Cours 2
Du code source au programme éxécutable.

1. Rappels

a. Hiérarchie des langages.

- Langage machine : format directement lisible par le processeur
- Assembleur : une instruction assembleur = une instruction en langage machine – mais c'est plus lisible.
- C, Cobol, Ada, Java : langages évolués. Une instruction C = un ensemble d'instructions en langage machine.

b. Traducteurs

Ce sont des programmes qui transforment un code à un niveau de la hiérarchie (code *source*) en un code à un niveau plus bas, plus près de la machine (code *cible*). Certaines erreurs ne sont compréhensibles que si l'on sait comment une instruction est traduite.

c. Programme et fichier

Un programme source (C, assembleur,...) est l'ensemble des instructions qui vont être traduites en un fichier en langage machine directement éxécutable par le S.E. **Un programme source peut être réparti dans plusieurs fichiers**. C'est en fait la meilleure méthode dès qu'il devient un peu gros. (En turbo C, un programme réparti sur plusieurs fichiers s'appelle un projet)

1. Les étapes de la traduction

Le S.E. utilise plusieurs étapes de traduction pour passer du code source en langage évolué à un programme exécutable. Les étapes successives sont :

- **Pré-compilation** (fichier par fichier) : traitement des macro-instructions. Le résultat est encore en langage source.
  #define TAILLE 30 // remplacement de toutes les occurrences de TAILLE par 30 (expansion)
  #define FOO TAILLE + 3 // FOO sera expansé par 30 + 3
  #define MIN(A, B) ((A) > (B)) ? (B) : (A) // MIN (x-y,x+y) sera expansé en ((x-y) > (x+y)) ? (x+y) : (x-y)
  #include <stdio.h> // le fichier stdio.h à l'emplacement des headers standards est copié ici

  #if VERSION = 4
  // code spécifique de la version 4
  #endif // obligatoire

  #ifdef MAINFILE
  // code à inclure dans le seul fichier principal
  #endif //obligatoire
- **Compilation** (fichier par fichier) : le source est traduit en assembleur.
  A cette étape, chaque variable reçoit un **emplacement mémoire** pour stocker sa valeur (**définition**). Une utilisation de la variable dans la suite du programme est une **référence**. Certaines références visent des variables ou des fonctions dont la définition se trouve dans un autre fichier source ou un fichier de bibliothèque : ce sont des **références externes**. Les déclarations dans les headers permettent de connaitre au moins la taile des références externes.
  La position du code assembleur dans le programme complet n'est pas connue, puisqu'on ne sait pas dans quel ordre les différents fichiers seront regroupés, ni quelle sera la taille des autres fichiers traduits. Les références qui ne sont pas encore définitivement fixées sont appelées **relogeables**.
- **Assemblage** (fichier par fichier) : traduction de l'assembleur en langage machine. Les références relogeables ou non résolues sont notées dans un tableau d'en-tête (tableau des références non résolues)
- **Édition de liens** (tous les fichiers ensemble) : un fichier objet d'initialisation est ajouté (récupération de la ligne de commande, de l'environnement, certaines routines ou variables de traitement d'erreur, etc.), de même que les fonctions de bibliothèque. Tous les fichiers objet sont rassemblés en un seul code. Il faut fournir la liste des bibliothèques utilisées (sauf la bibliothèque standard). La plupart des références sont

connues, sauf les **références absolues** (celles qui dépendent de l'adresse de chargement). Le résultat est un programme exécutable.
On peut aussi faire une édition de lien partielle : plusieurs fichiers objets sont rassemblés en un fichier objet plus gros, mais qui n'est pas encore éxécutable.

En turboC, les différentes étapes de création d'un exécutable sont accessibles par la fenêtre de projet, menu contextuel du fichier source (bouton droit)| edit Node Attributes, champ Translator.
Sous unix, ce sont des options :

cc –E foo.c (préprocesseur, génère foo.i),
cc –s foo.c (assemblage, génère foo.s),
cc –c foo.c ou cc –c foo.s (génère un fichier objet foo.o à partir d'un source c ou assembleur),
cc foo.c bar.s go.o –o bar (génère l'exécutable bar à partir de foo.c, bar.s et go.o)

Ms-Dos a la particularité de distinguer 2 types d'exécutables en langage machine. Les .exe suivent le schéma précédent, alors que les .com sont des programmes qui n'ont pas de référence absolue. Pour cela, on doit obligatoirement avoir cs=ds=es=ss, ce qui limite l'espace total disponible à 64 Ko.

2. Le portage sur différentes machines et les langages à machine virtuelle

Ce qui dépend de la machine ou du S.E. :

Le processeur définit le langage machine et les mnémoniques assembleur.
Le système définit le format des exécutables. En général, il fournit un éditeur de liens qui détermine les possibilités d'emploi des références en assembleur.
Le système détermine aussi la liste des appels système disponibles (fonctions utilisant le matériel, par ex. lecture clavier, disque).

Si deux compilateurs de langages différents traduisent de la même façon les noms de variable et de fonction et utilisent des méthodes d'appel de sous-programmes compatibles, on peut lier dans le même exécutable des fichiers objets générés par les deux compilateurs.

Pour disposer du même langage sur différentes machines ou différents systèmes, on doit à chaque fois réécrire le compilateur. En pratique, il y a souvent de petites différences, ce qui fait que le code passe parfois mal d'un compilateur à l'autre. Il y a deux solutions à ce problème : définir a priori le compilateur, et porter le même compilateur sur différentes machines. C'est le cas de gcc (compilateur C freeware) qui existe à l'identique sur la plupart des machines sous Unix (sauf la taille des types de base, qui dépend du processeur).

Si l'on veut la portabilité sur des systèmes très différents, on peut utiliser des machines virtuelles.
La technique des machines virtuelles est utilisée en particulier par Lisp (List Processor, un des plus anciens langages d'intelligence artificielle) et Java. On spécifie (décrit de façon abstraite) une machine avec sa taille de mot, ses registres, ses accès mémoire, son langage machine, son assembleur et son chargeur. Il faut alors écrire sur chaque type de machine réelle et chaque système d'exploitation un logiciel qui simule la machine virtuelle en respectant scrupuleusement ses spécifications. Tous les compilateurs génèrent le même code dans le langage de la machine virtuelle, quel que soit le système et le processeur – ce qui facilite l'écriture des compilateurs. Les programmes compilés sur une machine peuvent être exécutés sur une autre sans problème, ce qui est un avantage important. Par contre, la machine virtuelle constitue un intermédiaire supplémentaire lors de l'exécution, qui est moins rapide que celle d'un exécutable dans le langage de la machine réelle.

Par exemple, le JDK sous Dos comporte une machine virtuelle (java.exe), un compilateur/assembleur (javac.exe), un debogueur (jdb.exe), un désassembleur (javap.exe).