←!—

- @Author: JIANG Yilun
- @Date: 2022-04-24 14:28:58
- @LastEditTime: 2022-04-24 18:00:10
- @LastEditors: JIANG Yilun
- @Description:
- @FilePath: /Projet_cowsay_L1S2/rapport_cowsay_JIANGYilun.md
  —→

Projet Cowsay

JIANG Yilun

Sommaire

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 a 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 ou les premiers cours de “C” débuteront.

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` :

```
cowsay -h
```

Nous avons donc le résultat suivant:

```
cow{say,think} version 3.03, (c) 1999 Tony Monroe
Usage: cowsay [-bdgpstwy] [-h] [-e eyes] [-f cowfile]
          [-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:

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

Avec les résultats suivant:

```
 ------------------------------
< Hello, my name is JIANG Yilun >
 ------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Ensuit, nous utilise le commande `cowthink` :

```
cowthink "Hello, my name is JIANG Yilun"
```

Avec les résultats suivant:

```
 ------------------------------
( Hello, my name is JIANG Yilun )
 ------------------------------
        o   ^__^
         o  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

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

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

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

```
$ cowsay -f sheep hello
 _______
< hello >
 -------
  \
   \
        __
       UooU\.'@@@@@@`.
       \__/(@@@@@@@@@@)
            (@@@@@@@@)
            `YY~~~~YY'
             ||    ||
```

On peut aussi utiliser le vase avec des tuyaux:

```
$ ll | cowsay
 ________________________________________
/ total 8 -rw-r--r--@ 1 yilunjiang staff \
| 3.6K Apr 24 15:03                      |
\ rapport_cowsay_JIANGYilun.md           /
 ----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

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

```
$ ll | cowsay -n
 ____________________________________________________________________________________
/ total 16                                                                           \
\ -rw-r--r--@ 1 yilunjiang  staff   4.2K Apr 24 15:09 rapport_cowsay_JIANGYilun.md /
 ------------------------------------------------------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

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:

```
$ cowsay -e -- "Hello, my name is JIANG Yilun"
 ______________________________
< Hello, my name is JIANG Yilun >
 ------------------------------
        \   ^__^
         \  (--)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Nous avons même réussi à lui faire cracher sa langue:

```
$ cowsay -T U "Hello, my name is JIANG Yilun"
 ______________________________
< Hello, my name is JIANG Yilun >
 ------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U ||----w |
                ||     ||
```

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

```
###
 # @Author: JIANG Yilun
 # @Date: 2022-04-24 15:15:21
 # @LastEditTime: 2022-04-24 17:55:56
 # @LastEditors: JIANG Yilun
 # @Description:
 # @FilePath: /Projet_cowsay_L1S2/cow_kindergarten.sh
###

if [ $# -eq 0 ]; then
    temp=10
    while [ $temp -gt 0 ]; do
        clear
        cowsay $temp
        sleep 1
        temp=$((temp-1))
    done
else
```

```
    temp=$1
    while [ $temp -gt 0 ]; do
        clear
        cowsay $temp
        sleep 1
        temp=$((temp-1))
    done
fi
```

cow_primaryschool

```
###
 # @Author: JIANG Yilun
 # @Date: 2022-04-24 15:33:12
 # @LastEditTime: 2022-04-24 17:54:01
 # @LastEditors: JIANG Yilun
 # @Description:
 # @FilePath: /Projet_cowsay_L1S2/cow_primaryschool.sh
###

i=1
if [ $# -eq 0 ]; then
    echo "Saissez un nombre:"
    read nombre
    while [ $i -le $nombre ]; do
        clear
        cowsay $i
        sleep 1
        i=$((i+1))
    done
else
    nombre=$1
    while [ $i -le $nombre ]; do
        clear
        cowsay $i
        sleep 1
        i=$((i+1))
    done
fi
```

cow_highschool

```
###
 # @Author: JIANG Yilun
 # @Date: 2022-04-24 15:37:56
 # @LastEditTime: 2022-04-24 17:52:24
 # @LastEditors: JIANG Yilun
 # @Description:
 # @FilePath: /Projet_cowsay_L1S2/cow_highschool.sh
###

i=1

if [ $# -eq 0 ]; then
    echo "Saissez un nombre:"
    read nombre
    while [ $i -le $nombre ]; do
        clear
        cowsay $(($i*$i))
        sleep 1
        i=$((i+1))
    done
else
    nombre=$1
    while [ $i -le $nombre ]; do
        clear
        cowsay $(($i*$i))
        sleep 1
        i=$((i+1))
    done
fi
```

cow_college

```
###
 # @Author: JIANG Yilun
 # @Date: 2022-04-24 15:41:00
 # @LastEditTime: 2022-04-24 17:44:13
 # @LastEditors: JIANG Yilun
 # @Description:
 # @FilePath: /Projet_cowsay_L1S2/cow_college.sh
###

# nombres de Finonacci

i=0
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"
```

```
                cowsay -T U "$m"
            else
                echo "$m est un nombre premier"
                cowsay "$m"
            fi
        fi
    done
}

if [ $# -eq 0 ]; then
    echo "donnez le dernier nombres premiers à calculer"
    read n
    i=2     #le premier nombre premier
    a=$(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
    m=3
    echo "voici sa suite de nombres premiers de $i à $n"
    while [ $m -le $n ]
    do
        echo m:$m
        i=2
        nbr_premier $m
        m=$((m+1))
        sleep 1
    done
else
    n=$1
    i=2     #le premier nombre premier
    a=$(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
    m=3
    echo "voici sa suite de nombres premiers de $i à $n"
    while [ $m -le $n ]
    do
        echo m:$m
        i=2
        nbr_premier $m
        m=$((m+1))
        sleep 1
    done
fi
```

smart_cow

```
###
 # @Author: JIANG Yilun
 # @Date: 2022-04-24 16:27:30
 # @LastEditTime: 2022-04-24 16:40:09
```

```
 # @LastEditors: JIANG Yilun
 # @Description:
 # @FilePath: /Projet_cowsay_L1S2/smart_cow.sh
###


if [ $# -eq 0 ]; then
    echo "Donner l'expression à calculer:"
    read expression
    cowsay -e $(echo "$expression" | bc) $expression
else
    cowsay -e $(echo "$1" | bc) $1
fi
```

crazy_cow

```
###
 # @Author: JIANG Yilun
 # @Date: 2022-04-24 16:44:04
 # @LastEditTime: 2022-04-24 17:57:02
 # @LastEditors: JIANG Yilun
 # @Description:
 # @FilePath: /Projet_cowsay_L1S2/crazy_cow.sh
###

for var in "$@"
do
    if [[ "$var" == "-h" || "$var" == "--help" ]]; then
        echo "Usage: $0 [OPTION]... [FILE]..."
        echo "Print a crazy cow."
    elif [[ "$var" == "-v" || "$var" == "--version" ]]; then
        echo "crazy_cow.sh version 1.0"
    elif [[ "$var" == "-a" || "$var" == "--addition" ]]; then
        sh cow_primaryschool.sh ${@: -1}
    elif [[ "$var" == "-c" || "$var" == "--countdown" ]]; then
        sh cow_kindergarten.sh ${@: -1}
    elif [[ "$var" == "-s" || "$var" == "--square" ]]; then
        sh cow_highschool.sh ${@: -1}
    elif [[ "$var" == "-f" || "$var" == "--finonacci" ]]; then
        sh cow_college.sh ${@: -1}
    elif [[ "$var" == "-p" || "$var" == "--premiere" ]]; then
        sh cow_university.sh ${@: -1}
    elif [[ "$var" == "-S" || "$var" == "--smart" ]]; then
        sh smart_cow.sh ${@: -1}
    fi
done
```

C

Question 1

`affiche_vache` :

```c
int affiche_vache()
{
    printf("\n");
    printf("    \\   ^__^\n");
    printf("     \\  (oo)\\_______\n");
    printf("        (__)\\       )\\/\\\n");
    printf("            ||---w |\n");
    printf("            ||     ||\n");
    printf("\n");
    return 0;
}

int main()
{
    affiche_vache();
}
```

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

```
$ gcc newcow.c && ./a.out


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

Question 2

```c
/*
 * @Author: JIANG Yilun
 * @Date: 2022-04-24 18:07:27
 * @LastEditTime: 2022-04-24 18:44:29
 * @LastEditors: JIANG Yilun
 * @Description:
 * @FilePath: /Projet_cowsay_L1S2/newcow.c
 */
```

```c

#include <stdio.h>
#include <string.h>

int affiche_vache (char *eyes, char *tongue)
{
    if (eyes == NULL && tongue == NULL){
        printf("\n");
        printf("    \\   ^__^\n");
        printf("     \\  (oo)\\_______\n");
        printf("        (__)\\       )\\/\\\n");
        printf("            |\u2500\u2500---w |\n");
        printf("            ||     ||\n");
        printf("\n");
        return 0;
    }
    else if (eyes == NULL && tongue \u2260 NULL){
        printf("\n");
        printf("    \\   ^__^\n");
        printf("     \\  (oo)\\_______\n");
        printf("        (__)\\       )\\/\\\n");
        printf("         %s |\u2500\u2500---w |\n",tongue);
        printf("            ||     ||\n");
        printf("\n");
        return 0;
    }
    else if (eyes \u2260 NULL && tongue == NULL){
        printf("\n");
        printf("    \\   ^__^\n");
        printf("     \\  (%s)\\_______\n",eyes);
        printf("        (__)\\       )\\/\\\n");
        printf("            |\u2500\u2500---w |\n");
        printf("            ||     ||\n");
        printf("\n");
        return 0;
    }
    else
    {
        printf("\n");
        printf("    \\   ^__^\n");
        printf("     \\  (%s)\\_______\n",eyes);
        printf("        (__)\\       )\\/\\\n");
        printf("         %s |\u2500\u2500---w |\n",tongue);
        printf("            ||     ||\n");
        printf("\n");
        return 0;
    }
}

int main (int argc, char *argv[])
{
    char *eyes = NULL;
```

```
    char *tongue = NULL;
    char *message = NULL;
    char *tail = NULL;
    for (int i = 1; i < argc; i++)
    {
        if (strcmp(argv[i], "-e") == 0 || strcmp(argv[i], "--eyes") == 0)
        {
            eyes = argv[i+1];
        }
        if (strcmp(argv[i], "-t") == 0 || strcmp(argv[i], "--tongue") == 0)
        {
            tongue = argv[i+1];
        }
    }
    affiche_vache(eyes, tongue);
}
```

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

```
$ gcc test.c && ./a.out -e AA -t U

   \   ^__^
    \  (AA)\_______
       (__)\       )\/\
        U |├ ---w |
          ||     ||
```

Question 3

```
/*
 * @Author: JIANG Yilun
 * @Date: 2022-04-24 18:07:27
 * @LastEditTime: 2022-04-24 21:10:24
 * @LastEditors: JIANG Yilun
 * @Description:
 * @FilePath: /Projet_cowsay_L1S2/newcow.c
 */

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

int affiche_vache(int *length, char *message, char *eyes, char *tongue, int *tail)
{
    printf(" -");
    for (int i = 0; i < *length; i++)
    {
```

```c
        printf("-");
    }
    printf("\n");
    printf("< %s >\n", message);
    printf(" -");
    for (int i = 0; i < *length; i++)
    {
        printf("-");
    }
    printf("\n");
    printf("    \\   ^__^\n");
    printf("     \\  (%s)\\_______\n", eyes);
    printf("        (__)\\       )\\/\\");
    for (int i = 0; i < *tail; i++)
    {
        printf("/\\");
    }
    printf("\n");
    printf("         %s ||----w |\n", tongue);
    printf("            ||     ||\n");
    printf("\n");
    return 0;
}

void update() { printf("\033[H\033[J"); }

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

int main(int argc, char *argv[])
{
    char *eyes = "oo"; // default eyes
    char *tongue = "  "; // default tongue
    char *message = "--help to display help"; // default message
    int tail = 1; // default tail
    for (int i = 1; i < argc; i++)
    {
        if (strcmp(argv[i], "-e") == 0 || strcmp(argv[i], "--eyes") == 0)
        {
            eyes = argv[i + 1];
        }
        if (strcmp(argv[i], "-T") == 0 || strcmp(argv[i], "--tongue") == 0)
        {
            tongue = argv[i + 1];
        }
        if (strcmp(argv[i], "-m") == 0 || strcmp(argv[i], "--message") == 0)
        {
            message = argv[i + 1];
        }
        if (strcmp(argv[i], "-t") == 0 || strcmp(argv[i], "--tail") == 0)
        {
            tail = atoi(argv[i + 1]);
        }
```

```
        if (strcmp(argv[i], "-h") == 0 || strcmp(argv[i], "--help") == 0)
        {
            printf("\n");
            printf("Usage: newcow [OPTION]...\n");
            printf("\n");
            printf("Options:\n");
            printf("  -e, --eyes=STRING  eyes of the cow (default: oo)\n");
            printf("  -t, --tongue=STRING tongue of the cow (default:  )\n");
            printf("  -m, --message=STRING message to display (default: none)\n");
            printf("  -h, --help          display this help and exit\n");
            printf("\n");
            return 0;
        }
    }
    int length = strlen(message) + 1;
    affiche_vache(&length, message, eyes, tongue, &tail);
}
```

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

S'il n'y a pas de message d'entrée:

```
$ gcc newcow.c && ./a.out -e AA -T U
 ------------------------
< --help to display help >
 ------------------------
    \   ^__^
     \  (AA)\_______
        (__)\       )\/\
         U |├ ---w |
           ||     ||
```

Si je veux obtenir des informations d'aide:

```
$ gcc newcow.c && ./a.out -h

Usage: newcow [OPTION]...

Options:
  -e, --eyes=STRING  eyes of the cow (default: oo)
  -t, --tongue=STRING tongue of the cow (default:  )
  -m, --message=STRING message to display (default: none)
  -h, --help          display this help and exit
```

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

```
$ gcc newcow.c && ./a.out -e AA -T UU -m "Hello, my name is JIANG Yilun"
 -------------------------------
< Hello, my name is JIANG Yilun >
 -------------------------------
    \   ^__^
     \  (AA)\_______
        (__)\       )\/\
         UU |+-ww-w |
            ||     ||
```

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

```
$ gcc newcow.c && ./a.out -e AA -T UU -m "Hello, my name is JIANG Yilun" -t 10
 -------------------------------
< Hello, my name is JIANG Yilun >
 -------------------------------
    \   ^__^
     \  (AA)\_______
        (__)\       )\/\/\/\/\/\/\/\/\/\/\
         UU |+-ww-w |
            ||     ||
```