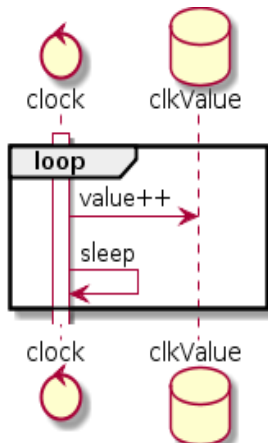


TP : getting started with ADA language

Project : the Black Box

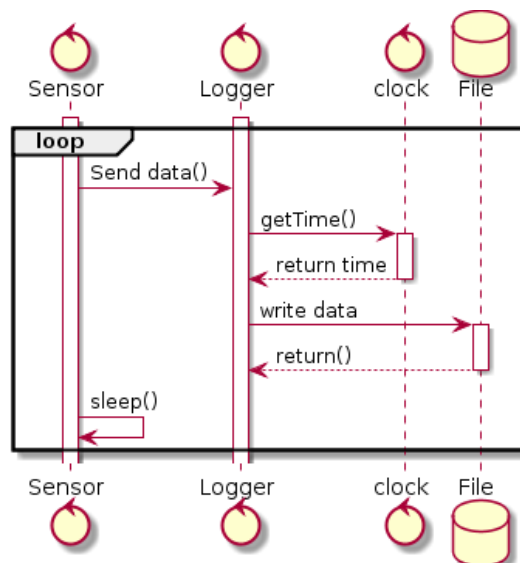
Ce projet consiste à mettre en œuvre un data logger. Le principe est de simuler 3 capteurs qui envoient une mesure (nombre entier entre 1 et 100) à une tâche dédiée qui enregistre la mesure dans un fichier. Cette mesure doit être estampillée avec la valeur d'une horloge qui doit être créée également.

1. Description UML



La tâche horloge correspond à un processus indépendant. Qui incrémente la valeur de l'horloge régulièrement. Cette valeur est stockée dans une variable globale au programme. Il n'est pas nécessaire de contrôler l'accès à la ressource car cette tâche est la seule à venir modifier la valeur.

Le déroulement du programme et l'interaction entre les processus sont représentés ci-dessous. On peut voir qu'une tâche Sensor génère des données puis fait appel au Logger. (Il y en a 3 en réalité, un seul est représenté pour simplifier le schéma). Le Logger a un rôle important puisqu'il sert d'intermédiaire entre les Capteurs et le fichiers où sont inscrites les mesures. Pour chaque données reçues, le Logger estampille la données avec la valeur de l'horloge décrite précédemment. Ces données estampillées sont ensuite inscrites dans le fichier.



2. Tâche d'horloge

La tâche mis en place permet de simuler une horloge. Il s'agit d'une tâche périodique qui incrémente une variable.

La période est définie lors de l'initialisation de la tâche, par le programme principal. Cette initialisation est réalisée à l'aide d'une instruction « **accept** » qui est bloquante. Ainsi la tâche d'horloge attend de recevoir sa période par le processus principal avant de commencer.

Quand la période est envoyée, la tâche commence son déroulement. Le principe est le suivant : Une variable « **horloge** » globale est définie. A chaque tour de boucle du processus horloge, cette variable est incrémentée. La tâche se met ensuite au repos pendant la durée définie par la période.

3. Tâche Capteur

Cette tâche permet de simuler un capteur. Elle se caractérise par un nom qui permettra de la différencier dans le fichier de log. Chaque tâche possède également une période, et un générateur de nombres aléatoires. La tâche capteur est défini comme étant un type de tâche. De cette façon, il est possible d'instancier plusieurs capteurs.

Lors du lancement du programme, le processus principal crée 3 capteurs pour lesquels il définit un nom et une période. Cela permet d'initialiser les tâches. Après initialisation, le travail commence seulement.

A chaque tour de boucle, la tâche génère un nombre aléatoire qui représente la mesure qu'elle va envoyer. Elle récupère ensuite la valeur de la variable qui sert d'horloge et envoi ces informations à la tâche « Logger ». La tâche sert met alors en sommeil pendant la période qui a été définie lors de la création du capteur. A la fin de la période de sommeil, la tâche effectue les mêmes actions.

4. Tâche Logger

Cette tâche a été créée pour pouvoir centraliser l'accès au fichier de log. De cette manière il n'y a qu'un seul processus qui écrit dans le fichier et les données ne risquent pas d'être corrompues.

Le fonctionnement est le suivant :

Lorsque la tâche est lancée pour la première fois, elle crée et ouvre un fichier nommé « **black_box.log** ». Quand le fichier est ouvert, cette tâche se met en attente de recevoir des appels en provenance des autres tâches capteurs. (utilisation d'une instruction « **accept** »). Quand les autres tâches font appel au Logger, ce dernier récupère la valeur de la mesure, le nom du capteur et la valeur de l'horloge et inscrit toutes ces informations dans le fichier ouvert.

Lorsque chaque tâche fait appel à la fonction write du Logger, la demande est mise en attente d'être traitée. Toutes les demandes en attentes seront traitées dans l'ordre où elle sont arrivées, il s'agit d'une file de type First In First Out (FIFO).

EXERCICES

4. input/output
fichier « input_output.adb ». Le programme d'origine a été modifié pour prendre en charge une chaîne de caractère de taille variable. On utilise ici, le type `Unbounded_String`.
5. Procedure
fichier « proced1.adb ». On crée 3 procédures qui sont appelées successivement pour afficher dans la console différents textes.
6. Function
Fichier « funct.adb ». Le programme affiche le carré du nombre 12 et demande à l'utilisateur de saisir un nombre. Le carré de ce nombre est ensuite affiché dans la console.
7. Task
Fichier « task1.adb ». Lors des premières exécutions du programme (CF ANNEXE 1), on peut remarquer que les tâches correspondent aux threads. Le processeur tente d'exécuter toutes les tâches en simultané. Le processeur donne donc la main pendant une courte période pour chaque tâche. C'est pourquoi on peut voir que l'affichage de toutes les tâches se mélange.
Après l'ajout d'un délai au début de chaque tâche on diffère le lancement des tâches. Cela permet à chaque tâche de se terminer avant qu'une autre soit lancée. Donc lors de l'exécution, on peut voir que chaque tâche affiche son exécution en un seul bloc.(CF ANNEXE2)
8. rendez-vous
Fichier « hotdog.adb »
Dans ce programme, nous mettons en œuvre une tâche avec un point de synchronisation. La tâche Gourmet, attend de recevoir un appel de la part d'une autre tâche. Quand un appel est passé, le code est exécuté. Lorsque la boucle est terminée, la tâche termine son exécution.
9. protect area of memory
Fichier « protect.adb ».
Dans ce programme, des variables sont modifiées par plusieurs tâches. Et pour éviter que les variables soient corrompues (plusieurs écriture/lecture en simultané), une procédure protégée est mise en place. Cela signifie que lorsque cette procédure est lancée, il faut attendre la fin de son exécution pour pouvoir lancer une autre exécution (par une autre tâche par exemple).

ANNEXE 1

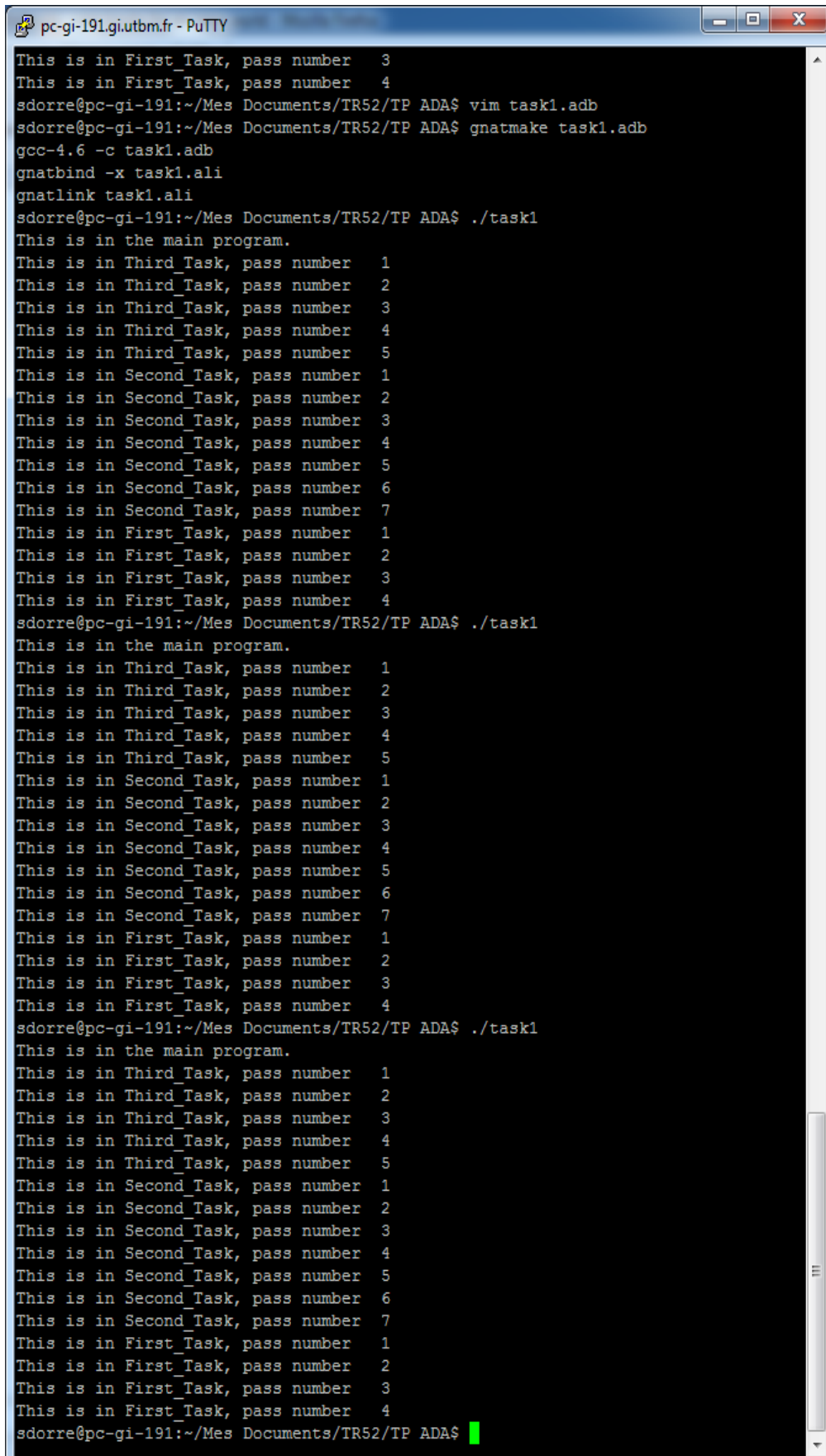
```
pc-gi-191.gi.utbm.fr - PuTTY
This is in Second_Task, pass number 7
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ ./task1
This is in Third_Task, pass number This is in the main program.
This is in First_Task, pass number 1This is in Second_Task, pass number 1
This is in Third_Task, pass number 2
This is in Third_Task, pass number 3
This is in Third_Task, pass number 4
This is in Third_Task, pass number 5
1
This is in First_Task, pass number 2
This is in First_Task, pass number 3
This is in First_Task, pass number 4

This is in Second_Task, pass number 2
This is in Second_Task, pass number 3
This is in Second_Task, pass number 4
This is in Second_Task, pass number 5
This is in Second_Task, pass number 6
This is in Second_Task, pass number 7
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ ./task1
This is in Third_Task, pass number This is in the main program.
This is in First_Task, pass number 1This is in Second_Task, pass number 1
This is in Third_Task, pass number 2
This is in Third_Task, pass number 3
This is in Third_Task, pass number 4
This is in Third_Task, pass number 5
1
This is in First_Task, pass number 2
This is in First_Task, pass number 3
This is in First_Task, pass number 4

This is in Second_Task, pass number 2
This is in Second_Task, pass number 3
This is in Second_Task, pass number 4
This is in Second_Task, pass number 5
This is in Second_Task, pass number 6
This is in Second_Task, pass number 7
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ ./task1
This is in First_Task, pass number This is in Third_Task, pass number 1This is
in Second_Task, pass number 1
This is in Third_Task, pass number 2
This is in Third_Task, pass number 3
This is in Third_Task, pass number 4
This is in Third_Task, pass number 5
This is in the main program.

This is in Second_Task, pass number 2
This is in Second_Task, pass number 3
This is in Second_Task, pass number 4
This is in Second_Task, pass number 5
This is in Second_Task, pass number 6
This is in Second_Task, pass number 7
1
This is in First_Task, pass number 2
This is in First_Task, pass number 3
This is in First_Task, pass number 4
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ █
```

ANNEXE 2



```
pc-gi-191.gi.utbm.fr - PuTTY
This is in First_Task, pass number 3
This is in First_Task, pass number 4
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ vim task1.adb
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ gnatmake task1.adb
gcc-4.6 -c task1.adb
gnatbind -x task1.ali
gnatlink task1.ali
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ ./task1
This is in the main program.
This is in Third_Task, pass number 1
This is in Third_Task, pass number 2
This is in Third_Task, pass number 3
This is in Third_Task, pass number 4
This is in Third_Task, pass number 5
This is in Second_Task, pass number 1
This is in Second_Task, pass number 2
This is in Second_Task, pass number 3
This is in Second_Task, pass number 4
This is in Second_Task, pass number 5
This is in Second_Task, pass number 6
This is in Second_Task, pass number 7
This is in First_Task, pass number 1
This is in First_Task, pass number 2
This is in First_Task, pass number 3
This is in First_Task, pass number 4
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ ./task1
This is in the main program.
This is in Third_Task, pass number 1
This is in Third_Task, pass number 2
This is in Third_Task, pass number 3
This is in Third_Task, pass number 4
This is in Third_Task, pass number 5
This is in Second_Task, pass number 1
This is in Second_Task, pass number 2
This is in Second_Task, pass number 3
This is in Second_Task, pass number 4
This is in Second_Task, pass number 5
This is in Second_Task, pass number 6
This is in Second_Task, pass number 7
This is in First_Task, pass number 1
This is in First_Task, pass number 2
This is in First_Task, pass number 3
This is in First_Task, pass number 4
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$ ./task1
This is in the main program.
This is in Third_Task, pass number 1
This is in Third_Task, pass number 2
This is in Third_Task, pass number 3
This is in Third_Task, pass number 4
This is in Third_Task, pass number 5
This is in Second_Task, pass number 1
This is in Second_Task, pass number 2
This is in Second_Task, pass number 3
This is in Second_Task, pass number 4
This is in Second_Task, pass number 5
This is in Second_Task, pass number 6
This is in Second_Task, pass number 7
This is in First_Task, pass number 1
This is in First_Task, pass number 2
This is in First_Task, pass number 3
This is in First_Task, pass number 4
sdorre@pc-gi-191:~/Mes Documents/TR52/TP ADA$
```