Compte-rendu du projet Cowsay

Mathias Rivet

29 avril 2023

Les codes sources ainsi que des enregistrements d'exécution supplémentaires sont disponible à l'adresse : https://github.com/programindfr/INF203-Projet

Table des matières

1 Préliminaires

Argument	Usage	Exemple

`-n`	Permet d'afficher le texte sans qu'il ne soit formaté.	`mathias@pc:~$ cowsay -n`
		`Hello`
		`World !`
		`Ctrl+D`
		`  _________`
		`/ Hello   \`
		`\ World ! /`
		`  _________`
		`         \   ^__^`
		`          \  (oo)\_______`
		`             (__)\       )\/\`
		`                 ||----w |`
		`                 ||     ||`

-W	Permet de choisir le nombre de colonnes avant un retour à la ligne.	mathias@pc:~$ cowsay -W 5 Hello World !
		  ------ 	
		/ Hell \
		| o |
		| Worl |
		\ d ! /
		  ------ 	
		  \ ^__^
		  \ (oo)_______
		  (__)\ )\/\
		  |----w |
		  | ||

-b	La vache se transforme en cyborg.	mathias@pc:~$ cowsay -b Hello World !
		  --------------- 	
		< Hello World ! >
		  --------------- 	
		  \ ^__^
		  \ (==)_______
		  (__)\ )\/\
		  |----w |
		  | ||

-d	La vache semble morte.	mathias@pc:~$ cowsay -d Hello World !
		  --------------- 	
		< Hello World ! >
		  --------------- 	
		  \ ^__^
		  \ (xx)_______
		  (__)\ )\/\
		  U |----w |
		  | ||

-g	La vache a des yeux avides.	mathias@pc:~$ cowsay -g Hello World !
		  --------------- 	
		< Hello World ! >
		  --------------- 	
		  \ ^__^
		  \ ($$)_______
		  (__)\ )\/\
		  |----w |
		  | ||

`-p`	La vache est devenue folle.	`mathias@pc:~$ cowsay -p Hello World !`
		`< Hello World ! >`
		`\ ^__^`
		`\ (@@)\_______`
		`(__)\ )\/\`
		`||----w |`
		`|| ||`

`-s`	La vache est sous l'emprise de stupéfiants.	`mathias@pc:~$ cowsay -s Hello World !`
		`< Hello World ! >`
		`\ ^__^`
		`\ (**)\_______`
		`(__)\ )\/\`
		`U ||----w |`
		`|| ||`

`-t`	Affiche une vache fatiguée.	`mathias@pc:~$ cowsay -t Hello World !`
		`< Hello World ! >`
		`\ ^__^`
		`\ (--)\_______`
		`(__)\ )\/\`
		`||----w |`
		`|| ||`

`-w`	La vache est surexcitée.	`mathias@pc:~$ cowsay -w Hello World !`
		`< Hello World ! >`
		`\ ^__^`
		`\ (OO)\_______`
		`(__)\ )\/\`
		`||----w |`
		`|| ||`

`-y`	Affiche une jeune vache.	`mathias@pc:~$ cowsay -y Hello World !`
		`< Hello World ! >`
		`\ ^__^`
		`\ (..)\_______`
		`(__)\ )\/\`
		`||----w |`
		`|| ||`

`-e`	L'utilisateur peut choisir lui-même deux caractères pour les yeux de la vache.	`mathias@pc:~$ cowsay -e "><" Hello World !`
		`< Hello World ! >`
		`\ ^__^`
		`\ (><)\_______`
		`(__)\ )\/\`
		`||----w |`
		`|| ||`

`-T`	L'utilisateur peut choisir lui-même deux caractères pour la langue de la vache.	`mathias@pc:~$ cowsay -T \"\" Hello World !`
		`< Hello World ! >`
		`\ ^__^`
		`\ (oo)\_______`
		`(__)\ )\/\`
		`"" ||----w |`
		`|| ||`

`-l`	L'option `-l` liste l'ensemble des `cowfiles` présents à l'endroit pointé par la variable d'environnement `COWPATH`.	`mathias@pc:~$ cowsay -l`
		`Cow files in /usr/share/cowsay/cows: apt`
		`↪ bud-frogs bunny calvin cheese cock cower`
		`↪ daemon default dragon dragon-and-cow duck`
		`↪ elephant elephant-in-snake eyes`
		`↪ flaming-sheep fox ghostbusters gnu`
		`↪ hellokitty kangaroo kiss koala kosh`
		`↪ luke-koala mech-and-cow milk moofasa moose`
		`↪ pony pony-smaller ren sheep skeleton`
		`↪ snowman stegosaurus stimpy suse three-eyes`
		`↪ turkey turtle tux unipony unipony-smaller`
		`↪ vader vader-koala www`

`-f`	L'option `-f` permet de choisir quel `cowfile` afficher. Si un chemin est donné, il sera interprété comme un chemin relatif jusqu'au `cowfile`, sinon la variable d'environnement `COWPATH` sera utilisée.	`mathias@pc:~$ cowsay -f tux Hello World !`
		`< Hello World ! >`
		`\`
		`\`
		`.--.`
		`|o_o |`
		`|:_/ |`
		`// \ \`
		`(| | )`
		`/`_ _/`\`
		`\___)=(___/`

-h	L'option -h permet d'afficher le message d'aide de cowsay.	mathias@pc:~$ cowsay -h 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]

2 Bash

2.1 cow kindergarten

Script

```
#!/bin/bash

for i in $(seq 10)
do
    clear
    if [ $i -eq 10 ]        # test si i est égal à 10
    then
        cowsay -T "U " $i    # si oui la vache tire la langue
    else
        cowsay $i
        sleep 1
    fi
done
```

Exemple

```
mathias@pc:~$ chmod +x cow_kindergarten
mathias@pc:~$ ./cow_kindergarten
 ____
< 10 >
 ----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
```

2.2 cow primaryschool

Script

```
#!/bin/bash

for i in $(seq $1)
do
    clear
    if [ $i -eq $1 ]        # test si i est égal au premier argument
    then
        cowsay -T "U " $i   # si oui la vache tire la langue
    else
        cowsay $i
        sleep 1
```

```
    fi
done
```

Exemple

```
mathias@pc:~$ chmod +x cow_primaryschool
mathias@pc:~$ ./cow_primaryschool 20
 ____
< 20 >
 ----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
```

2.3 cow highschool

Script

```
#!/bin/bash

for i in $(seq $1)
do
    clear
    if [ $i -eq $1 ]                    # test si i est égal au premier argument
    then
        cowsay -T "U " $(( $i * $i ))   # si oui la vache tire la langue
    else
        cowsay $(( $i * $i ))           # on affiche le carré de i
        sleep 1
    fi
done
```

Exemple

```
mathias@pc:~$ chmod +x cow_highschool
mathias@pc:~$ ./cow_highschool 10
 _____
< 100 >
 -----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
```

2.4 cow college

Script Pour une raison non élucidée, `cowsay 0` attend un input depuis `stdin`, ce qui bloque l'exécution. Il faut donc rajouter un espace pour corriger le problème.

```
#!/bin/bash

f0=0                                    # On définit f1 et f0.
f1=1

if [ $1 -eq 1 ]                         # Pour n égal à 1, c'est déjà la fin
                                        # et la vache tire la langue.
```

```
then
    cowsay -T "U " "$f0 "                # On ajoute un espace sinon cowsay
                                         # reste bloqué pour afficher 0.
elif [ $1 -ge 2 ]                        # Pour n supérieur ou égal à 2, f0
                                         # et f1 sont affichés.
then
    cowsay "$f0 "                        # On ajoute aussi un espace pour 0.
    sleep 1
    clear
    cowsay $f1
    sleep 1
    while
        fn=$(($f0 + $f1))
        [ $fn -lt $1 ]                   # Est-ce que fn+1 est supérieur à n ?
    do                                   # Non alors on peut l'afficher.
        clear
        f0=$f1                           # On incrémente d'un cran f0 et f1.
        f1=$fn
        cowsay $f1
        if [ $(($f0 + $f1)) -lt $1 ]     # Si ce n'est pas le dernier
                                         # affichage on attend.
        then
            sleep 1
        fi
    done                                 # Oui alors la vache tire la langue
                                         # en affichant fn.
    clear
    cowsay -T "U " $f1
fi
```

Exemple

```
mathias@pc:~$ chmod +x cow_college
mathias@pc:~$ ./cow_college 0
mathias@pc:~$ ./cow_college 1
 ____
< 0  >
 ____
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
mathias@pc:~$ ./cow_college 2
 ___
< 1 >
 ___
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
mathias@pc:~$ ./cow_college 60
 ____
< 55 >
 ____
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
```

```
                ||      ||
```

2.5 cow university

Script On utilise ici la méthode du crible d'Ératosthène.

```
#!/bin/bash

i=2                                             # 0 et 1 ne sont pas des nombres premiers.
while [ $(($i * $i)) -lt $1 ]                   # On cherche la racine de n.
do
    i=$(($i + 1))
done

lastPrime=2                                     # On mémorise le dernier nombre premier
                                                # pour l'afficher en tirant la langue.
for k in $(seq 2 $1)
do
    isPrime=0                                   # Booleen pour savoir si le nombre est
                                                # premier.
    for j in $(seq 2 $i)
    do
        if [ $(($k % $j)) -eq 0 -a $k -ne $j ]  # Si k modulo j est égal à 0 et que k est
                                                # différent de j alors k n'est pas premier
                                                # (pour j quelconque fixé entre 2 et
                                                # racine de n).
        then
            isPrime=1                           # Booleen indiquant que k n'est pas
                                                # premier.
        fi
    done
    if [ $isPrime -eq 0 ]                       # Si k est premier on l'affiche.
    then
        if [ $k -gt 2 ]                         # Pas de pause avant le premier affichage.
        then
            sleep 1
        fi
        lastPrime=$k                            # On mémorise le nombre premier actuel.
        clear
        cowsay $lastPrime                       # On affiche le nombre premier actuel.
    fi
done

if [ $1 -ge 2 ]                                 # 0 et 1 ne sont pas premiers.
then
    clear
    cowsay -T "U " $lastPrime                   # La vache tire la langue pour le dernier
                                                # nombre premier.
fi
```

Exemple

```
mathias@pc:~$ chmod +x cow_university
mathias@pc:~$ ./cow_university 0
mathias@pc:~$ ./cow_university 1
mathias@pc:~$ ./cow_university 100
 ____
< 97 >
 ----
        \   ^__^
```

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

2.6 smart cow

Script Les deux points clés de ce script sont les fonctions `wc` et `cut`. Le fait que `bash` interprète les variables selon le contexte est utile pour réduire la taille du code et les répétitions. Ainsi le caractère opérateur est interprété selon le contexte ce qui réduit la taille du code par 5.
BONUS : la vache a aussi appris à calculer le modulo de deux nombres !

```bash
#!/bin/bash

length=$(($(echo "$1" | wc -c) - 1))    # On récupère la taille de la chaine de
                                         # caractère sans compter le caractère \n de
                                         # fin.
for i in $(seq $length)
do
    char=$(echo "$1" | cut -c $i)        # La variable char contient le caractère de la
                                         # chaine qui se situe à la i-ème position.
    if [ "$char" = '+' -o "$char" = '-' -o "$char" = '*' -o "$char" = '/' -o "$char" = '%'
    ↪  ]
                                         # Si char est un opérateur alors on sait comment
                                         # découper la chaine puisqu'on connait sa
                                         # position.
    then
        deb=$(echo "$1" | cut -c 1-$(($i - 1)))
                                         # deb contient les caractères de la position 1 à
                                         # i-1 (la position juste avant l'operateur).
        fin=$(echo "$1" | cut -c $(($i + 1))-$length)
                                         # fin contient les caractères de la position i+1
                                         # à length (position juste après l'opérateur
                                         # à la position du dernier caractère).
        cowsay -e "$(($deb $char $fin))" "$deb $char $fin"
                                         # Selon le contexte, char est interprété comme un
                                         # opérateur arithmétique ou un caractère.
    fi
done
```

Exemple

```
mathias@pc:~$ chmod +x smart_cow
mathias@pc:~$ ./smart_cow '3 + 11'
 --------
< 3 + 11 >
 --------
        \   ^__^
         \  (14)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
mathias@pc:~$ ./smart_cow 3-11
 --------
< 3 - 11 >
 --------
        \   ^__^
         \  (-8)\_______
            (__)\       )\/\
```

```
                ||----w |
                ||     ||
```

2.7 crazy cow

Script La `crazy_cow` affiche les termes de la suite de Conway jusqu'au terme de rang `n` avec `n` un nombre donné en argument du script. La vache marque une pause entre chaque terme et tire la langue pour afficher le dernier terme. Le script de `crazy_cow` est inspiré de `cow_college` pour l'aspect de suite définie par récurrence et de `smart_cow` pour l'aspect d'opérations sur des chaînes de caractères. Attention, il se pourrait que la vache soit surexcitée par le calcul des différents termes.

```
#!/bin/bash

suivant(){                                  # On définit une fonction pour le calcul du
                                            # terme suivant.
    length=$(($(echo "$1" | wc -c) - 1))    # Calcule la taille du terme actuel.
    xSuivant=""
    nb=0
    j=1
    while [ $j -le $length ]                # Itération sur tous les caractères.
    do
        chiffre=$(echo "$1" | cut -c $j)    # Caractère du chiffre à compter.
        nb=1
        while [ "$chiffre" = "$(echo "$1" | cut -c $(($j + $nb)))" ]
                                            # Tant que le caractère suivant correspond au
                                            # même chiffre on compte le nombre
                                            # d'occurences.
        do
            nb=$(($nb + 1))
        done
        xSuivant="$xSuivant$nb$chiffre"     # On ajoute le nombre d'occurences du chiffre
                                            # et le chiffre lui-même.
        j=$(($j + $nb))                     # On passe au chiffre suivant en ignorant les
                                            # occurences du chiffre actuel.
    done
    echo $xSuivant                          # Équivalent à return.
}

xn=1                                        # Initialisation de la suite.
for i in $(seq 0 $1)                        # On affiche jusqu'au rang n.
do
    clear
    if [ $i -lt $1 ]                        # Est-ce qu'il faut calculer la suite ou tirer
                                            # la langue.
    then
        cowsay -w $xn
        sleep 1
        xn=$(suivant $xn)
    else
        cowsay -T "U " $xn
    fi
done
```

Exemple

```
mathias@pc:~$ chmod +x crazy_cow
mathias@pc:~$ ./crazy_cow 0
 ___
< 1 >
 ---
```

```
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
mathias@pc:~$ ./crazy_cow 3
 ------
< 1211 >
 ------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
mathias@pc:~$ ./crazy_cow 10
 ----------------------------
< 11131221133112132113212221 >
 ----------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
```