -p 'Selectionnez le numéro de votre arme:
' nbarme
    case "$nbarme" in
        q|Q) exit;;
        1|2|3) char "$nom" "$description" "$pvmax" "$pv"
                sed -n $nbarme'p' ./lib/armes.txt >> char.txt
                start;;
        *) newchar;;
    esac
}

#Affiche la feuille de personnage
function perso {
    nb=0
    declare -a affichage
    while read ligne ; do
        affichage[$nb]=$ligne
        nb=$((nb+1))
    done < $DEF_PATH"/char.txt"
    echo -e "\033[1;30;47mFiche de personnage :
Nom : ${affichage[0]}
Description : ${affichage[1]}
PV Max : ${affichage[2]}
PV Courrant : ${affichage[3]}
Arme : ${affichage[4]}"
    echo -e "Lieu : "$(basename "${perso[5]}") "\033[0m"
}
```

init.sh

```
#Place les objets dans les salles
function initObjects {
    touch $1/armes.txt $1/enigmes.txt $1/mobs.txt $1/potions.txt
    $1/enigmes.txt $1/description.txt #inutile mais demandé par l'énoncé.
    if [ $((($RANDOM%2)) -eq 0 ) ]; then
        echo $(sed -n $((1 + ($RANDOM % $(cat $DEF_PATH"/lib/armes.txt"
| wc -l))))"p" $DEF_PATH"/lib/armes.txt") >> $1/armes.txt;
    fi
    if [ $((($RANDOM%2)) -eq 0 ) ]; then
        echo $(sed -n $((1 + ($RANDOM % $(cat $DEF_PATH"/lib/mobs.txt"
| wc -l))))"p" $DEF_PATH"/lib/mobs.txt") >> $1/mobs.txt;
```

```

        ((nombreMobs++))
    fi
    if [ $((($RANDOM%2)) -eq 0 )]; then
        echo $(sed -n $((1 + ($RANDOM % $(cat
$DEF_PATH"/lib/potions.txt" | wc -l)))) "p" $DEF_PATH"/lib/potions.txt") >>
$1/potions.txt;
    fi
    if [ $((($RANDOM%2)) -eq 0 )]; then
        echo $(sed -n $((1 + ($RANDOM % $(cat
$DEF_PATH"/lib/enigmes.txt" | wc -l)))) "p" $DEF_PATH"/lib/enigmes.txt") >>
$1/enigmes.txt;
        ((nombreEnigmes++))
    fi
    echo $(sed -n $((1 + ($RANDOM % $(cat $DEF_PATH"/lib/description.txt"
| wc -l)))) "p" $DEF_PATH"/lib/description.txt") > $1/description.txt;
}

#Crée la carte du jeu
function initPlacement {
    find "$DEF_PATH/entree" -type d | (while read A ; do initObjects $A;
done
    if [[ "$nombreMobs" -gt 12 && "$nombreEnigmes" -gt 8 ]]; then echo
"Génération de la carte terminée."; else initPlacement; fi
)
}

#Initialise la carte
function initMap {
    rm -Rf $DEF_PATH/entree
    mkdir -p
$DEF_PATH/entree/{hall/{chambre/{armoire,bibliotheque},cuisine/{coin,garde_
manger/abattoir},salon/{cheminee,etage},escalier/{cave,crypte}},jardin/{cab
ane,allee/foret/{grotte,champignonniere}}}
    if [ $? != 0 ]; then echo "Impossible de créer la carte."; exit; fi

    #sortie:
    rand=$((3 + ($RANDOM % 15 )));
    find "$DEF_PATH/entree" -type d | while read A ; do
    let "cpt+=1"; if [ $rand -eq $cpt ]; then mkdir $A/sortie; fi
    done

    #passage secret:
    rand=$((3 + ($RANDOM % 15 )));
    rand2=$((3 + ($RANDOM % 15 )));
    cpt=0
    cpt2=0
    find "$DEF_PATH/entree" -type d | while read A ; do
    let "cpt+=1"; if [ $rand -eq $cpt ]; then
        find "$DEF_PATH/entree" -type d | while read B ; do
        let "cpt2+=1"; if [ $rand2 -eq $cpt2 ]; then
            ln -s $B $A; #echo "Passage entre $B et $A";
        fi; done
    fi; done

    nombreMobs=0;
    nombreEnigmes=0;
    initPlacement
}

#Initialise la partie
function start {

```

```
initMap
cd entree
pwd >> "$DEF_PATH"/char.txt
clear
game
}
```

game.sh

```
function game {
    echo -en "\e[1;34m$\e[0m"
    nb=0
    declare -a perso pieces mobs potions enigmes armes

    #On récupère les infos du fichier char.txt
    while read ligne ; do perso[$nb]=$ligne; let "nb+=1";
done < "$DEF_PATH"/char.txt
lieu=$(basename "${perso[5]}");
if [ $lieu = sortie ]; then win; fi
declare -i pvMax="${perso[2]}"
declare -i pv="${perso[3]}"
arme="${perso[4]}"
nomArme=$(cut -d : -f 1 $DEF_PATH"/char.txt" | sed -n '5p')
degArme=$(cut -d : -f 3 $DEF_PATH"/char.txt" | sed -n '5p')
nb=0
echo -e "-----\n|${perso[0]}\n|PV:
\e[1;32m$pv/$pvMax\e[0m\n|Arme:  \e[0;34m$nomArme\e[0m\n-----
\n|Lieu:  \e[0;34m$lieu\e[0m"

    mobsCount=0 #On récupère les monstres
    if [ -f "${perso[5]}/mobs.txt ]; then
        while read ligne ; do mobs[$mobsCount]=$ligne; let
"mobsCount+=1";
        done < "${perso[5]}/mobs.txt; fi
    potionsCount=0 #On récupère les potions disponibles
    if [ -f "${perso[5]}/potions.txt ]; then
        while read ligne ; do potions[$potionsCount]=$ligne; let
"potionsCount+=1";
        done < "${perso[5]}/potions.txt; fi
    enigmesCount=0 #On récupère les énigmes disponibles
    if [ -f "${perso[5]}/enigmes.txt ]; then
        while read ligne ; do enigmes[$enigmesCount]=$ligne; let
"enigmesCount+=1";
        done < "${perso[5]}/enigmes.txt; fi
    armesCount=0 #On récupère les armes disponibles
    if [ -f "${perso[5]}/armes.txt ]; then
        while read ligne ; do armes[$armesCount]=$ligne; let
"armesCount+=1";
        done < "${perso[5]}/armes.txt; fi
    nb=0 #On récupère les pièces disponibles
    for i in $(ls -F | grep / 2>/dev/null); do pieces[$nb]=$i; let
"nb+=1"; done

    echo -e "|Arme(s): \e[1;33m  $armesCount\e[0m\n|Monstre(s):
\e[1;33m$mobsCount\e[0m\n|Potion(s): \e[1;33m
$potionsCount\e[0m\n|Enigme(s): \e[1;33m $enigmesCount\e[0m\n-----
----";
```

```

cat description.txt
echo "Actions:"; nb=0;
if [ ! $mobsCount -eq 0 ]; then for i in "${mobs[@]}"; do echo $nb)
Attaquer : $i" | cut -d: -f1,2; let "nb+=1"; done
elif [ ! $enigmesCount -eq 0 ]; then for i in "${enigmes[@]}"; do
echo $nb) Résoudre l'énigme : $i" | cut -d: -f1,2; let "nb+=1"; done
else for i in "${pieces[@]}"; do echo $nb) Aller : $i"; let "nb+=1";
done; fi
if [ ! $armesCount -eq 0 ]; then echo "A) Changer d'Arme"; fi
echo "C) Chercher un Passage Secret"
echo "M) Menu Principal"
[ ! $potionsCount -eq 0 ] && echo "P) Utiliser Potion";
echo "Q) Quitter"
if [ "$lieu" != "entree" ]; then echo "R) Reculer d'une pièce"; fi

read choice
case "$choice" in
a|A) armes;;
q|Q) exit;;
c|C) secret;;
m|M) menu;;
p|P) potion;;
r|R) reculer;;
0|1|2|3|4|5) choice $choice;;
*) echo "Choix non valide..."
esac
#On enregistre la configuration du joueur:
char "${perso[0]}" "${perso[1]}" "${perso[2]}" "$pv" "$arme" `pwd`
game #On relance
}

```

actions.sh

```
#Déplace le personnage dans la pièce reçue en paramètre
function move {
    clear
    if [ "${pieces[$1]}" ]; then cd "${perso[5]}/${pieces[$1]}";
    else echo "Cette pièce n'existe pas"; fi
}

#Fait combattre le personnage avec le mob reçu en paramètre
function fight {
    clear
    nomMob=$(echo ${mobs[$1]} | cut -d: -f1)
    desMob=$(echo ${mobs[$1]} | cut -d: -f2)
    declare -i pvMob=$(echo ${mobs[$1]} | cut -d: -f3)
    declare -i degMaxMob=$(echo ${mobs[$1]} | cut -d: -f4)
    echo -e "Vous entrez en phase de combat avec: $nomMob,\n$desMob";
    while [ "$pvMob" -gt 0 ]; do
        echo -e "Vos Pv: \e[1;33m$pvMob\e[0m | $nomMob Pv:
\e[1;33m$pvMob\e[0m"
        if [ $((RANDOM%2)) -eq 0 ]; then ((pvMob-=degArme)); echo -e
"
        \e[1;32mVous infligez \e[1;33m$degArme\e[1;32m dommages.\e[0m";
        else echo " Vous ratez votre attaque..."; fi
        sleep 1;
        if [ $pvMob -gt 0 ]; then
            if [ $((RANDOM%3)) -eq 0 ]; then ((pv-=degMaxMob));
echo -e "
        \e[1;31m$nomMob vous inflige \e[1;33m$degMaxMob\e[1;31m
dommages.\e[0m";
            else echo -e "
        $nomMob rate son attaque!"; fi
            sleep 1;
        fi
        if [ $pv -le 0 ]; then clear; echo "Vous êtes mort... $nomMob
vous a vaincu."; rm -rf $DEF_PATH/char.txt; menu; fi
        done
        sed $((($1+1))"d" `pwd`/mobs.txt -i
        clear; echo -e "\e[1;34mVous avez vaincu $nomMob !\e[0m"
    }

#Fait résoudre au personnage l'énigme reçue en paramètre
function resolve {
    clear
    echo "${enigmes[$1]}?" | cut -d: -f2
    echo -n "1)"; echo "${enigmes[$1]}?" | cut -d: -f3
    echo -n "2)"; echo "${enigmes[$1]}?" | cut -d: -f4
    echo "A) Annuler"
    echo "Q) Quitter"
    read choice
    case "$choice" in
        q|Q) exit;;
        a|A) game;;
        1|2) let "choice+=2"; if [ $(echo "${enigmes[$1]}" | cut -d: -
f$((($choice))) = $(echo "${enigmes[$1]}" | cut -d: -f5) ]; then
            sed $((($1+1))"d" `pwd`/enigmes.txt -i; clear; game
"Félicitations, vous avez réussi l'énigme!\n";
            else
                clear; echo "Vous avez ratez l'énigme et perdez
3PV."; ((pv-=3));
                if [ $pv -le 0 ]; then clear; echo "Vous êtes
mort... $nomMob vous a vaincu."; rm -rf $DEF_PATH/char.txt; menu; fi
                fi;;
        *) echo "Choix non valide..."
    esac
}
```

```

        esac
    }

function choice { if [ ! $mobsCount -eq 0 ]; then fight $1; elif [ !
$enigmesCount -eq 0 ]; then resolve $1; else move $1; fi }

#Utilise le passage secret si il est disponible
function secret {
    if [ $(ls -F | grep @ -c) -eq 0 ]; then clear; echo "Il n'y a pas de
passage secret ici.";
    else clear; echo "Vous avez trouvé un passage secret, il semble mener
vers " $(ls -F | grep @ | cut -d@ -f1) ".";
    echo -e "Osez vous l'emprunter?\nO) Oui\nN) Non"
    read choice
    case "$choice" in
        q|Q) quitter;;
        o|O) cd $(ls -F | grep @ | cut -d@ -f1);;
        n|N) break;;
        *) echo "Choix non valide..."
    esac
    fi
}

#Retourne dans la pièce précédente
function reculer {
    if [ "$lieu" != "entree" ]; then cd ..; clear;
    else clear; echo "Vous ne pouvez pas reculer..."; fi
}

#Fait boire au personnage la potion reçue en paramètre
function drink {
    pvSup=$(echo ${potions[$1]} | cut -d: -f3)
    let "pv+=pvSup"
    [ $pv -gt $pvMax ] && ((pv=$pvMax));
    clear; echo -e "\e[1;34mVous avez utilisé la potion:
${potions[$1]}.\e[0m" | cut -d: -f1,2
    sed $((($1+1))"d" `pwd`/potions.txt -i
}

#Liste les potions disponibles
function potion {
    clear; nb=0
    echo "Potion(s) Disponible(s):"
    for i in "${potions[@]}"; do
        echo "-$i" | cut -d: -f1;
        echo "$i" | cut -d: -f2;
        let "nb+=1";
    done; nb=0;
    if [ ! $potionsCount -eq 0 ]; then
        for i in "${potions[@]}"; do echo $nb") Utiliser : $i" | cut -
d: -f1,2; let "nb+=1"; done;
        echo "A) Annuler"
        echo "Q) Quitter"
        read choice
        case "$choice" in
            q|Q) quitter;;
            a|A) game;;
            0|1|2|3|4|5) drink $choice;;
            *) echo "Choix non valide..."
        esac
    else echo "Vous ne trouvez aucune potion dans la pièce..."; fi
}

```

```

}

#Echange l'arme du personnage
function switch {
    if [ "${armes[$1]}" ]; then
        clear;
        sed $((($1+1))"d" `pwd`/armes.txt -i
        char "${perso[0]}" "${perso[1]}" "${perso[2]}" "$pv"
        "${armes[$1]}" `pwd`
        echo "Vous êtes désormais équipé d'une nouvelle arme."; game
    else clear; echo "Choix non valide..."; fi;
}

#Liste les armes disponibles
function armes {
    clear; nb=0
    if [ ! $armesCount -eq 0 ]; then
        echo "Arme(s) Disponible(s):"
        for i in "${armes[@]}"; do
            echo "-$i" | cut -d: -f1;
            echo "$i" | cut -d: -f2;
            let "nb+=1";
        done; nb=0;
        for i in "${armes[@]}"; do echo $nb") Equiper : $i" | cut -d: -
f1,2; let "nb+=1"; done;
        echo "A) Annuler"
        echo "Q) Quitter"
        read choice
        case "$choice" in
            q|Q) quitter;;
            a|A) game;;
            0|1|2|3|4|5) switch $choice;;
            *) clear; echo "Choix non valide..."
        esac
    else echo "Vous ne trouvez aucune arme dans la pièce..."; fi
}

```