Master d'informatique | **module PC2R** | **Université Pierre et Marie Curie**
Documents autorisés | *Durée 2 heures* | **Année 2012-2013**

Examen réparti du 08 janvier 2013

Exercice : Les lampes et son interrupteur (Esterel)

Un système d'éclairage est composé de 3 lampes numérotées de 1 à 3 et d'un interrupteur, et fonctionne de la façon suivante :

- Au départ, toutes les lampes sont éteintes.
- L'utilisateur actionne l'interrupteur (en émettant un `CLIC`). Successivement, la lampe 1 s'allume en émettant le signal valué `LAMPE(1)` pendant `n` tick, puis la lampe 2, puis la lampe 3 (voir l'exemple d'exécution ci-dessous avec `n = 5`).
- Il peut aussi actionner l'interrupteur avec un double `CLIC` (en émettant deux `CLIC` dans un laps de temps de 3 ticks, voir l'exemple d'exécution). Les 3 lampes s'allume ensemble en émettant chacune son signal pendant `n` tick. On voit afficher `LAMPE(6)` indiquant que les 3 lampes sont allumées en même temps.
- A chaque `CLIC` ou double `CLIC`, ce dernier annule le cycle précédent (dans l'exemple, la lampe 3 a été interrompue).

Vous pouvez et/ou devez ajouter la déclaration des signaux, variables ou autres jugés utiles pour les modules. On évitera de faire des attentes actives.

Question 1 Compléter le module `lampe` permettant de répondre au comportement énoncé ci-dessus.

Question 2 Compléter le module `interrupteur` permettant de gérer le `CLIC` ou le double `CLIC`.

Question 3 Compléter le module `systeme` permettant de mettre en place le système et répondre au comportement énoncé ci-dessus.

Modules

```
module systeme :

input CLIC;
     % ?????????????????????????????????????? %
end module

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

module interrupteur :

input CLIC;
     % ?????????????????????????????????????? %
end module

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

module lampe :

constant numero, n : integer;
    % ??????????????????????????????????????? %
end module
```

Exemple d'exécution

```
$ ./systeme
systeme> ;CLIC;;;;;;;;;;;;CLIC;;CLIC;;;;;;
--- Output:
--- Output: LAMPE(1)
--- Output: LAMPE(1)
--- Output: LAMPE(1)
--- Output: LAMPE(1)
--- Output: LAMPE(1)
--- Output: LAMPE(2)
--- Output: LAMPE(2)
--- Output: LAMPE(2)
--- Output: LAMPE(2)
--- Output: LAMPE(2)
--- Output: LAMPE(3)
--- Output: LAMPE(3)
--- Output: LAMPE(1)
--- Output: LAMPE(1)
--- Output: LAMPE(6)
--- Output: LAMPE(6)
--- Output: LAMPE(6)
--- Output: LAMPE(6)
--- Output: LAMPE(6)
--- Output:
systeme>
```

Problème : Modélisation postale (Client-Serveur en 1 ou 2 langages)

Conseil : Lisez l'énoncé de cet exercice en entier avant de commencer à répondre aux questions.

Dans cet exercice, pour des raisons pratiques, il n'y a aucune gestion d'erreurs à effectuer ni aucun comportement hors limite. Exemples : si on attend 2 chiffres, alors on aura bien exactement 2 chiffres ; les hôtes distants sont toujours joignables ; les ports sont toujours libres quand on en a besoin.

On propose de modéliser un bureau de poste entièrement géré par des machines communiquant via TCP/IP.

Le *chef* du bureau de poste (modélisé par un serveur qui écoute sur 10.0.0.1) gère l'ordonnancement des tâches de tous les autres *employés* (modélisés par des clients-serveurs). Le *chef* peut être contacté par les *employés* via le port `_CHEF`. Il répond au protocole textuel suivant avec exactement une requête ou réponse par ligne :

Requête	Réponse
`VACANT/`*id*`/`	`OK`
`PAUSE*/`*id*`/`*durée-demandée*`/`	*durée-accordée*
`BUSY**/`*id*`/`	`OK`
`HUMAN*/`	*vacant id*
`TEMPER/`	*température du bureau de poste*
`DISPAT/`*id*`/`*poids*`/`*prix*`/`	`OK`
`MMM***/`	*montant-de-la-recette*

- *id* est remplacé par le numéro de l'*employé*, compris entre 42 et 99.
- Toutes les durées sont en minutes entières et sont toujours sur 2 chiffres (quitte à ajouter un zéro devant).
- Chaque *employé* a droit à 42 minutes de pause au total durant la durée de vie du *chef* (qui lui n'a jamais de pause).
- À chaque fois qu'un humain entre dans le bureau de poste, un détecteur envoie `HUMAN*/` au chef et celui-ci répond avec le numéro d'un *employé* disponible.
- La température initiale du bureau de poste est de 20˚C.
- Lorsque la température dépasse 22˚C, on déduit une unité de la recette par envoi effectué.
- *poids* est remplacé par le poids de la lettre (`LETTER`) ou du colis (`PACKAG`).
- Tous les poids sont en grammes et limités à 9876 grammes, ils sont tous représentés sur exactement 4 chiffres (quitte à ajouter des 0 devant).
- Tous les prix sont des nombres entiers représentés sur exactement 3 chiffres, et correspondent à (1 + poids / 100) pour les lettres et (2 + poids / 100) pour les colis, arrondis à l'entier inférieur.
- **Remarque : les mots (nom de la commande, arguments) dans le protocole ont une taille toujours fixe, profitez-en.**

Les *employés* (10.0.0.XXX) peuvent être contactés par n'importe qui via le port `_EMPL`. Ils répondent au protocole textuel suivant avec exactement une requête ou réponse par ligne :

Requête	Réponse
`LETTER/`*poids*`/`	*prix*
`PACKAG/`*poids*`/`	*prix*
`STATUS/`	`AVAILABLE` ou `BUSY` ou `AWAY`

- Lorsqu'un *employé* reçoit `STATUS`, il répond avec son statut qui doit être cohérent avec celui qu'il a déclaré au *chef*.
- Lorsqu'un *employé* a géré 100 envois, il fait augmenter la température du bureau de poste de 1˚C à chaque kilogramme supplémentaire envoyé. Lorsqu'un *employé* est en surchauffe, il demande une pause d'au moins une minute à son *chef*. Chaque minute de pause effectuée fait baisser la température de la pièce de 1˚C, dans la limite du nombre de ˚C que l'*employé* a contribué. Lorsqu'un *employé* a fini sa pause, il doit déclarer son retour au *chef*.

Question 4 Pour cette question, il faudra donner les parties intéressantes du code d'un serveur (le *chef*) qui gère ses *employés* en répondant aux spécifications données plus haut. Vous devez choisir un (seul) langage parmi OCaml, Java et C.

Si vous utilisez des variables globales, il faut les expliciter. Si vos noms de variables ne sont pas explicites, il faut les documenter.

1. Écrire le code qui gère les statuts des *employés*.
2. Écrire le code qui gère la température (donc des poids des envois aussi) du bureau de poste.
3. Écrire le code qui gère la sélection d'un *employé* lorsque le *chef* reçoit `HUMAN*/`.
4. Écrire le code qui traite les requêtes selon le protocole.
5. Expliciter les fonctions (ou procédures ou classes/méthodes) qui manquent pour compléter le serveur afin de pouvoir le lancer. Vous pouvez utiliser le support de cours comme référence, par exemple : *cours 9, page 42, lignes 4 à 8*.
6. Donner la commande permettant de compiler et de lancer le serveur.

Question 5 Pour cette question, il s'agira de donner le code pour créer les *employés* (qui sont chacun à la fois client et serveur). Vous devez choisir un (seul) langage parmi OCaml, Java et C (différent de celui ou le même que vous avez choisi pour la question précédente).

1. Écrire le code de gestion des pauses (pour pauser, vous pouvez utiliser la fonction `PC2R.sleep:int->unit` en OCaml, `PC2R_sleep(int seconds)` en C, `PC2R.sleep(int seconds)` en Java).
2. Écrire le code qui traite les requêtes selon le protocole. N'oubliez pas de gérer les statuts des *employés* (vis-à-vis d'eux-mêmes et vis-à-vis du *chef*).
3. Écrire le code pour créer N ($1 \leq N \leq 200$) *employés* répondant aux spécifications données plus haut et dont les adresses IPv4 vont de 10.0.0.42 à 10.0.0.(42+N-1).
4. Donner la commande permettant de compiler votre code, et donner celle permettant de créer 58 *employés*.

Question 6 Votre système permet-il d'empêcher une augmentation de la température de plus de 2°C ? Si oui, expliquez comment. Si non, que proposez-vous pour éviter que tous les *employés* ne soient en pause en même temps tout en maintenant la température du bureau de poste en dessous de cette température ? Votre réponse doit être brève mais surtout convaincante.

Question 7 Que proposez-vous pour que lorsque tous les *employés* sont en pause en même temps, celui le plus apte à être remis prématurément au travail soit appelé à mettre fin à sa pause lorsqu'un humain entre dans le bureau de poste ? **L'efficacité** de votre système sera particulièrement appréciée ; n'hésitez pas à l'expliciter si elle n'est pas évidente : vous devez être convaincant.