

# Jeu de l'oie

## Premier TD linux de LMI1

### Remarques :

↪ Date de remise de l'énoncé du TD : semaine du 12 mars 2007

↪ Date d'évaluation du TD : semaine du 26 mars 2007

Renseignez-vous auprès de votre professeur afin de connaître ses modalités de remise avant la date fatidique<sup>1</sup>.

DANS ce TD nous allons implémenter un **jeu de l'oie** ultra-basique. Ce jeu permettra à un joueur de jouer seul à condition qu'il se soit au préalable muni d'un dé. En effet, dans ce jeu, nous nous contenterons d'afficher un plateau de jeu et de faire avancer l'unique pion sur ce plateau<sup>2</sup>.

Le plateau de jeu se présente comme une "simple" suite de nombres<sup>3</sup>. L'arrivée étant marquée par une \* par exemple. Un peu comme suit :

1	2	3	4	5	6	7	8	9	10
36	37	38	39	40	41	42	43	44	11
35	64	65	66	67	68	69	70	45	12
34	63	84	85	86	87	88	71	46	13
33	62	83	96	97	98	89	72	47	14
32	61	82	95	*	99	90	73	48	15
31	60	81	94	93	92	91	74	49	16
30	59	80	79	78	77	76	75	50	17
29	58	57	56	55	54	53	52	51	18
28	27	26	25	24	23	22	21	20	19

Pour utiliser le programme<sup>4</sup>, il faudra entrer le nombre donné par le dé et le programme affichera l'avancement du pion (il suffit de réafficher le plateau de jeu en remplaçant l'un des nombres par le pion). Il est clair qu'il faudra tenir compte des règles du jeu, du style : "*vous pouvez avancer de 3 cases*"

## 1 Les fonctionnalités

### 1.1 Fonctionnement du jeu

Le programme affichera à chaque étape le plateau de jeu suivi des règles (voir infra) et demandera la valeur du dé (c'est-à-dire la lecture d'un entier compris entre 1 et 6 par exemple). Après cette lecture, le pion est déplacé sur le plateau et l'on passe à l'étape suivante.

<sup>1</sup>N'oubliez pas qu'une date se compose, dans notre cas, du jour et d'une *heure*

<sup>2</sup>J'avais dit que ce serait très basique

<sup>3</sup>Le lecteur attentif remarque déjà que l'ordre n'est pas *si simple*

<sup>4</sup>Je n'ose pas dire "jouer"

## 1.2 Règles du jeu

Le pion est avancé du nombre de cases correspondant à la valeur du dé. Si cette valeur est plus grande que la distance séparant le dé de l'arrivée, le pion avance jusqu'à l'arrivée et recule ensuite d'un nombre de cases égal à la différence entre la valeur initiale du dé et la distance (en nombre de case) jusqu'à l'arrivée.

Si le pion se trouve sur une *case spéciale*, on exécute l'action associée à la case. Ce traitement de la case spéciale ne se fait qu'une seule fois par tour. Ce qui veut dire qu'atterrir sur une case spéciale grâce à une avancée ou un recul spécial n'accorde aucune nouvelle faveur ou défaveur.

Les **cases spéciales** sont

- ↪ Le numéro de la case est un multiple de 7, j'avance de 7 cases
- ↪ 10, c'est la moitié, je retourne sur la case 5
- ↪ 19, je retourne à la case 9
- ↪ 23, j'avance de 3
- ↪ 24, j'avance de 4
- ↪ 25, j'avance de 5
- ↪ 36, tant pis, je retourne à la case 26
- ↪ 42, c'est la réponse, je me rends à la case 100
- ↪ 51,52,53,54,55, j'avance de la même valeur que le dé
- ↪ 65,67, je peux relancer le dé
- ↪ 71,73,75,77, j'avance de la somme des chiffres du numéro de la case
- ↪ 82,83,85,87, je recule de la somme de mes chiffres
- ↪ 99, pas de bol, je retourne à la case départ

## 2 Aspects pratiques

Ce TD n'est pas compliqué si il est découpé en petites difficultés successives. Essayez de résoudre les différents points ci-dessous avant de vous lancer dans la solution complète de l'exercice<sup>5</sup>.

### 2.1 Ecrire du texte

Il y a deux manières de faire pour écrire du texte à l'écran (voir [1] 91-92). La première est d'utiliser l'instruction `int 0x80` et de consulter le fichier `unistd.h` ... dans lequel on trouve

```
#define __NR_write 4
```

Un `man 2 write` m'apprend que `write` permet d'écrire *count* octet dans le descripteur de fichier *fd* (1 pour `stdout`). Les octets que je veux écrire se trouvant dans le buffer *buf*

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t count);
```

<sup>5</sup>Ceci est simplement un conseil, vous travaillez comme vous l'entendez

Il me suffit alors d'écrire quelque chose du genre

```

...
section .data
buf DB "Mon_texte"
section .texte
...
mov EAX,4 ; numero de l'appel systeme
mov EBX,1 ; stdout
mov ECX,buf
mov EDX,9 ; nombre de bytes à ecrire
int 0x80h
...

```

Ceci devrait écrire votre texte à l'écran ...

Une autre manière d'écrire du texte à l'écran est d'utiliser la *libc* (la librairie C standard) avec des fonctions C standard. Vous aurez alors envie d'utiliser l'une des fonctions C permettant d'écrire du texte ; `write`, `puts`, ... et vous relirez le slide 91 dans [1].

## 2.2 Lire un caractère, `readchar`

Lire un caractère à l'écran se fait grâce à la fonction `read` ou `getchar` ... et se code de la même manière que dans 2.1.

## 2.3 Parcours ... du tableau

Les considérations techniques étant levées, il reste à traiter de la logique de programmation. Vous aurez remarqué que le parcours du tableau n'est pas vraiment en adéquation avec le parcours du plateau de jeu. Vous pouvez créer un tableau faisant le lien entre les deux. Lorsque vous avancez de 3 cases. Vous avancez de trois dans le tableau "de lien" et y lisez la position dans le plateau de jeu.

## Table des matières

<b>1</b>	<b>Les fonctionnalités</b>	<b>1</b>
1.1	Fonctionnement du jeu . . . . .	1
1.2	Règles du jeu . . . . .	2
<b>2</b>	<b>Aspects pratiques</b>	<b>2</b>
2.1	Ecrire du texte . . . . .	2
2.2	Lire un caractère, <code>readchar</code> . . . . .	3
2.3	Parcours ... du tableau . . . . .	3

## Webographie-bibliographie

[1] Pierre BETTENS et les professeurs du cours. Slides de microprocesseurs, 2004-2007. <http://esi.namok.be>.