Hes·so Ecole d'ingénieurs et d'architectes de Fribourg

Hes·so Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2: Travail écrit no 4. 5,5/6

Nom : Ghidoni **Prénom :** Matteo **Classe : T/2** **Date : 12.06.2014**

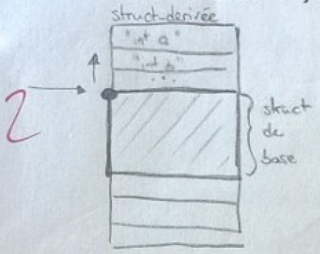
10

Problème n° 1 (programmation orienté-objet)

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

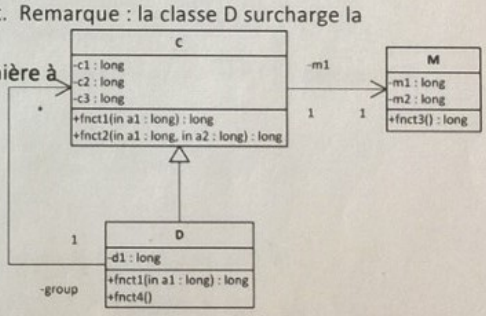
En C n'existe pas la définition de classe. On peut par contre regrouper tous les elements d'une classe dans une meme structure (attributs et methodes). Chaque methode doit avoir comme paramètre un pointeur de sa propre structure pour pouvoir utilisée ses elements.

2. Décrivez succinctement le mécanisme permettant à une méthode d'une classe dérivée d'obtenir la référence sur son objet (l'objet dérivé) à partir de la référence sur l'objet de la classe de base.



pour obtenir la reference de la classe derivée, à partire de la reference de la classe de base il faudra soustraire l'espace memoire occupé par les element de la classe derivée. déclaré avant

3. Pour le diagramme de classes ci-contre :
   a. Déclarez les classes C, D et M en langage C orienté-objet. Remarque : la classe D surcharge la fonction «`fnct1`» de la classe C.
   b. Implémentez la fonction «`fnct1`» de la classe D de manière à ce qu'elle retourne le produit de « `a1 * d1 * c1`»



a)
```
struct C {
    long c1;
    long c2;
    long c3;
    long (*fnct1)(struct C *oref, long a1);
    long (*fnct2)(struct C *oref, long a1, long a2);
    struct M * m_ref;
}
struct D {
    long d1;
    void (*fnct4)(struct D *oref);
    struct C m_base;
    struct C * group;
}
struct M {
    long m1;
    long m2;
    long (*fnct3)(struct M *oref);
}
```

b)
```
long fnct1(struct C *oref, long a1) {
    struct D * d_ref = container_of(oref, struct D
        m_base);
    return a1 * d_ref->d1 * oref->c1;
}
```

Interruptions : Logicielles

Hes·so Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2: Travail écrit no 4.

8

Problème n° 2 (Toolchain)

1. Expliquez succinctement le principe de fonctionnement d'un Makefile

(cible)

Un makefile permet de generer un fichier executable en partant par les fichiers "d'entrées" qu'il le compose. Dans un makefile peut-il avoir plusieurs cibles à "générer". + - - - compile plusieurs fichiers et il les reuni dans le fichier executable finale

1,1

2. Décrivez la fonction du mot clef « `.PHONY:` »

Indique les cible factices et force ainsi leur reconstruction

1

3. Définissez une règle permettant de générer les fichiers objets « `.o` » à partir de leur source « `.c` »

Admettons que y a une variable SRCS contenant tous les fichiers de source (.c)

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

%.o : %.c

0,7

4. Décrivez la fonction de la règle « `-include $(OBJS :.o=.d)` »

~~création des dependances~~

inclu dans le makefile les dependances crée.

? permet de continuer l'execution du make aussi si un des fichier n'existe pas

0,1

5. Indiquez la signification de la commande « `$(MAKE) -C <dirname> <targetname>` » placée dans un Makefile:

Cette commande est utilisé pour l'appelle des "sous-makefile" ✓

+ - - - - - -

0,1

6. Indiquez la signification de la commande shell « `$> make HOST=apf27 clean all` »:

On declare une variable HOST qui prends la valeur "apf27" et on lance notre makefile à les cibles "clean all"

1

7. Indiquez en une phrase la méthode pour débugger une application fonctionnant sur une cible à partir d'une machine hôte.

On peut faire du "remote debugging" avec l'outil gdb ✓ (gdbserver)

1

8. Indiquez la fonction des utilitaires ci-dessous:

a. gcc : compilateur ✓
b. as : assembleur ✓
c. ld : linkeur ✓
d. ar : création des bibliothèques ✓
e. strip : elimine les symbole d'un code objet

2

Interruptions : Logicielles

Hes·so Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 3 (Vérification)

1. Citez 2 mécanismes permettant de valider la bonne réalisation et le bon fonctionnement d'applications logicielles durant les différentes phases de leur développement

2

- test unitaires
- reviews

2. Citez une méthode permettant d'identifier le goulet d'étranglement dans une application logicielle. Citez l'utilitaire de la chaîne d'outils GNU permettant de mettre en œuvre cette méthode.

1

Il faut faire un test de performance du code

L'utilitaire est le gprof

3. Citez une méthode permettant de garantir qu'un composant logiciel a été correctement testé. Citez l'utilitaire de la chaîne d'outils GNU permettant de mettre en œuvre cette méthode.

1

Il faut faire un control de qualité des tests avec un test de couverture du code

L'utilitaire est le cgov

4. Décrivez succinctement le principe des tests unitaires.

2

Ils permettent de valider le bon fonctionnement des différents composantes d'un logicie avec un programme codé par le développeur

5. Implémentez un test unitaire permettant de valider/vérifier le bon fonctionnement de la fonction « `double sqrt (double x)` » ci-dessous (2 tests positifs et 2 tests négatifs).

```
/*
 * The sqrt() function computes the square root of x.
 * The following error conditions are defined for this function:
 *   - if x is NAN: NAN is returned and errno is set to EDOM
 *   - if x is negative: NAN is returned and errno is set to EDOM
 */
double sqrt (double x);
```

```
void t_sqrt() {
1  CU_ASSERT( sqrt(25.0) == 5 ); // positif
0,5 CU_ASSERT( sqrt(NAN) == 1); // positif
0,5 CU_ASSERT( sqrt(-4) != NAN); // negatif
1  CU_ASSERT( sqrt(NAN) != NAN); // negatif
}
```

Interruptions : Logicielles

Hes·so Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 4 (Documentation & DMA)

10

1. Citez 4 éléments importants devant faire partie de la documentation d'un logiciel pour le développeur

2
- specifications (analyse, architecture, ...)
- guides
- code (API)
- aide (exemple d'utilisation)

2. Citez et décrivez succinctement un logiciel permettant de documenter du code source en langage C

1 Doxygen. Au travers d'un fichier de configuration c'est possible définir comment le code doit être documenté.

3. Décrivez succinctement l'utilité d'un SCM (Source Code Management Tool) tel que GIT ou SVN

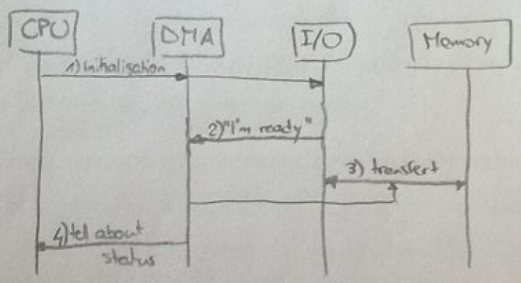
2 Permet d'avoir access à tout le code present sur le depot (pull) et permet la création des branches. Permet aussi de laisser des commentaire à chaque fois que on "push" du code (ou on modifie). → Gestion des versions

4. Décrivez succinctement la fonction d'un DMA

2 Le controleur DMA permet de diminuer la charge de travail du processeur pour ce qui concere les transferts entre memoire et peripheriques I/O

5. Citez les 4 phases principales d'un transfert DMA entre une mémoire et un périphérique

2



6. Citez 2 domaines d'application des DMA

1
- serial (ex. UART)
- Ethernet

Interruptions : Logicielles

Hes·so Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2: Travail écrit no 4.

8

Problème n° 5 (Mémoire cache et MMU)

1. Décrivez succinctement la fonction de la mémoire cache

1 La mémoire cache permet de réduir le temps d'access à la mémoire principale. Les données qui sont utilisé plusieurs fois sont donc placée dans la mémoire cache.

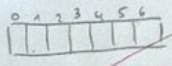
2. Citez et décrivez succinctement les deux principes qui sont à l'origine des mémoires caches

1 - localité temporelle : ✓

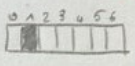
1 - localité spatiale ✓

3. Décrivez succinctement le principe des mémoires caches complétement associatives

0 ligne 22 à inserer

mémoire cache : 

22 % 7 = 1

⇒ 

⇒ avec une mémoire cache completement associative une ligne peut être placée à n'importe quel endroit

4. Citez 2 algorithmes utilisés pour le remplacement des lignes dans la mémoire cache

1 - random

- FIFO

5. Décrivez succinctement le fonctionnement de la MMU et citez ses 3 fonctions principales

La MMU est placé entre la CPU et la mémoire principale est s'occupe, au travers une translation d'adresse, de la communication CPU ↔ mémoire

0 3 fonctions :
- convertir les adresses virtuelles en adresses physiques ✓
- limité l'acces direct pour le processeur vers la mémoire principale ✓
- gestion de la mémoire cache ✓

6. Décrivez succinctement la traduction d'une adresse virtuelle en adresse physique

1 Pour convertir une adresse virtuelles en adresse physiques la MMU utilise une table de conversion situe dans la mémoire principale.
Les bits plus hauts de l'adresse virtuelle sont utilisé pour avoir acces à cette table qui contient les adresses physiques

7. Citez le mécanisme implémenté par les MMU pour accélérer la translation d'adresses virtuelles en adresses physiques

1 Les MMUs utilisent des mémoires cache où ils stoquent la table de translation