Compte-rendu

Projet Cowsay

Auteurs: Nguyen Ho Minh Khanh, Haiban Alshouni
Groupe: IMA 2

Préliminaire

1. La syntaxe générale de Cowsay:

```
cowsay [-e eye_string] [-f cowfile] [-h] [-l] [-n] [-T tongue_string] [-W column] [-bdgpstwy]
```

2. Tableau d'options disponibles de Cowsay:

Option	Fonctions
-e [string]	La fonction prend les deux premiers caractères du mot donné et les mets sur le visage de la vache comme des paires d'œil.
-T [string]	La fonction prend les deux premiers caractères du premier mot et les mets sous le visage de la vache comme une langue et affiche le reste de la phrase dans la bulle.
-W [int]	Cela permet de définir la largeur de la bulle quel que soit la longueur de notre texte. Si int = 0 ou 1, la vache dira uniquement 0.
-f [string]	Cela permet spécifier un .cow fichier qui va être utilisé comme l'image alternatif de la vache.
-h	Cela affiche les informations sur le versions du Cowsay etaussi la syntaxe générale et tous les arguments que nous pouvons utiliser avec Cowsay.
-l	Afficher tous les fichiers des autres animaux dans le repertoire

Option	Fonctions
	COWPATH.
-n	L'option désactive le retour à la ligne, cela permettra à la vache d'afficher la chaine dans une seule ligne ou d'afficher le texte sous le format de "ASCII art".
-b	L'option active le "Borg mode", ce qui utilisera "==" comme les yeux de la vache.
-d	Cela affiche une vache morte avec "xx" comme ses yeux.
-g	L'option affiche une vache avide en utilisant "$$" comme ses yeux.
-p	L'option permet d'afficher une vache paranoïde avec des yeux "@@".
-s	Cela affiche une vache défoncée, avec des yeux "**" et une langue "U".
-t	Cela affiche une vache fatiguée, avec des yeux "--".
-w	L'option affiche une vache anxieuse, avec des yeux "OO".
-y	L'option affiche une jeune vache en utilisant ".." comme ses yeux.

3. Quelques exemples

```
cowsay -e UU -T W hello
 _______
< hello >
 -------
        \   ^__^
         \  (UU)\_______
            (__)\       )\/\
             W ||----w |
               ||     ||
```

```
cowsay -d J"'"suis mort
 _____________
< J'suis mort >
 -------------
        \   ^__^
         \  (xx)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
```

```
cowsay -g -W 6 "money money money money money"
 _______
/ money \
| money |
| money |
| money |
\ money /
 -------
        \   ^__^
         \  ($$)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Partie Bash

- **cow_kindergarten.sh**

```bash
#!/bin/bash
clear
for i in $(seq 1 10)
do
    if [ $i -eq 10 ]
    then
        cowsay -T U $i
    else
        cowsay $i
        sleep 1
        clear
    fi
done
```

- **cow_primaryschool.sh**

```
#!/bin/bash
n=$1
clear
for i in $(seq 1 $n)
do
    if [ $i -eq $n ]
    then
        cowsay -T U $i
    else
        cowsay $i
        sleep 1
        clear
    fi
done
```

- **cow_highschool.sh**

```
#!/bin/bash
n=$1
clear
for i in $(seq 1 $n)
do
    if [ $i -eq $n ]
    then
        cowsay -T U $(($n**2))
    else
        cowsay $(($i**2))
        sleep 1
        clear
    fi
done
```

- **cow_college.sh**

```
#!/bin/bash
n=$1
s1=0
s2=1
clear
while [ $s2 -lt $n ]
do
    cowsay "$s1 "
    sleep 1
    clear
    s2=$(expr $s2 + $s1)
    s1=$(expr $s2 - $s1)
done
cowsay -T U "$s1 "
```

- **cow_university.sh**
  Pour le cow_university, l'idée est que pour chaque nombre inférieur à n, on utilise une boucle pour déterminer si n divise un nombre compris entre 2 et n - 1, sinon, n est premier.

```
#!/bin/bash
#!/bin/bash
n=$1
clear
if [ $n -ge 2 ]
then
    cowsay 2
    sleep 1
    clear
fi
for i in $(seq 1 $n)
do
    for j in $(seq 2 $(expr $i - 1))
    do
        if [ $(expr $i % $j) -eq 0 ]
        then
            break
        fi
        if [ $j -eq $(expr $i - 1) ]
        then
            cowsay $i
            sleep 1
            clear
        fi
    done
done
```

- **smart_cow.sh**

  Caution: pour le smart_cow, si vous voulez tester la multiplication *, nous vous recommandons de
  mettre le calcul entre les parenthèses, sinon l'interpreteur ne peut pas distinguer le wildcard * avec l'opération *.

```
#!/bin/bash
clear
if [ $# -eq 0 ]
then
    exit 0
else
    if [ $(($1)) -ge 10 ]
    then
        cowsay -e $(($1)) $1
    #si le résultat est une seule chiffre,
    #on va ajouter un espace pour formatter les yeux
    else
        cowsay -e "$(($1)) " $1
    fi
fi
```

- **crazy_cow.sh**
  En utilisant la formule très bien connu de Einstein, la vache va calculer l'énergie de la masse prise en argument.

```
#!/bin/bash
n=$1
clear
cowsay "Avec E = mc^2 on a..."
sleep 2
clear
cowsay "$1 * (299 792 458)^2 = ..."
sleep 2
clear
cowsay -e 00 $(($1 * (299792458 ** 2)))
```

Partie C

1.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void affiche_vache(char eyes[], char tongue[])
{
    printf(" ^__^\n");
    printf(" (%s)\\_______\n", eyes);
    printf(" (__)\\       )\\/\\\n");
    printf("  %s  ||----w |\n", tongue);
    printf("     ||     ||\n");

}
```

2.

Nous avons décidé de créer la fonction "basic_options", qui prend en arguments les variables "eyes" et "tongue", et puis changer la valeur de ces variables.

La valeur défauts de "yeux" est "oo", et celle de "tongue" est " ".

La fonction rend en compte aussi les options orignales de cowsay, par exemple l'option "-d" ou "-w", l'appel de fonction avec ses options changera en même temps les yeux et la langue de la vache.

Il y a une option bonus "-m" qui sera expliquée dans la question suivante.

3.

Nous avons fait une fonction qui afficher un troupeau de vaches, avec au plus 5 vaches sur une ligne.

Pour afficher ce troupeau de vaches, l'utilisateur doit exécuter le programme en ajoutant une option "-m" (qui signifie "multiple vaches") avec un nombre de vaches voulu.

Voici le code:

```c
void basic_options(int c, int *num, char * v[], char eyes[], char tongue[])
{
for (int i = 0; i < c; i++)
    {
        if (strcmp(v[i], "-b") == 0)
        {
            strcpy(eyes, "==");
            break;
        }
        else if (strcmp(v[i], "-d") == 0)
        {
            strcpy(eyes, "xx");
            strcpy(tongue, "U");

        }
        else if (strcmp(v[i], "-g") == 0)
        {
            strcpy(eyes, "$$");

        }
        else if (strcmp(v[i], "-p") == 0)
        {
            strcpy(eyes, "@@");

        }
        else if (strcmp(v[i], "-s") == 0)
        {
            strcpy(eyes, "**");
            strcpy(tongue, "U");

        }
        else if (strcmp(v[i], "-t") == 0)
        {
            strcpy(eyes, "--");

        }
        else if (strcmp(v[i], "-w") == 0)
        {
            strcpy(eyes, "OO");

        }
        else if (strcmp(v[i], "-y") == 0)
        {
            strcpy(eyes, "..");

        }
```

```
        else if (strcmp(v[i], "-e") == 0 || strcmp(v[i], "-eyes") == 0)
        {
            strcpy(eyes, "");
            strncat(eyes, v[i + 1], 2);
        }
        else if (strcmp(v[i], "-T") == 0)
        {
            strcpy(tongue, "");
            strncat(tongue, v[i + 1], 1);
        }
        //option "-m" affiche un troupeau de vaches
        else if (strcmp(v[i], "-m") == 0)
        {
            *num = (atoi(v[i + 1]));
        }
    }
}
```

```c
//Affichage de plusieurs vaches sur une ligne (max 5 vaches)
void affiche_mvache(int num, char eyes[], char tongue[])
{
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf(" ^__^               \n");
            break;
        }
        else
            printf(" ^__^              ");          //premier ligne
    }

    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf(" (%s)\\_______      \n", eyes);
            break;
        }
        else
            printf(" (%s)\\_______      ", eyes);   //deuxieme ligne
    }

    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf(" (__)\\       )\\/\\\n");
            break;
        }
        else
            printf(" (__)\\       )\\/\\  ");       //troisieme ligne
    }

    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf("  %s  ||----w |     \n", tongue);
            break;
        }
        else
            printf("  %s  ||----w |     ", tongue); //quatrieme ligne
    }
```

```
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf("     ||     ||    \n");
            break;
        }
        else
            printf("     ||     ||     " );         //cinquieme ligne
    }
}


//Affichage d'un troupeau de vaches
void affiche_troupeau(int num, char eyes[], char tongue[])
{

    if (num <=5)
        affiche_mvache(num, eyes, tongue);
    else
    {
        for (int i = 1; i <= num/5;i++)
            affiche_mvache(5,eyes, tongue);
        affiche_mvache(num % 5, eyes,tongue);
    }
}
```

4.

La fonction update() efface l'écran de la console, la chaine "\033" est un code ASCII signifiant "ESC". La chaine "\033[H" déplace la souris en haut à gauche de l'écran et "\033[J" efface l'écran à partir de la position de la souris, alors en combinant ces chaines, on aura la fonction update() qui efface l'écran de console.

La fonction gotoxy() affiche une chaine de caractères à une position précise sur l'écran de console, avec l'aide des coordonnées x et y qui sont prises comme arguments de la fonction. Mais nous devons faire attention que les coordonnées sont inversées, axe x est la position verticalement et axe y celle horizontalement.

Nous avons créé le moonwalk_cow, une vache qui fait le "moonwalk" de Micheal Jackson, pour que la vache puisse bouger, nous devons créer deux fonctions distinctes, moonwalk_vache1 et moonwalk_vache2, qui font les animations de "moonwalk". Pour chaque

mouvement, ces deux fonctions sont appellées alternativement en changeant la position de la vache avec la fonction gotoxy.

Ensuite, pour que le programme soit plus "wild", la fonction moonwalk_troupeau a été implémentée. On doit implémenter les fonctions moonwalk_mvache pour afficher plusieurs vaches sur une ligne, les fonctions moonwalk_troupeau pour faire les animations avec un troupeau plus grand, de même façcon qu'on a fait affiche_troupeau ,et enfin la fonction moonwalk_troupeau qui affiche un troupeau de vaches faisant le moonwalk.

Voici le code des fonctions dans le fichier wildcow.c

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

//Fonctions pour rafraîchir l'écran de console et
//pour mettre la souris à position voulue
void update(){printf("\033[H\033[J");}

void gotoxy(int x,int y){printf("\033[%d;%dH",x,y);}

//-------------------------------------------------------------

void moonwalk_vache1(int i) //1ere animation de la vache
{
    gotoxy(1, i);
    printf(" ^__^\n");
    gotoxy(2, i);
    printf(" (==)\\_______\n");
    gotoxy(3, i);
    printf(" (__)\\       )\\/\\\n");
    gotoxy(4, i);
    printf("     /|----w |\n");
    gotoxy(5, i);
    printf("     \\|     \\|\n");

}

void moonwalk_vache2(int i) //2eme animation de la vache
{
    gotoxy(1, i);
    printf(" ^__^\n");
    gotoxy(2, i);
    printf(" (==)\\_______\n");
    gotoxy(3, i);
    printf(" (__)\\       )---\n");
    gotoxy(4, i);
    printf("     |/----w /\n");
    gotoxy(5, i);
    printf("     |\\     |\\\n");

}

void moonwalk_cow() //l'animation complète de moonwalk_cow
{
    int step = 1;
```

```
    for (int i = 1; i < 9; i++)
    {

        update();
        moonwalk_vache1(step);
        sleep(1);
        update();
        step += 5;
        moonwalk_vache2(step);
        sleep(1);
        update();
        step += 5;
    }
    moonwalk_vache1(step);
}

//-------------------------------------------------------------------------------

void moonwalk_mvache1(int l, int c, int num) //1ere animation de mutiple vaches
{
    gotoxy(l, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf(" ^__^               \n");
            break;
        }
        else
            printf(" ^__^              ");          //premier ligne
    }
    gotoxy(l+1, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf(" (==)\\_______      \n");
            break;
        }
        else
            printf(" (==)\\_______      ");   //deuxieme ligne
    }
    gotoxy(l+2, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
```

```
        {
            printf(" (__)\\       )\\/\\\n");
            break;
        }
        else
            printf(" (__)\\       )\\/\\  ");       //troisieme ligne
    }
    gotoxy(l+3, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf("     /|----w |\n");
            break;
        }
        else
            printf("     /|----w |     "); //quatrieme ligne
    }
    gotoxy(l+4, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf("     \\|     \\|\n");
            break;
        }
        else
            printf("     \\|     \\|     ");         //cinquieme ligne
    }
}

//-------------------------------------------------------------------------
void moonwalk_mvache2(int l, int c, int num) //2eme animation de mutiple vaches
{
    gotoxy(l, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf(" ^__^              \n");
            break;
        }
        else
            printf(" ^__^             ");          //premier ligne
    }
    gotoxy(l+1, c);
```

```
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf(" (==)\\_______      \n");
            break;
        }
        else
            printf(" (==)\\_______      ");   //deuxieme ligne
    }
    gotoxy(l+2, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf(" (__)\\       )---\n");
            break;
        }
        else
            printf(" (__)\\       )---  ");       //troisieme ligne
    }
    gotoxy(l+3, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf("     |/----w /\n");
            break;
        }
        else
            printf("     |/----w /     "); //quatrieme ligne
    }
    gotoxy(l+4, c);
    for (int i = 1; i <= num; i++)
    {
        if (i == num)
        {
            printf("     |\\     |\\\n");
            break;
        }
        else
            printf("     |\\     |\\     ");          //cinquieme ligne
    }
}
//-----------------------------------------------------------------------
```

```
void moonwalk_troupeau1(int c, int num) //1ere animation du troupeau
{

    if (num <= 5)
        moonwalk_mvache1(1, c, num);
    else
    {
        int l = 1;
        for (int j = 1; j <= num / 5; j++){
            moonwalk_mvache1(l, c, 5);
            l += 5;
            gotoxy(l, c);}
        moonwalk_mvache1(l, c, num % 5);
        }
    }

void moonwalk_troupeau2(int c, int num) //2eme animation du troupeau
{

    if (num <= 5)
        moonwalk_mvache2(1, c, num);
    else
    {
        int l = 1;
        for (int j = 1; j <= num / 5; j++){
            moonwalk_mvache2(l, c, 5);
            l += 5;
            gotoxy(l, c);}
        moonwalk_mvache2(l, c, num % 5);
        }
    }

//-----------------------------------------------------------------------

void moonwalk_troupeau(int n) //l'animation complète de moonwalk_troupeau
{
    int step = 1;
    for (int i = 1; i < 7; i++)
    {

        update();
        moonwalk_troupeau1(step, n);
        sleep(1);
        update();
        step += 5;
        moonwalk_troupeau2(step, n);
```

```
        sleep(1);
        update();
        step += 5;
    }
    moonwalk_troupeau1(step, n);
}
```

5.

Pour le reading_cow, notre vache va bouger pendant la lecture du fichier afin de mieux formatter l'affichage de la vache et de bulle de text.

Nous vous avons fourni un fichier "text.txt" pour tester le reading_cow.

Voici quelques images d'illustration

```
 _
<
 _
  \   ^__^
   \  (oo)\_______
      (__)\       )\/\
          ||----w |
          ||     ||
```

```
 ___
< b >
 ---
    \   ^__^
     \  (oo)\_______
        (__)\       )\/\
         o  ||----w |
            ||     ||
```

```
 ____
< bo >
 ----
    \   ^__^
     \  (oo)\_______
        (__)\       )\/\
         n  ||----w |
            ||     ||
```

```
 ___________________
< bonjour hello hi >
 -------------------
                  \   ^__^
                   \  (oo)\_______
                      (__)\       )\/\
                          ||----w |
                          ||     ||
```

Et le code reading_cow

```c
#include <stdio.h>
#include <stdlib.h>
#include<unistd.h>

//-------------------------------------------------------------

//Fonctions pour rafraîchir l'écran de console et
//pour mettre la souris à position voulue
void update(){printf("\033[H\033[J");}

void gotoxy(int x,int y){printf("\033[%d;%dH",x,y);}

//-------------------------------------------------------------

//Fonction pour afficher la vache
void reading_vache(int x, int y, char c)
{
    gotoxy(x, y);
    printf(" \\   ^__^               \n");
    gotoxy(x+1, y);
    printf("  \\  (oo)\\_______      \n");
    gotoxy(x+2, y);
    printf("     (__)\\       )\\/\\\n");
    gotoxy(x+3, y);
    printf("      %c  ||----w |     \n", c);
    gotoxy(x+4, y);
    printf("         ||     ||    \n");
}

void read_file (char nom[]){
    FILE *f, *des;
    f = fopen(nom, "r");
    des = stdout;
    char c;
    int position = 3;
    fscanf(f, "%c", &c);
    update();
    gotoxy(1, 1);
    printf(" _\n<\n -\n");
    reading_vache(4, position, ' ');
    while (!feof(f))
    {
        //Les codes suivants sont pour but d'afficher le bulle de text
        //contenant les mots dans le fichier
        sleep(1);
        gotoxy(1, position);
```

```
        printf("_");
        gotoxy(2, position);
        printf("%c", c);
        gotoxy(3, position);
        printf("-");
        fscanf(f, "%c", &c);
        position++;
        gotoxy(1, position);
        printf("_");
        gotoxy(2, position);
        printf(" >");
        gotoxy(3, position);
        printf("-\n");
        reading_vache(4, position, c);

    }
    reading_vache(4, position, ' ');
}
```

Automates

Premièrement, pour afficher la vache dans les états différents, nous avons créé 3 fonctions cow_die, cow_sad et cow_normal, chaque fonction représente un état différent de la vache. Puis implémenter la fonction afficher_vache qui prend en argument le nombre de l'état et faire l'appel de fonction convenable.

Pour l'état liferocks, la fonction cow_normal va afficher un vache "rock" qui semble très énergétique, pour l'état lifesucks, une vache horrible va être affiché sur l'écran, enfin pour le byebyelife, nous avons designé une vache inversée avec les yeux correspondent à l'option "-d" du cowsay.

```c
//etat byebyelife (0)
void cow_die(){
    gotoxy(4, 8);
    printf("     \\\n");
    gotoxy(5, 8);
    printf("      \\       //     //      \n");
    gotoxy(6, 8);
    printf("       \\      \\\\\----M \\      \n");
    gotoxy(7, 8);
    printf("          (--)/_______)      \n");
    gotoxy(8, 8);
    printf("          (xx)/        \\/\\ \n");
    gotoxy(9, 8);
    printf("          v--v               \n");
}

//etat lifesucks (1)
void cow_sad(){
    gotoxy(4, 8);
    printf("      \\    __               \n");
    gotoxy(5, 8);
    printf("       \\ <(@@)>_______      \n");
    gotoxy(6, 8);
    printf("          (**)\\       )\\   \n");
    gotoxy(7, 8);
    printf("           U  /\\----w \\ \\ \n");
    gotoxy(8, 8);
    printf("             /  \\   /  \\   \n");
}

//etat liferocks (2)
void cow_normal(){
    gotoxy(4, 8);
    printf("      \\   ^__^              \n");
    gotoxy(5, 8);
    printf("       \\  (==)\\_______      \n");
    gotoxy(6, 8);
    printf("          (__)\\       )\\/\\\n");
    gotoxy(7, 8);
    printf("              ||----w |     \n");
    gotoxy(8, 8);
    printf("              ||     ||    \n");
}

void affiche_vache(int etat){
    if (etat == 0)
```

```
        cow_die();
    else if (etat == 1)
        cow_sad();
    else if (etat == 2)
        cow_normal();
}
```

Images de la vache

- liferocks

```
\   ^__^
 \  (==)\_______
    (__)\       )\/\
        ||----w |
        ||     ||
```

- lifesucks

```
\    __
 \ <(@@)>_______
    (**)\       )\
     U  /\----w \ \
       /  \   /  \
```

- byebyelife

```
\       //     //
 \      \\----M \
    (--)/_______)
    (xx)/        \/\
    v--v
```

Deuxièmement, nous avons déclaré les variables globales, fitness et stock, en dehors de la fonction main.

Dans la fonction fitness_update, pour créer une valeur aléatoire digestion comprises entre -3 et 0, il vaut mieux de créer une valeur comprises entre 0 et 3 et la soustraire. Donc on a utilisé rand() % 4 pour que la valeur renvoie soit toujours entre 0 et 3 (la valeur de modulo 4).

Dans la fonction stock_update, de même façon, et avec l'aide d'une formule que nous avons

trouvé sur Internet, pour que la valeur crop soit comprises entre -3 et 3, on va créer des valeurs crop comprises entre 0 et 6, puis diminuer de 3, on obtiendra la valeur entre -3 et 3. L'explication de cette formule est facilement représentée par le calcul de l'intervalle, évidemment, pour x appartenant à [0, 6], x-3 doit être dans [-3, 3]. Alors nous utilisons rand() % 7, mais un problème apparaît. Les nombres renvoies par rand() % 7 sont toujours 4, donc rand() % 7 - 3 vaut toujours 1, nous ne pouvons pas encore trouver la raison de cette erreur. Heureusement, avec une petite modification, (rand() % 10) % 7 - 3 marche bien comme l'on veut.

```
//les variables globales
int stock = 5;
int fitness = 5; //crop et fitness sont comprises entre 0 et 10

void stock_update(int lunchfood){
    srand(time(NULL));
    int crop = (rand() % 10) % 7 - 3; //variable aléatoire crop entre -3 et 3
    stock = stock + crop - lunchfood;
    if (stock > 10)
        stock = 10;
    else if (stock < 0)
        stock = 0;
}

void fitness_update(int lunchfood){
    srand(time(NULL));
    int digestion = rand() % 4; //variable aléatoire digestion entre -3 et 0
    fitness = fitness + (lunchfood - digestion);
    if (fitness > 10)
        fitness = 10;
    else if (fitness < 0)
        fitness = 0;
}
```

Bonus

- Nous avons réalisé la fonction parole_update pour que le jeu soit plus intéressant. Pendant l'état liferocks, la vache va nous dire les citations motivantes, comme "Reste inspiré ,reste fou" (Steve Jobs), ou encore "Rock'n roll babe", alors que dans l'état lifesucks, elle va nous dire les choses démotivantes, comme "l'avenir de la vie est la mort". Pour chaque état dans le jeu, nous avons créé 5 paroles différentes et la fonction va générer une valeur aléatoire pour choisir la parole de la

vache.

```
char parole[50];
void parole_update(){
    srand(time(NULL));
    int n = rand() % 5; //génere un entier aléatoire entre 0 et 5 pour choisir les paroles
    if (fitness >= 4 && fitness <= 6){ // parole pour l'état liferocks
        if (n == 0){
            strcpy(parole, "        Rock'n roll babe!!       ");}
        else if (n == 1){
            strcpy(parole, "         LA VIE EN ROSE!         ");}
        else if (n == 2){
            strcpy(parole, "      Rien n'est impossible!     ");}
        else if (n == 3){
            strcpy(parole, "    Reste inspiré. Reste fou!    ");}
        else if (n == 4){
            strcpy(parole, "Quoi que tu fasses, fais le bien!");}}
    else{
        if (n == 0){
            strcpy(parole, "     I don't feel good, bro!     ");}
        else if (n == 1){
            strcpy(parole, "     Life is meaningless :(      ");}
        else if (n == 2){
            strcpy(parole, "    La vie n'est pas en rose!    ");}
        else if (n == 3){
            strcpy(parole, "L'avenir de la vie c'est la mort!");}
        else if (n == 4){
            strcpy(parole, "Si on est pessimiste, on est fini");}
    }

}
```

- Enfin, voici le code de la fonction main, avec le boucle while qui affiche les informations pendant le jeu.

```c
int main(){
    int etat = 2;
    int day = 1;
    int lunchfood;
    strcpy(parole, " Yo what's up dude, I'm lil'cow! ");
    while (etat != 0) // initialisation de jeu
    {
        update();
        printf(" _____\n");
        printf("< %s >\n", parole);
        printf(" ------------------------------------\n");
        affiche_vache(etat);
        printf("\n");
        printf("/-------------------------\\\n");
        printf("| Days: %d        Stock: %d  |\n", day, stock);
        printf("\\-------------------------/\n");
        printf("Feed the cow: ");
        scanf("%d", &lunchfood);

        //Une boucle while pour vérifier si la quantité de lunchfood est valide

        while (lunchfood > stock || lunchfood < 0)
        {
            if (lunchfood > stock)
                printf("not enough food!");
            else
                printf("invalid amount of food!");
            scanf("%d", &lunchfood);
        }

            //mettre à jour les variables globales

        fitness_update(lunchfood);
        stock_update(lunchfood);

            //mettre à jour l'état de la vache

        if (fitness == 0 || fitness == 10)
            etat = 0;
        else {
            if (fitness >= 4 && fitness <= 6)
                etat = 2;
            else
                etat = 1;
            day++;      //incrémenter le nombre du jour si la vache survive
            parole_update();    //mettre à jour les parole de la vache
```

```
        }
    }
    update();
    strcpy(parole, "         See ya next time!       ");
    printf(" _____________________________________\n");
    printf("< %s >\n", parole);
    printf(" -------------------------------------\n");
    affiche_vache(0);
    printf("\n");
    printf("/-------------------------\\\n");
    //les conditions pour formater l'affichage du résultat de joueur
    if (day >= 10)
        printf("|       Days lived: %d     |\n", day);
    else
        printf("|       Days lived: %d      |\n", day);
    printf("\\-------------------------/\n");

    return 0;
}
```

Images du jeu

```
 _____________________________________
< Si on est pessimiste, on est fini >
 -------------------------------------
            \    __
             \ <(@@)>_______
                (**)\       )\
                 U  /\----w \ \
                   /  \   /  \

/-------------------------\
| Days: 2       Stock: 0  |
\-------------------------/
Feed the cow:
```

```
 ______________________________________
<          LA VIE EN ROSE!             >
 --------------------------------------
             \   ^__^
              \  (==)\_______
                 (__)\       )\/\
                     ||----w |
                     ||     ||

/--------------------------\
| Days: 5        Stock: 0  |
\--------------------------/
Feed the cow:
```

```
 ______________________________________
<          See ya next time!           >
 --------------------------------------
            \
             \       //     //
              \      \\----M \
                 (--)/_______)
                 (xx)/        \/\
                 v--v

/--------------------------\
|       Days lived: 12     |
\--------------------------/
```

FIN