

Eval : 1h, réponses courtes et connaissances rapides sur les cours

## Cours Latex 1 :

Créé par Leslie Lamport. C'est un outil qui va permettre de construire son texte avec des macro commandes, qui commencent toujours par un backslash. Il sera équivalent à un texte de programme car composé de commandes. La première commande d'un document est \documentclass, moins de perte de temps dans la mise en page. {article} par exemple est le type de classe du document, {babel} est un support multilingue pour Latex. [utf8]{inputenc} est l'encodage d'entrée.

Dès qu'on fait documentclass on est dans l'entête et dès qu'on fait begin{document} on est dans le body. \section donne la décomposition logique du document, avec section et subsection et subsubsection. Nous avons aussi les chapters. Les formules mathématiques sont entourées par des \$ au début et à la fin. Avec \forall on peut afficher le symbole "pour tous", mais seulement en mode mathématique. On finit par end{document}.

Pour compiler on peut faire exemple1.tex. Ensuite le fichier produit était en .dvi etc. Mais maintenant nous utilisons pdf-latex. Les caractères spéciaux sont déspecialisés avec un antislash. Une commande peut avoir de 1 à 9 paramètre avec :

- Option obligatoire = entre {}
- Option = le premier (valeur par défaut)

\documentclass[a4paper]{article} par exemple avec a4paper en option.

Les environnements : \begin{itemize} et \item permettent de numérotter chaque élément entre ces deux blocs pour faire des listes à puces etc.

Les paquetages sont des extensions du système latex original. Elles permettent de définir de nouvelles commandes et environnements. Ces paquetages sont trouvables sur (CTAN). verbatim est par exemple un paquetage qui permet d'insérer du code c++ dans un document comme listings.

Les classes de document :

- book
- article
- beamer
- lettre

\ = retour en début de ligne.

\par = changement de paragraphe.

\ = espace insécable (M.\~Meynard donnera M. Meynard)

\parindent = indentation de paragraphe.

\parskip = espace vertical entre paragraphes.

\newpage = insère un saut de page.

\\$ = insérer une ou plusieurs ligne de math.

x\_i = x indice i

x^2 = x<sup>2</sup>

Eval : 1h, réponses courtes et connaissances rapides sur les cours

Les tableaux :

```
\begin{tabular}{||c|r|} | c et r représente les trois colonnes.  
\hline = génère une ligne horizontale  
\end{tabular}
```

Les flottants : tabular fait un tableau et table fait un tableau flottants on encapsule dans begin{table}. Il va flotter dans le texte afin qu'il soit placé correctement de manière propre. \caption ajoute la légende et {\label{adrsmontab} Mon tableau} donne un lien vers le numéro de page et vers le tableau.

\includegraphic{monimage} insère des images et nécessite le package graphicx. Généralement on omettra l'extension du fichier lors de l'inclusion. Entre crochets on peut mettre une hauteur ou une largeur donnée, on même une échelle avec scale.

Pour les images \begin{figure}[htbp] met soit au top soit bottom soit previous etc. on peut tout labéliser.

Minipage permet d'effectuer une mise en page en plusieurs colonnes. Le premier paramètre optionnel précise le positionnement tandis que le second indique la largeur.

Pour citer une réf bibliographique on va faire \cite{VER1875}  
\tableofcontents = table des matières  
\listoftables = liste des tables flottantes  
\listoffigures = liste des figures flottantes  
makeindex = production d'un index ou glossaire (un peu lourd)

---

Cours Latex Beamer :

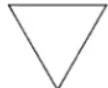
Beamer fait référence aux polycopiés et présentations. On commence avec \documentclass{beamer}. \pgfpagesuselayout{4 on 1} nous permet d'avoir 4 frames sur une même page. \usetheme permet de choisir un thème. \tableofcontent[hideallsubsections] permet de cacher dans la table toutes les sous sections. \begin{frame}{mon titre de trame} \note permet de faire des notes que nous devons imprimer, mais que les spectateurs ne verront pas. \begin{itemize}{<+-| alert@+>} permet de faire apparaître les choses les unes après les autres. Le titre de frame étant le titre de la page de diapo. \setbeamertemplate{caption}[numbered] permet de numérotter les tables et les figures. Beamer permet de produire des diaporama de présentation, un polycopié pour le locuteur, des transparents physiques sur des feuilles, des articles.

\transblindhorizontal arrive des côtés etc.. Tikz permet de faire du dessin vectoriel directement sur latex. On peut aussi utiliser des librairies pour Tikz afin de faire des graphes etc, ainsi que des arbres.

- par défaut, les dimensions sont 1cm vers la droite, 1cm vers le haut

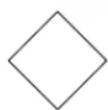
```
\tikz \draw (0,0) -- (1,1);
```

- coord. pol. par un angle et une longueur : (60:10pt)



```
\tikz \draw (0,0) [color=yellow] (60:1) -- (120:1) -- (0,0)
```

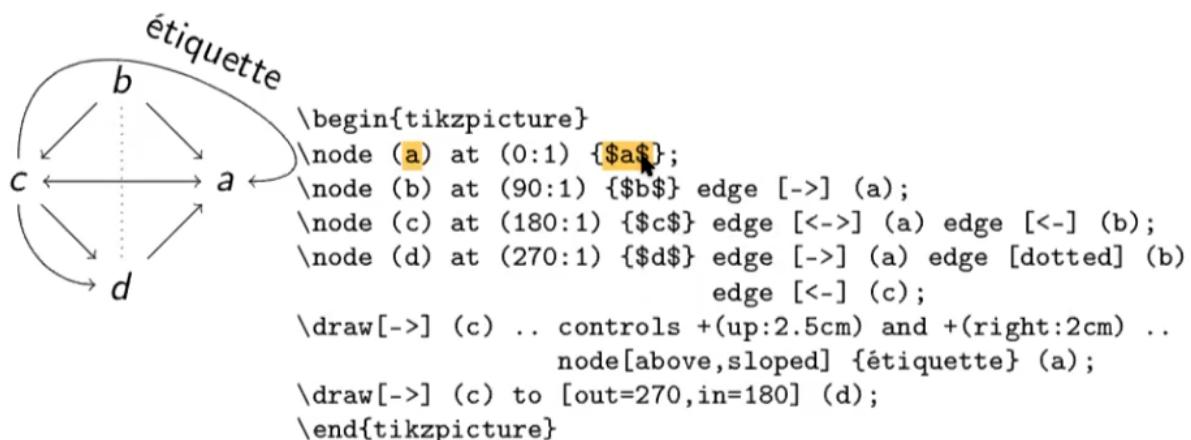
- coordonnées cart. relatives au dernier point utilisé en préfixant par ++



```
\tikz \draw (0,0) -- ++(.5,.5) -- ++(-.5,.5) -- ++(-.5,-.5) -- (0,0);
```

- coordonnées 3-dimensions (x,y,z) ; ancre relative à une position dans une forme ; ...

On fait une liste de points successif (path) fill on remplit shades pour un remplissage ombré, on finit par un cycle pour montrer qu'on revient au point de départ. Les node on peut faire des noeuds



## Cours Git :

Git créé pour le développement de linux. Récupérations possibles après des modifications.

Le dépôt local contient toutes les versions. Linux : sudo apt-get git-core gitk

On va faire les commandes suivantes : git (et ce que l'on veut faire : add ...).

**git status** : état du dépôt.

**git pull** : récupérer la dernière version du serveur.

**git add** : fichier ajoute un fichier ou prend en compte les modifications.

**git commit** : valider les modifications et commente plusieurs add avant un commit.

Eval : 1h, réponses courtes et connaissances rapides sur les cours

**git push** : 1 ou plusieurs commit avant un push.

Lors du premier envoi il faut utiliser “**git push origin master**” pour indiquer qu'il s'agit de la branche master que l'on souhaite synchroniser.

Il faut refaire un git add des tous les fichiers avant un commit. Dans le git commit utiliser a pour all et envoyer tous les fichiers en même temps.

Caractéristique :

Proposer sa contribution (pull request) au dépôt principal.

forker un projet ou le cloner permet de rapatrier localement tout l'historique du projet.

Communication avec un dépôt distant via le protocole git (port 9418).

Le répertoire racine d'un dépôt contient un répertoire .git qui mémorise le dépôt et de nombreuses informations :

- Le répertoire objects contient toutes les versions de fichiers.
- Le fichier config contient des propriétés dans des rubriques telles que remote.
- Les fichiers .gitignore contiennent eux aussi des expressions régulières correspondant au noms de fichier que doit ignorer git; il est sous contrôle de version :  
git add .gitignore

La configuration peut être globale à l'utilisateur : (~/.gitconfig) ou locale au dépôt (racine/.git/config).

A faire une fois pour tous les dépôts et les dépôts à venir :

**git config --global user.name “Michel Meynard”**

**git config --global user.email “ton mail”**

Les validations futures utiliseront cette identité sauf si une identité locale à un dépôt a été redéfini :

**git config --local user.name “bla”**

**git config --local user.email “mail”**

Chaque commit :

- est identifié par une chaîne de 40 car. hexadécimaux (somme de contrôle SHA-1);
- est effectué par un utilisateur ayant un nom et un email (config);
- a une date;
- a un message indiquant textuellement les modifications;
- a une liste des modifications effectuées

Afficher les dernières modifications

```
$ git log -p -2
commit ddf4c11106f9cfe973a4c080cb14c69a96cdc3df
Author: Marianne Huchard <huchard@lirmm.fr>
Date:   Wed Nov 19 09:09:06 2014 +0100

        ajout article
        pour chacun des fichiers modifiés
diff --git a/Etat-Art/DocumentEtatArt/etatArt.tex b/Etat-A
```

commande  
id  
user  
date  
message de commentaire  
avant      après  
pour chacun des fichiers modifiés

Eval : 1h, réponses courtes et connaissances rapides sur les cours

Git log contient plein d'option :

- p (patch) affiche les différences entre les fichiers
- s"int fact(" cherche une string
- graph pour voir les embranchements
- pretty=oneline permet d'afficher qu'une ligne par commit
- author=="michel M" que les commit d'un utilisateur en particulier
- since=2.weeks , until, before, after en fonction des dates

L'ajout de répertoire par git add rep ajoute tous les fichiers et répertoires inclus sauf s'ils correspondent aux expressions régulières contenues dans le fichier .gitignore de la racine du dépôt local. La suppression de fichiers ou répertoires ajoutés et non validés est réalisée par : git reset HEAD fich1 fich2 rep3 .. Attention, ces fichiers ne sont pas supprimés de l'arborescence. Git rm supprime les fichiers de l'arborescence.

Après un commit erroné git reset permet de revenir en arrière.

Pour collaborer à plusieurs, plusieurs dépôts distants peuvent coexister. On peut utiliser https, ssh, git et local.

Chaque dépôt distant à un nom (origin) et une url selon le protocole. La commande git remote permet de gérer la liste des dépôts distants qui partagent certaines branches de votre projet.

git remote add origin [git@gitlab.info-ufr.univ-montp2.fr](mailto:git@gitlab.info-ufr.univ-montp2.fr):mi.. ajouter un dépôt distant.

GitLab est un serveur de version Git et un site Web permettant également :

- ajouter éditer valider des fichiers sur le dépôt distant depuis le site web
- éditer un wiki
- signaler des problèmes et leur résolution
- gérer des groupes et des permissions sur les projets

2 interfaces : web : administration, gestion du dépôt distant.

Console : gestion de version courante sur le dépôt local.

2 méthodes d'authentification sont possibles : via ssh ou via https :

### SSH

Créer localement un couple de clés publique et privée :

```
ssh-keygen -t rsa -C "michel.meynard@univ-montp2.fr"
cat ~/.ssh/id_rsa.pub
```

copier la clé publique à la souris, dans GitLab sur le web, aller dans profile/SSH keys, cliquer sur add SSH Key, coller la clé publique

```
cd ~/.ssh/
chmod 0600 id_rsa
cd ~/test
git push -u origin master
```

Eval : 1h, réponses courtes et connaissances rapides sur les cours

## Cours Promise : Promesses javascript :

Créés avec JavaScript 2015, Une promise est une promesse d'existence d'une valeur qui peut être déjà disponible ou qui le sera plus tard, ou qui ne le sera jamais.

- Permet la réalisation des traitements asynchrones et les enchaîner,
- Une fois créée elle est dans un des états suivants : pending( en attente) promesse ni tenue ni rompue, (tenue) opération a réussi, (rompue) opération a échoué, (acquittée) tenue ou rompue.
- Les méthodes asynchrones renvoient des valeurs comme les méthodes synchrones mais ce sont des promesses.

Un objet promise est construit avec comme paramètre une fonction anonyme "exécuteur" à 2 paramètres :

- Les 1 paramètres formels seront des fonctions utilisées lorsque la promesse sera tenue ou en échec, résolve ou reject.

```
mapromesse = new Promise(function(resolve, reject) { ... } );
```

Pour résolve, la callback récupère le résultat souhaité dans l'exécuteur

Pour reject, la callback récupère souvent un message d'erreur (string)

Lors de la création de la promesse avec new, le code de l'exécuteur est lancé sauf les appels à resolve et reject qui sont en attente de liaison avec les callbacks effectifs. Ils sont en attente de l'événement liaison.

```
function mafonasync() {
    return new Promise((successCallback, failureCallback) => {
        console.log("promesse constructeur");
        let r=Math.random();
        if ( r > .5 ) {
            successCallback("Réussite");
        } else {
            failureCallback("Échec");
        }
    })
}
mafonasync().then(console.log, console.log);
console.log("la suite");
```

On peut conserver la promesse de retour dans une variable :

let mapromesse=mafonasync(); Cependant en cas d'échec une erreur aura lieu car toute promesse rompue doit être attrapée (catch). Il faut au minimum indiquer une callback pour reject au moment de l'appel.

```
x= mafonasyn().catch(console.log);
```

```
...
```

```
x.then(console.log);
```

La plupart du temps, on consomme des promesses renvoyées par une fonction pouvant éventuellement échouer (connexion BD, requête HTTP). On enchaîne souvent l'attachement des callbacks afin de construire une chaîne de promesses.

Eval : 1h, réponses courtes et connaissances rapides sur les cours

L'API fetch fournit une interface pour la récupération de ressources à travers le réseau. Elle remplace XMLHttpRequest en proposant néanmoins un ensemble de fonctionnalités plus souples et plus puissantes. Elle utilise les promesses. La méthode fetch() prend un argument obligatoire, le chemin vers la ressource souhaitée. Elle retourne une promesse qui résout la Response de cette requête, qu'elle soit couronnée de succès ou non. Par défaut, fetch() effectue une simple requête GET comme dans l'exemple suivant équivalent à un curl ou wget.

La méthode text() de Response retourne une promise qui se résout lorsque la lecture du flot de texte de la réponse est terminée et stockée dans une USVstring. En effet, la réponse de la promesse retournée par fetch contient un corps qui est un ReadableStream. La méthode json() appliquée à cette réponse retourne une promesse dont la valeur est un objet js contenant :

- url origin headers qui concernent http
- form, files args qui sont les données envoyées
- data qui contient le texte du body

Afin de proposer d'autres types de requêtes que celles par défaut, il faut fournir un second paramètre init à fetch sous la forme d'un objet js ayant des propriétés parmi les suivantes.

**method** : une chaîne indiquant la méthode de la requête "POST"

**headers** : soit un objet ou un tableau de tableaux de 2 chaînes spécifiant les entêtes de la requête.

**body** : le corps de la requête qui peut contenir un fichier à téléverser ou un FormData.

**referrer** : header de requête HTTP

**referrerPolicy** : contrôle la quantité d'infos sur le referrer.

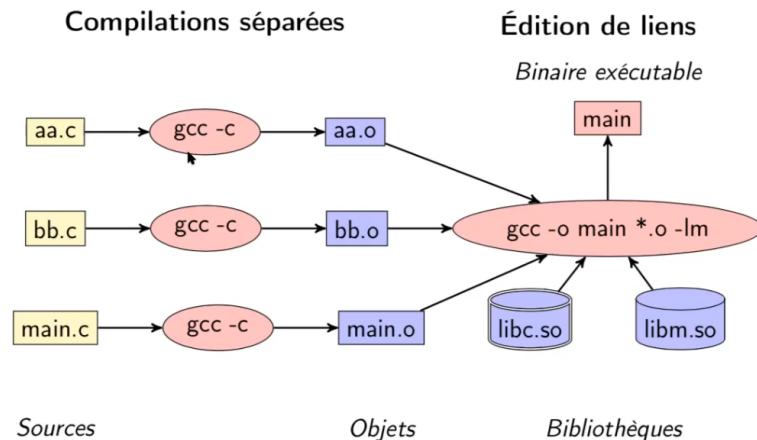
**mode** : une chaîne consiste à ajouter des en-tête HTTP afin de permettre à un navigateur d'accéder à des ressources d'un serveur situé sur une autre origine que le site courant.

**credentials** : une chaîne spécifiant l'envoi de cookie ou d'identifiants HTTP, aux sites de la même origine.

---

## Cours Librairies et make :

Langage compilé différent de langage interprété. Variables et fonctions peuvent être déclarées plusieurs fois mais définies une seule fois. Les fichiers d'en-tête ne doivent contenir que des déclarations. Une variable définie dans une fonction n'est pas initialisée par défaut, est située dans le segment pile, a une durée de vie égale au bloc d'instructions dans lequel elle est définie.. Une variable locale static définie une seule fois et doit être initialisée. Un objet dynamique n'est pas une variable il est référencé par un pointeur et n'est pas initialisé (sauf si malloc). Les variables globales ne sont pas recommandées, mais il est recommandé de toujours initialiser les variables.



Pour obtenir le résultat du prétraitement on fait gcc -E main.c. L'édition de lien réalisée lors de la compilation ne résout pas les liens vers les fonctions de bibliothèques dynamiques mais insère des appels systèmes pour réaliser la liaison à l'exécution. La commande ldd permet de lister l'ensemble des bibliothèques partagées requises par un exécutable.

- -fPIC : position independant code
- -shared : librairie partagée
- -Wl : fait passer l'option -soname libtruc.so.1 à l'éditeur de lien
- -soname : nom interne de la librairie inscrit par ld (Link eDit)
- -L : pour les références avec des versions différentes

Pour compiler mon programme en utilisant ma librairie truc :

```
gcc -o monprog monprog.c -ltruc; ./monprog
```

Fabriquer une librairie statique : **ar rs libmabib.a a.o b.o**

r : remplace les o

s : maj la table des symboles

Pour utiliser mabib : **gcc -static -o monprog monprog.o -lmabib -L.; ./monprog**

Make : gestion de projets multi-fichiers. Maintenance des dépendances inter-fichiers dans un fichier de fabrication : le makefile. Reconstruction optimale du projet : les fichiers non modifiés depuis la dernière construction ne sont pas compilés. Couplage possible avec un gestionnaire de versions. Le makefile est un fichier texte décrivant la façon de construire le projet et les objets dont il dépend : liste d'entrées séparés par des lignes vides.

Dépendances : indique les fichiers composants dont dépend le fichier cible.

```
monprog : hello2.o princ.o
@echo debut edition de liens : princ et hello2
gcc -o monprog hello2.o princ.o
@echo fin edition de liens
```

Eval : 1h, réponses courtes et connaissances rapides sur les cours

La construction d'une cible est récursive. Un make de chacun de ses composants est lancé. Construction de la cible du makefile situé dans le répertoire courant. Chaque règle est précédée d'un caractère de tabulation \t et non d'espaces multiples. Commentaires : commencent par # et finissent par \n.

Une macro variable est toujours définie au début du makefile et est évaluée avant la vérification de la cible. Les macros permettent de faire varier les constructions en ne modifiant que leurs définitions.

---

## Cours gestion de projet :

Iso : un projet est un processus unique qui consiste en un ensemble d'activités coordonnées et maîtrisées comportant des dates de début et de fin, dans le but d'atteindre un objectif, conforme à des exigences spécifiques.

AFNOR : un projet est un ensemble d'actions à réaliser pour satisfaire un objectif défini, dans le cadre d'une mission précise et pour la réalisation desquelles on a identifié un début et une fin.

### 2 Types de projets :

- projet d'ingénierie : visant un résultat pour un client
- projet produit : conception d'un modèle qui donnera lieu à la fabrication répétitive.

**Maître d'ouvrage** : personne physique ou morale qui sera propriétaire de l'ouvrage. Fixe les objectifs, les budgets et les délais et paie les dépenses liées à la réalisation.

**Maître d'œuvre** : personne physique ou morale qui réalise l'ouvrage et qui assure la responsabilité de la qualité , des délais et du coup.

**Equipe de projet** : ensemble des personnes placées sous l'autorité du chef de projet.

**Directeur de projet** : responsable devant le maître d'ouvrage du respect du cahier des charges par le maître d'œuvre.

**Chef de projet** : responsable devant le maître d'œuvre de l'avancement du projet.

**Concepteur** : informaticien gestionnaire ou organisateur qui conçoit avec un outil de modélisation dans les phases d'études préalables ou détaillées.

**Les acteurs** : développeur informaticien écrivant des programmes, support technique, contrôleur qualité, sont d'autres membres potentiels de l'équipe de projet.

Eval : 1h, réponses courtes et connaissances rapides sur les cours

Organisation :

- Le maître d'ouvrage établit un cahier des charges
- Après appel d'offres il passe contrat avec le maître d'oeuvre
- Ce dernier est responsable de la conduite de projet
- Lors de chaque fourniture contractuelle d'un livrable, le maître d'ouvrage procède à la recette en prononçant l'acceptation ou le refus.

Un livrable est un élément du tout qui doit être fourni et qui doit être reçu par le maître d'ouvrage, la recette du livrable c'est une phase dans laquelle le maître d'ouvrage va accepter ou refuser le livrable.

Il y a la gestion du temps, des ressources, et de la production.

Le cycle de management est le suivant :

- Analyser
- Organiser
- Piloter
- Aller au pas 1

Découpage d'un projet :

Le découpage s'appuie sur :

- L'approche cartésienne : réduction de la difficulté globale
- L'approche systémique : liens et flux entre sous-systèmes

Deux découpages non exclusifs peuvent être :

- Temporel : le cycle de vie est une succession d'étapes de phases, d'activités de tâches.
- Structuré : maîtriser des petits modules, répartir les responsabilités, réduire les délais grâce au parallélisme.

Découpage structurel :

- Domaine : sous-ensemble du système d'information qui ne repose pas forcément sur la répartition administrative de l'organisation.
- Des sous-domaines, modules peuvent être définis.
- Chaque domaine possède des informations et des processus propres.
- Le découpage peut s'appuyer sur une vision statique : données, objets, datagrammes
- Le découpage peut s'appuyer sur une vision dynamique : traitements , communication, diagramme de séquence, actigramme.

Cycle de vie classique : Merise

- Schéma directeur (pour l'ensemble des domaines)
- étude préalable par domaine
- conception
- étude détaillée (spécifications externes)
- étude technique (spécifications internes)
- réalisation (programmation)
- mise en oeuvre (installation)
- qualification (tests dans l'environnement opérationnel)

Eval : 1h, réponses courtes et connaissances rapides sur les cours

Un modèle de vie spécifie l'ordre et la manière dans lesquels les étapes sont réalisées :

**cascade** : les phases sont effectuées simplement les unes après les autres chaque phase :

- produit un livrable préalablement défini
- se termine à une date précise
- ne se termine qu'après validation du livrable

**En v** : amélioration du modèle en cascade

- en cas d'anomalie, limite le retour à l'étape précédente.

**Incrémental** : découpage du projet en lots dont le premier est le noyau auquel viennent se raccrocher les incréments suivants. (extensions, plugins)

**Itératif** : cette approche reprend les différentes étapes du cycle en V. Elle s'appuie sur une succession de cycles découpés en quatre phase :

1. Déterminer les objectifs du cycle, les solutions pour les atteindre et les contraintes
2. Evaluer les solutions en choisir une, développer une série de tests pour vérifier l'adéquation
3. Développer et vérifier la solution retenue
4. Évaluer le prototype avec la MOA et planifier le cycle suivant.

Les méthodes agiles : elles sont itératives et incrémentales en partie :

- **objectif** : construire progressivement le système de manière participative
- **caractéristique** : chaque cycle est court et aboutit à une version du système que l'utilisateur final expérimente, comment.
- **Incrémental** : Le modèle est incrémental lorsque chaque cycle introduit de nouvelles fonctionnalités.
- **RUP** : Rational Unified Process utilise UML et conjugue plusieurs approches.
- **RAD** : Rapid Application Developpement possède 5 phases linéaires dont celles de construction est itérative.

**DSDM** : La phase d'étude du métier utilise des ateliers réunissant équipe du projet et clients qui doivent spécifier conjointement

**XP** : Dans eXtreme Programming; des itérations de livraisons englobent des itérations de développement, on utilise le pair programming

**Scrum** : Méthode agile la plus utilisée de nos jours.

Le manifeste pour le développement logiciel agile regroupe 4 valeurs et 12 principes de base repris ensuite par toutes les méthodes agiles. 4 valeurs :

- aux individus et leurs interactions plutôt qu'aux processus et aux outils ;
- à un logiciel fonctionnel plutôt qu'à une documentation exhaustive ;
- à la collaboration avec les clients plutôt qu'à la négociation contractuelle ;
- à l'adaptation au changement plutôt qu'à l'exécution d'un plan.

Eval : 1h, réponses courtes et connaissances rapides sur les cours

12 principes :

- Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
  - Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
  - Livrez fréquemment un logiciel fonctionnel, dans des cycles de quelques semaines à quelques mois, avec une préférence pour les plus courts.
  - Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
  - Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont elles ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
  - La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.
  - Un logiciel fonctionnel est la principale mesure de progression d'un projet.
  - Les processus agiles encouragent un rythme de développement soutenable.  
Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
  - Une attention continue à l'excellence technique et à un bon design.
  - La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
  - Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
  - À intervalles réguliers, l'équipe réfléchit aux moyens possibles de devenir plus efficace. Puis elle s'adapte et modifie son fonctionnement en conséquence.
- 

Cours MVC :

Patron d'architecture qui est utilisé par l'ensemble des frameworks.

Un patron de conception est une structure de code reconnue comme bonne pratique en réponse à un problème de conception d'un logiciel. Solution standard, utilisable dans la conception de différents logiciels. Un patron = une manière d'organiser des modules ou des classes afin de répondre à un besoin récurrent.

**Structure d'un patron de conception :**

Nom (identifiant du problème), description (du problème à résoudre), solution (les éléments de la solution), conséquences (résultats issus de la solution).

L'orthogonalité :

- Chaque patron doit correspondre à une approche différente, qui ne répète pas les idées d'autres patrons

Eval : 1h, réponses courtes et connaissances rapides sur les cours

- Le concepteur analyse un problème complexe et en résout chaque aspect à l'aide d'un patron
- Il combine ensuite les patrons pour construire une solution
- Certains auteurs de patrons proposent d'avantages d'orthogonalité que les patrons.

**Factory** : classe qui va fabriquer des objets de différentes classes.

**Adaptateur** : Convertir l'interface d'une classe en une autre interface exploitée par une application.

**Pont** : Permet de découpler une abstraction de son implémentation.

**Monteur** : Séparer le processus de construction d'un objet du résultat obtenu.

**Chaîne de responsabilité** : vise à découpler l'émission d'une requête de la réception et le traitement de cette dernière en permettant à plusieurs objets de la traiter successivement.

**Commande** : Emboîte une demande dans un objet, permettant de paramétriser, mettre en file d'attente, journaliser et annuler des demandes.

**Décorateur** : permet d'attacher dynamiquement des responsabilités à un objet.

**Iterator** : classe associée à une classe itérable.

**Proxy** : substitut d'un objet qui permet de contrôler l'utilisation de ce dernier. Protège un autre objet.

**Singleton** : classe qui possède une instance et ne possède pas de constructeur public, il faut une méthode get quand il est créé sinon il faut pouvoir le créer.

**Observer** : Établit une relation un à plusieurs entre des objets, indique des objets qui observent des événements.

**MVC** : architecture des applications avec interface graphique. Le modèle est chargé de fournir et de sauvegarder les données. Le contrôleur reçoit des messages provenant de l'interaction utilisateur ou de déclencheur du modèle et répond à ceux-ci en consultant et/ou modifiant le modèle et en mettant à jour la vue.

**Inversion de contrôle** : framework prend en charge l'exécution principale du programme coordonne et contrôle l'activité de l'application.

On souhaite améliorer la modularité de notre conception MVC :

- La connexion à la base de données doit être fermée à la fin du parcours de la table par la vue
- On crée une classe singleton Connexion qui permet de n'ouvrir qu'une seule fois la connexion dans le contrôleur même en cas de vues multiples
- Cette connexion sera fermée par le contrôleur
- Le modèle définit une fonction getEtudiant() permettant de sélectionner certains étudiants selon certains critères.

- Si la mise en page est la même dans tout le site (header, nav, main, footer), il est souhaitable de factoriser au moins l'en-tête et le pied de page dans des fichiers inclus par chaque vue

Aller plus loin :

- Il est parfois souhaité qu'une seule page index.php représente tout le site ! Il faudra utiliser alors un paramètre GET (action ou page) pour définir la page à atteindre : index.php?page=listeEtudiantC. Le fichier index.php peut être vu alors comme un super contrôleur du site qu'on appellera routeur
- Un mécanisme de réécriture d'URL (URL rewriting) existe dans la configuration d'Apache qui permet de router sans utiliser de paramètres mais en allongeant le chemin

## Cours Entité - Association :

**Solution conceptuelle** : Indépendante du rôle respectif de l'homme et de la machine et des solutions techniques choisies.

**Solution logique** : Précise les rôles respectifs de l'homme et de la machine et les ressources utilisées.

**Solution physique** : Adaptée à un ensemble de moyens matériels.

Un modèle global des données :

- idée de stabilité des données
- les données devront être partagées
- une conception synthétique des données

Un formalisme lisible :

- rôle moteur de l'utilisateur (nécessité de bien communiquer)
- permet la communication entre spécialistes.
- le formalisme adopté est intuitif

Un monde d'individus (var, instances, entité, objet)

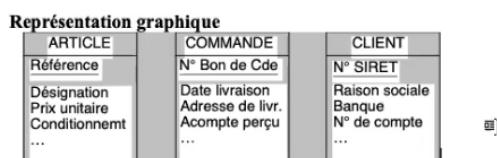
Rangés en types (catégories, classes, sortes) d'objets semblables.

**Individu** : objet abstrait ou concret.

**Propriété** : attribut spécifique à un individu permettant de le décrire

Une valeur, une atomicité(ni calculée, ni agrégée).

**Individu type** : classe d'individus ayant les mêmes propriétés. Chaque occurrence d'un individu type est unique et identifié par une propriété-type particulière nommée identifiant.



**Remarques :**

- la propriété est la **plus petite** unité d'information
- propriété particulière: l'**identifiant**
  - permet de distinguer les objets
  - c'est à dire de repérer un objet particulier parmi ceux de sa classe

**Relation** : Les individus peuvent être en relation (n-aire), associés. (verbes du discours)

Eval : 1h, réponses courtes et connaissances rapides sur les cours

**Relation-type** : ensemble des relations de même ordre définies sur plusieurs individus types.

Les relations n'existent que parce qu'elles relient des objets (individus-types).

Les relations, les individus et les propriétés sont regroupées en types.

Relations :

- Toutes les propriétés d'une relation doivent être valuées
- chaque lien doit être nécessaire : on préfère les relations binaires
- S'il est possible de mettre une propriété soit dans une relation soit dans un individu y participant, on choisira cette seconde solution
- Chaque relation doit respecter ses cardinalités de lien
- Une instance de relation est identifiée par l'ensemble des identifiants des individus y participant

Propriété calculée ou agrégée : montant total d'une commande, numéro de sécurité sociale...

#### Transformation des associations (relations du MCD)

- 1) association dont aucun lien n'a (1,1) pour cardinalités → table R
- Ses propriétés (s'il y en a) deviennent des attributs de R
  - On recopie dans la nouvelle table R les identifiants (clés)  $K_j$  de chacun des individus  $I_j$  participant à l'association.  
Pour chacun des  $I_j$  participant à l'association avec une **cardinalité minimale nulle** (i.e., toutes les occurrences d' $I_j$  ne participeront pas forcément à l'association) :
    - on renomme  $K_j$ , soit  $K'_j$
    - on pose la Dépendance d'Inclusion :  $K'_j \subseteq K_j$

Remarque :

Si aucun des liens de l'association n'a (0,1) pour cardinalités

Alors L'ensemble des  $K_j$  (ou  $K'_j$ ) constituent la clé primaire de la nouvelle table

Sinon pour chacun des liens de cardinalités (0,1), le  $K_j$  correspondant constitue une clé candidate de R

---

Cours MongoDB :

MongoDb est un sgbd, (not only sql) permettant de stocker des documents dans des collections.

- SGBD orienté documents, répartissables sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données.
- Il est écrit en c++ et fait partie de la mouvance NoSQL
- Le serveur et les outils sont distribués sous licence SSPL
- MongoDB permet de manipuler des objets structurés au format BSON (JSON sérialisé et encodé en binaire).

Eval : 1h, réponses courtes et connaissances rapides sur les cours

- Sans schéma prédéterminé (index peuvent être ajouté à tout moment)
- Les données prennent la forme de documents enregistrés eux-mêmes dans des collections, une collection contenant un nombre quelconque de documents sauf si plafonnée
- Les collections sont comparables aux tables , et les documents aux lignes des bases de données relationnelles
- Le seul champ commun et obligatoire est le champ de clé principale(\_id)
- MongoDB ne permet ni les requêtes très complexes standardisées ni les JOIN mais permet de programmer des requêtes spécifiques en Javascript

Permet la réPLICATION et la réPARTITION de données sur un ensemble de serveurs à des fins de résistances aux pannes et de répartition de la charge.

Un serveur de données mongod doit être lancé en arrière plan en définissant la localisation de la BD : **\$ mongod --dbpath data/db/**

Un client permettant d'interagir avec la ligne de commande est lancé dans un autre terminal : **\$ mongo**

C'est dans ce client CLI que l'on va explorer quelques commandes dans un langage très proche de Javascript :

- **show dbs** pour lister les BDs disponibles
- **use mabd** définit la BD courante
- **db** affiche la BD courante
- **db.etudiants.fin()** affiche tous les documents de la collection étudiants.

Utiliser l'indexation et l'accès à une propriété js :

- **db.etudiant.find()[1].prenom**

compter les éléments :

- **db.etudiant.fin().count()**

itérer sur la collection et exécuter un traitement js sur chaque document :

- **db.etudiant.find().forEach(doc => print(doc.prenom))**

La méthode findOne() permet de ne récupérer que le premier document satisfaisant les critères de recherche.

- une expression régulière à la Perl **sans guillemets** peut être utilisée :  
> db.etudiants.find({nom: /du.\*/i}) // tous les du... DU...  
→ Du...
- \$eq, \$ne, \$lt, \$lte, \$gt, \$gte pour comparer des valeurs  
> db.etudiants.find({nom: {\$gt: "C"}}) // ni A... ni B...
- appartenance ou non \$in, \$nin  
> db.etudiants.find({prenom: {\$in: ["Paul", "Michel"]}})
- existence d'une propriété  
> db.etudiants.find({prenom: {\$exists: true}})
- opérateurs logiques \$or, \$not, \$nor (le et logique est le connecteur par défaut). La requête suivante retourne les étudiants de prénom Paul et qui n'ont pas de nom ou un nom qui commence par Du...

Eval : 1h, réponses courtes et connaissances rapides sur les cours

Distinct : supprime les doublons à la place de find.

Les jointures entre les collections :

Souvent les concepteurs de collections préfèrent avoir des objets métiers très imbriqués afin d'éviter les jointures. Mais elles sont parfois indispensables.

Ceci est réalisé par la méthode aggregate qui contient un tableau d'actions exécutées en pipeline.

L'action \$unwind permet de distribuer le tableau des ues sur chaque etudiant.

- Insertion d'un document  
> db.ues.insert({code: "HAIN303I", nom: "Systèmes  
↳ d'exploitation"}) // crée la collection et insère
- Mise à jour d'un document à l'aide d'un filtre de sélection (1<sup>e</sup> paramètre) et d'une action d'affectation d'une nouvelle valeur de propriété (set) ou de suppression de la propriété (unset)  
> db.etudiants.update({nom:  
↳ "Durand"}, {\$set:{ues:[ObjectId("605a3a14507f6aa68c7c5a96")]},  
• Suppression de documents à l'aide d'un filtre de sélection :  
> db.etudiants.remove({nom: "Martin"})  
• Suppression de collection :  
> db.ues.drop()

- MongoDB est une alternative aux SGBD relationnels
- Il permet de conserver des collections de documents beaucoup moins structurées que les tables
- Certaines propriétés peuvent être présentes dans certains documents seulement.
- L'imbrication des documents est permis grâce aux listes et aux objets
- Les documents sont beaucoup plus proches des objets métiers : un événement contient des tournois qui contiennent des tours, poules, match ..
- L'API pour réaliser des requêtes est assez spécifique et complexe (find, aggregate ..)
- La capacité de MongoDB à utiliser plusieurs serveurs répartis et possédant des fragments de la BD est très prisée.
- Il existe d'autres SGBD NoSql comme Neo4J basé sur des graphes;

---

## Cours Doctrine :

Outil permettant de faire de la persistance d'objets en php. C'est un middleware situé entre la couche métier et la couche données. Il est possible de l'utiliser avec des frameworks. Une correspondance objet-relationnel est un patron de conception qui permet de manipuler une base de données relationnelle en utilisant des objets PHP persistants. Ainsi le code PHP est plus homogène, ce qui facilite sa programmation et son débogage. **Une entité** est un objet PHP identifié par un attribut unique qui est lié à la clé primaire de la ligne qui lui correspond. Une classe d'entités possède des entités chaque entité correspondant à une ligne d'une table relationnelle. Génération de la BD à partir de classes d'entités métiers ou construction du modèle objet à partir d'une BDD existante. Pas de schéma XML intermédiaire entre modèle objet et BD. DQL doctrine query language.

Eval : 1h, réponses courtes et connaissances rapides sur les cours

- Les entités sont liées entre elles par des associations : une association matérialise sur les objets PHP une contrainte d'intégrité référentielle de la BD.
- Des transactions ACID sont présentes.
- L'actualisation des associations dans les entités est réalisée de manière fainéante, au fur et à mesure des accès.
- code first, database first, model first qui sont créer le code puis créer la BD, ou construire le modèle objet à partir d'une base de donnée existante. Ou permet de tout construire à partir d'un modèle UML.
- La commande CURL permet de télécharger l'exécutable php installer.phar
- Puis on passe via un tube ce même fichier à l'interprète php qui exécute l'installeur.
- Le fichier composer.phar est installé et est disponible.
- 

Doctrine s'installe, comme Propel, en utilisant Composer. On crée le fichier composer.json suivant :

```
{  
  "require": {  
    "doctrine/orm": "*"  
  }  
}
```

EntityManager est une classe PHP qui fournit le point d'accès au cycle de vie complet des entités. C'est une classe singleton utilisant le patron de conception factory.

- \$entityManager->persist(\$product) ; permet de tamponner l'entité product dans le gestionnaire d'entités
- cette méthode persist sera utilisée plusieurs fois avant de lancer la transaction de BD qui effectuera l'ensemble des requêtes (insert ou update ou delete).
- \$entityManager->flush() ; lance la transaction sur la BD

L'état final de l'entité sera enregistré lors du flush().

Avant de rendre persistantes les entités, il faut les définir.

- la classe d'entité en PHP : ensemble d'attributs protégés ayant chacun une paire d'accesseur/mutateur (get/set) et un attribut id qui correspond à la clé primaire et qui lui ne possède qu'un accesseur (dans le répertoire src)
- chaque classe peut posséder des méthodes métiers
- les métadonnées de la classe qui représentent la liaison entre classe et table et entre attribut et colonne
- Ces métadonnées peuvent être fournies en XML, en YAML (XML indenté) dans un fichier externe ou en PHP grâce à des annotations (\*\* @Entity ...) dans le fichier définissant la classe d'entité

Afin de générer le schéma de BD en SQL (CREATE TABLE ...) et de l'exécuter, il faut utiliser la commande doctrine suivante depuis un terminal :

```
$ vendor/bin/doctrine orm:schema-tool:create --dump-sql  
CREATE TABLE products (id INTEGER NOT NULL, name VARCHAR(255)  
NOT NULL, PRIMARY KEY(id));
```

Eval : 1h, réponses courtes et connaissances rapides sur les cours

La méthode `find($entityName, $id)` de l'`entityManager` permet de récupérer une entité par son identifiant. Le formulaire web suivant permet d'afficher un produit.

On peut utiliser `findBy()` ou `findOneBy()` sur un dépôt en fournissant un tableau de clé valeur où chaque clé est un nom d'attribut accepté.

DQL != SQL.

- Les associations entre entités permettent de modéliser les relations existant entre les objets métiers
- Elles sont, pour certaines, liées à des contraintes d'intégrité référentielles dans la BD mais **pas toujours**
- e.g. l'héritage entre classe d'entités ne donnera pas forcément lieu à une relation dans la BD ↗
- certaines BD n'implémentent pas certaines contraintes d'intégrité

Quelques règles :

- une association unidirectionnelle ne possède qu'un côté propriétaire (*owning side*)
- une association bidirectionnelle possède un côté propriétaire (*owning side*) et un côté inverse
- l'entité du côté inverse utilise l'attribut `mappedBy` pour désigner l'attribut de l'entité propriétaire
- l'entité du côté propriétaire utilise l'attribut `inversedBy` pour désigner l'attribut de l'entité inverse
- seuls les changements (ajout, suppression) côté propriétaire seront pris en compte par doctrine lors du `flush` : effectuer les changements des deux côtés ou bien seulement du côté propriétaire ↗

---

## Cours Doctrine Fin :

Association unidirectionnelle many-to-many :

- Des utilisateurs sont affiliés à différents groupes d'utilisateurs
- Dans la BD, une table de jointure permet de représenter les associations multiples de groupes et d'utilisateurs.

Plusieurs implémentations d'héritages possibles :

- Mapped Superclass : permet de définir les attributs et les associations communs dans une superclasse qui ne donnera pas lieu à création de table mais chaque entité fille donnera une table contenant les champs de sa superclasse.
- Single Table Inheritance : dans ce patron toute la hiérarchie est représentée dans une unique table qui contient un champ discriminant la classe. Les champs spécifiques à chaque sous-classe doivent absolument être NULL puisqu'ils existeront pour tout le monde.

Eval : 1h, réponses courtes et connaissances rapides sur les cours

- Class Table Inheritance : chaque classe fille correspondra à une table la représentant et autant de tables que nécessaire pour représenter ses ancêtres avec des clés étrangères les reliant.

Afin de générer les métadonnées associées à une BD préexistante, on peut utiliser la commande doctrine afin de générer un fichier de métadonnées xml, php .. Mais cette technique ne permet pas de récupérer les associations inverses, les héritages, les clés étrangères etc. Chaque table doit avoir une clé primaire.

- L'utilisation d'un ORM comme Doctrine ne peut être envisagé que dans le cadre d'un développement à plein temps d'un projet.
  - En effet, le nombre de concepts employés ainsi que le vocabulaire utilisé nécessitent un long temps d'apprentissage.
  - Ses concepts proviennent d'ORM leaders dans d'autres langages et son apprentissage reste utile.
- 

## Cours débogage GDB :

Débogueur : programme qui analyse un code pour y trouver des bogues.

Le débogage consiste à chercher et corriger les erreurs dans le code. Résultat incorrect d'un programme, arrêt inopiné, fuite mémoire. Cela ne gère pas les erreurs de syntaxe ou de sémantique. Il gère uniquement les problèmes à l'exécution.

**gdb monprog** : démarrer une session de débogage

**break main** : poser un point d'arrêt

**run toto 24** : lancer le programme

**print argc** : évaluer une expression lorsque l'exécution s'interrompt sur un point d'arrêt

**next** : avancer d'une ligne dans l'exécution de la fonction

**step** : avancer d'une instruction dans l'exécution, en particulier , en entrant dans une fonction appelée dans l'instruction courante

**continue** : sortir du mode (pas à pas) en continuant l'exécution jusqu'au prochain point d'arrêt

**ctrl-d** : quitter l'exécution courante

**quit ou ctrl-d** : quitter la session

- jdb (Java Debugger)
- codeView (Débogueur d'environnement de développement intégré Microsoft Visual C++)
- bashdb (débogueur shell indépendant du projet GNU)
- gdb (GNU Debugger)
- llDb (utilisable sur Mac OSX)
- Xdebug (extension PHP permettant le débogage à distance)
- débogueur JavaScript **graphique** intégré au navigateur Chrome ou Firefox

Eval : 1h, réponses courtes et connaissances rapides sur les cours

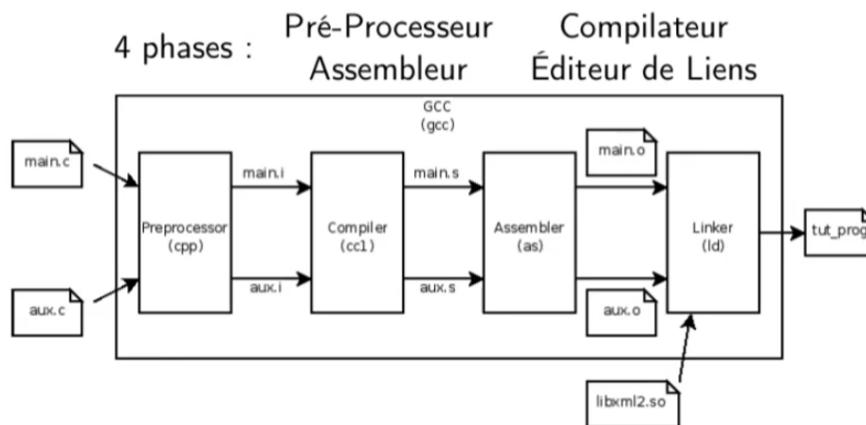
### Deux types d'utilisation du débogueur GDB :

- GDB permet de déboguer un programme en cours d'exécution en le déroulant instruction par instruction ou en examinant et modifiant ses données)
- Il permet également un débogage post-mortem en analysant un fichier core qui représente l'image mémoire d'un programme terminé anormalement

Utilisation de GDB :

```
gcc -g -o monprog mobprog.c; gdb monprog
```

Compilation en C++ :



Les symboles de débogage sont des données que l'on rajoute dans l'exécutable binaire, afin que le débogueur puisse lire ces dernières. Sans symboles de débogage, gdb ne sait pas quelle adresse dans l'exécutable correspond à quelle ligne de fonction du code source. Ils font office de dictionnaire qui contient :

- L'adresse liée à la fonction ou à la variable
- Le type de donnée
- Le nom du fichier et son numéro de ligne où cette donnée est définie

**gdb list** : permet de lister la prochaine ligne à exécuter et ses suivantes.

Les commandes suivantes peuvent être abrégées (souvent les 2 premières lettres i.e. br main) :

Commande	Arguments	Description
run	(arg1 arg2 ...)	Exécute le programme
kill	-	Arrête le programme
quit	-	Quitte gdb
help	( all / [commande] )	Affiche l'aide
cd	[dir]	Change de répertoire
file	[fichier]	Charge le fichier
attach	[PID]	Attache GDB sur un PID
shell	[commande]	Exécute une commande shell

Eval : 1h, réponses courtes et connaissances rapides sur les cours

Les breakpoints servent à surveiller le fonctionnement du programme. Il permet de mettre en pause l'exécution du programme.

- break [fonction] : Met en pause l'exécution de l'appel de la fonction
- break [fichier] :[n] : Met en pause l'exécution à la ligne n du fichier passé en argument
- break [+/- n] : Met en pause l'exécution à la position courante +/- n ligne(s)
- break [ligne/fonction] if [condition] : Met en pause l'exécution uniquement si la condition est vérifiée
- clear ([n° ligne]) : Détruit les breakpoints

On utilise watch (les watchpoints) pour s'arrêter dès que la variable est utilisée.

- VARIABLES, Locals : affiche les valeurs des variables globales et locales
- BREAKPOINTS affiche les points d'arrêts
- WATCH permet de définir des expressions complexes dont la valeur est calculée en temps réel MAIS **ne permet pas** d'arrêter le programme sur modification d'une variable
- CALLSTACK affiche la trace des fonctions appelées (backtrace)

Les avantages d'un débogueur graphique :

- Ne pas apprendre les commandes CLI
  - Même interface pour déboguer du Java, Python, Js ...
  - On reste dans l'environnement de développement aussi les corrections vont beaucoup plus vite
  - Mais par contre, certaines commandes sophistiquées des débogueurs ne sont pas implémentées (watchpoints de GDB, attachements à un processus,...)
-