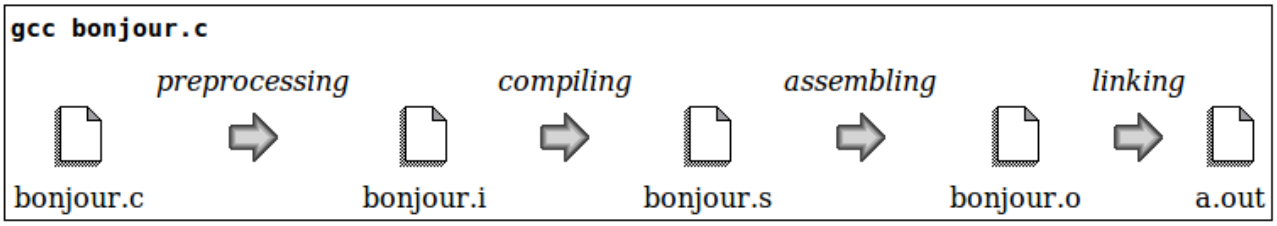
Les étapes de la compilation



gcc effectue quatre étapes pour produire l'exécutable :

1. **passage au pré-processeur (*preprocessing*)** (**gcc -E bonjour.c > bonjour.i**) :

Le pré-processeur réalise plusieurs opérations de **substitution** sur le code C, notamment:

- suppression des commentaires (// et /* */) qui sont utiles au programmeur, mais inutiles pour l'ordinateur.
- inclusion des fichiers .h dans le fichier .c (#include)
- traitement des directives de compilation qui commencent par un caractère #. Ces directives sont define elif else endif error if ifdef ifndef include line pragma undef, vous les aborderez dans la suite du cours.
- Le pré-processeur contrôle la syntaxe du programme.

Les erreurs de compilation sont bloquantes, mais pas les avertissements (warning).Pour afficher les avertissements , il faut utiliser l'option -Wall (W pour Warning).

2. **compilation en langage assembleur (gcc -S bonjour.c > bonjour.s) :**

Le code du langage C est transformé en code Assembleur. C'est un code qui est lisible ecrit en assembleur.

3. **conversion du langage assembleur en code machine (*gcc -c bonjour.c > bonjour.o*) :**

Le code assembleur qui est lisible est transformé en code machine binaire. Il suffit de donner à gcc l'option -c suivie du nom du fichier qui contient le code assembleur pour obtenir un fichier *objet* bonjour.o. Ce fichier binaire ne peut pas être directement édité et lu par un humain. On peut le rendre lisible en utilisant une commande od -x bonjour.o. La commande od pour octal dump lit le fichier binaire et affiche sur la sortie standard les *octets* en hexadécimal avec l'option -x .

4. **édition des liens (*linking*)**

Le fichier bonjour.o produit par l'étape 3 est incomplet, il ne contient pas le code de la fonction printf. En effet, ce code est dans une bibliothèque. Afin que le programme puisse être compilé, nous avons inclus son prototype mais le code de la fonction n'est pas présent dans bonjour.o.

L'édition des liens va réunir le fichier objet et les fonctions contenues dans les bibliothèques, pour produire le programme complet : l'exécutable a.out dans notre exemple.

Une **unité de gestion mémoire** (**MMU** pour *Memory Management Unit*) est un composant informatique responsable de l'accès à la mémoire demandée par le processeur.

la **segmentation** est une technique gérée par l'unité de segmentation de la MMU, utilisée sur les systèmes d'exploitation modernes, qui divise la mémoire physique (dans le cas de la segmentation pure) ou la mémoire virtuelle (dans le cas de la segmentation avec pagination) en *segments* caractérisés par leur adresse de début et leur taille (*décalage*).

Le **Zend Framework** est un framework pour PHP 5 créé en mars 2006 par Zend Technologies .Il a été développé dans le but de simplifier le développement Web tout en recommandant les bonnes pratiques et la conception orientée objets en offrant des outils aux développeurs. ZF permet aussi d'utiliser nativement le principe de MVC (Modèle-Vue-Contrôleur) mais ne l'oblige pas.

Les 7 couches de modèle OSI :

1 - La couche physique

La couche physique s'occupe de la transmission des bits de façon brute sur un canal de communication. Cette couche doit garantir la parfaite transmission des données

Cette couche doit normaliser les caractéristiques électriques (un bit 1 doit être représenté par une tension de 5 V), les caractéristiques mécaniques (forme des connecteurs, de la topologie...), les caractéristiques fonctionnelles des circuits de données et les procédures d'établissement, de maintien et de libération du circuit de données.

L'unité d'information de cette couche est le bit, représenté par une certaine différence de potentiel.

2 - La couche liaison de données

Son rôle est un rôle de "liant" : elle va transformer la couche physique en une liaison a priori exempte d'erreurs de transmission pour la couche réseau. Elle fractionne les données d'entrée de l'émetteur en trames, transmet ces trames en séquence et gère les trames d'acquittement renvoyées par le récepteur.
La couche liaison de données doit donc être capable de reconnaître les frontières des trames.
La couche liaison de données doit être capable de renvoyer une trame lorsqu'il y a eu un problème sur la ligne de transmission.

De manière générale, un rôle important de cette couche est la détection et la correction d'erreurs intervenues sur la couche physique.

Cette couche intègre également une fonction de contrôle de flux pour éviter l'engorgement du récepteur.

L'unité d'information de la couche liaison de données est la trame qui est composées de quelques centaines à quelques milliers d'octets maximum.

3 - La couche réseau

C'est la couche qui permet de gérer le sous-réseau, i.e. le routage des paquets sur ce sous-réseau et l'interconnexion des différents sous-réseaux entre eux. Au moment de sa conception, il faut bien déterminer le mécanisme de routage et de calcul des tables de routage (tables statiques ou dynamiques...).

La couche réseau contrôle également l'engorgement du sous-réseau. On peut également y intégrer des fonctions de comptabilité pour la facturation au volume, mais cela peut être délicat.

L'unité d'information de la couche réseau est le paquet.

4 - Couche transport

Cette couche est responsable du bon acheminement des messages complets au destinataire. Le rôle principal de la couche transport est de prendre les messages de la couche session, de les découper s'il le faut en unités plus petites et de les passer à la couche réseau, tout en s'assurant que les morceaux arrivent correctement de l'autre côté. Cette couche effectue donc aussi le réassemblage du message à la réception des morceaux.

Cette couche est également responsable de l'optimisation des ressources du réseau : en toute rigueur, la couche transport crée une connexion réseau par connexion de transport requise par la couche session, mais cette couche est capable de créer plusieurs connexions réseau par processus de la couche session pour répartir les données, par exemple pour améliorer le débit. A l'inverse, cette couche est capable d'utiliser une seule connexion réseau pour transporter plusieurs messages à la fois grâce au multiplexage. Dans tous les cas, tout ceci doit être transparent pour la couche session.

Cette couche est également responsable du type de service à fournir à la couche session, et finalement aux utilisateurs du réseau : service en mode connecté ou non, avec ou sans garantie d'ordre de délivrance, diffusion du message à plusieurs destinataires à la fois... Cette couche est donc également responsable de l'établissement et du relâchement des connexions sur le réseau.

Un des tous derniers rôles à évoquer est le contrôle de flux.

C'est l'une des couches les plus importantes, car c'est elle qui fournit le service de base à l'utilisateur, et c'est par ailleurs elle qui gère l'ensemble du processus de connexion, avec toutes les contraintes qui y sont liées.

L'unité d'information de la couche réseau est le message.

5 - La couche session

Cette couche organise et synchronise les échanges entre tâches distantes. Elle réalise le lien entre les adresses logiques et les adresses physiques des tâches réparties. Elle établit également une liaison entre deux programmes d'application devant coopérer et commande leur dialogue (qui doit parler, qui parle...). Dans ce dernier cas, ce service d'organisation s'appelle la gestion du jeton. La couche session permet aussi d'insérer des points de reprise dans le flot de données de manière à pouvoir reprendre le dialogue après une panne.

6 - La couche présentation

Cette couche s'intéresse à la syntaxe et à la sémantique des données transmises : c'est

elle qui traite l'information de manière à la rendre compatible entre tâches communicantes. Elle va assurer l'indépendance entre l'utilisateur et le transport de l'information.

Typiquement, cette couche peut convertir les données, les reformater, les crypter et les compresser.

7 - La couche application

Cette couche est le point de contact entre l'utilisateur et le réseau. C'est donc elle qui va apporter à l'utilisateur les services de base offerts par le réseau, comme par exemple le transfert de fichier, la messagerie...

MVC

Le modèle

Le modèle représente le comportement de l'application : traitements des données, interactions avec la base de données, etc. Il décrit ou contient les données manipulées par l'application. Il assure la gestion de ces données et garantit leur intégrité.

La vue :

La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle.

Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, etc). Ces différents événements sont envoyés au contrôleur. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.

Le contrôleur :

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer.

Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle, ce dernier avertit la vue que les données ont changé pour qu'elle se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. .

Par exemple, dans le cas d'une base de données gérant les emplois du temps des professeurs d'une école, une action de l'utilisateur peut être l'entrée (saisie) d'un nouveau cours. Le contrôleur ajoute ce cours au modèle et demande sa prise en compte par la vue. Une action de l'utilisateur peut aussi être de sélectionner une nouvelle personne pour visualiser tous ses cours. Ceci ne modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des cours de cette personne.

Une Vue (BD) :

Une vue est une table virtuelle, c'est-à-dire dont les données ne sont pas stockées dans une table de la base de données, et dans laquelle il est possible de rassembler des

informations provenant de plusieurs tables. On parle de "vue" car il s'agit simplement d'une représentation des données dans le but d'une exploitation visuelle. Les données présentes dans une vue sont définies grâce à une clause SELECT

POO :

La **programmation orientée objet** consiste à modéliser informatiquement un ensemble d'éléments d'une partie du monde réel (chien ,cahier,voiture ...que l'on appelle *domaine*) en un ensemble d'entités informatiques. Ces entités informatiques sont appelées **objets**. Il s'agit de données informatiques regroupant les principales caractéristiques des éléments du monde réel (taille, couleur, ...).

L'encapsulation

L'encapsulation est un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure en cachant l'implémentation de l'objet, c'est-à-dire en empêchant l'accès aux données par un autre moyen.
L'encapsulation permet de définir des niveaux de visibilité des éléments d'un classe.
L'encapsulation est le fait qu'un objet renferme ses propres attributs et ses méthodes.

Le masquage des informations

L'utilisateur d'une classe n'a pas forcément à savoir de quelle façon sont structurées les données dans l'objet, cela signifie qu'un utilisateur n'a pas à connaître l'implémentation. Ainsi, en interdisant l'utilisateur de modifier directement les attributs, et en l'obligeant à utiliser les fonctions définies pour les modifier (appelées **interfaces**).

L'héritage

L'**héritage** est un principe propre à la programmation orientée objet, permettant de créer une nouvelle classe à partir d'une classe existante.provient du fait que la classe dérivée (la classe nouvellement créée) contient les attributs et les méthodes de sa superclasse (la classe dont elle dérive). L'intérêt majeur de l'héritage est de pouvoir définir de nouveaux attributs et de nouvelles méthodes pour la classe dérivée, qui viennent s'ajouter à ceux et celles héritées.

Héritage multiple

Certains langages orientés objet, tels que le C++, permettent de faire de l'héritage multiple, ce qui signifie qu'ils offrent la possibilité de faire hériter une classe de deux superclasses.

Polymorphisme

C'est un mécanisme qui permet à une sous classe de redéfinir une méthode dont elle a hérité tout en gardant la même signature de la méthode héritée.

Un même message peut ainsi déclencher des traitements différents selon l'objet auquel il fait appel.

Un message polymorphe poserait un problème à la compilation statique car on ne saurait identifier précisément la méthode qu'il vise.

On ne pourra le savoir qu'au moment de l'exécution du programme. C'est la compilation dynamique qui permettra de résoudre ce problème.

MVC

Struts est un framework open source pour développer des applications Web J2EE utilisant l'architicture Modèle Vue Contrôleur (MVC).
Isoler la donnée elle-même de sa présentation .

Distinguer la consultation de la modification .

différence entre MVC 1 et MVC 2

Dans l'architecture MVC2 il n'ya qu'un seul contrôleur qui reçoit toutes les demande de l'application et est chargée de choisir l'action appropriées pour réponse à chaque demande.

Diffrence entre c++ et java.

- le garbage collector en Java, est inexistant en C++

l'heritage mutiple du C++ est "remplacé" par les interfaces en Java

Spring

-

JEE :

-

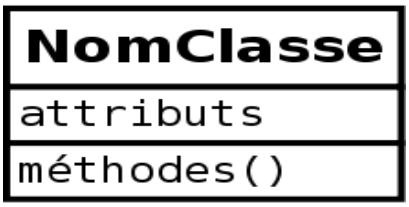
differencd entre swing awt

-

diagramme de classes

Le **diagramme de classes** est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci.

Il montre la structure interne du système.



Une classe est représentée par un rectangle séparée en trois parties :

- la première partie contient le nom de la classe
- la seconde contient les attributs de la classe
- la dernière contient les méthodes de la classe

in: Paramètre d'entrée passé par valeur. Les modifications du paramètre ne sont pas disponibles pour l'appelant. C'est le comportement par défaut.
out: Paramètre de sortie uniquement. Il n'y a pas de valeur d'entrée et la valeur finale est disponible pour l'appelant.

inout: Paramètre d'entrée/sortie. La valeur finale est disponible pour l'appelant.
Le type du paramètre (<Type>) peut être un nom de classe, un nom d'interface ou un type de donné prédéfini.

La notion de visibilité indique qui peut avoir accès à l'attribut.

Elle ne peut prendre que 3 valeurs possibles :

Caractère	Rôle	Mot clé	Description
+	accès public	public	Toutes les autres classes ont accès à cet attribut.
#	accès protégé	protected	Seules la classe elle-même et les classes filles (héritage) ont accès à cet attribut.
-	accès privé	private	Seule la classe elle-même a accès à cet attribut.

Visibilité nomFonction(*directionParamètreN nomParamètreN* : *typeParamètreN*) : *typeRetour*

- Association : relation logique entre deux classes

Multiplicité ou cardinalité

La multiplicité associée à une terminaison d'association, d'agrégation ou de composition déclare le nombre d'objets susceptibles d'occuper la position définie par la terminaison d'association

composition

La composition, également appelée agrégation composite, décrit une contenance structurelle entre instances. Ainsi, la destruction de l'objet composite implique la destruction de ses composants. Une instance de la partie appartient toujours à au plus une instance de l'élément composite. Graphiquement, on ajoute un losange plein (◆) du côté de l'agrégat

agregation

Lorsqu'une association entre deux instances d'une classe a en plus une particularité dont le sens est du style : "*une instance est composée d'une ou plusieurs autres instances* ", on peut alors qualifier cette association d'*agrégation*.

On peut dire également qu'une agrégation est une association de type " Composé– Composant ". Où, l'instance composé est l'agrégat et les composants sont les instances agrégées.
Une agrégation peut être perçue comme une association. Cependant une association ne peut être une agrégation.
Si une association a les caractéristiques suivantes, elle peut alors être représentée par une agrégation :
L'association a une sémantique de style " *est composée de* ... " ou " *est une partie de* ... ".
Il existe une forte différence de granularité entre une classe (l'agrégat) et d'autres classes (les agrégés).

La suppression d'un objet agrégat ferait disparaître les objets agrégés.
La modification d'un attribut d'un objet agrégat porte aussi sur aussi sur les attributs des objets agrégés.
La définition d'une méthode de l'objet agrégat repose sur celles des objets agrégés et peuvent porter d'ailleurs le même nom.

diagramme de cas d'utilisation :

Un cas d'utilisation décrit l'interaction des éléments qui se trouvent à l'extérieur d'un

système (acteurs) avec les système lui-même
Acteur : entité externe qui agit sur le système ; Le terme acteur ne désigne pas seulement les utilisateurs humains mais également les autres systèmes.
Cas d'utilisation : ensemble d'actions réalisées par le système en réponse à une action d'un acteur.

Méthodes et classes abstraites

Une méthode est dite abstraite lorsqu'on connaît son entête mais pas la manière dont elle peut être réalisée (*i.e.* on connaît sa déclaration mais pas sa définition).
Ou bien une classe declaré avec le mots abstract.
Une classe abstrait on ne peut pas l'instancier.
Une classe est dite abstraite lorsqu'elle définit au moins une méthode abstraite ou lorsqu'une classe parent contient une méthode abstraite non encore réalisée.
On ne peut instancier une classe abstraite : elle est obligé à se spécialiser . Une classe abstraite peut très bien contenir des méthodes concrètes.
Une classe abstraite pure ne comporte que des méthodes abstraites. En programmation orientée objet, une telle classe est appelée une interface.
abstraite pure = interface.
Classe abstrait trops générale

trigger

Les trigger sont similaire au procédure stocker. Ils peuvent contenir du code SQL, PL/SQL et même du code Java. Par contrairement au procédure, le trigger est déclencher sur des évenements précis.

Option BEFORE/AFTER :elle précise le moment de l'exécution du trigger .

index

Un index est un objet complémentaire (mais non indispensable) à la base de données permettant d'"indexer" certaines colonnes dans le but d'améliorer l'accès aux données par le SGBDR.

Toutefois la création d'index utilise de l'espace mémoire dans la base de données, et, étant donné qu'il est mis à jour à chaque modification de la table à laquelle il est rattaché, peut alourdir le temps de traitement du SGBDR lors de la saisie de données. Par conséquent il faut que la création d'index soit justifiée et que les colonnes sur lesquelles il porte soient judicieusement choisies (de telle façon à minimiser les doublons). De cette façon certains SGBDR créent automatiquement un index lorsqu'une clé primaire est définie.

CREATE INDEX Nom_de_l_index ON Nom_de_la_table (Nom_de_champ [ASC/DESC], ...)
Sans index :
- parcours séquentiel
- parcours par dichotomie si les valeurs sont triées

Avec index :
- Utilisation de l'index -> accès direct à l'enregistrement voulu
- les mises à jours coûtent cher, car il faut reconstruire l'index pour qu'il soit utilisable

Par défaut, oracle crée un index secondaire sur la clé primaire des tables