



Systèmes Embarqués 1 & 2: Travail écrit no 4.

Classe : T-2/I-2

Nom :

Prénom :

Date : 08.06.2015

Problème n° 1 (programmation orienté-objet)

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

La programmation OO pas supportée en C, on crée des struct contenant attr. et pointeurs de fonctions, référence à struct dans param méthodes

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

Elle sert à trouver la classe parente de la classe qui la contient. Elle permet d'accéder à ses attributs et méthodes.

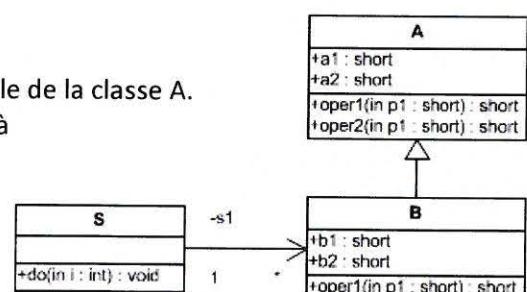
```
#define container_of(ptr, type, member) \
((char*) ((type *) (ptr)) - offset_of(type, member)))
```

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;
short oper1(struct A*, short p1);
short oper2 (struct A*, short, p1);
}
struct B {
struct A m_base;
short b1;
short b2;
}



Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 2 (Toolchain)

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

Il contient les instructions de compilation pour le compilateur afin que celui-ci génère automatiquement le fichier binaire exécutable

exemple : file.c a le header file.h inclus au début
↳ MF agrège les deux fichiers pour la compilation (décrit les dépendances)

Il décrit aussi le comportement à suivre pour gérer les fichiers (effacer les fichiers intermédiaires,

- 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

OBJS = \$(SRCS:.c=.o)

all : \$(EXEC)

.C .O

\$(CC) \$(CFLAGS) -o \$* .

[...]

- 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.

La cible doit posséder un serveur GDB, et le host utilise GDB pour s'y connecter. J-Tag est l'une des interfaces actuellement utilisées.



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

'Elles permettent de passer en revue la structure du projet, de prévoir les interactions entre ses composants et d'éviter qu'un problème dans ces domaines ne soit trop complexe à trouver et régler par après. Elles ont lieu à la fin de la phase de design.'

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

'Chaque composant est testé dans une mesure aussi complète que possible, indépendamment des autres. Cela permet d'isoler les problèmes dans le composant lui-même des problèmes d'interaction.'

'Ces tests sont des programmes qui tentent un maximum de fonctionnalités du composant.'

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 log10Test()  
{  
    CU_ASSERT(log10(NAN)==NAN);  
    CU_ASSERT(log10(1)==1);  
    CU_ASSERT(log10(-2)==NAN);  
    CU_ASSERT(log10(0)==-HUGE_VAL);  
}
```



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.

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

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

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

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.
 2. Citez les deux principes qui sont à l'origine des mémoires caches et donnez un exemple.
 3. Donnez la définition de l'abréviation MMU et décrivez succinctement sa fonction dans un système à µP.
 4. Décrivez succinctement la fonction de la TLB (Translation Lookaside Buffer)
 5. On parle de cache physique et virtuelle. Expliquez succinctement la différence entre ces 2 types.