46 pt
5.6

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Nom : Sendoura Gonçalves **Prénom :** Hugo

Classe : T-2/I-2
Date : 08.06.2015

Problème n° 1 (programmation orienté-objet)

1. Décrivez succinctement le principe d'orienté-objet en langage C.

En C les objets ne sont pas supporté. Pour créer un objet il faut faire une structure dans laquelle on va mettre ses attributs et les méthodes de l'objets. Les méthodes devront avoir en paramètre la référence de la structure (objet). ✓

2

2. Décrivez succinctement l'utilité de la macro `container_of` et donnez son implémentation.

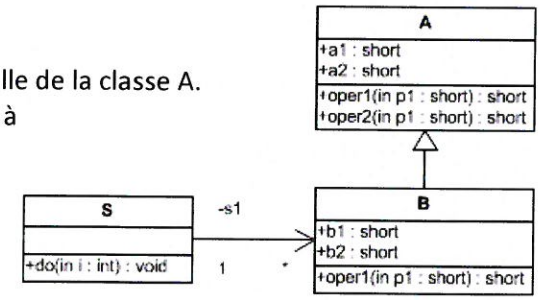
Container_of permet d'avoir la référence sur une classe dérivé afin d'avoir accès à ses attributs et méthodes. ✓

```
#define container_of (ptr, type, member) \
((char*) (((type *) (ptr)) - offset_of (type, member)))
```

✓

2

3. Pour le diagramme de classes ci-contre :
   a. Déclarez les classes A, B et S en langage C orienté-objet.
      Remarque : la méthode «`oper1`» de la classe B surcharge celle de la classe A.
   b. Implémentez la fonction «`oper1`» de la classe B de manière à ce qu'elle retourne le produit de « `p1 * a2 * b2`»




```
a) struct A {
    short a1;
    short a2;
1   short (*oper1) (struct A* oref, short p1);
    short (*oper2) (struct A* oref, short p1);
   }
   Struct B{
1  short b1;
   short b2;
   struct A m_base;
   }
1/2 Struct S{
   struct B*group;
   void (*do) (struct S * oref, int : );
   }
```

```
b) short operation1 (struct A* oref, short p1){
3   struct B* B_ref = container_of(oref, struct B, m_base);
    return (p1 * oref->a2 * B_ref->b2);
   }
```

✓

Hugo Sendoura 13h06

10

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 2 (Toolchain)

1. Expliquez à l'aide d'un exemple le principe de fonctionnement d'un Makefile.

un Makefile permet de créer un fichier executable automatiquement.
Il prend plusieurs fichiers pour en faire un seul executable.

1 1/2 En entré → file1.c , file2.c
en Sorti un fichier executable → my-app

exemple pas expressif

2. Concevez un Makefile pour la génération d'une application composée de 3 fichiers (`my_app.c`, `file1.c`, `file2.c`). Le programme exécutable sera appelé «`my_app`». Le compilateur «`gcc`» sera utilisé pour la génération de l'application avec les flags de compilation «`-Wall -Wextra -O1 -std=c11`». Le Makefile devra également permettre d'effacer tous les fichiers générés.
Il est impératif d'utiliser des variables pour spécifier les flags de compilation et les fichiers sources.

`Makefile:`

```
EXEC = my-app
CC = gcc
CFLAGS = -Wall -Wextra -O1 -std=c11
SRCS = file1.c file2.c my-app.c
OBJS = $(SRCS:.c=.o)

all: $(EXEC)

.c .o
	$(CC) $(CFLAGS) -o $* .o$<

$(EXEC): $(OBJS)
	$(CC) $(LDFLAGS) -o $@ $^

clean: $(EXEC)
	rm -f $(EXEC)
	rm -f *.o

.PHONY all clean
```

2 2 2 2

3. Décrivez succinctement la méthode à mettre en œuvre pour débugger une application chargée sur une cible à partir d'une machine hôte. Citez une ou deux interfaces permettant de connecter la machine hôte à la cible pour de telles opérations.

2 pour faire du "remote debuging" il faut utiliser le gdb et le gdb serveur. On peut le faire avec jtag.

(10)

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 3 (Vérification)

1. Décrivez l'objectif des revues de construction, indiquez où il se situe (quelle phase) dans le processus de développement logiciel et quels types de documents sont examinés

Les revue de construction se font à la fin de la phase de design et au début de la phase de codage. On fait une revu des .h et de la documentation faite en phase de design pour éviter de perdre du temps avec des erreurs en phase de codage.

3

2. Décrivez succinctement le principe des tests unitaires et citez une méthode permettant d'en garantir l'efficacité.

Le test unitaire est code fait par le développeur pour tester un code quelle parte
Il sert a testé tout les cas possible. (Chaque ligne de code)
On utilise le gcov pour vérifier que toutes les lignes de code ont été vérifier.

2'/2

3. Implémentez un test unitaire permettant de valider/vérifier le bon fonctionnement de la fonction ci-dessous (2 tests positifs et 2 tests négatifs).

```
/**
 * This function returns the base 10 logarithm of x.
 *   - if x is NAN: NAN is returned
 *   - if x is 1: 0 is returned
 *   - if x is negative: NAN is returned
 *   - if x is 0: -HUGE_VAL is returned
 */
double log10 (double x);
```

```
Void test_log10(){
    CU_ASSERT ( log10(NAN) == NAN);
    CU_ASSERT ( log10(1) == 0);
    CU_ASSERT (log10(-5) == NAN);
    CU_ASSERT (log10(0) == -HUGE_VAL);
    CU_ASSERT(log10(100) == 10);
}
```

2

3'/2

9

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 4 (Documentation & DMA)

1. On constate que les codes sources ont souvent un en-tête sous la forme d'un commentaire. Expliquez à quoi sert cet en-tête et indiquez les informations données par en tel en-tête. Rédigez un en-tête pour le fichier «fibonacci.c» qui calcule et affiche la suite de Fibonacci.

Cette en-tête sert à décrire brièvement à quoi sert le code, qui l'a codé, quand, pour une méthode on indiquera les paramètre et la valeur de retour ainsi qu'une brève description.

```
/**
* @file fibonacci.c
* @brief permet calculer la suite de fibonacci
* @date 07.06.2015
* @author Hugo Serdoura
*/
```

3

2. Décrivez succinctement l'utilité d'outils tel que Git ou Subversion

Permet de faire des versions du code ✓
Permet d'avoir accès à tout le code source ok, mais rien de neuf!
Permet de faire des "branch" ok (≠ version)
Permet de commenter chaque ajout ("push") ok

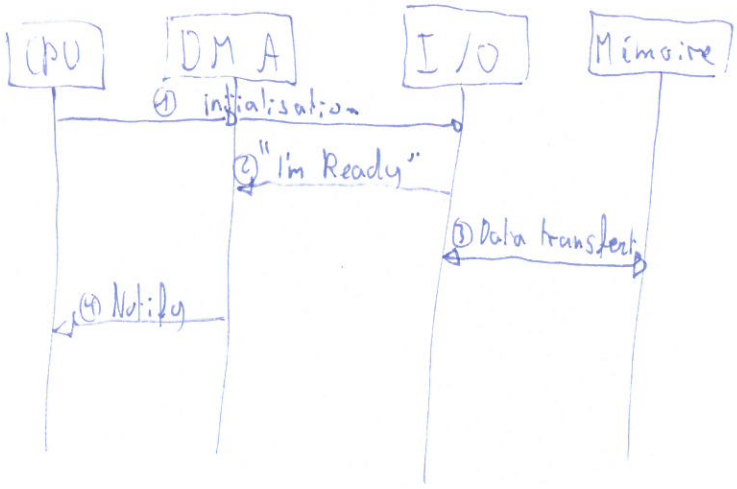
1 1/2

3. Donnez la définition de l'abréviation DMA et décrivez succinctement sa fonction dans un système à µP

Direct Memory Acces. Permet de décharger le µP quand il y a de gros ou fréquent transfert de données entre la mémoire et les périphériques d'entrées/sorties

1

4. Expliquez à l'aide d'une figure les phases principales d'un transfert DMA entre un périphérique d'entrée/sortie et la mémoire principale



3

9

Systèmes Embarqués 1 & 2: Travail écrit no 4.

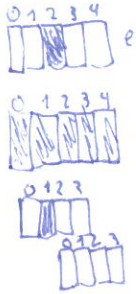
Problème n° 5 (Mémoire cache et MMU)

1. L'utilisation de la mémoire cache s'est popularisée sur les µP modernes. Décrivez succinctement son utilité et indiquez les 3 types d'architecture des mémoires caches.

la mémoire cache est une mémoire rapide. les données souvent utilisées sont placées dans le cache cela permet d'accélérer les accès mémoire.

3

- Direct ou
- complètement associatif
- n voie



2. Citez les deux principes qui sont à l'origine des mémoires caches et donnez un exemple.

proximité spatial : si une bout de la mémoire est utilisé il est fort probable qu'un bout a proximité soit utilisé sous peu

2

proximité temporel : si un bout de la mémoire est utilisé : il est fort probable qu'il soit réutilisé dans peu de temps

3. Donnez la définition de l'abréviation MMU et décrivez succinctement sa fonction dans un système à µP

Memory Management Unit. Le MMU se met entre le CPU et la mémoire et permet de convertir les adresses virtuel en adresse physique grâce à une table de translation.

2

4. Décrivez succinctement la fonction de la TLB (Translation Lookaside Buffer)

Le TLB est une table de conversion qui permet de convertir les addresses virtuelle en adresse physique. Cette table est placé dans la mémoire cache afin d'accélérer cette translation.

1

5. On parle de cache physique et virtuelle. Expliquez succinctement la différence entre ces 2 types.

0

La mémoire physique est la mémoire "réelle" et la mémoire virtuelle est une mémoire qui permet d'accélérer les accès mémoire. Il faut tout de même écrire la mémoire virtuelle dans la physique a un moment donné.

(8)