

Système Exécutable et compilation

Gilles Roussel
<http://igm.univ-mlv.fr/~rousse1/L22/>

Licence 2

6 mars 2011

Qu'est-ce qu'un exécutable ?

- Un fichier
- Ayant des droits d'exécution
- Contenant du code machine ou des données à interpréter
 - 2 premiers octets (*Magic Number*) du fichier permettent de savoir comment traiter le contenu

Quel fichier est exécuté ?

- Fichier dont le chemin d'accès est précisé
 - `/bin/ls` (chemin absolu)
 - `./ls` si vous êtes dans le répertoire `/bin` (chemin relatif)
- Pourtant je peux taper `ls` sans préciser le chemin ?
- Le fichier trouvé dans une liste de répertoires
 - Défini dans la variable d'environnement `PATH`
`# echo $PATH`
`/usr/local/bin:/usr/bin:/bin`
 - Premier fichier trouvé est exécuté. La commande `which` permet de savoir quel est le fichier effectivement exécuté.
`# which ls`
`/bin/ls`

J'ai donné les droits en exécution, pourtant la commande ne s'exécute pas !

- Il faut avoir les droits en exécution sur l'ensemble des répertoires de son chemin d'accès
- Droits sur les répertoires donnés par la commande `ls -ld .`
`drwxrwxr-x 4 rousse1 rousse1 4096 fév 10 14:28 .`
- `r` indique que le contenu du répertoire peut être listé ; c'est-à-dire qu'il est possible de réaliser un appel à `ls` avec le répertoire en argument ;
- `x` indique qu'il est possible d'accéder au répertoire ; en particulier, il est possible de réaliser un appel à `cd` avec le répertoire en argument ;
- `w` indique qu'il est possible de créer un fichier dans le répertoire

Parenthèse : « À quel groupe j'appartiens ? »

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichiers

d'exécution

Préprocesseurs

Compilation

assemblage

La commande `id` permet de connaître les groupes auxquels vous appartenez et votre groupe courant

```
# id
uid=500(rousseau) gid=501(rousseau) groupes=501(rousseau),502(test)
```

Votre groupe principal est précisé dans le fichier `/etc/passwd`

```
rousseau:x:500:501:Gilles Roussel:/home/rousseau:/bin/bash
```

Les autres groupes sont précisés dans le fichier `/etc/group`

```
rousseau:x:501:
test:x:502:rousseau
```

La commande `newgrp` permet de changer votre groupe courant

```
# newgrp test
# id
uid=500(rousseau) gid=502(test) groupes=501(rousseau),502(test)
```

Gilles Roussel

Système

Parenthèse : « Où sont passés les mots de passe ? »

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichiers

d'exécution

Préprocesseurs

Compilation

assemblage

- Les “mots de passe” étaient à l'origine dans `/etc/passwd` et `/etc/group` à la place du `x`

```
rousseau:x:500:501:Gilles Roussel:/home/rousseau:/bin/bash
```

- Ils sont maintenant dans `/etc/shadow` et `/etc/gshadow`
 - Ils ne sont pas stockés en texte clair
 - Stockage d'une valeur de *hashage*

Gilles Roussel

Système

Exécution

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichiers

d'exécution

Préprocesseurs

Compilation

assemblage

- Système associe un processus au programme
 - Un programme peut être associé à plusieurs processus
- Un processus est associé à :
 - une identité
 - un espace mémoire (virtuel) propre au processus
 - des droits dépendants de l'utilisateur et du propriétaire du fichier
 - diverses autres choses

Gilles Roussel

Système

Comment produire un fichier binaire exécutable ?

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichiers

d'exécution

Préprocesseurs

Compilation

assemblage

- Écriture en binaire !!!
- Écrire en assembleur et générer le binaire :
 - non portable
 - difficile à écrire
 - difficile à maintenir
- Utilisation d'un langage de haut niveau et d'un compilateur vers l'architecture cible

Gilles Roussel

Système

Un programme C

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichier

d'ex-écutables

Préprocesseur

Compilation

aspirée

rien.c

```
#include <stdlib.h>
int main(int argc, char* argv[]) {
    return EXIT_SUCCESS;
}
```

Gilles Roussel

Système

Un programme assembleur

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichier

d'ex-écutables

Préprocesseur

Compilation

aspirée

rien.s

```
.file "rien.i"
.text
.globl main
.type main, @function
main:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
movq %rsp, %rbp
.cfi_offset 6, -16
.cfi_def_cfa_register 6
movl %edi, -4(%rbp)
movq %rsi, -16(%rbp)
movl $0, %eax
leave
ret
.cfi_endproc
.LFE0:
.size main, -main
```

Gilles Roussel

Système

Un programme binaire

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichier

d'ex-écutables

Préprocesseur

Compilation

aspirée

rien

```
177  E  L  F 001 001 001  \0  \0  \0  \0  \0  \0  \0  \0  \0
002  \0 003  \0 001  \0  \0  \0 214 202 004  \b  4  \0  \0  \0
   8  \a  \0  \0  \0  \0  \0  \0  4  \0  \0  \a  \0  (  \0
....
```

Gilles Roussel

Système

Compilation : la version courte

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichier

d'ex-écutables

Préprocesseur

Compilation

aspirée

```
# make rien
cc rien.c -o rien
# ls -l rien*
-rwxrwxr-x 1 roussel roussel 4620 fév  5 15:25 rien
-rw-rw-r-- 1 roussel roussel  49 fév  5 10:08 rien.c
```

rien.c → **Compilateur** → rien

make ne recompile pas rien, s'il est plus récent que rien.c

```
# make rien
make: 'rien' est à jour.
```

Gilles Roussel

Système

Compilation : la version longue

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier

d'as-liste

Préprocesseur

Compilation

argument

```
# cpp rien.c rien.i # Préprocesseur
# cc1 -s rien.i # Compilateur
# as rien.s -o rien.o # Assembleur
# ld -o rien rien.o /usr/lib/crt1.o \
  /usr/lib/crti.o /usr/lib/crtn.o -lc \
  -dynamic-linker /lib/ld-linux.so.2 # Édition de liens

# ls -l rien*
-rwxrwxr-x 1 roussel roussel 3587 fév  5 15:25 rien
-rw-rw-r-- 1 roussel roussel  49 fév  5 10:08 rien.c
-rw-rw-r-- 1 roussel roussel 117 fév  5 16:26 rien.i
-rw-rw-r-- 1 roussel roussel 709 fév  5 16:34 rien.o
-rw-rw-r-- 1 roussel roussel 363 fév  5 16:26 rien.s
```

Gilles Roussel

Système

Pourquoi la ligne d'édition de liens est si longue ?

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier

d'as-liste

Préprocesseur

Compilation

argument

```
# ld -o rien rien.o /usr/lib/crt1.o \
  /usr/lib/crti.o /usr/lib/crtn.o -lc \
  -dynamic-linker /lib/ld-linux.so.2
```

- elle précise le nom de l'exécutable à créer et le nom du fichier objet contenant mon programme ;
- elle précise les noms d'autres fichiers objets contenant des fonctions nécessaires à la construction du programme exécutable ;
- elle précise le nom d'une bibliothèque (*library*) contenant des fonctions de base (`printf()`, etc.) ;
- elle dit que ces fonctions doivent être liées dynamiquement à l'exécution.

Gilles Roussel

Système

Mais encore ?

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier

d'as-liste

Préprocesseur

Compilation

argument

- Le fichier objet `rien.o` contient uniquement le code de la fonction `main()` que j'ai écrite.
La commande `nm` permet de connaître les noms accessibles dans le fichier objet (adresse, type, nom) :

```
# nm rien.o
00000000 T main
```
- Il n'est pas dans un format exécutable.
La commande `file` permet de connaître le format des données contenues dans le fichier

```
# file rien.o
rien.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not stripped

# file rien
rien: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically linked (uses shared libs), not stripped
```

Gilles Roussel

Système

Mais encore ?

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier

d'as-liste

Préprocesseur

Compilation

argument

- Il y a besoin d'autres « morceaux » pour pouvoir utiliser `rien.o`.
En particulier le fichier `/usr/lib/crt1.o` :
 - prépare l'exécution (organisation mémoire) ;
 - contient le nom `_start` (recherché pour commencer le programme) ;
 - appelle `main()`.
- L'édition de liens se charge de recoller les morceaux.

Gilles Roussel

Système

Mais encore ?

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier d'ex-é-é-é

Préprocesseur

Compilation

argente

- Les bibliothèques sont précisées au moyen de l'option **-l**
- Option valide pour `ld` et `gcc`
- **-lc** correspond à la recherche d'un fichier de nom :
 - **libc.a** : bibliothèque statique
 - **libc.so** : bibliothèque dynamique
- Liste des répertoires de recherche (par défaut) configuré dans `ld`

```
# ld --verbose
....
SEARCH_DIR("/usr/i386-redhat-linux/lib");
SEARCH_DIR("/usr/local/lib"); SEARCH_DIR("/lib");
SEARCH_DIR("/usr/lib");
...
```

Gilles Roussel

Système

Mais encore ?

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier d'ex-é-é-é

Préprocesseur

Compilation

argente

- Les fonctions de bibliothèque qui ne font pas partie du programme initial peuvent être liées :
 - **dynamiquement** (à l'exécution) : grâce à la bibliothèque `ld-linux`
 - **statiquement** (à la compilation) : dans le fichier exécutable
- ```
gcc -static -o rien rien.o
```

Gilles Roussel

Système

## Liaison statique ou dynamique

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier d'ex-é-é-é

Préprocesseur

Compilation

argente

- Liaison statique
  - Exécutable de grande taille
  - Liaison à une version particulière des bibliothèques
- Liaison dynamique
  - Exécutable de plus petite taille
  - Sur-coût à l'exécution
  - Besoin des bibliothèques
  - Suivi des versions
- **ldd** permet de connaître les bibliothèques dynamiques nécessaires à un programme.

Gilles Roussel

Système

## Compilation : la version longue

Système

Gilles Roussel

Exécutable

Processus

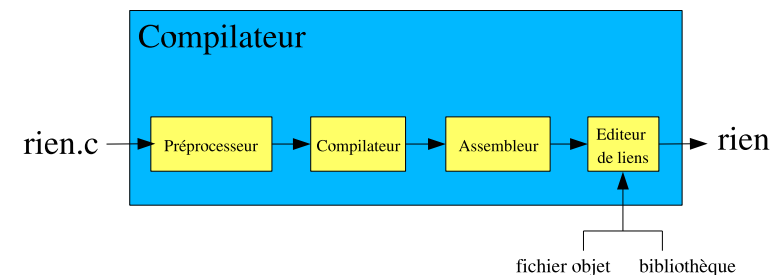
Compilation

Fichier d'ex-é-é-é

Préprocesseur

Compilation

argente



Gilles Roussel

Système

## Un autre programme ?

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichier  
d'en-tête

Préprocesseur

Compilation  
séparée

### deux.c

```
int main(int argc , char* argv []) {
 printf ("Racine de 4=%d\n",sqrt(4));
 return 0;
}
```

```
make deux
cc deux.c -o deux
./deux
Racine de 4 = 0
```

Gilles Roussel

Système

## Pourquoi ?

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichier  
d'en-tête

Préprocesseur

Compilation  
séparée

- Les types de `printf()` et `sqrt()` sont déduits de leur utilisation
- Il faut ajouter la déclaration de `printf()` et `sqrt()`
  - à la main

```
extern int printf (char * format, ...);
extern double sqrt (double v);
```
  - en utilisant le fichier `.h` d'en-tête (*header*) associés à `printf()` et `sqrt()`

```
#include<stdio.h>
#include<math.h>
```
- Il faut ajouter des options de compilation pour vérifier que toutes les fonctions externes sont bien déclarées et bien utilisées

```
-ansi -Wall
```

Gilles Roussel

Système

## Ajouter les options de compilation

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichier  
d'en-tête

Préprocesseur

Compilation  
séparée

- Créer un fichier de compilation `makefile` minimal pour donner des indications à `make`

### makefile

```
CFLAGS=-ansi -Wall
```

```
make deux
cc -ansi -Wall deux.c -o deux
deux.c: In function 'main':
deux.c:2: attention : déclaration implicite de la fonction
'printf'
deux.c:2: attention : déclaration implicite de la fonction
'sqrt'
deux.c:2: attention : format int, arg double (arg 2)
```

Gilles Roussel

Système

## Éliminer tous les "attention" !

Système

Gilles Roussel

Exécutables

Processus

Compilation

Fichier  
d'en-tête

Préprocesseur

Compilation  
séparée

### deux\_prime.c

```
#include<stdio.h>
#include<math.h>

int main(int argc, char* argv[]) {
 printf ("Racine de 4=%f\n",sqrt(4));
 return 0;
}
```

```
make deux_prime
cc -ansi -Wall deux_prime.c -o deux_prime
./deux_prime
Racine de 4 = 2.000000
```

Gilles Roussel

Système

## Comment sont ajoutées les déclarations du fichier d'en-tête ?

| Système             |                          |
|---------------------|--------------------------|
| Gilles Roussel      | Par le préprocesseur cpp |
| Exécutable          |                          |
| Processus           |                          |
| Compilation         |                          |
| Fichier d'en-tête   |                          |
| Préprocesseur       |                          |
| Compilation séparée |                          |

```
cpp deux.c.
...
1 "/usr/include/stdio.h" 1 3 4
...
extern int printf (__const char *__restrict __format, ...);
...
2 "deux.c" 2

int main(int argc, char* argv[])
 printf("Racine de 4 = %d\n",sqrt(4));
 return 0;
```

|                |         |
|----------------|---------|
| Gilles Roussel | Système |
|----------------|---------|

## Comment sont trouvés les fichiers d'en-tête ?

| Système             |  |
|---------------------|--|
| Gilles Roussel      |  |
| Exécutable          |  |
| Processus           |  |
| Compilation         |  |
| Fichier d'en-tête   |  |
| Préprocesseur       |  |
| Compilation séparée |  |

- Dans un ensemble de répertoires prédéfinis

```
cpp --verbose
/usr/local/include
/usr/lib/gcc/i386-redhat-linux/3.4.2/include
/usr/include
```
- Dans des répertoires précisés avec l'option -I

```
cpp -I/home/roussel/include ...
```
- Si le nom du fichier à inclure est entre guillemets (") le fichier est également recherché dans le répertoire courant

```
#include <stdio.h>
#include "monfichier.h"
```

|                |         |
|----------------|---------|
| Gilles Roussel | Système |
|----------------|---------|

## Que contient un fichier d'en-tête ?

| Système             |  |
|---------------------|--|
| Gilles Roussel      |  |
| Exécutable          |  |
| Processus           |  |
| Compilation         |  |
| Fichier d'en-tête   |  |
| Préprocesseur       |  |
| Compilation séparée |  |

- **Jamais de code exécutable !**
- Des demandes d'inclusion d'autres fichiers .h

```
#include <features.h>
```
- Des déclarations de fonctions externes

```
extern int fclose (FILE *__stream);
```
- Des déclarations de variables globales externes

```
extern struct _IO_FILE *stdin;
```
- Des définitions de constantes ou macros

```
#define EOF (-1)
#define getc(_fp) _IO_getc (_fp)
```
- Des définitions de types

```
typedef struct _IO_FILE FILE;
```

|                |         |
|----------------|---------|
| Gilles Roussel | Système |
|----------------|---------|

## Les instructions du préprocesseur

| Système             |  |
|---------------------|--|
| Gilles Roussel      |  |
| Exécutable          |  |
| Processus           |  |
| Compilation         |  |
| Fichier d'en-tête   |  |
| Préprocesseur       |  |
| Compilation séparée |  |

- # en début de ligne, évaluées au fur et à mesure (ordre)
- inclusion de fichier : **#include**
- définition de constante ou de macro : **#define**
- suppression d'une définition : **#undef**

```
#undef EOF
```
- conditionnelles : **#ifdef**, **#ifndef**, **#else**, **#elif** et **#endif**

```
#ifdef OPTION1
...
#elif OPTION2
...
#else
...
#endif
```

|                |         |
|----------------|---------|
| Gilles Roussel | Système |
|----------------|---------|

## Les conditionnelles

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier d'en-tête

Préprocesseur

Compilation séparée

- Utilisées, en particulier, pour éviter les inclusions multiples de fichiers d'en-tête

### exemple.h

```
#ifndef _EXEMPLE_H
define _EXEMPLE_H
....
/* Contenu de exemple.h */
....
#endif /* _EXEMPLE_H */
```

Gilles Roussel

Système

## Compilation séparée

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier d'en-tête

Préprocesseur

Compilation séparée

- Exécutable construit à partir de plusieurs « morceaux » compilés séparément
- Fichiers plus petits, plus lisibles
- Évite de tout recompiler à chaque fois (si c'est bien fait !)
- Permet de partager des « morceaux »
- Création de bibliothèques de fonctions

Gilles Roussel

Système

## Compilation séparée

Système

Gilles Roussel

Exécutable

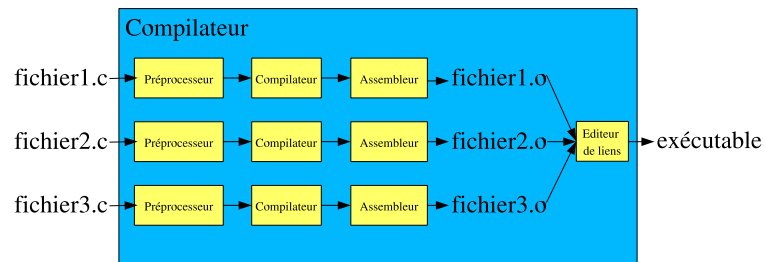
Processus

Compilation

Fichier d'en-tête

Préprocesseur

Compilation séparée



- Les fichiers se « connaissent » via les fichiers d'en-tête
- Au moins un .h par .c qui est utilisé par un autre .c

Gilles Roussel

Système

## Qui doit inclure un fichier .h ?

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier d'en-tête

Préprocesseur

Compilation séparée

- Tout fichier qui utilise les fonctions et variables déclarées dans le fichier d'en-tête.  
Permet, en particulier, de vérifier que les types des arguments utilisés pour appeler une fonction correspondent à ceux des paramètres déclarés.
- Le fichier qui définit les fonctions et variables déclarées dans le fichier d'en-tête.  
Permet, en particulier, de vérifier que les types des déclarations sont bien cohérents avec ceux des définitions.
- Tout fichier utilisant les types déclarés dans les fichiers d'en-tête.

Gilles Roussel

Système

## Un exemple simple

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier  
d'ex-écut

Préprocesseur

Compilation  
séparée

f.c

```
#include "f.h"

int f(int i) {
 return 2*i*i;
}
```

f.h

```
#ifndef _F_H
#define _F_H 1

extern int f(int i);
#endif
```

Gilles Roussel

Système

## Un exemple simple

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier  
d'ex-écut

Préprocesseur

Compilation  
séparée

main.c

```
#include <stdio.h>
#include "f.h"

int main(int argc , char* argv []){
 printf ("%d\n",f(2));
 printf ("%d\n",f(4));
 return 0;
}
```

Gilles Roussel

Système

## Compilation

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier  
d'ex-écut

Préprocesseur

Compilation  
séparée

```
make main
cc -ansi -Wall main.c -o main
/tmp/ccy1DjDf.o(.text+0x25): In function 'main':
: undefined reference to 'f'
collect2: ld a retourné 1 code d'état d'exécution
make: *** [main] Erreur 1
```

make ne sait pas quels fichiers sont nécessaires pour construire main!

Gilles Roussel

Système

## Un nouveau makefile

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier  
d'ex-écut

Préprocesseur

Compilation  
séparée

makefile

```
CFLAGS=-ansi -Wall

main: main.c f.c f.h
 gcc $(CFLAGS) -o main main.c f.c
```

La cible main dépend de main.c, f.c et f.h  
Pour créer main il faut compiler main.c et f.c ensemble.

```
make
cc -ansi -Wall -o main main.c f.c
```

main cible par défaut (make ⇔ make main)  
Si main.c ou f.c ou f.h est plus récent que main alors main est recompilé

Gilles Roussel

Système

## Il n'y a pas de compilation séparée !

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier  
d'en-tête

Préprocesseur

Compilation  
séparée

Pour faire de la compilation séparée il faut :

- créer des **fichiers objets** (.o) qui seront recompilés lorsque les fichiers sources (.c et .h) changent
- « **lier** » les **fichiers objets** entre eux pour créer l'exécutable chaque fois que l'un d'eux change

Gilles Roussel

Système

## Un meilleur makefile

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier  
d'en-tête

Préprocesseur

Compilation  
séparée

### makefile

```
CFLAGS=-ansi -Wall
```

```
main: main.o f.o
 gcc -o main main.o f.o
```

```
make
```

```
cc -ansi -Wall -c -o main.o main.c
```

```
cc -ansi -Wall -c -o f.o f.c
```

```
gcc -o main main.o f.o
```

On utilise **gcc** plutôt que **ld** pour éviter les multiples arguments

Gilles Roussel

Système

## Ce n'est pas encore parfait !

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier  
d'en-tête

Préprocesseur

Compilation  
séparée

- Si f.h est modifié, make dit qu'il n'y a rien à faire !
- Il faut ajouter des dépendances pour la création des fichiers objets

### makefile

```
CFLAGS=-ansi -Wall
```

```
main: main.o f.o
 gcc -o main main.o f.o
```

```
main.o: main.c f.h
```

```
f.o: f.c f.h
```

Gilles Roussel

Système

## Générer automatiquement les dépendances

Système

Gilles Roussel

Exécutable

Processus

Compilation

Fichier  
d'en-tête

Préprocesseur

Compilation  
séparée

### makefile

```
CFLAGS=-ansi -Wall
```

```
SRCS=f.c main.c
```

```
main: main.o f.o
 gcc -o main main.o f.o
```

```
depend:
 makedepend -- $(CFLAGS) -- $(SRCS)
```

Utilisation d'une cible inconditionnelle (la commande associée est toujours exécutée) **depend** et de la commande **makedepend** qui ajoute les dépendances au makefile

Gilles Roussel

Système

## Générer automatiquement les dépendances

Système

Gilles Roussel

Essentielle

Processus

Compilation

Fichier

d'en-tête

Préprocesseur

Compilation  
séparée

```
make depend
```

```
makedepend -- -ansi -Wall -- f.c main.c
```

```
makefile
```

```
CFLAGS=-ansi -Wall
```

```
SRCS=f.c main.c
```

```
main: main.o f.o
```

```
gcc -o main main.o f.o
```

```
depend:
```

```
makedepend -- $(CFLAGS) -- $(SRCS)
```

Gilles Roussel

Système