Maison du Libre

Les petits hackers

Atelier Unix/Linux (culture générale)

Philosophie Unix :
Avant de parler de philosophie, voici l'histoire d'Unix, dans ce schéma :
http://upload.wikimedia.org/wikipedia/commons/7/77/Unix_history-simple.svg

La philosophie Unix est de faire des applications simples (pas simplistes) :
http://fr.wikipedia.org/wiki/Philosophie_d%27Unix

Qu'est ce que Unix ?
http://fr.wikipedia.org/wiki/Unix

et sa particularité :
http://fr.wikipedia.org/wiki/Unix#Particularit.C3.A9s_des_syst.C3.A8mes_Unix
notamment celle de considérer comme fichier un bon nombre d'objets du système (entrée, sortie).

Le système de fichiers est un point important et demande une « standardisation » :
http://fr.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

=> déplacez vous dans votre système en utilisant le gestionnaire de fichier ou par un terminal, utilisez dans un shell (Bash…) les commandes **cd** et **ls** pour explorer les différents répertoires et fichiers qui existent.

Démarrage de Linux

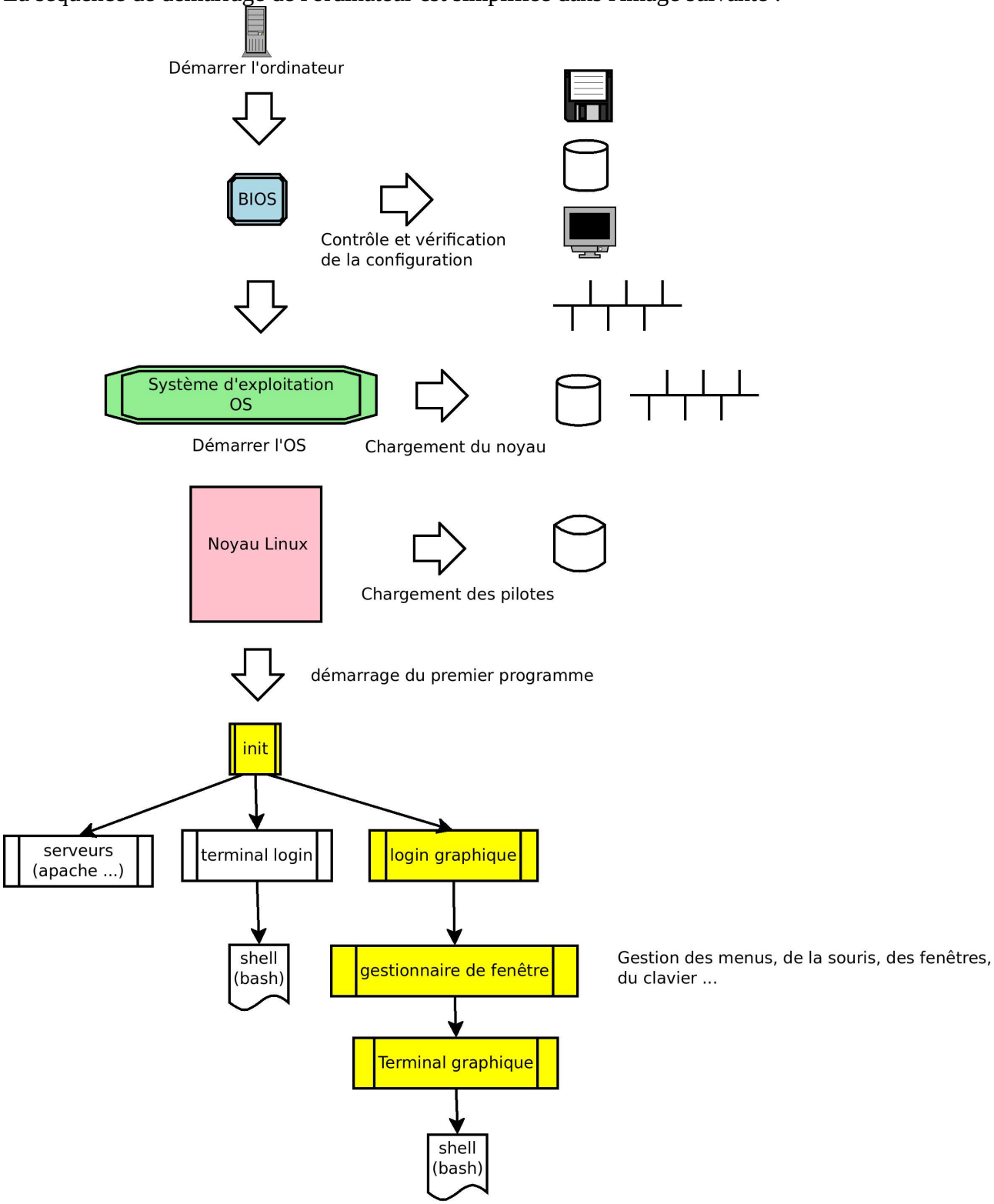
Démarrage d'un ordinateur

Votre PC est constitué d'une carte mère :
http://fr.wikipedia.org/wiki/Carte_m%C3%A8re

Sur cette carte mère, il existe une puce appelée BIOS (c'est un logiciel ou firmware:
http://fr.wikipedia.org/wiki/Firmware) :
http://fr.wikipedia.org/wiki/Basic_Input_Output_System
ce logiciel est démarré au lancement du PC et permet de découvrir, de configurer et de tester les éléments matériels de la carte mère avant de laisser la main à un système d'exploitation en lui permettant de démarrer à partir d'un disque dur, clé USB, CDROM, réseau.

Le BIOS est aujourd'hui en cours de remplacement par son successeur UEFI :
http://fr.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface
Il existe des BIOS open source :
http://fr.wikipedia.org/wiki/OpenBIOS
Si vous voulez étudier le fonctionnement d'un BIOS « open source » :
http://www.openfirmware.info/Welcome_to_OpenBIOS ou
http://www.coreboot.org/
Téléchargez OpenBIOS et explorez les différents répertoires et fichiers (ça ressemble un peu à un noyau Linux)

La séquence de démarrage de l'ordinateur est simplifiée dans l'image suivante :

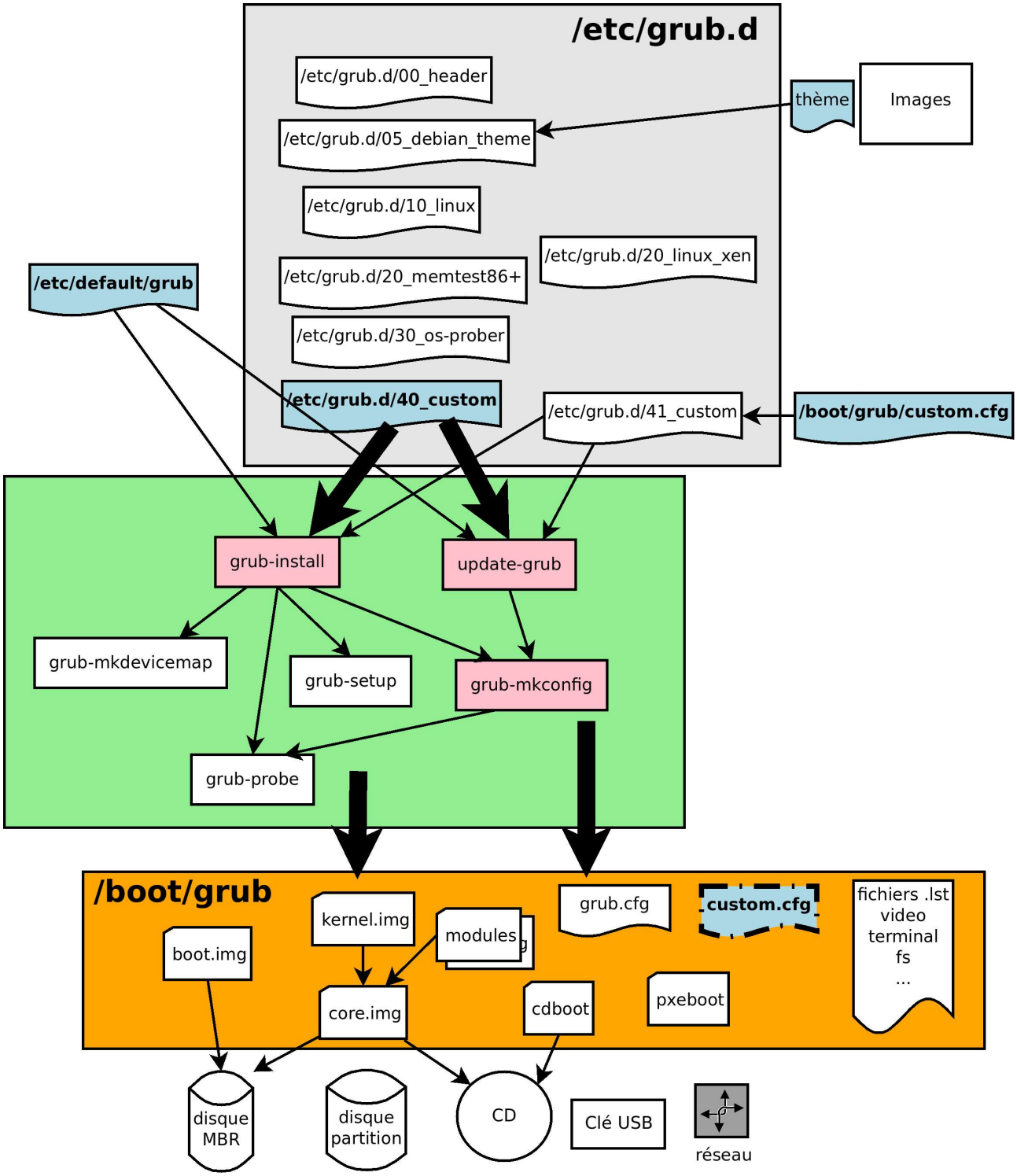


La boîte verte dans le schéma correspond à un choix de périphériques (disque dur, clé USB, CDROM, réseau) sur lequel il faudra aller chercher les instructions pour charger le noyau du système d'exploitation. Cette phase et ces instructions sont souvent regroupés dans une application appelée « chargeur » (bootloader) :
http://fr.wikipedia.org/wiki/Chargeur_d%27amor%C3%A7age
Il existe des chargeurs « open source » (libre pour les puristes) comme les plus connus :

1. GRUB : http://fr.wikipedia.org/wiki/GNU_GRUB
2. LiLo

3. IsoLinux de SysLinux

Voici comment fonctionne Grub2 :



Grub2 est composé de plusieurs fichiers de configuration, dans le cadre gris en haut, et des outils dans le cadre vert qui compilent et configurent tous les paramètres qui seront dans le répertoire /boot/grub (cadre orange) et préparent le support de démarrage (disque dur, clé USB…) : le bout de code qui chargera GRUB pour afficher un menu de présentation et chargera le noyau sélectionné.

Une fois le noyau sélectionné (noyau Linux de préférence), celui-ci est chargé en mémoire, décompressé (si nécessaire) et prend la main sur la machine (il est chargé à partir du répertoire **/boot** de votre système installé).

Suivant la façon dont le noyau Linux a été compilé, il ne connaît pas tous les périphériques présents

sur la machine, ni les formats de systèmes de fichiers par exemple. Pour cela, le noyau va charger en mémoire une image contenant les pilotes de périphériques qui ne sont pas intégrés au noyau appelé **initrd** (init ram disk : http://fr.wikipedia.org/wiki/Initrd):
http://manpages.ubuntu.com/manpages/lucid/fr/man4/initrd.4.html
Cette image est chargée le temps du démarrage, avant lancer **/sbin/init**, voir premier schéma.
Le chargement en mémoire se fait en utilisant un RAM disque (http://fr.wikipedia.org/wiki/RAM_disque),
Exercice: pour créer un RAM disque sous linux : http://doc.ubuntu-fr.org/tmpfs

Pour voir le contenu du fichier initrd :
$ cp /boot/initrd.img-3.13.0-43-generic ~/travail/initrd/
$ file initrd.img-3.13.0-43-generic
initrd.img-3.13.0-43-generic: gzip compressed data, from Unix
$ gunzip < initrd.img-3.13.0-43-generic > initrd.img-3.13.0-43
$ file initrd.img-3.13.0-43
initrd.img-3.13.0-43 : ASCII cpio archive (SVR4 with no CRC)
$ mkdir init
$ cd init
$ cpio -iv < ../initrd.img-3.13.0-43
[…]
$ ls -l

Le fichier **init** est le fichier qui est démarré. Explorez les répertoires et fichiers.

Ce fichier **initrd** permet ensuite au noyau Linux de pouvoir avoir accès à la partition, au système de fichiers et à l'application **/sbin/init** qui sera exécuté pour démarrer le système d'exploitation :
http://fr.wikipedia.org/wiki/Init

Compilation noyau Linux

Pourquoi compiler son noyau, “Source Debian: Manuel d'installation pour la distribution Debian GNU/Linux” :

1. gérer des périphériques spéciaux, ou des conflits de périphériques dans les noyaux par défaut ;
2. activer des options qui ne sont pas incluses dans le noyau par défaut, permettre la gestion de la mémoire haute par exemple ;
3. optimiser le noyau en enlevant les pilotes inutiles, ce qui peut accélérer le démarrage de la machine ;
4. créer un noyau monolithique à la place d'un noyau modulaire
5. utiliser une version de développement du noyau ;
6. mieux connaître le noyau Linux

Télécharger/installer les sources
apt-get install linux-source-3.x.y
cd /usr/src/linux-source-3.x.y/linux-source-3.x.y

Installer les outils pour compiler :
apt-get install fakeroot debconf-utils dpkg-dev debhelper build-essential kernel-package libncurses5-dev

Configuration

Si vous voulez récupérer la configuration de votre noyau actuel pour compiler un nouveau noyau avec les mêmes caractéristiques, il suffit de copier le fichier de configuration de votre noyau qui se trouve dans /boot et de le mettre dans le répertoire des sources linux sous forme **.config** :
cp /boot/config-3.13.0-43-generic .config
C'est un fichier caché (n'oubliez pas le point) **.config**

Modifier le fichier .config
make menuconfig
Compiler
make && make modules_install

Cela créer un fichier image appelé bzImage :
http://en.wikipedia.org/wiki/Vmlinux

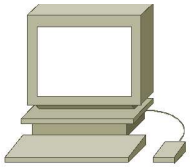
Ne pas installer le noyau pour l'instant, découvrez et explorez les répertoires et les fichiers en se basant sur l'architecture décrite dans les liens suivants :
http://fr.wikipedia.org/wiki/Noyau_Linux (utilisez la carte du noyau :
http://fr.wikipedia.org/wiki/Noyau_Linux#mediaviewer/File:Linux_kernel_map.png)
http://fr.wikipedia.org/wiki/Documentation_du_noyau_Linux
http://en.wikipedia.org/wiki/Linux_kernel#Architecture

Interface

Interface texte

Le terminal informatique (à voir plus tard)
http://fr.wikipedia.org/wiki/Terminal_informatique

Interface graphique



matériel

X-Window (X11)
MIR
Wayland

Système de fenêtrage

Interface avec le matériel
Fournit les biblothèques de base pour dessiner des points/traits ...

X11, GTK, QT, OpenGL/Clutter ...

Metacity basé sur GTK pour Gnome
Mutter, basé sur OpenGL
Kwin, basé sur QT pour KDE
OpenBox basé sur GTK pour LXDE
Window Maker
AfterStep
Enlightenment
...

Gestionnaire de fenêtres
(Window Manager)

Gestion:
- barre des tâches
- menus
- bureaux virtuels
- déplacement des fenêtres
- icônes
- modification du fond d'écran
- ...

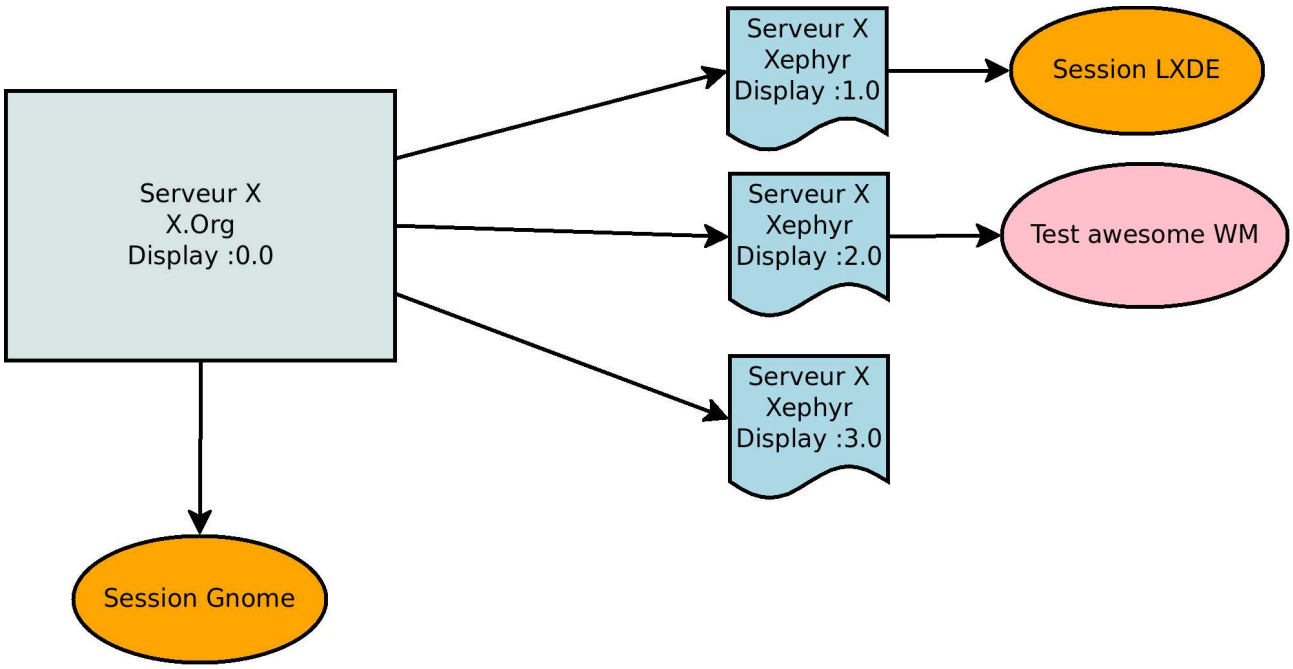
Gnome
KDE
LXDE
Unity
Enlightenment
Xfce
GnuStep
twm

Environnement Graphique
Environnement de Bureau

Fournit des applications ergonomiquement et graphiquement cohérentes dans différentes langues (look and feel) pour l'utilisateur final

L'interface graphiques est basée sur plusieurs couches :
- le système de fenêtrage, première couche : http://fr.wikipedia.org/wiki/Syst%C3%A8me_de_fen%C3%Aatrage, le plus connu sous Linux est X-Window (X11) : http://fr.wikipedia.org/wiki/X_Window_System
- le gestionnaire de fenêtre : http://fr.wikipedia.org/wiki/Gestionnaire_de_fen%C3%Aatres,
- et enfin l'environnement de bureau ou graphique : http://fr.wikipedia.org/wiki/Environnement_de_bureau, les plus connus sont KDE (http://fr.wikipedia.org/wiki/KDE)et GNOME (http://fr.wikipedia.org/wiki/GNOME)

Pour tester les différents gestionnaires (ou développer un gestionnaire de fenêtre), il est pratique d'utiliser un outil qui va ouvrir un serveur graphique (serveur X imbriqué) dans le serveur graphique courant (serveur X) :



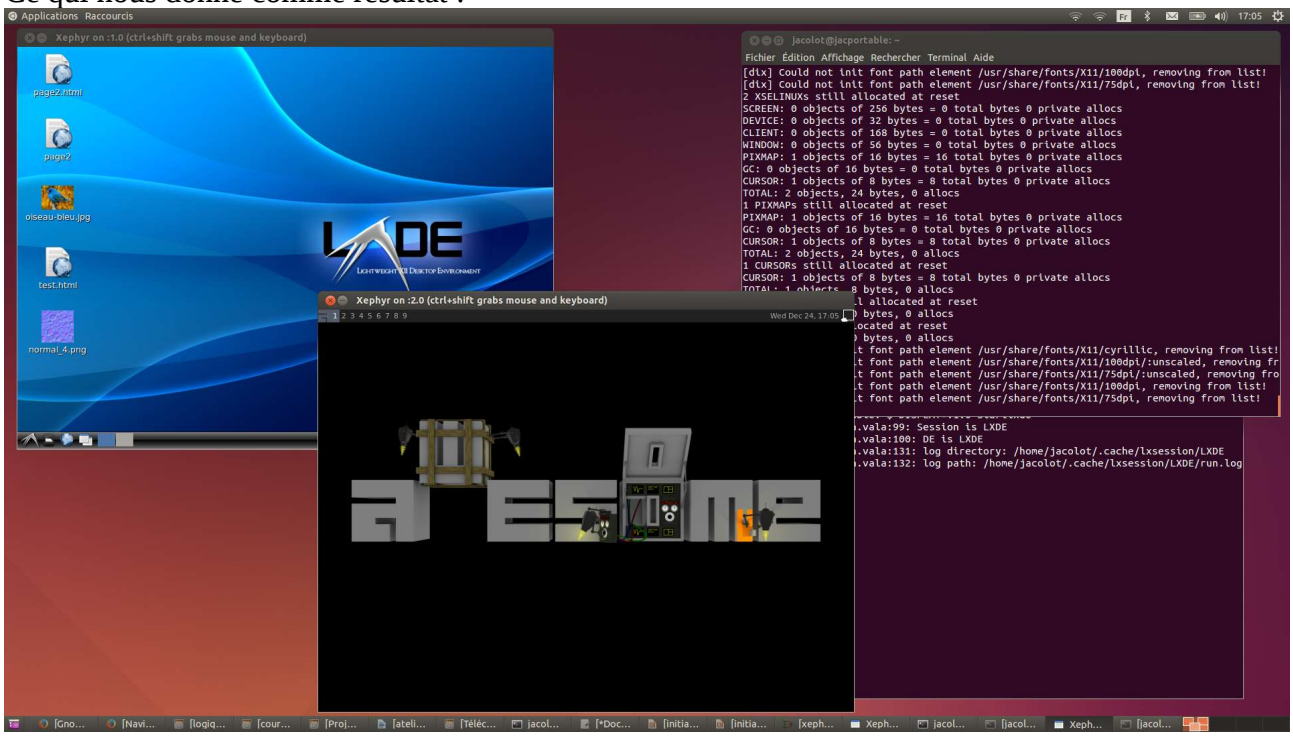
Installation de Xephyr :
$ sudo apt-get install xserver-xephyr
Installation d'un gestionnaire de fenêtre (Window Manager) :
$ sudo apt-get install awesome
ou de :
$ sudo apt-get install lxde

Démarrage de Xephyr à partir d'une console :
$ Xephyr -ac -screen 800x600 :1

A partir d'une autre console :
$ DISPLAY=:1.0 startlxde

$ Xephyr -ac -screen 800x600 :2
$ DISPLAY=:2.0 awesome

Ce qui nous donne comme résultat :



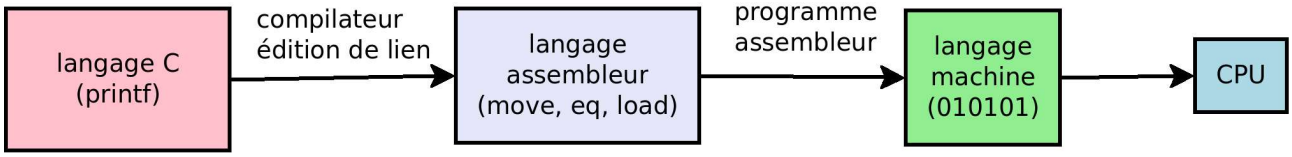
plus d'information sur :
http://en.wikipedia.org/wiki/Xephyr
http://fr.wikipedia.org/wiki/LXDE
http://fr.wikipedia.org/wiki/Awesome_%28logiciel%29

Liste des gestionnaires de fenêtres (à tester, window manager en anglais) :
https://wiki.debian.org/fr/WindowManager

Depuis plusieurs années maintenant les environnements graphiques sont « standardisés » dans le sens où les applications installées dans les menus sont disponibles et utilisables par n'importe quel gestionnaire de fenêtres qui supportent ce standard :
Desktop Group (XDG) (projet Freedesktop.org) : favoriser la collaboration entre les différents environnements de bureau libre: http://fr.wikipedia.org/wiki/Freedesktop.org

Compilateur et interpréteur

Compilation



Voir les liens suivants pour les définitions (pas besoin de détail pour l'instant) :
http://fr.wikipedia.org/wiki/Compilateur
http://fr.wikipedia.org/wiki/%C3%89dition_de_liens
http://fr.wikipedia.org/wiki/Assembleur

Exemple
Programme en C (le langage C : http://fr.wikipedia.org/wiki/C_%28langage%29)
$ vi hello.c
#include<stdio.h> // inclure fichier pour l'utilisation de printf

```
// fonction calcul
int calcul(int a, int b) {
        return a+b;
}

// point d'entrée du programme
void main() {
        int result = calcul(1,2);
        printf("Hello\nResultat: %d\n", result);
}
```

Compilation et édition de lien
$ gcc -o hello hello.c
Type du fichier exécutable (http://fr.wikipedia.org/wiki/Fichier_ex%C3%A9cutable)
$ file hello
Exécution du programme
$ ./hello

Voir le contenu en assembleur

pour pouvoir lire les instructions : http://fr.wikipedia.org/wiki/Jeu_d%27instructions_x86, http://fr.wikipedia.org/wiki/Liste_de_programme_Hello_world#Assembleur_x86.2C_sous_Linux.2C_.C3.A9crit_pour_l.27assembleur_NASM

```
$ gcc -S hello.c
$ more hello.s
        .file   "hello.c"
        .text
        .globl  calcul
        .type   calcul, @function
calcul:
.LFB0:
        .cfi_startproc
        pushq  %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq   %rsp, %rbp
        .cfi_def_cfa_register 6
        movl   %edi, -4(%rbp)
        movl   %esi, -8(%rbp)
        movl   -8(%rbp), %eax
        movl   -4(%rbp), %edx
        addl   %edx, %eax
        popq   %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   calcul, .-calcul
        .section        .rodata
.LC0:
        .string  "Hello\nResultat: %d\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB1:
        .cfi_startproc
        pushq  %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq   %rsp, %rbp
        .cfi_def_cfa_register 6
        subq   $16, %rsp
        movl   $2, %esi
        movl   $1, %edi
        call    calcul
        movl   %eax, -4(%rbp)
        movl   -4(%rbp), %eax
        movl   %eax, %esi
        movl   $.LC0, %edi
        movl   $0, %eax
```

```
        call    printf
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 4.8.2-19ubuntu1) 4.8.2"
        .section        .note.GNU-stack,"",@progbits
```

Exemple arduino avec Blink:
Voici le fichier en langage machine qui sera chargé sur l'arduino pour faire clignoter la LED (difficile à lire!) :

```
:100000000C945C000C9479000C9479000C947900A9
:100010000C9479000C9479000C9479000C9479007C
:100020000C9479000C9479000C9479000C9479006C
:100030000C9479000C9479000C9479000C9479005C
:100040000C949A000C9479000C9479000C9479002B
:100050000C9479000C9479000C9479000C9479003C
:100060000C9479000C947900000000070002010054
:1000700000030406000000000000000000102040864
:1000800010204080010204081020010204081020002
:10009000040404040404040202020202020203032E
:1000A000030303030000000250028002B000000CC
:1000B0000000240027002A0011241FBECFEFD8E043
:1000C000DEBFCDBF11E0A0E0B1E0EEE1F4E002C0A0
:1000D00005900D92A230B107D9F711E0A2E0B1E08E
:1000E00001C01D92AB30B107E1F70E9400020C94F1
:1000F0000D020C94000061E0809100010C949101CC
:10010000CF93DF93C0E0D1E061E088810E94CA0113
:1001100068EE73E080E090E00E94070160E0888173
:100120000E94CA0168EE73E080E090E0DF91CF9119
:100130000C9407011F920F920FB60F9211242F9368
:100140003F938F939F93AF93BF93809103019091BF
:100150000401A0910501B09106013091020123E054
:10016000230F2D3720F40196A11DB11D05C026E8EF
:10017000230F0296A11DB11D20930201809303015C
:100180090930401A0930501B093060180910701AB
:100190090910801A0910901B0910A010196A11D59
:1001A000B11D8093070190930801A0930901B093BA
:1001B0000A01BF91AF919F918F913F912F910F9025
:1001C0000FBE0F901F9018953FB7F89480910701CC
:1001D00090910801A0910901B0910A0126B5A89B50
:1001E00005C02F3F19F00196A11DB11D3FBF662725
:1001F000782F892F9A2F620F711D811D911D42E06A
:10020000660F771F881F991F4A95D1F70895CF92DF
:10021000DF92EF92FF92CF93DF936B017C010E94FC
:10022000E400EB01C114D104E104F10479F00E946F
:10023000E4006C1B7D0B683E7340A0F381E0C81A9C
:10024000D108E108F108C851DC4FECCFDF91CF9124
:10025000FF90EF90DF90CF900895789484B58260FE
:1002600084BD84B5816084BD85B5826085BD85B55A
```

```
:10027000816085BDEEE6F0E0808181608083E1E809
:10028000F0E0108280818260808380818160808341
:10029000E0E8F0E0808181608083E1EBF0E0808144
:1002A00084608083E0EBF0E0808181608083EAE716
:1002B000F0E080818460808380818260808380819F
:1002C000816080838081806880831092C10008955E
:1002D000833081F028F4813099F08230A1F00895C4
:1002E0008630A9F08730B9F08430D1F48091800055
:1002F0008F7D03C0809180008F7780938000089568
:1003000084B58F7702C084B58F7D84BD08958091B8
:10031000B0008F7703C08091B0008F7D8093B000D4
:100320000895CF93DF9390E0FC01E458FF4F2491B0
:10033000FC01E057FF4F8491882349F190E0880F3A
:10034000991FFC01E255FF4FA591B4918C559F4F29
:10035000FC01C591D4919FB7611108C0F8948C91AC
:10036000209582238C93888182230AC0623051F4C5
:10037000F8948C91322F309583238C938881822B33
:10038000888304C0F8948C91822B8C939FBFDF915B
:10039000CF9108950F931F93CF93DF931F92CDB703
:1003A000DEB7282F30E0F901E859FF4F8491F901B9
:1003B000E458FF4F1491F901E057FF4F04910023D7
:1003C000C9F0882321F069830E9468016981E02FC8
:1003D000F0E0EE0FFF1FEC55FF4FA591B4919FB7D2
:1003E000F8948C91611103C01095812301C0812B79
:1003F0008C939FBF0F90DF91CF911F910F91089524
:100400000E942D010E947B00C0E0D0E00E9480008D
:0E0410002097E1F30E940000F9CFF894FFCF8F
:02041E000D00CF
:00000001FF
```

et voici un extrait du programme Blink en assembleur (remarquez les méthodes setup et loop), l'assembleur est différent de celui du processeur x86 vu ci-dessus (http://fr.wikipedia.org/wiki/Atmel_AVR, les instructions du langage d'assemblage http://www.atmel.com/images/doc0856.pdf) :

```
[...]
000000f6 <setup>:
  f6:   61 e0           ldi     r22, 0x01       ; 1
  f8:   80 91 00 01     lds     r24, 0x0100
  fc:   0c 94 91 01     jmp     0x322  ; 0x322 <pinMode>

00000100 <loop>:
 100:   cf 93           push    r28
 102:   df 93           push    r29
 104:   c0 e0           ldi     r28, 0x00       ; 0
 106:   d1 e0           ldi     r29, 0x01       ; 1
 108:   61 e0           ldi     r22, 0x01       ; 1
 10a:   88 81           ld      r24, Y
 10c:   0e 94 ca 01     call    0x394  ; 0x394 <digitalWrite>
 110:   68 ee           ldi     r22, 0xE8       ; 232
 112:   73 e0           ldi     r23, 0x03       ; 3
 114:   80 e0           ldi     r24, 0x00       ; 0
 116:   90 e0           ldi     r25, 0x00       ; 0
```

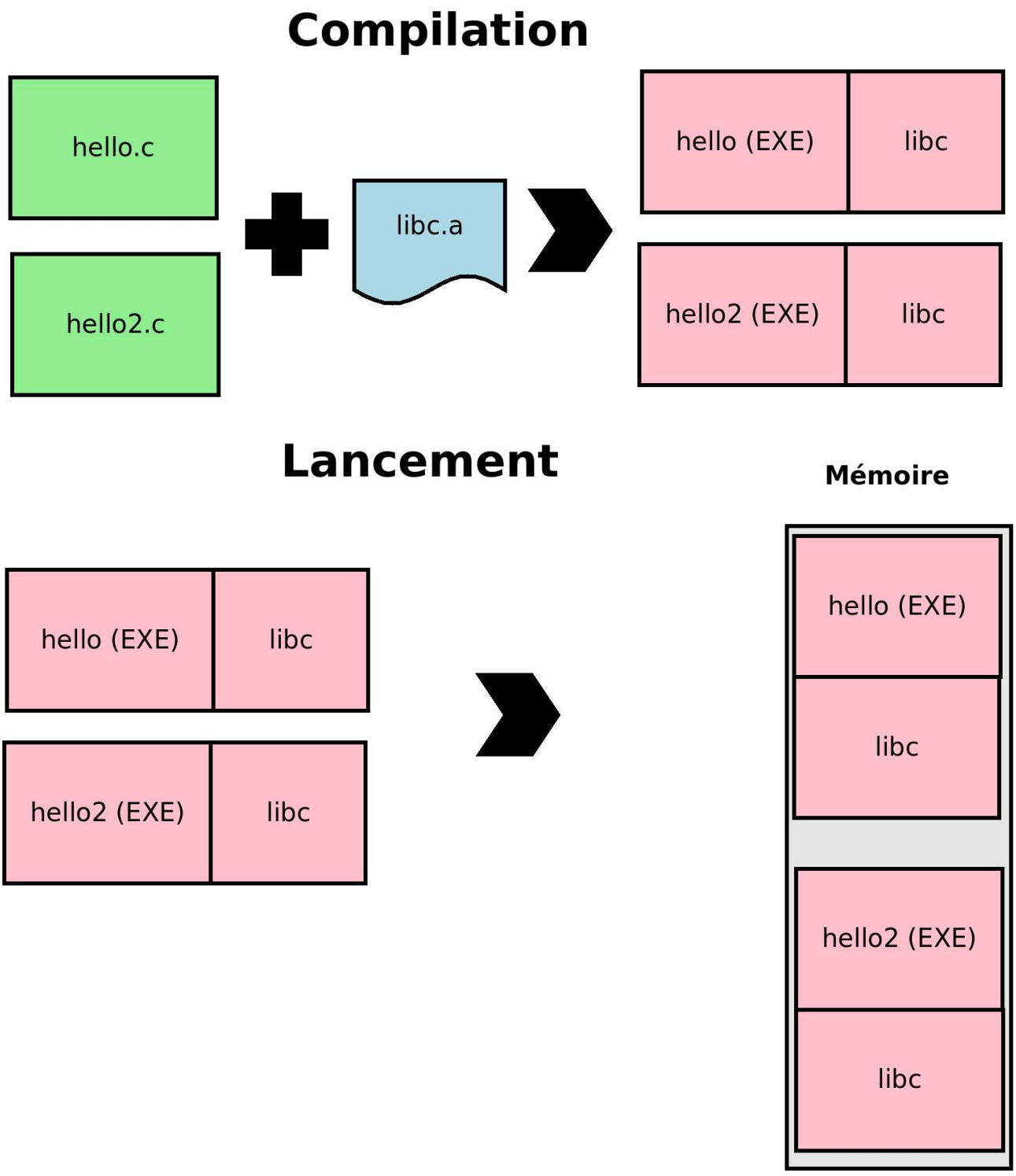
```
 118:   0e 94 07 01    call    0x20e  ; 0x20e <delay>
 11c:   60 e0          ldi     r22, 0x00       ; 0
 11e:   88 81          ld      r24, Y
 120:   0e 94 ca 01    call    0x394  ; 0x394 <digitalWrite>
 124:   68 ee          ldi     r22, 0xE8       ; 232
 126:   73 e0          ldi     r23, 0x03       ; 3
 128:   80 e0          ldi     r24, 0x00       ; 0
 12a:   90 e0          ldi     r25, 0x00       ; 0
 12c:   df 91          pop     r29
 12e:   cf 91          pop     r28
 130:   0c 94 07 01    jmp     0x20e  ; 0x20e <delay>
[...]
```

Sous Linux, une fois que le programme exécutable est prêt, pour le démarrer, il faut le charger en mémoire, et lire les instructions :
$ file hello
Le fichier exécutable « dynamically linked (uses shared libs) »
$ ldd hello

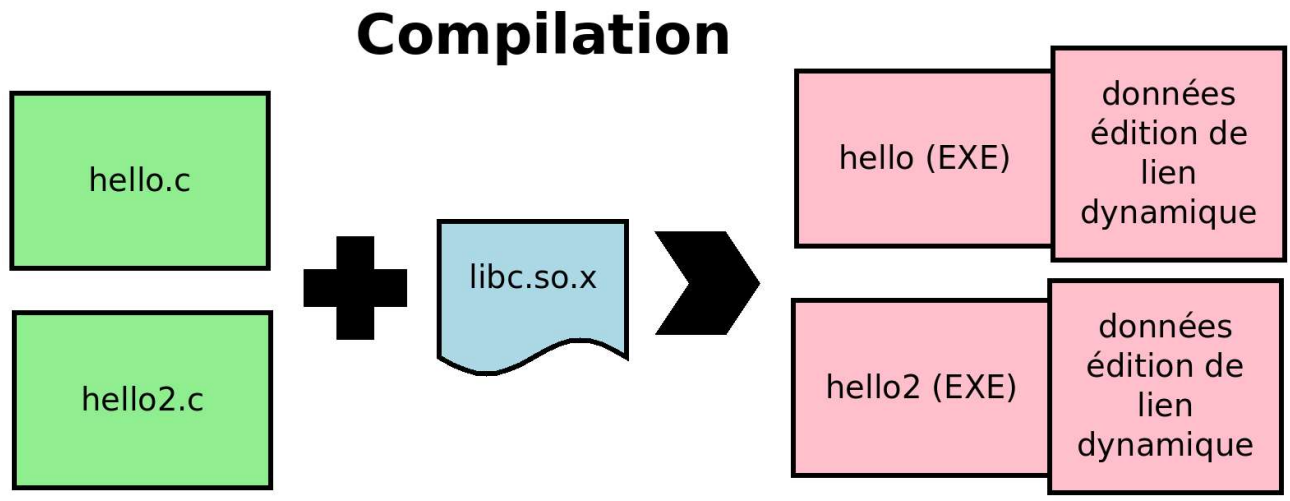
```
        linux-vdso.so.1 =>  (0x00007fffa12ab000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f7b5ba43000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f7b5be29000)
```

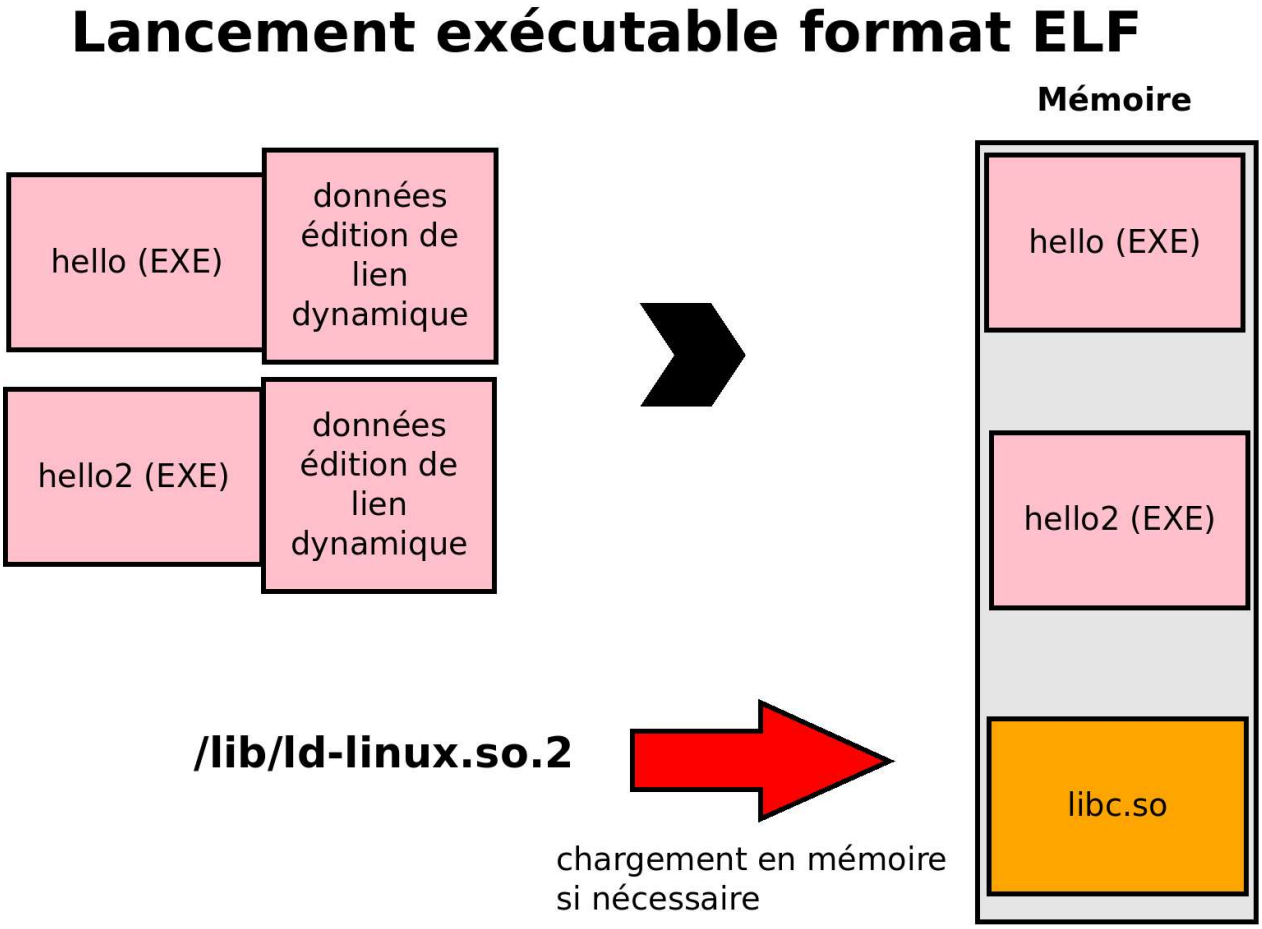
Qu'est ce que les bibliothèques partagées ?
http://fr.wikipedia.org/wiki/Biblioth%C3%A8que_logicielle#La_technique

Par défaut, sous Linux, un exécutable utilise la technique des bibliothèques partagées.
Mais pour tester la méthode dite « statique », voici les options du compilateur pour produire un fichier exécutable qui contient toutes les bibliothèques dont il a besoin pour fonctionner :
$ gcc -static -o hello2 hello.c
Le fichier exécutable **hello2** est plus gros que le fichier **hello**
$ file hello2
… statically linked …
Sur l'image ci-dessous, le résultat du chargement en mémoire des programmes statiques :

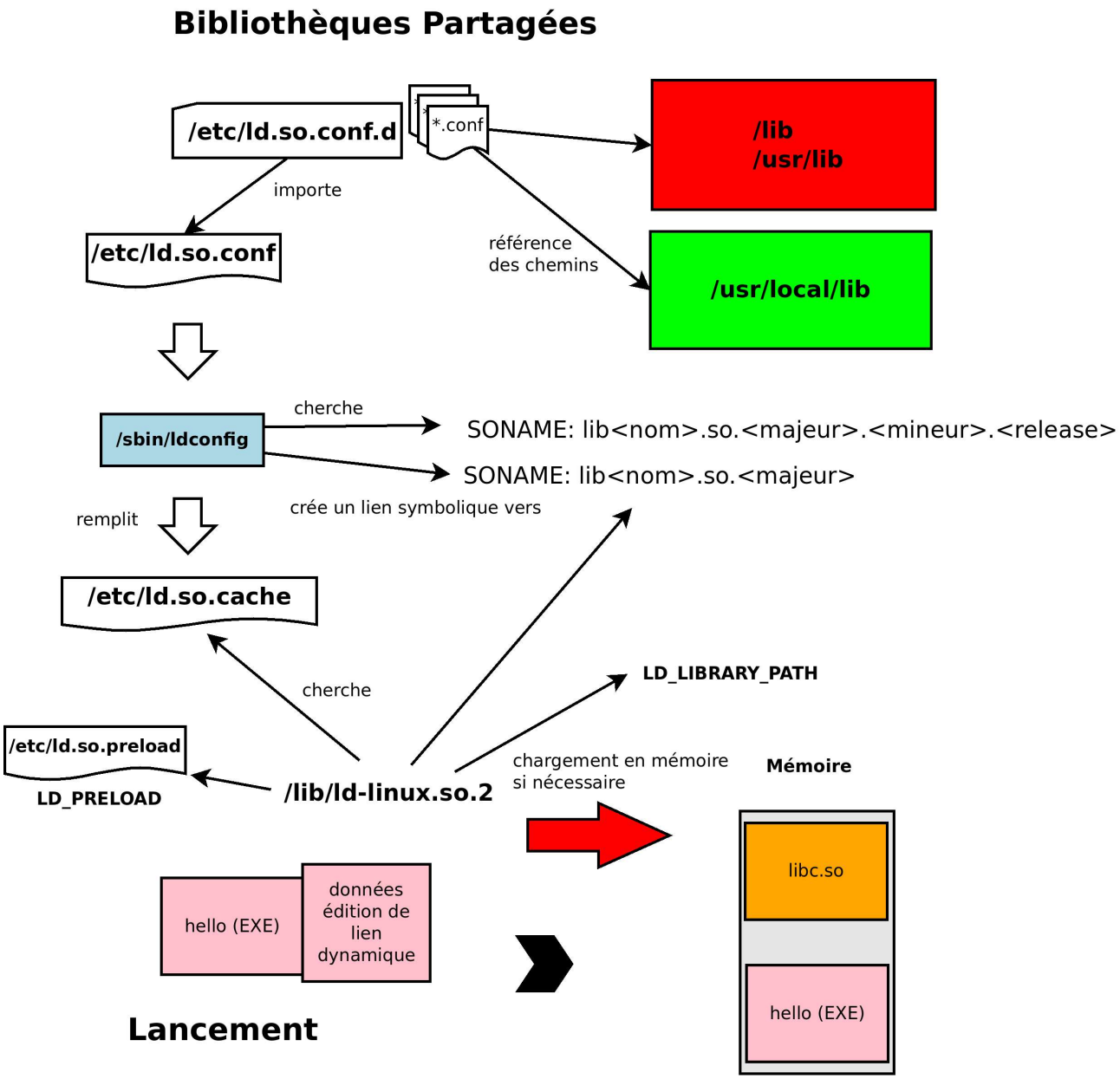


Les bibliothèques dynamiques permettent un gain de place, voir ci-dessous :





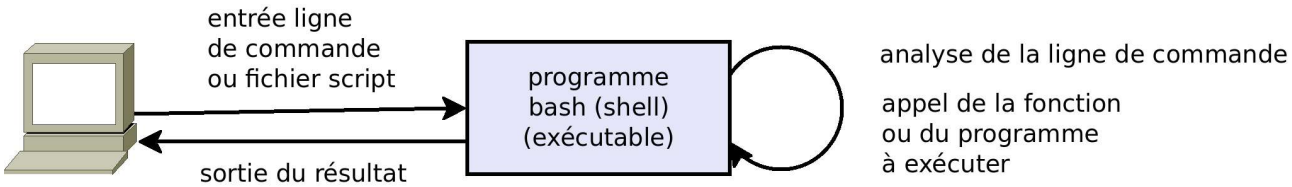
Pour cela, il faut quelques fichiers de configuration et un chargeur : le chargeur est aussi un programme qui va lire le fichier exécutable en mémoire, faire le lien avec la bibliothèque dynamique (soit déjà en mémoire ou soit à charger en mémoire) et lancer l'exécution du programme : http://fr.wikipedia.org/wiki/Chargeur_%28informatique%29
Voir image ci-dessous



Interpréteur

http://fr.wikipedia.org/wiki/Interpr%C3%A8te_%28informatique%29

Exemple avec un shell (http://fr.wikipedia.org/wiki/Interface_syst%C3%A8me),
le programme Bash (http://fr.wikipedia.org/wiki/Bourne-Again_shell)



Le shell prend ses commandes sur l'entrée standard (dans le mode de fonctionnement en ligne de commande). Le programme analyse les caractères et les mots de cette entrée et ensuite en fonction du résultat, il va soit afficher une erreur (l'entrée n'est pas correcte, fonction inconnue, ou programme introuvable), ou exécuter la fonction ou le programme.

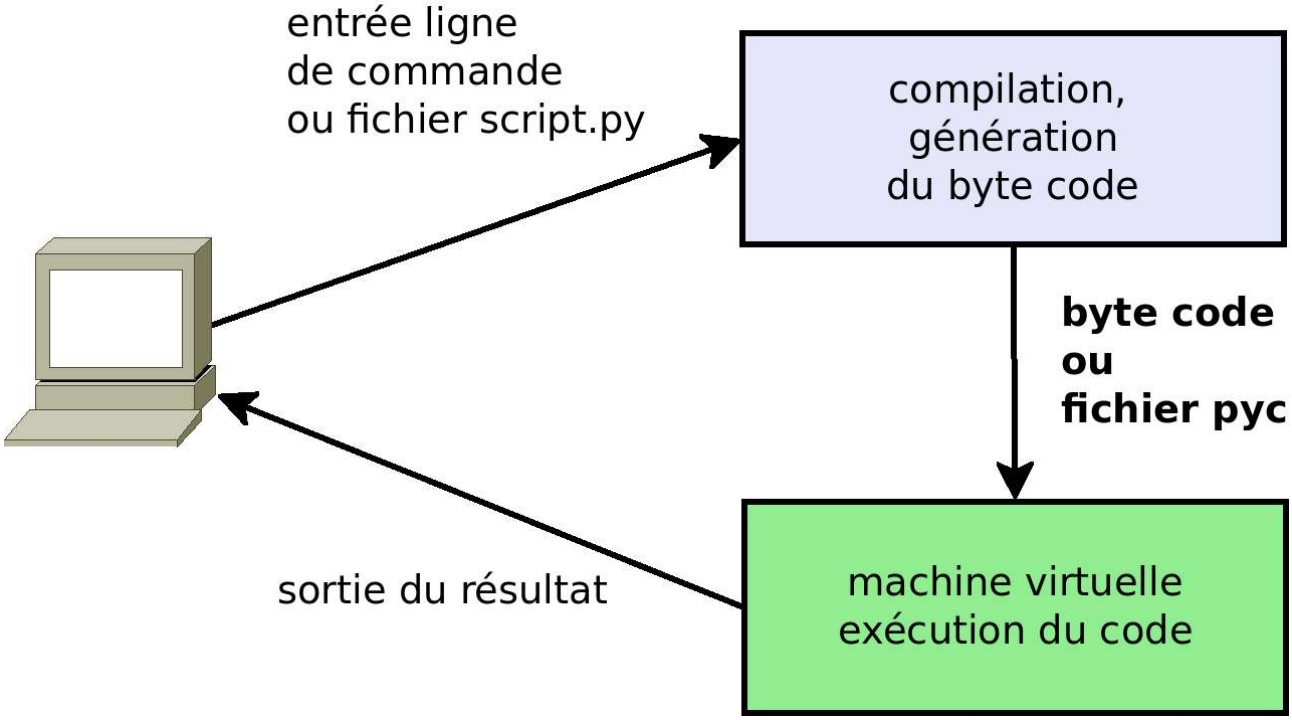
Le shell contient des fonctions internes (appelées builtins) qui sont exécutées par le shell. Les autres commandes sont des programmes du système d'exploitation.
Pour faire la différence (ou presque) entre les deux, il faut utiliser la commande type :
$ **type** cd
cd est une primitive du shell
$ **which** cd
# rien, pas de résultat

Attention, toutefois, il existe des fonctions qui sont aussi des commandes :
$ type pwd
pwd est une primitive du shell
$ which pwd
/bin/pwd
$ man pwd
[…]
NOTE: your shell may have its own version of pwd, which usually supersedes the version described here.
Please refer to your shell's documentation for details about the options it supports.
[…]

Mais par défaut, c'est la fonction du shell qui est utilisée.

Pour les plus curieux, vous pouvez télécharger les sources du bash :
$wget http://ftp.gnu.org/gnu/bash/bash-4.3.tar.gz
$ tar xf bash-4.3.tar.gz
$ cd bash-4.3 #
$ cd builtins # toutes les fonctions intégrées

Exemple d'interpréteur avec un langage de programmation Python
(http://fr.wikipedia.org/wiki/Python_%28langage%29)



Démarrage de l'interpréteur python 3 :

```
$ python3
>>>def add(a,b) :
…       return a + b
>>>add(1,2)
3
>>> import dis
>>> dis.dis(add)
  2           0 LOAD_FAST                0 (a)
              3 LOAD_FAST                1 (b)
              6 BINARY_ADD
              7 RETURN_VALUE
```

Le bytecode python3 est référencé sur : https://docs.python.org/3/library/dis.html

Pour les plus curieux, téléchargez le code source de python3 :
$ wget https://www.python.org/ftp/python/3.4.2/Python-3.4.2.tar.xz
$ tar xf Python-3.4.2.tar.xz
$ cd Python-3.4.2/Python # et dans Python-3.4.2/Python/Include fichier opcode.h

Annexes

Le hacker : http://fr.wikipedia.org/wiki/Hacker_%28programmation%29
http://fr.wikipedia.org/wiki/Hacker_%28s%C3%A9curit%C3%A9_informatique%29

Comment devenir un hacker : http://files.jkbockstael.be/hacker-howto-fr.html