Architecture

Jean Goubault-Larrecq

LSV/CNRS UMR 8643 & ENS Cachan

Plan

1. Exemples, motivation.
2. Architecture matérielle des ordinateurs.
3. Les instructions du langage machine.
4. Efficacité, caches, MMU, et autres subtilités.
5. Architecture logicielle des ordinateurs (systèmes d'exploitation).

Qu’est-ce qu’un programme?

- Quelques exemples simples sous Unix.

  (C’est pareil sous les autres systèmes d’exploitation... mais parfois moins clair!)

- Un programme est:
  - ... un fichier source?
  - ... un fichier exécutable?
  - ... autre chose?

- Compilation, assemblage, édition de liens, chargement, exécution: quid?

Exemple: Le programme `cat` sous Unix

`cat` est un programme simple qui affiche (et concatène) les fichiers en argument.

```
bash$ cat a
Ceci est mon premier fichier.
bash$ cat b
Ceci est mon premier fichier.
bash$ cat a b
Ceci est mon premier fichier.
Ceci est mon second fichier.
```

Comment ça marche?

Le shell "bash" (d'invite `bash$`):

1. a lu la commande utilisateur `cat a b`;
2. a recherché un fichier `cat` dans le `PATH`;

   ```
   bash$ which cat
   /bin/cat
   ```
   ← c'est l'exécutable.
3. l'a chargé en mémoire;
4. a demandé au système d'exploitation (Unix) de l'exécuter, en lui passant "`a`" et "`b`" en arguments.

Regardons à quoi ressemble `/bin/cat/`

En le chargeant sous un éditeur de texte:

```
^?ELF^A^A^A^@^@^@^@^@^@^@^@^@^B^@^C^@^A^@^@^@\200\214^D^H4^@
^@^@1N^@^@^@^@^@^@4^@ ^@^E^@(^@^U^@^T^@^F^@^@^@4^@^@^@4\200^D
^H4\200^D^H^@^@^@^@^@^@^E^@^@^@^D^@^@^@^C^@^@^@^@^@^@\200
^D^H\200^D^H^S^@^@^@^S^@^@^@^D^@^@^@^A^@^@^@^A^@^@^@^@^@^@^@
^@\200^D^H^@\200^D^H?I^@^@?I^@^@^E^@^@^@^@^P^@^@^A^@^@^@@I^@
^@@^D^H@^D^H\214^A^@^@^B^@^@^F^@^@^@^@^P^@^@^B^@^@^@DJ^@^@
D^D^HD^D^H^@^@^@\210^@^@^@^F^@^@^@^D^@^@^@/lib/ld-linux.so.1
^@^@%^@^@^@@^@^@^@$^@^@^@+^@^@^@*^@^@^@,^@^@^@G^@^@^@^@^@^@^@
^S^@^@^@^H^@^@^@^@^@^@^@^T^@^@^@^O^@^@^@^C^@^@^@7^@^@^@^@^@^@
^@'^@^@^@)^@^@^@8^@^@^@^@^@^@^@=^@^@^@.^@^@^@9^@^@^@5^@^@^@^\
^@^@^@%^@^@^@<^@^@^@4^@^@^@]^@^@^@;^@^@^@^@^@^@3^@^@^@^_^@
^@^@:^@^@^@^Q^@^@^@ ^@^@^@6^@^@^@1^@^@^@2^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A
^@^@^@^F^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^E^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@      ^@^@^@^@^@^@^@^M^@^@^@^@^@^@^@^@^@^@
^@^N^@^@^@^@^@^@^@^X^@^@^@...
```

(On ne peut pas dire que ce soit clair...)

Regardons à quoi ressemble `/bin/cat/` (2)

Utilisons une autre baguette magique: le debugger `gdb`.

```
bash$ gdb /bin/cat
(gdb) x/11i main
```

```
0x8048530 <main>:     push %ebp              0x55
0x8048531 <main+1>:   mov  %esp,%ebp         0x89 0xe5
0x8048533 <main+3>:   sub  $0xc,%esp         0x83 0xec 0x0c
0x8048536 <main+6>:   push %edi              0x57
0x8048537 <main+7>:   push %esi              0x56
0x8048538 <main+8>:   push %ebx              0x53
0x8048539 <main+9>:   mov  0x8(%ebp),%edi    0x8b 0x7d 0x08
0x804853c <main+12>:  mov  $0x1,%ebx         0xbb 0x01 0x00 0x00 0x00
0x8048541 <main+17>:  cmp  %edi,%ebx         0x39 0xfb
0x8048543 <main+19>:  jge  0x66 <main+102>   0x7d 0x51
0x8048545 <main+21>:  add  $0xfffffff8,%esp  0x83 0xc4 0xf8
```

(En fait, j'ai triché: j'ai fait `gdb mycato`, où `mycato` est un programme qui fait pareil que `cat`, et qu'on reverra plus tard...)

C’est quoi tous ces trucs bizarres?

– `0x8048541` est une adresse, écrite en hexadécimal (en base 16, avec les chiffres 0123456789abcdef; `0x` signifie que le reste est écrit en hexa).

– `<main+17>:` est une indication donnée par `gdb` (voir plus tard).

– `cmp %edi, %ebx` est un opcode (code d’opération; ici “`compapre` value in registers `%edi` and `%ebx`).

– `0x39 0xfb` sont les deux octets stockés aux adresses `0x8048541` et `0x8048542`;
ces deux octets signifient `cmp %edi, %ebx` pour le processeur Pentium.