

←!—

- @Author: JIANG Yilun
 - @Date: 2022-04-24 14:28:58
 - @LastEditTime: 2022-04-24 21:15:04
 - @LastEditors: JIANG Yilun
 - @Description:
 - @FilePath: /Projet_cowsay_L1S2/rapport_cowsay_JIANGYilun.md
-

Projet Cowsay

JIANG Yilun

Sommaire

Projet Cowsay

Sommaire

1. Présentation du Projet cowsay
2. Objectif du projet
 1. Préliminaires
 2. Bash
 3. C
 4. Automates

Préliminaires

Bash

cow_kindergarten
cow_primaryschool
cow_highschool
cow_college
cow_university
smart_cow
crazy_cow

C

Question 1
Question 2
Question 3

Automates

1. Présentation du Projet cowsay

Le projet débute au premier jour du cours INF203 et s'achève lors la dernière semaine de cours. Cette dernière fait date de rendu (dimanche soir minuit de la dernière semaine). Vous pouvez progresser sur le projet à votre rythme, mais nous vous recommandons de prendre de l'avance par rapport au cours, du moins aucun retard. Par exemple, la partie "Bash" devra être achevée au moment où les premiers cours de "C" débiteront.

2. Objectif du projet

L'objectif du projet est de découvrir le monde merveilleux de "cowsay". Au cours du projet, vous réaliserez les objectifs suivants:

1. Préliminaires

Découvrir la commande cowsay a travers son manuel (manpage) et l'ensemble des options qu'elle contient.

2. Bash

Implémenter un script Bash qui fait réciter a la vache la suite des nombres premiers, des nombres de Fibonacci, ou toute autre suite exotique de votre choix.

3. C

Recoder cowsay en C, avec de nouvelles fonctionnalités additionnelles de votre choix (comme par exemple la longueur de la queue).

4. Automates

En s'appuyant sur la théorie des automates, implementer un "cow-Tamagoshi" qu'il s'agit de nourrir et faire survivre aussi longtemps que possible.

Préliminaires

Découvrir du code `cowsay` :

```
1 | cowsay -h
```

Nous avons donc le résultat suivant:

```
1 | cow{say,think} version 3.03, (c) 1999 Tony Monroe
2 | Usage: cowsay [-bdgpstwy] [-h] [-e eyes] [-f cowfile]
3 |               [-l] [-n] [-T tongue] [-W wrapcolumn] [message]
```

Nous apprenons donc que le projet cowsay contient en fait deux commandes, l'une appelée Cowsay et l'autre Cowthink. Cowsay utilise des lignes droites pour relier la vache aux mots prononcés, tandis que cowthink utilise des cercles.

Par exemple, nous utilise `cowsay` en première:

```
1 | cowsay "Hello, my name is JIANG Yilun"
```

Avec les résultats suivant:

```

1 | -----
2 | < Hello, my name is JIANG Yilun >
3 | -----
4 |      \   ^__^
5 |       \  (oo)\_______
6 |          (__)\       )\/\
7 |             ||----w |
8 |             ||     ||

```

Ensuite, nous utilisons la commande `cowthink` :

```

1 | cowthink "Hello, my name is JIANG Yilun"

```

Avec les résultats suivants :

```

1 | -----
2 | ( Hello, my name is JIANG Yilun )
3 | -----
4 |      o   ^__^
5 |       o  (oo)\_______
6 |          (__)\       )\/\
7 |             ||----w |
8 |             ||     ||

```

En fait, le `cowsay` ne se limite pas à la forme de la vache. Après nous utilisons la commande `cowsay -l`, nous pouvons constater que nous avons en fait de nombreux modèles à choisir :

```

1 | $ cowsay -l
2 | Cow files in /opt/homebrew/Cellar/cowsay/3.04_1/share/cows:
3 | beavis.zen blowfish bong bud-frogs bunny cheese cower daemon default dragon
4 | dragon-and-cow elephant elephant-in-snake eyes flaming-sheep ghostbusters
5 | head-in hellokitty kiss kitty koala kosh luke-koala meow milk moofasa moose
6 | mutilated ren satanic sheep skeleton small stegosaurus stimp supermilker
7 | surgery three-eyes turkey turtle tux udder vader vader-koala www

```

Par exemple, on peut utiliser la forme `sheep` :

```

1 $ cowsay -f sheep hello
2
3 < hello >
4
5 \
6  \
7
8      _
9      UooU\.'@@@@@'.
10     \_/(@@@@@@@@@@)
11         (@@@@@@@@)
12         `YY~~~~YY'
           ||    ||

```

On peut aussi utiliser le vase avec des tuyaux:

```

1 $ ll | cowsay
2
3 -----
4 / total 8 -rw-r--r--@ 1 yilunjiang staff \
5 | 3.6K Apr 24 15:03 |
6 \ rapport_cowsay_JIANGYilun.md /
7
8 \      ^__^
9  \    (oo)\_____
10      (__)\       )\/\
11           ||----w |
12           ||

```

En fait, la sortie de cowsay est très pauvre - pratiquement impossible à visualiser très bien. Mais si nous ajoutons la commande -n:

```

1 $ ll | cowsay -n
2
3 -----
4 / total 16
5 \ -rw-r--r--@ 1 yilunjiang staff 4.2K Apr 24 15:09 rapport_cowsay_JIANGYilun.md /
6
7 \      ^__^
8  \    (oo)\_____
9      (__)\       )\/\
10           ||----w |
11           ||

```

De cette façon, les informations décrites peuvent être lues de manière plus visuelle.

La commande cowsay a en fait ces petits extras, par exemple, nous pouvons changer les yeux de la vache:

```

1 $ cowsay -e -- "Hello, my name is JIANG Yilun"
2
3 < Hello, my name is JIANG Yilun >
4 -----
5      \   ^__^
6       \  (oo)\_______
7          (__)\       )\/\
8              ||----w |
9              ||     ||

```

Nous avons même réussi à lui faire cracher sa langue :

```

1 $ cowsay -T U "Hello, my name is JIANG Yilun"
2
3 < Hello, my name is JIANG Yilun >
4 -----
5      \   ^__^
6       \  (oo)\_______
7          (__)\       )\/\
8              U ||----w |
9              ||     ||

```

Bash

En fait, le code présenté ci-dessous a été modifié une deuxième fois (après avoir vu la vache folle) et comporte deux sections distinctes : une avec un argument et une sans.

cow_kindergarten

```

1 ###
2 # @Author: JIANG Yilun
3 # @Date: 2022-04-24 15:15:21
4 # @LastEditTime: 2022-04-24 17:55:56
5 # @LastEditors: JIANG Yilun
6 # @Description:
7 # @FilePath: /Projet_cowsay_L1S2/cow_kindergarten.sh
8 ###
9
10 if [ $# -eq 0 ]; then
11     temp=10
12     while [ $temp -gt 0 ]; do
13         clear
14         cowsay $temp
15         sleep 1
16         temp=$((temp-1))
17     done
18 else

```

```

19     temp=$1
20     while [ $temp -gt 0 ]; do
21         clear
22         cowsay $temp
23         sleep 1
24         temp=$((temp-1))
25     done
26 fi

```

cow_primaryschool

```

1  ###
2  # @Author: JIANG Yilun
3  # @Date: 2022-04-24 15:33:12
4  # @LastEditTime: 2022-04-24 17:54:01
5  # @LastEditors: JIANG Yilun
6  # @Description:
7  # @FilePath: /Projet_cowsay_L1S2/cow_primaryschool.sh
8  ###
9
10 i=1
11 if [ $# -eq 0 ]; then
12     echo "Saissez un nombre:"
13     read nombre
14     while [ $i -le $nombre ]; do
15         clear
16         cowsay $i
17         sleep 1
18         i=$((i+1))
19     done
20 else
21     nombre=$1
22     while [ $i -le $nombre ]; do
23         clear
24         cowsay $i
25         sleep 1
26         i=$((i+1))
27     done
28 fi

```

cow_highschool

```
1  ###
2  # @Author: JIANG Yilun
3  # @Date: 2022-04-24 15:37:56
4  # @LastEditTime: 2022-04-24 17:52:24
5  # @LastEditors: JIANG Yilun
6  # @Description:
7  # @FilePath: /Projet_cowsay_L1S2/cow_highschool.sh
8  ###
9
10 i=1
11
12 if [ $# -eq 0 ]; then
13     echo "Saissez un nombre:"
14     read nombre
15     while [ $i -le $nombre ]; do
16         clear
17         cowsay $((i*i))
18         sleep 1
19         i=$((i+1))
20     done
21 else
22     nombre=$1
23     while [ $i -le $nombre ]; do
24         clear
25         cowsay $((i*i))
26         sleep 1
27         i=$((i+1))
28     done
29 fi
```

cow_college

```
1  ###
2  # @Author: JIANG Yilun
3  # @Date: 2022-04-24 15:41:00
4  # @LastEditTime: 2022-04-24 17:44:13
5  # @LastEditors: JIANG Yilun
6  # @Description:
7  # @FilePath: /Projet_cowsay_L1S2/cow_college.sh
8  ###
9
10 # nombres de Finonacci
11
12 i=0
13 j=1
```

```

14
15 if [ $# -eq 0 ]; then
16     echo "Saissez un nombre:"
17     read nombre
18     while [ $j -lt $nombre ]; do
19         cowsay $j
20         temp=$((i+j))
21         i=$j
22         j=$temp
23         sleep 1
24     done
25 else
26     nombre=$1
27     while [ $j -lt $nombre ]; do
28         cowsay $j
29         temp=$((i+j))
30         i=$j
31         j=$temp
32         sleep 1
33     done
34 fi

```

cow_university

```

1  ###
2  # @Author: JIANG Yilun
3  # @Date: 2022-04-24 15:55:25
4  # @LastEditTime: 2022-04-24 17:42:31
5  # @LastEditors: JIANG Yilun
6  # @Description:
7  # @FilePath: /Projet_cowsay_L1S2/cow_university.sh
8  ###
9
10 nbr_premier() {
11     while [ $i -le $m ]
12     do
13         p=$((m%i))
14         if [ $p -eq 0 ]
15         then
16             break
17         else
18             i=$((i+1))
19         fi
20     if [ $i -eq $m ]
21     then
22         if [ $m -eq $n ]
23         then
24             echo "$m est un nombre premier"

```



```

25         cowsay -T U "$m"
26     else
27         echo "$m est un nombre premier"
28         cowsay "$m"
29     fi
30 fi
31 done
32 }
33
34 if [ $# -eq 0 ]; then
35     echo "donnez le dernier nombres premiers à calculer"
36     read n
37     i=2      #le premier nombre premier
38     a=$(bc <<< "scale=0; sqrt($n)") #scale=0 n'affiche pas les décimale, scale=1 la
première, etc... sqrt() calcule la racine carré. marche grace à la commande bc
39     m=3
40     echo "voici sa suite de nombres premiers de $i à $n"
41     while [ $m -le $n ]
42     do
43         echo m:$m
44         i=2
45         nbr_premier $m
46         m=$((m+1))
47         sleep 1
48     done
49 else
50     n=$1
51     i=2      #le premier nombre premier
52     a=$(bc <<< "scale=0; sqrt($n)") #scale=0 n'affiche pas les décimale, scale=1 la
première, etc... sqrt() calcule la racine carré. marche grace à la commande bc
53     m=3
54     echo "voici sa suite de nombres premiers de $i à $n"
55     while [ $m -le $n ]
56     do
57         echo m:$m
58         i=2
59         nbr_premier $m
60         m=$((m+1))
61         sleep 1
62     done
63 fi

```

smart_cow

```

1  ###
2  # @Author: JIANG Yilun
3  # @Date: 2022-04-24 16:27:30
4  # @LastEditTime: 2022-04-24 16:40:09

```

```

5 # @LastEditors: JIANG Yilun
6 # @Description:
7 # @FilePath: /Projet_cowsay_L1S2/smart_cow.sh
8 ###
9
10
11 if [ $# -eq 0 ]; then
12     echo "Donner l'expression à calculer:"
13     read expression
14     cowsay -e $(echo "$expression" | bc) $expression
15 else
16     cowsay -e $(echo "$1" | bc) $1
17 fi

```

crazy_cow

```

1 ###
2 # @Author: JIANG Yilun
3 # @Date: 2022-04-24 16:44:04
4 # @LastEditTime: 2022-04-24 17:57:02
5 # @LastEditors: JIANG Yilun
6 # @Description:
7 # @FilePath: /Projet_cowsay_L1S2/crazy_cow.sh
8 ###
9
10 for var in "$@"
11 do
12     if [[ "$var" = "-h" || "$var" = "--help" ]]; then
13         echo "Usage: $0 [OPTION]... [FILE]..."
14         echo "Print a crazy cow."
15     elif [[ "$var" = "-v" || "$var" = "--version" ]]; then
16         echo "crazy_cow.sh version 1.0"
17     elif [[ "$var" = "-a" || "$var" = "--addition" ]]; then
18         sh cow_primaryschool.sh ${@: -1}
19     elif [[ "$var" = "-c" || "$var" = "--countdown" ]]; then
20         sh cow_kindergarten.sh ${@: -1}
21     elif [[ "$var" = "-s" || "$var" = "--square" ]]; then
22         sh cow_highschool.sh ${@: -1}
23     elif [[ "$var" = "-f" || "$var" = "--finonacci" ]]; then
24         sh cow_college.sh ${@: -1}
25     elif [[ "$var" = "-p" || "$var" = "--premiere" ]]; then
26         sh cow_university.sh ${@: -1}
27     elif [[ "$var" = "-S" || "$var" = "--smart" ]]; then
28         sh smart_cow.sh ${@: -1}
29     fi
30 done

```

C

Question 1

affiche_vache :

```
1 int affiche_vache()
2 {
3     printf("\n");
4     printf("    \\  ^__^\\n");
5     printf("    \\  (oo)\\_______\\n");
6     printf("    (__)\\        )\\/\\\\n");
7     printf("    |  ---w  |\\n");
8     printf("    ||      ||\\n");
9     printf("\n");
10    return 0;
11 }
12
13 int main()
14 {
15     affiche_vache();
16 }
```

Après la compilation, nous avons pu obtenir les résultats suivants:

```
1 $ gcc newcow.c && ./a.out
2
3     \  ^__^
4     \  (oo)\\_______
5     (__)\        )\\/\\
6         |  ---w  |
7         ||      ||
```

Question 2

```
1 /*
2  * @Author: JIANG Yilun
3  * @Date: 2022-04-24 18:07:27
4  * @LastEditTime: 2022-04-24 18:44:29
5  * @LastEditors: JIANG Yilun
6  * @Description:
7  * @FilePath: /Projet_cowsay_L1S2/newcow.c
8  */
```

```

9
10 #include <stdio.h>
11 #include <string.h>
12
13 int affiche_vache (char *eyes, char *tongue)
14 {
15     if (eyes == NULL && tongue == NULL){
16         printf("\n");
17         printf("    \\  ^__^\\n");
18         printf("    \\  (oo)\\_______\\n");
19         printf("        (__)\\        )\\/\\\\n");
20         printf("            |  ---w  |\\n");
21         printf("            ||      ||\\n");
22         printf("\n");
23         return 0;
24     }
25     else if (eyes == NULL && tongue != NULL){
26         printf("\n");
27         printf("    \\  ^__^\\n");
28         printf("    \\  (oo)\\_______\\n");
29         printf("        (__)\\        )\\/\\\\n");
30         printf("            %s |  ---w  |\\n", tongue);
31         printf("            ||      ||\\n");
32         printf("\n");
33         return 0;
34     }
35     else if (eyes != NULL && tongue == NULL){
36         printf("\n");
37         printf("    \\  ^__^\\n");
38         printf("    \\  (%s)\\_______\\n", eyes);
39         printf("        (__)\\        )\\/\\\\n");
40         printf("            |  ---w  |\\n");
41         printf("            ||      ||\\n");
42         printf("\n");
43         return 0;
44     }
45     else
46     {
47         printf("\n");
48         printf("    \\  ^__^\\n");
49         printf("    \\  (%s)\\_______\\n", eyes);
50         printf("        (__)\\        )\\/\\\\n");
51         printf("            %s |  ---w  |\\n", tongue);
52         printf("            ||      ||\\n");
53         printf("\n");
54         return 0;
55     }
56 }
57
58 int main (int argc, char *argv[])
59 {
60     char *eyes = NULL;

```

```

61     char *tongue = NULL;
62     char *message = NULL;
63     char *tail = NULL;
64     for (int i = 1; i < argc; i++)
65     {
66         if (strcmp(argv[i], "-e") == 0 || strcmp(argv[i], "--eyes") == 0)
67         {
68             eyes = argv[i+1];
69         }
70         if (strcmp(argv[i], "-t") == 0 || strcmp(argv[i], "--tongue") == 0)
71         {
72             tongue = argv[i+1];
73         }
74     }
75     affiche_vache(eyes, tongue);
76 }

```

Après la compilation, nous avons pu obtenir les résultats suivants:

```

1  $ gcc test.c && ./a.out -e AA -t U
2
3      \   ^__^
4      \  (AA)\_____
5         (__)\       )\/\
6            U     ||---w   ||
7                ||         ||

```

Question 3

```

1  /*
2   * @Author: JIANG Yilun
3   * @Date: 2022-04-24 18:07:27
4   * @LastEditTime: 2022-04-24 21:10:24
5   * @LastEditors: JIANG Yilun
6   * @Description:
7   * @FilePath: /Projet_cowsay_L1S2/newcow.c
8   */
9
10 #include <stdio.h>
11 #include <string.h>
12 #include <stdlib.h>
13
14 int affiche_vache(int *length, char *message, char *eyes, char *tongue, int *tail)
15 {
16     printf(" -");
17     for (int i = 0; i < *length; i++)
18     {

```

```

19     printf("-");
20 }
21 printf("\n");
22 printf("< %s >\n", message);
23 printf(" -");
24 for (int i = 0; i < *length; i++)
25 {
26     printf("-");
27 }
28 printf("\n");
29 printf("    \ \  ^_^ \n");
30 printf("    \ \  (%s) \ \ _____ \n", eyes);
31 printf("    ( _ ) \ \          ) \ \");
32 for (int i = 0; i < *tail; i++)
33 {
34     printf("/ \");
35 }
36 printf("\n");
37 printf("        %s | | ---w | \n", tongue);
38 printf("        | |          | | \n");
39 printf("\n");
40 return 0;
41 }
42
43 void update() { printf("\033[H\033[J"); }
44
45 void gotoxy(x, y) { printf("\033[%d;%dH", x, y); }
46
47 int main(int argc, char *argv[])
48 {
49     char *eyes = "oo"; // default eyes
50     char *tongue = " "; // default tongue
51     char *message = "--help to display help"; // default message
52     int tail = 1; // default tail
53     for (int i = 1; i < argc; i++)
54     {
55         if (strcmp(argv[i], "-e") == 0 || strcmp(argv[i], "--eyes") == 0)
56         {
57             eyes = argv[i + 1];
58         }
59         if (strcmp(argv[i], "-T") == 0 || strcmp(argv[i], "--tongue") == 0)
60         {
61             tongue = argv[i + 1];
62         }
63         if (strcmp(argv[i], "-m") == 0 || strcmp(argv[i], "--message") == 0)
64         {
65             message = argv[i + 1];
66         }
67         if (strcmp(argv[i], "-t") == 0 || strcmp(argv[i], "--tail") == 0)
68         {
69             tail = atoi(argv[i + 1]);
70         }

```

```

71     if (strcmp(argv[i], "-h") == 0 || strcmp(argv[i], "--help") == 0)
72     {
73         printf("\n");
74         printf("Usage: newcow [OPTION]...\n");
75         printf("\n");
76         printf("Options:\n");
77         printf("  -e, --eyes=STRING  eyes of the cow (default: oo)\n");
78         printf("  -t, --tongue=STRING tongue of the cow (default: )\n");
79         printf("  -m, --message=STRING message to display (default: none)\n");
80         printf("  -h, --help          display this help and exit\n");
81         printf("\n");
82         return 0;
83     }
84 }
85 int length = strlen(message) + 1;
86 affiche_vache(&length, message, eyes, tongue, &tail);
87 }

```

On peut ajouter d'argument "eyes" ou argument "tongue".

S'il n'y a pas de message d'entrée:

```

1  $ gcc newcow.c && ./a.out -e AA -T U
2  -----
3  < --help to display help >
4  -----
5      \   ^__^
6       \  (oo)\_______
7          (__)\       )\/\
8             U   ||---w  |
9                ||      ||

```

Si je veux obtenir des informations d'aide:

```

1  $ gcc newcow.c && ./a.out -h
2
3  Usage: newcow [OPTION]...
4
5  Options:
6  -e, --eyes=STRING  eyes of the cow (default: oo)
7  -t, --tongue=STRING tongue of the cow (default: )
8  -m, --message=STRING message to display (default: none)
9  -h, --help          display this help and exit

```

Bien sûr, la possibilité d'afficher des messages est essentielle:

```

1 $ gcc newcow.c && ./a.out -e AA -T UU -m "Hello, my name is JIANG Yilun"
2 -----
3 < Hello, my name is JIANG Yilun >
4 -----
5      \   ^__^
6         \  (AA)\_____
7            (__) \        )\/\
8               UU  ||---w |
9                  ||     ||

```

En même temps, nous pouvons définir la longueur du tail:

```

1 $ gcc newcow.c && ./a.out -e AA -T UU -m "Hello, my name is JIANG Yilun" -t 10
2 -----
3 < Hello, my name is JIANG Yilun >
4 -----
5      \   ^__^
6         \  (AA)\_____
7            (__) \        )\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
8               UU  ||---w |
9                  ||     ||

```

Automates

Je n'ai pas réussi à concevoir un automate, car je ne pense pas avoir identifié toutes les possibilités. Mais j'ai créé un mini-jeu "CowSay" qui n'utilise pas d'automate.



Les joueurs doivent répondre à des questions de mathématiques pour obtenir de la nourriture, avec une autre chance d'obtenir de la nourriture toutes les heures.

Ce jeu comporte cinq modèles mathématiques : déterminer si un nombre est premier ou non, ajouter, soustraire, multiplier et trouver le mod entre des nombres.

De plus, des événements aléatoires se produiront dans le scénario du jeu et le joueur peut obtenir des effets négatifs, tels que la perte de nourriture ou de vies.