Compilation automatisée : make et le Makefile

Introduction et Principes :

- compilation séparée : un binaire, ou une bibliothèque peuvent comporter plusieurs parties, compilées séparément, et assemblées pour constituer un binaire unique
  (une bibliothèque est assimilée à un binaire sans main ici)

- la recompilation d'un fichier source est considérée comme inutile tant qu'il existe un fichier objet à jour, c'est à dire créé après la dernière modification du source.

=> Complément (prochain cours) : **gestion des fichiers d'en-tête pour éviter l'inclusion multiple**

Exemple :

Un binaire utilise le contenu de trois fichiers sources : programme.c , entrees.c et sorties.c

Pour compiler programme.c, on peut faire :

gcc [options] programme.c entrees.c sorties.c

si entrees.o et sorties.o existent, et sont à jour,
(entrees.c et sorties.c n'ayant pas subi de modification depuis la dernière compilation) ,

alors, une commande équivalente pourrait être :

gcc [options] programme.c entrees.o sorties.o

Ce qui signifie que gcc compilera programme.c séparément,
et le liera programme.o aux fichiers entrees.o et sorties.o existant.

Intérêt :

Au lieu de trois recompilations, une seule est nécessaire

Inconvénient majeur :

Il faut que la personne qui fait la compilation
se souvienne des dernières modifications, ce qui peut devenir complexe
quand le nombre de fichiers.o augmente

Idée : automatiser la recompilation (ou non) en utilisant
la date de dernière modification des fichiers.

Outil dédié à l'automatisation des tâches : make

Avantages :

- permettra de s'affranchir des oublis/erreurs
- minimisera le nombre d'opérations nécessaires
- vérification automatique des fichiers .o existant

Définitions essentielles :

Cible : l'action à réaliser, symbolisée d'ordinaire par le nom du fichier à générer
(ou un nom (pas d'obligation sur le nom) attribué à une tâche spécifique)

Dépendance : la relation qui unit deux cibles : la cible A dépend de la cible B
si la modification de B affecte aussi A

Exemple : fichier.o dépend de fichier.c , mais aussi de tous les fichiers dont il dépend,
y compris les fichiers d'en-tête. Ainsi, modifier un fichier d'en tête dont
dépend fichier.o obligera à recompiler fichier.o pour que celui-ci
soit à nouveau à jour.

À jour , **ou actualisé** :

Se dit d'un fichier plus récent que tous les fichiers dont il dépend.

Exemple : si file.c inclut juste stdio.h, alors file.o sera à jour si il est plus récent que file.c et stdio.h (qui a peu de chance d'être modifié sans prendre de risques)

Makefile :

Fichier décrivant la création d'une ou plusieurs cibles, dans lequel le programmeur énonce leurs dépendances et les règles qui président à la compilation correcte des fichiers requis.

Usage : chaque répertoire contient un Makefile distinct indiquant à l'outil make comment compiler le code local.

La version GNU de make cherchera par défaut (et dans l'ordre) dans le répertoire courant, le fichier :

- Makefile
- makefile

Remarque : on peut aussi utiliser tout autre fichier pouvant servir de makefile avec l'option : -f nom_du_makefile

Exemple : certains logiciels libres sont portables sur plusieurs OS,
et les auteurs on prévu un Makefile par OS/Architecture :

Makefile.linux , Makefile.darwin , Makefile.Windows ...etc

Sous Linux, on lancerait la commande :

make -f Makefile.linux

Pour lancer la compilation.

Exemple de makefile