Contrôle de Systèmes d'Exploitation
(1h30 - Documents de cours autorisés)
18 mars 2022

Exercice 1 : Questions générales

Répondre aussi brièvement et clairement que possible à chaque question.

1. À quoi sert un système d'exploitation ?
2. Qu'est-ce qu'une tâche ? Quelles sont les différences entre un processus lourd et un processus léger ?
3. L'ordonnanceur est un élément important des systèmes d'exploitation. Quel est son rôle ?
4. Qu'est-ce que le tourniquet (round-robin) ?
5. Un utilisateur a écrit un sous-programme de temporisation sensé durer une minute. Pour cela, il a simplement utilisé une boucle à compteur de sorte à exécuter n instructions où n est tel que $n/p = 60$ sachant que p est le nombre d'instructions exécutées en une seconde par le micro-processeur. Lorsqu'il compare le fonctionnement de ce sous-programme avec sa montre, l'utilisateur s'aperçoit que le résultat n'est pas celui attendu. Pourquoi ? La temporisation exécutée par le sous-programme est-elle trop longue ou trop courte ?

Exercice 2 : Virtualisation

Question 1 : *Quels sont les intérêts et les inconvénients de la paravirtualisation ?*

Question 2 : *Quels sont les avantages à utiliser les conteneurs Docker ? A-t-on réellement un système virtualisé ?*

Question 3 : *Ecrire le contenu d'un fichier Dockerfile qui :*
- *exécute le système minimal alpine ;*
- *exécute la commande apk pour installer gcc ;*
- *et compile le fichier toto.c.*

Quelle commande doit-on exécuter pour lancer un terminal dans le système alpine du conteneur ? Comment faire pour que le fichier compilé (toto.c → a.out) soit exécuté au lancement du conteneur ?

Exercice 3 : Micro-noyaux

Question 1 : *Quels sont les avantages des micro-noyaux ? A quoi servent les IPC ?*

Question 2 : *Qu'est-ce que le TCB ? A quoi sert le buffer dans le TCB des threads du micro-noyau L4 ?*

Question 3 : *Un micro-noyau comme L4 peut être utilisé comme hyperviseur d'une machine virtuelle exécutant un système Linux. Que se passe-t-il lorsqu'un processus du système Linux exécute l'appel système* `fork` *? Qui contrôle le processus créé ?*

Exercice 4 : Systèmes d'exploitation pour l'IoT

Question 1 : *Les systèmes d'exploitation pour l'IoT doivent avoir une empreinte mémoire faible. Expliquez pourquoi ?*

Nous proposons de programmer une application qui collecte la luminosité d'une pièce en utilisant le capteur OPT-3001 d'un Ti Sensortag.

Le protocole CoAP (Constrained Application Protocol) est un protocole de communication de niveau applicatif pour systèmes IoT. Il s'apparente au protocole HTTP et transmet les données sur IPv6. Le système Contiki possède une pile de communication contenant CoAP. Pour émettre des données via le protocole CoAP, la démarche est la suivante :

1. Définition du message (ici une requête de transmission) :

```
coap_message_t request;
```

2. Initialisation du message :

```
coap_init_message(&request, COAP_TYPE_CON, COAP_POST, 0);
```

3. Identification de la requête, ici la température :

```
coap_set_header_uri_path(&request, "/light");
```

4. Envoi de la requête et attente de la réponse :

```
COAP_BLOCKING_REQUEST(&server_ep, &request, receipt_func);
```

La variable `server_ep` identifie la connexion CoAP avec le serveur. Nous considérons que cette connexion est déjà établie. La fonction de réception `receipt_func` est appelée automatiquement lors de la réception du message par notre application. Cette fonction est définie par le code suivant :

```
void receipt_func(coap_message_t *response)
{
  const unsigned char *data;
  if(response == NULL) {
    puts("Request␣timed␣out");
    return;
  }
  int len = coap_get_payload(response, &data);
  printf("light:␣%s␣lux", (char *)data);
}
```

Question 2 : *Ecrire en langage C pour le système d'exploitation Contiki le code du thread* `light_collect` *qui émet une requête de luminosité périodiquement toutes les 10 ms en utilisant le protocole COAP.*

Le système de collecte dispose aussi du capteur de lumière OPT-3001. La quantité de lumière est lue par un thread Contiki nommé `local_light`. Ainsi, il est possible de comparer la luminosité où se situe l'instrument de collecte et la luminosité récupérée via le thead `light_collect`.

Question 3 : *Dans contiki, le capteur de luminosité se nomme* `opt_3001_sensor`. *Une lecture de la luminosité se fait en activant le capteur via la commande* `SENSORS_ACTIVATE()`. *Lorsque la lecture est terminée, un événement* `sensors_changed` *est émis. La fonction* `value` *sans paramètre associée au capteur permet de récupérer la valeur. Écrire en langage C, le code du thread* `local_light` *Contiki qui lit périodiquement la luminosité ambiante toutes les 10ms.*

Question 4 : *Ecrire les quelques lignes de code qui permettent de créer les 2 threads* `light_collect` *et* `local_light` *et qui lancent ces 2 threads.*