Architecture des Ordinateurs
Partie III : Liens avec le système d'exploitation
1. Modèles d'exécution

David Simplot
simplot@fil.univ-lille1.fr

Objectifs

- Faire le lien entre le matériel et ce que vous faîtes « au-dessus »
  - Système d'exploitation
  - Modèle d'exécution
    - Langage natif
    - Machine virtuelle
  - Compilateurs
  - Outils d'aide à la conception et de mise au point

Au sommaire...

- **Rôle d'un système d'exploitation**
- Chaîne de compilation
- Machines virtuelles

Rôle d'un Système d'Exploitation (1/3)

- Système d'Exploitation = *Operating System*
- Présenter au programmes une abstraction du matériel
  - Piloter un périphérique = très compliqué
    - Pilote = *Driver*
  - ⇒ le système d'exploitation propose une HAL
    - Hardware Abstraction Layer
    - Ex. fichiers, objets
- L'interface entre le système d'exploitation et les programmes de l'utilisateur est constitué d'un ensemble d'« instructions étendues » fournies par le système d'exploitation
  - Appels système

Rôle d'un Système d'Exploitation (2/3)

- Gestion des processus
  - Multi-tâches
    - Préemptif, non préemptif
  - Ordonnanceur de processus
  - Partage des ressources
  - Communications inter-processus
- Les entrées/sorties *(Voir Partie IV)*
  - Interruptions, DMA
  - « Bufferisation » des E/S
- Gestion de la mémoire *(Voir Partie IV)*
- Système de fichiers

Rôle d'un Système d'Exploitation (3/3)

- Chargement d'applications
  - Passer de l'état **« fichier exécutable »** à l'état **« programme qui s'exécute »** ☺
  - Le fichier contient une suite d'octets correspondant au programme en langage machine ainsi qu'aux données
    - En plus, on a des informations sur le programme
      - L'adresse de début du programme
      - La mémoire nécessaire
      - Les bibliothèques dynamiques utilisées
      - …
  - C'est le rôle du système d'exploitation d'effectuer le chargement
    - Copie du « code » en mémoire + liens + exécution

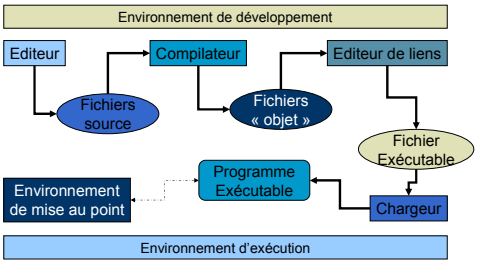
Au sommaire...

- Rôle d'un système d'exploitation
- **Chaîne de compilation**
- Machines virtuelles

Chaîne de Compilation (1/14)

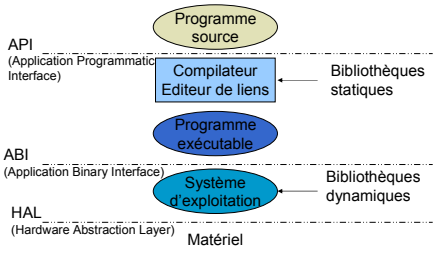
- Programme écrit en langage de « haut niveau »
  - Fortran, Pascal, Ada, Cobol, C, C++…
- Transformer le **« programme source »** en **« fichier exécutable »**, c'est le rôle de la compilation.
- On distingue deux phases :
  - Compilation
  - Edition de liens

Chaîne de Compilation (2/14)



Chaîne de Compilation (3/14)
Niveaux de compatibilité

- On distingue plusieurs niveaux de compatibilité



Chaîne de Compilation (4/14)
Niveaux de compatibilité (suite)

- Compatibilité source
  - Respect de l'API proposée
  - Repose généralement sur des normes
    - Ex. ANSI, POSIX, X-OPEN
  - Le portage d'un programme implique la recompilation des sources
  - Pb. de propriété industrielle
    - Open Source

Chaîne de Compilation (5/14)
Niveaux de compatibilité (suite)

- Compatibilité binaire niveau application
  - Programme sous forme de « binaires »
    - Programmes compilés dans un environnement déterminé sous forme chargeable
    - Plate-forme proposant l'interface ABI utilisée
  - Pb. pour les nouvelles architecture :
    - Coût de développement
    - ⇒ compatibilité binaire ascendante des plate-formes (matériel et logiciel)
  - Compatibilité binaire :
    - Architecture du processeur
    - Convention d'adressage et de communication (OS)
    - Interface avec l'OS et les bibliothèques (.so ou DLL)
    - Conventions de représentation des données (taille, LE ou BE)

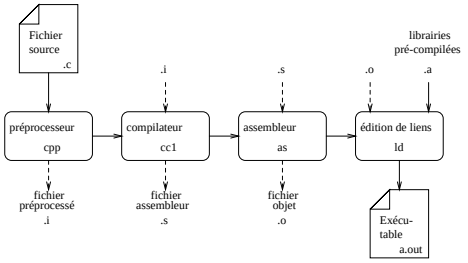
Chaîne de Compilation (6/14)
Niveaux de compatibilité (suite)

- Compatibilité binaire niveau OS
  - Portabilité des != OS sur != plateformes
    - On travaille sur l'interface OS/matériel
    - HAL = Hardware Abstraction Layer

Chaîne de Compilation (7/14)
Exemple GNU

- Chaîne de compilation du projet GNU
  - Le projet GNU a été lancé en 1984 afin de développer un système d'exploitation complet, semblable à Unix et qui soit un logiciel libre: le système GNU. (« GNU » est l'acronyme récursif the « **G**NU's **N**ot **U**nix »; on le prononce « gnou » avec un G audible) Des variantes du système d'exploitation GNU, basées sur le noyau « Linux », sont utilisées largement à présent; bien que ces systèmes soient communément appelés par le terme « Linux », ils le seraient plus exactement par « GNU/Linux ».

Chaîne de Compilation (8/14)
Exemple GNU (suite)



Chaîne de Compilation (9/14)
Exemple GNU (suite)

- Programme C très très simple :

```
#define MAX 2

int main(void)
{
   int a = MAX;

   a = a + 2;

   return 0;
}
```


Chaîne de Compilation (10/14)
Exemple GNU (suite)

- Première étage : préprocesseur

```
# 3 "ex.c"
int main(void)
{
   int a = 2;

   a = a + 2;

   return 0;
}
```


Chaîne de Compilation (11/14)
Exemple GNU (suite)

- Deuxième étape : génération de l'assembleur

```
   .file   "ex.c"
   .version       "01.01"
gcc2_compiled.:
.text
   .align 16
.globl main
   .type    main,@function
main:
   pushl   %ebp
   movl    %esp, %ebp
   subl    $4, %esp
   movl    $2, -4(%ebp)
   leal    -4(%ebp), %eax
```

Chaîne de Compilation (12/14)
Exemple GNU (suite)

```
    addl   $2, (%eax)
    movl   $0, %eax
    movl   %ebp, %esp
    popl   %ebp
    ret
.Lfe1:
    .size   main,.Lfe1-main
    .ident "GCC: (GNU) 2.96 20000731 (Linux-
    Mandrake 8.0 2.96-0.48mdk)"
```

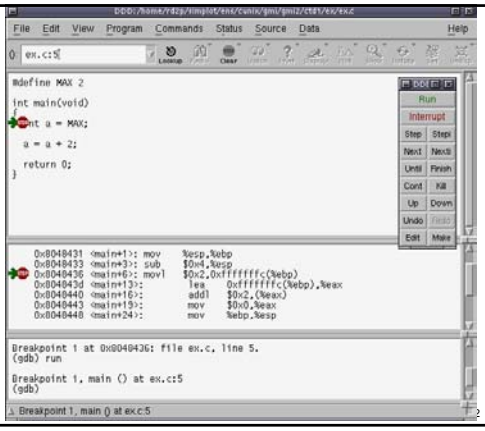

Chaîne de Compilation (13/14)
Exemple GNU (suite)

- Troisième étape : génération du .o

```
000000: 7f 45 4c 46 01 01 01 00   127 069 076 070 001 001 001 000  .ELF....
000008: 00 00 00 00 00 00 00 00   000 000 000 000 000 000 000 000  ........
000010: 01 00 03 00 01 00 00 00   001 000 003 000 001 000 000 000  ........
000018: 00 00 00 00 00 00 00 00   000 000 000 000 000 000 000 000  ........
000020: e8 00 00 00 00 00 00 00   232 000 000 000 000 000 000 000  è.......
000028: 34 00 00 00 00 00 28 00   052 000 000 000 000 000 040 000  4.....(.
000030: 09 00 06 00 00 00 00 00   009 000 006 000 000 000 000 000  ........
000038: 00 00 00 00 00 00 00 00   000 000 000 000 000 000 000 000  ........
000040: 55 89 e5 83 ec 04 c7 45   085 137 229 131 236 004 199 069  U.à.ì.ÇE
000048: fc 02 00 00 00 8d 45 fc   252 002 000 000 000 141 069 252  ü.....Eü
000050: 83 00 02 b8 00 00 00 00   131 000 002 184 000 000 000 000  ...¸....
000058: 89 ec 5d c3 08 00 00 00   137 236 093 195 008 000 000 000  .ì]Ã....
000060: 00 00 00 00 01 00 00 00   000 000 000 000 001 000 000 000  ........
000068: 30 31 2e 30 31 00 00 00   048 049 046 048 049 000 000 000  01.01...
000070: 00 47 43 43 3a 20 28 47   000 071 067 067 058 032 040 071  .GCC: (G
000078: 4e 55 29 20 32 2e 39 36   078 085 041 032 050 046 057 054  NU) 2.96
...
```


Chaîne de Compilation (14/14)
Exemple GNU (suite)

- Génération de l'exécutable
  - Prendre les fichiers .o
  - Déterminer le point d'entrée du programme
  - Faire les liens
    - Edition de liens = Link
- On peut mettre des informations permettant de débogguer le programme



Au sommaire...

- Rôle d'un système d'exploitation
- Chaîne de compilation
- **Machines virtuelles**

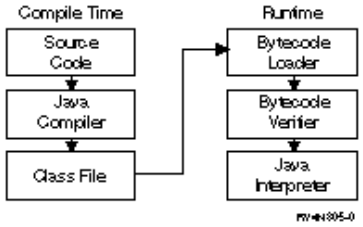
Machines Virtuelles (1/4)

- Objectif =
  - S'affranchir de la compabilité binaire !
  - On ne compile plus pour une plate-forme physique donnée, mais pour une Machine Virtuelle
- Langage Java
  - Développé par Sun Microsystem
  - S'inspire du C++ pour la syntaxe
  - S'inspire de Smalltalk pour la philosophie
  - Indépendance de la plate-forme matérielle
  - JVM = Java Virtual Machine

Write once, run everywhere

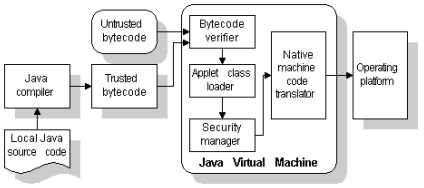
Machines Virtuelles (2/4)

- Chaîne de compilation et plate-forme d'exécution



Machines Virtuelles (3/4)

- Utilisation de code natif
  - JIT = Just-In-Time compiler



Machines Virtuelles (4/4)

- Java n'est pas le seul langage avec machine virtuelle :
  - Smalltalk
  - VisualBasic (génération de P-code)
  - Langages interprétés en général

Conclusion

- Reste à voir les techniques de génération de code