L2 Informatique — Université de Nantes
X4I0010 – Système d'exploitation — 2016/2017

Contrôle continu
Contrôle de travaux pratiques – Contrôle final (barème)

Dans un parc naturel nous avons introduit une population de 150 cerfs (deers) avec l'objectif d'encourager la chasse chez les habitants de la région. Dans le même temps nous souhaitons que la population de cerfs ne décroisse pas rapidement. Pour cette raison nous disposons d'un inspecteur chargé de vérifier l'état de la population une fois par mois.

À partir de la fréquence de reproduction des cerfs et de divers facteurs (météorologique, alimentation, prédateur, etc.) il est établi des règles indiquant les mois de l'année où la chasse des cerfs est autorisée dans le parc.

L'objectif de cet exercice est de simuler l'évolution de la population de cerfs dans le parc naturel au fil des mois.

Exercice 1 (Les processus [13 puntos])

Dans un premier temps, nous souhaitons publier les règles de chasse du parc et les périodes de reproduction des cerfs, pour chaque mois. Pour cela, nous allons utiliser deux processus différents :

— Un processus **A** qui n'a pas le droit d'afficher le mot "*hunting*", et
— Un processus **B** qui n'a pas le droit d'afficher le mot "*reproduction*".

Question 1 : En utilisant des processus et des signaux, complétez le programme `proc_2017.cpp` (à télécharger depuis MADOC) pour afficher les règles du parc (présents dans le fichier `np_rules.txt`) selon l'exemple suivent :

Contenu du fichier `np_rules.txt` :

```
000111111101
101011111110
```

La 1^{re} ligne (resp. la 2^{me}) de ce fichier liste, pour les 12 mois de l'année, s'il s'agit d'un mois de reproduction (resp. de chasse) ou non.
Ainsi, le texte affiché pour l'exemple précédent sera :

```
1 -> Only hunting
2 -> Nothing
3 -> Only hunting
4 -> Only reproduction
5 -> Reproduction and hunting
6 -> Reproduction and hunting
7 -> Reproduction and hunting
8 -> Reproduction and hunting
9 -> Reproduction and hunting
10 -> Reproduction and hunting
11 -> Only hunting
12 -> Only reproduction
```

[4 puntos] : Utilizar la función `signal(...)` para la asignación de *handlers* a señales.

```
signal (SIGUSR1, hand_SGL_test);
signal (SIGUSR2, hand_SGL_print_and_test);
```

[3 puntos] : Utilizar la función `fork()` para la creación de procesos independientes, teniendo en cuenta sus respectivos identificadores (PID).

```
pid_t pid;
if((pid = fork()) == -1)
{
   perror("Error forking");
   exit(1);
}
```

[3 puntos] : Saber como utilizar el PID para diferenciar las el proceder de cada proceso.

```
if (pid == 0)
{ /* process 0 */
   ...
}
else
{ /* process 1 */
   ...
}
```

[3 puntos] : Dar una solución respetanto las reglas impuestas.

Exercice 2 (Les threads et les sémaphores [7 puntos])

Le programme `thr_2017.cpp` (à télécharger depuis MADOC) contient une simulation, du comportement que nous souhaitons étudier dans le parc naturel en utilisant la fonction `sleep(...)`. Chaque acteur (cerfs, chasseurs, inspecteur) seront représentés par un fil d'exécution (*thread*) différent. Chacun d'eux doivent suivre les règles suivantes :

— Cerfs :

1. Les cerfs se reproduisent une seule fois pendant les 10 premiers jours de chaque mois indiqué dans les règles générales du parc (fichier `np_rules.txt`), selon l'équation suivante :

$$pop_{i+1} = 1.15 \cdot pop_i$$

   pop_k représentant la population des cerfs du mois k.

2. Les cerfs peuvent se reproduire le mois X si l'inspecteur a déjà publié les résultats de l'inspection du mois $X - 1$.

— Chasseurs :

1. Les chasseurs ont le droit de chasser seulement un jour dans la deuxième quinzaine de chaque mois indiqué dans les règles générales du parc (fichier `np_rules.txt`).
2. Les chasseurs ont le droit de chasser un maximum de 25 animaux par mois.

   (bonus) : Implémentez une fonction de probabilité (uniforme, par exemple) pour faire varier le nombre de cerfs chassé à chaque fois.
3. Les chasseurs ont de droit de chasser le mois X si l'inspecteur a déjà publié les résultats de l'inspection du mois $X - 1$.

— Inspecteur :

1. L'inspecteur publie les résultats de l'inspection le dernier jour de chaque mois à 23h59, en affichant la population actuelle des cerfs dans le parc naturel.

Question 1 : En utilisant des *threads* et des sémaphores, modifiez le programme `thr_2017.cpp` (sans utiliser la fonction `sleep(...)`) pour obtenir le résultat qu'il affiche tout en respectant les lois décrites précédemment.

Utilización de semáforos :

— **[1 punto]** Declaración de los semáforos necesarios.

```
sem_t synchro_reproduction;
sem_t synchro_hunting
sem_t synchro_inspection;
sem_t mutex_deers;
```

— **[1 punto]** Utilización de las funciones `sem_init(...)` y `sem_destroy(...)` para la creación y destrucción de los semáforos.

```
int main()
{
   ...
   /* Initialisation du semaphore */
   sem_init(&synchro_reproduction, 0, 1);
   sem_init(&synchro_hunting, 0, 0);
   sem_init(&synchro_inspection, 0, 0);
```

```
    sem_init(&mutex_deers, 0, 1);
    ...
    /* Destruction of semaphores */
    sem_destroy(&synchro_reproduction);
    sem_destroy(&synchro_hunting);
    sem_destroy(&synchro_inspection);
    sem_destroy(&mutex_deers);
}
```

— **[1 punto]** Utilización de funciones `sem_wait(...)` y `sem_post()` para la syncronización.

— **[2 puntos]** Correcta utilización de las funciones `sem_wait(...)` y `sem_post()` para la syncronización en cada *thread*.

Deer thread :

```
void * code_thread_deer(void * arg)
{
    ...
    sem_wait(&synchro_reproduction);
    sem_wait(&mutex_deers);
    ...
    sem_post(&mutex_deers);
    sem_post(&synchro_hunting);
    ...
}
```

Hunter thread :

```
void * code_thread_hunter(void * arg)
{
    ...
    sem_wait(&synchro_hunting);
    sem_wait(&mutex_deers);
    ...
    sem_post(&mutex_deers);
    sem_post(&synchro_inspection);
    ...
}
```

Instector thread :

```
void * code_thread_inspector(void * arg)
{
    ...
    sem_wait(&synchro_inspection);
    sem_wait(&mutex_deers);
    ...
    sem_post(&mutex_deers);
    sem_post(&synchro_reproduction);
```

```
    ...
}
```

— **[2 puntos]** Solución correcta respetando las reglas.

Question 2 (bonus) : Modifiez le fichier `np_rules.txt` afin que la population de cerfs survivre jusqu'à 1 janvier 2018. (Solo se tendrá en cuenta si la solución entregada es perfecta)