Maintenance applicative

Maintenance applicative : plan du module

- Gestion de la maintenance applicative
- Maintenabilité
- **Changement d’outils**

Maintenance applicative : plan du cours

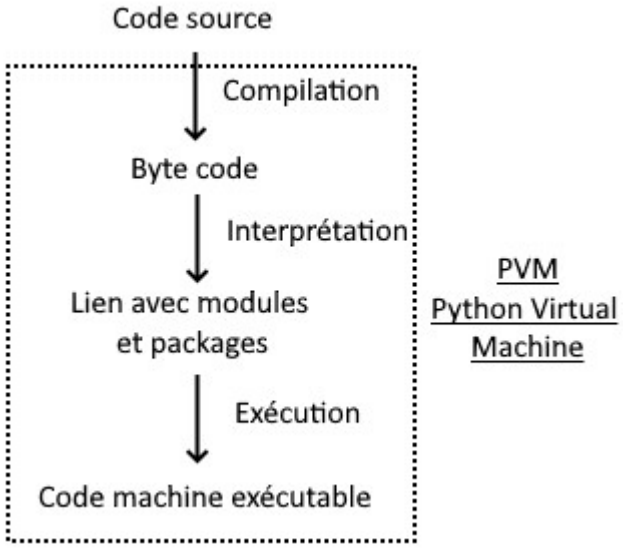
- Interpréteur
- Compilateur
- Virtualisation

Maintenance applicative : interpréteur processus d'interprétation

L'interpréteur Python est un exécutable (programme compilé utilisable sous Windows, Linux...), écrit en C (interpréteur de référence Cpython) qui :

- transforme un code source Python (fichiers .py) en code compilé : byte code (fichiers .pyc)
- transforme le bytecode en code exécutable en utilisant les modules et packages standards ou ceux importés
- Exécute le code. C'est la phase de runtime

La PVM gère la mémoire de son processus (donc de votre programme), le typage dynamique des données, possède un mécanisme d'interprétation mono-thread (GIL) ce qui limite l'usage du multi thread en Python.



Maintenance applicative : interpréteur environnement virtuel

- L'interpréteur possède un module venev d'environnement virtuel. Celui-ci permet de faire une photographie de l'interpréteur, de ses modules et packages.
- L'avantage est de pouvoir exécuter un programme toujours dans le même environnement, ce qui limite la maintenance adaptative.
- Exemple : virtualenv sous python

  virtualenv -p usrlocal/bin/python3.10 p6 est une commande de création de l'environnement virtuel (appelé p6 pour projet 6 par exemple) pour l'interpréteur Python et ses modules. L'intérêt est de figer les modules installés et leurs versions afin d'avoir toujours la même chaîne de génération. Dans un terminal, il suffit d'écrire source/p6/bin/activate pour se retrouver dans l'environnement du projet 6.

Maintenance applicative : interpréteur intégration dans/de l'interpréteur

Les modules en Python sont intégrés dans l'interpréteur par :

- les modules intégrés standard ou importés avec la commande pip
- les modules écrits en C ou C++ : des modules en langages compilés peuvent étendre les fonctionnalités de Python. Ces modules sont compilés en librairies partagées (.so sous Unix, .dll sous Windows) qui sont ensuite importables en tant que modules Python

L'interpréteur Python lui-même peut être intégré dans une application compilée : à l'aide d'une librairie et de fichiers include, on peut manipuler l'interpréteur Python dans un autre programme comme un objet avec ses méthodes et ses attributs. Cela permet d'exécuter des scripts Python dans un programme qui nécessiteraient leurs résultats dans son propre processus. En savoir plus

Maintenance applicative : interpréteur changement de versions et impacts

Le passage de Python 2 à Python 3 a apporté de nombreux changements. Parmi ceux-ci :

- Changement dans la division : En Python 2, la division entre deux entiers donne un résultat flottant. En Python 3, un résultat entier.
- Print : instruction en Python 2 mais fonction en Python 3
- Chaînes de caractères : str et bytes sont des types distincts en Python 3
- Nouvelles conventions de codage

Le prochain passage de Python 3.10 à Python 4 impliquera des changements et des choix dans les équipes de maintenance logicielle !

Maintenance applicative : interpréteur changement de versions et maintenance

2 choix pour les développeurs :

- Continuer à utiliser un environnement virtuel figé pour l'interprétation du programme afin de ne traiter que les demandes de maintenance corrective ou perfective

  Avantage : stabilité, pas de modification de code

  Inconvénient : les modules n'évoluent plus et les mises à jour récentes ne sont pas accessibles (notamment en sécurité)

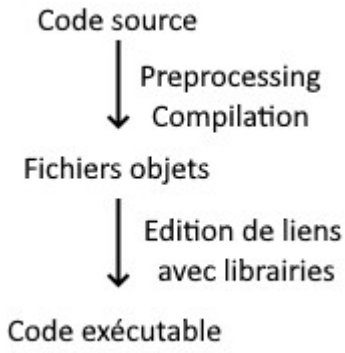
- Projet de maintenance adaptative : migration de l'ancienne version de l'interpréteur à la nouvelle.

  Avantage : mise à jour accessibles

  Inconvénients : une partie du code existant doit être réécrite (importance des tests et de la documentation technique). Si les programmes sont nombreux, les projets de migration seront nombreux et consommeront du temps et des ressources

Maintenance applicative : compilateur fonctionnement

- On devrait parler de chaîne de compilation plutôt que de compilateur. Un compilateur est un élément de la chaîne de compilation avec le préprocesseur et l'éditeur de liens. Mais le langage courant parle de compilateur pour les deux éléments.
- Un compilateur tels que gcc ou g++ transforme du code source en code machine exécutable
- Un compilateur utilise d'autres fonctions compilées rassemblées dans des librairies (aussi appelées bibliothèques) pour produire l'exécutable pendant l'édition de liens
- La chaîne de compilation produit le code machine mais ne l'exécute pas.



Maintenance applicative : compilateur les librairies

- Les librairies statiques : une librairie statique est incorporée dans l'exécutable pendant l'édition des liens. Elle est intégrée dans l'exécutable, ce le rend plus volumineux avec un fonctionnement indépendant. Fichiers : .a (linux) ou .lib (windows)

  Exemple : gcc main.c -static -o main.out compile statiquement main.c, le code de printf sera dans l'exécutable

- Les librairies dynamiques : une librairie dynamique n'est pas incluse dans l'exécutable qui contient des références aux fonctions de la librairie. Ces fonctions sont chargées depuis la librairie dynamique au moment de l'exécution. Cela permet de partager une même instance de la librairie entre plusieurs exécutables. Fichiers : .so (linux) ou .dll (windows)

  Exemple : gcc main.c -o main.out compile dynamiquement main.c, le code de printf est en mémoire partagée par l'OS et l'exécutable à une référence vers ce code qui sera « lié » à l'exécution

Maintenance applicative : compilateur
la maintenance des librairies

- Dans le cas statique, une modification apportée à la librairie n'est pas intégrée à l'exécutable existant. Il faut recompiler l'exécutable qui l'utilise avec la nouvelle version de la librairie. C'est un projet de maintenance corrective ou perfective.
- Dans le cas dynamique, la mise à jour d'une librairie est effectuée sans recompiler l'exécutable qui bénéficie de la nouvelle version. Il faut être prudent et veiller à la compatibilité des interfaces (API). Si la librairie dynamique n'est pas disponible ou si son API change, l'exécutable ne fonctionnera pas correctement (on parle de dépendance). C'est un projet de maintenance corrective ou perfective.

Maintenance applicative : compilateur création de librairies

Génération

- Sous linux : gcc -c -fPIC my_source_file.c puis gcc -shared -o my_library.so my_source_file.o
- Sous windows, gcc -c my_source_file.c puis gcc -shared -o my_library.dll my_source_file.o. Sous windows, gcc provient de l'installation de mingw.

Utilisation en maintenance adaptative : les versions compilateurs/librairie

- Il faut faire très attention à la version des compilateurs et des librairies. Un compilateur gcc 32 bits génère une librairie en 32 bits. Un compilateur gcc 64 bits génère une librairie en 64 bits.
- Une librairie 32 bits ne sera pas utilisable par un compilateur 64 bits et inversement. Dans un tel cas, le compilateur n'utilisera pas la librairie et affiche une erreur d'édition de lien (très générale). Il faudra sans doute vérifier la version du compilateur et de la librairie si l'erreur persiste. C'est une erreur non traditionnelle mais il faut absolument y penser !

Maintenance applicative : compilateur virtualisation

- Les machines virtuelles (VM) jouent un rôle important dans la maintenance adaptative.
- Elles fournissent un environnement stable pour héberger et exécuter les compilateurs et leurs versions de librairie, équivalent à l’environnement virtuel des interpréteurs.
- Les VM peuvent être facilement déplacées, redimensionnées, dupliquées ce qui facilite l'adaptation aux changements technologiques et aux besoins des utilisateurs.
- Par exemple, si un logiciel doit être migré vers un nouveau système d'exploitation ou une nouvelle librairie dynamique, une VM peut être configurée pour tester la migration. Si celle-ci est validée, la nouvelle VM prend également en charge la transition sans perturbation du service en production.

Maintenance applicative : compilateur conteneurisation

- Les containers représentent une autre solution que les VM, plus flexible et moins lourde. Mais les fonctionnalités sont plus limitées également.
- Les containers fournissent un environnement stable pour héberger et exécuter les compilateurs et leurs versions de librairie, équivalent à l'environnement virtuel des interpréteurs.
- Par exemple, un site Web qui authentifie des utilisateurs doit être maintenu en cohérence avec les protocoles de sécurité utilisés. Un container favorisera un environnement stable notamment au niveau des librairies.
- La maintenance adaptative utilise souvent un container pour évaluer un changement de version de librairie. Si la migration est validée, une VM sera en général utilisée pour le déploiement.

Maintenance applicative : compilateur conclusion

- L’environnement de développement doit être dupliqué dans un environnement virtuel afin de pouvoir reconstruire un exécutable en maintenant dans les mêmes conditions qu’en développement.
- La maintenance adaptative s’appuie sur les containers ou les VM pour valider des migrations d’un environnement ancien vers un nouveau. La difficulté concerne généralement les changements liés aux OS, aux librairies et aux modules.