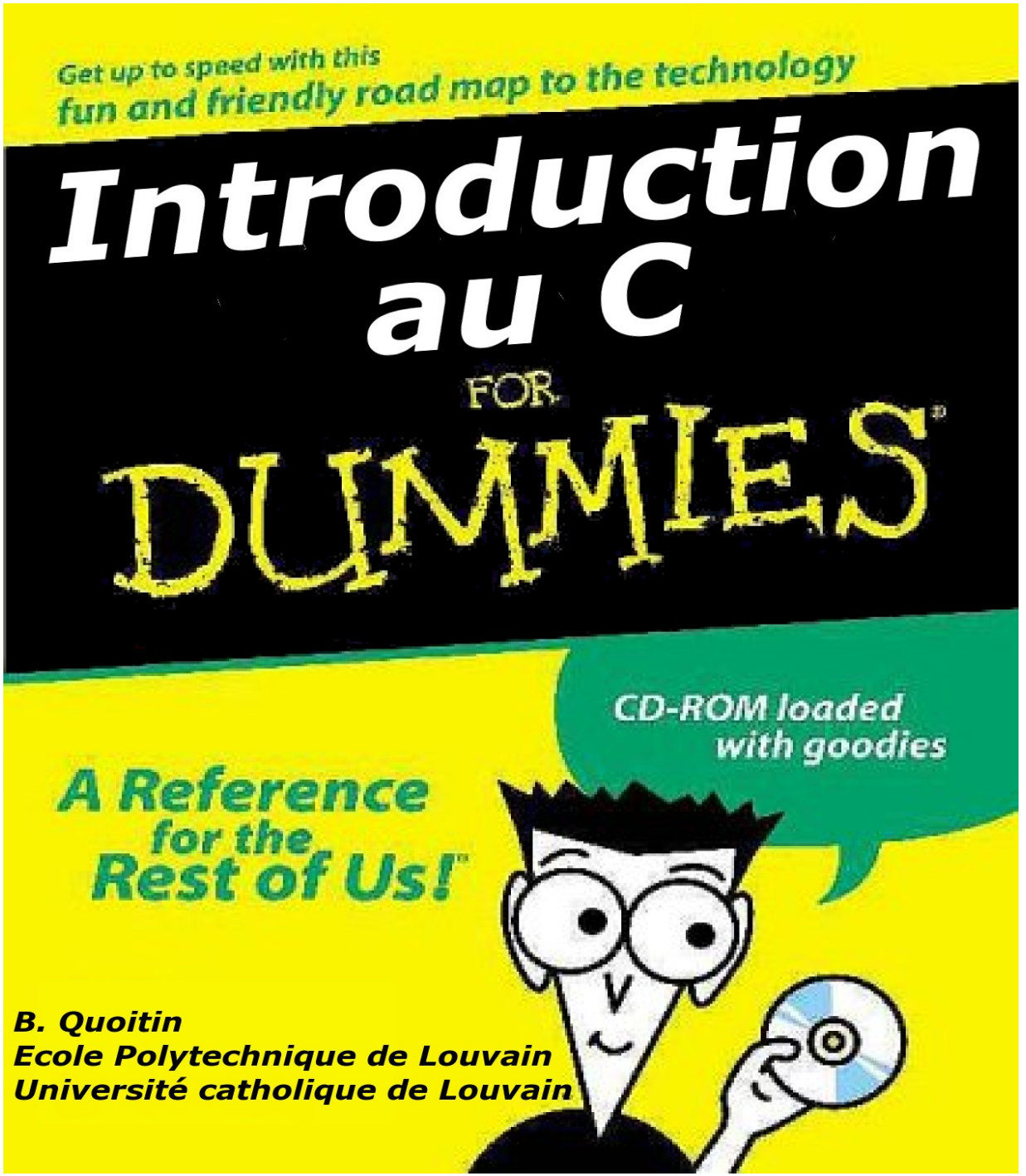
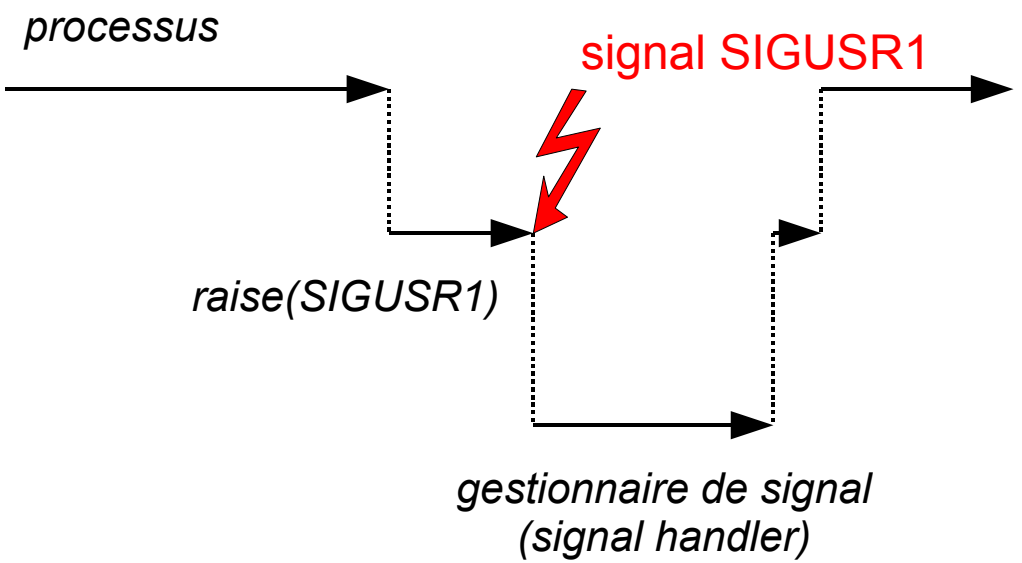

Get up to speed with this
fun and friendly road map to the technology
Introduction au C
FOR
DUMMIES
CD-ROM loaded with goodies
A Reference for the Rest of Us!
B. Quoitin
Ecole Polytechnique de Louvain
Université catholique de Louvain

Table des matières

UNIX Signals

UNIX Signals

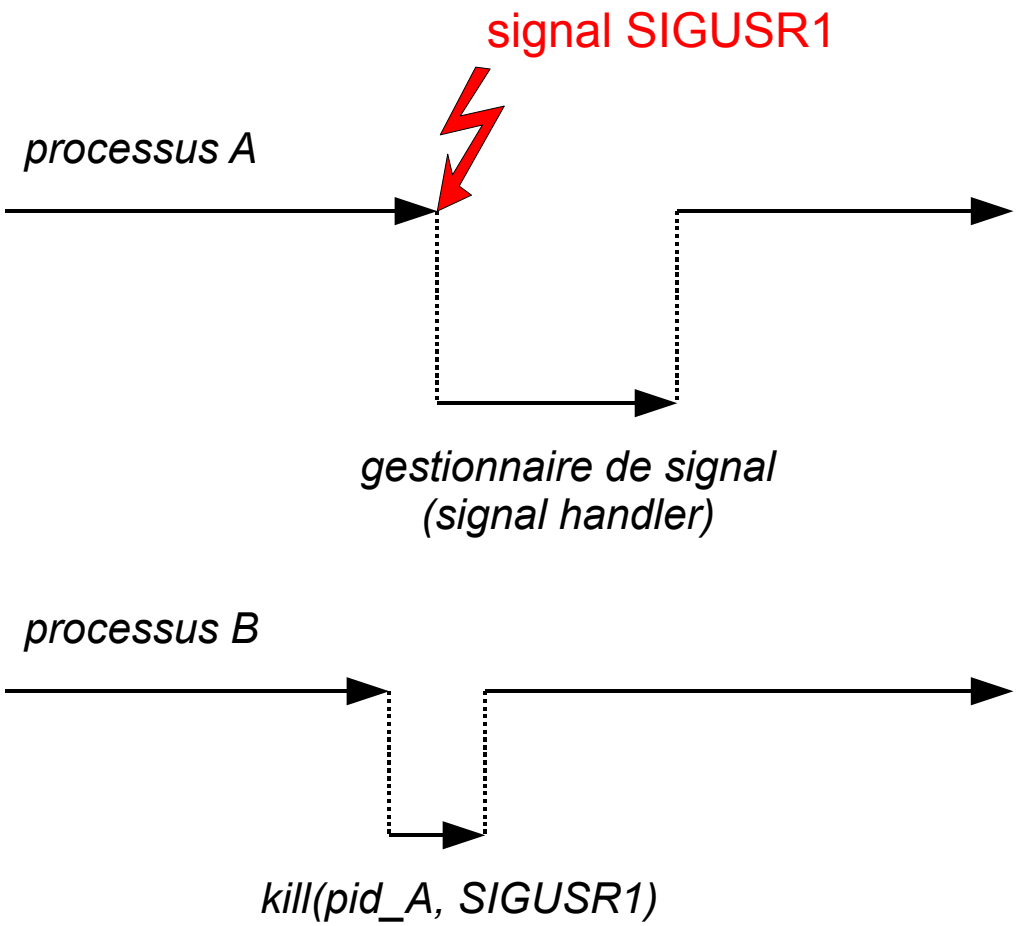
- Signal synchrone



Note: on décrit parfois les signaux comme des « interruptions software ».

UNIX Signals

- Signal asynchrone



UNIX Signals

- Gestionnaires par défaut
  - ignorer
  - terminer le processus (+ « core dump »)
  - suspendre le processus
  - continuer le processus
- Exemples de signaux
  - SIGFPE (division par 0, underflow, overflow)
  - SIGINT (ctrl-C)
  - SIGALRM (timer expiré)
  - SIGKILL (tue le processus)
  - SIGUSR1, SIGUSR2, ...

UNIX Signals

- Exemple

```c
#include <...> /* assert.h, signal.h, stdio.h, unistd.h */

int flag= 0;
int val= 0;

void signal_handler(int signum) {
  if (signum == SIGINT)        { flag= 1; }
  else if (signum == SIGUSR1) { val++; }
  else if (signum == SIGUSR2) { val--; }
}

int main() {
  assert(signal(SIGINT, signal_handler) != SIG_ERR);
  assert(signal(SIGUSR1, signal_handler) != SIG_ERR);
  assert(signal(SIGUSR2, signal_handler) != SIG_ERR);
  while (!flag)
    sleep(1);
  printf("La valeur est %d\n", val);
  return 0;
}
```

Note: il existe une interface de gestion des signaux plus récente: `sigaction()`

UNIX Signals

- Exemple

```
mortimer bqu$ ./sig_usr
(processus lancé, en attente de signaux)
```

```
mortimer bqu$ ps
  PID  TT  STAT      TIME COMMAND
 3426  p1  Ss+    0:00.22 bash
 5433  p3  S      0:04.91 xemacs sig_usr.c
 5870  p3  S+     0:00.01 ./sig_usr
 5505  p4  Ss     0:00.02 bash
mortimer bqu$ kill -SIGUSR1 5870
mortimer bqu$ kill -SIGUSR1 5870
mortimer bqu$ kill -SIGUSR1 5870
mortimer bqu$ kill -SIGUSR2 5870
mortimer bqu$ kill -SIGUSR1 5870
mortimer bqu$
```

UNIX Signals

- Exemple

```
mortimer bqu$ ./sig_usr
(processus lancé)
(Ctrl-C)
La valeur est 3
mortimer bqu$
```

UNIX Signals

- Exemple (`SIGALRM`)

```c
#include <...> // assert.h, signal.h, stdio.h, stdlib.h, unistd.h

void signal_handler(int signum) {
  if (signum == SIGALRM)
    printf("Les chaussettes de l'archiduchesse"
           " sont-elles sèches ?\n");
}

int main() {
  long delay;
  srandom(2009);
  assert(signal(SIGALRM, signal_handler) >= 0);
  while (1) {
    delay= 1 + (random() % 5);
    printf("Attend %ld secondes...\n", delay);
    alarm(delay);
    pause();
  }
  return 0;
}
```

UNIX Signals

- Exemple (`SIGALRM`)

```
mortimer bqu$ ./sig_alrm
Attend 2 secondes...
Les chaussettes de l'archiduchesse sont-elles sèches ?
Attend 2 secondes...
Les chaussettes de l'archiduchesse sont-elles sèches ?
Attend 4 secondes...
Les chaussettes de l'archiduchesse sont-elles sèches ?
Attend 3 secondes...
Les chaussettes de l'archiduchesse sont-elles sèches ?
Attend 4 secondes...
```

C Storage Classes

C Storage Classes

- Declaration

```
[ storage-class ] type identifier;
```

- Storage classes
  - sur la pile (`auto`)
  - si possible dans registre (`register`)
  - dans segment données (`auto`)
  - dans le segment `.bss` (`static`)
  - portée étendue (`extern`)

« durée » locale

« durée » globale

Note: `register` n'est pratiquement plus utilisé

C Storage Classes

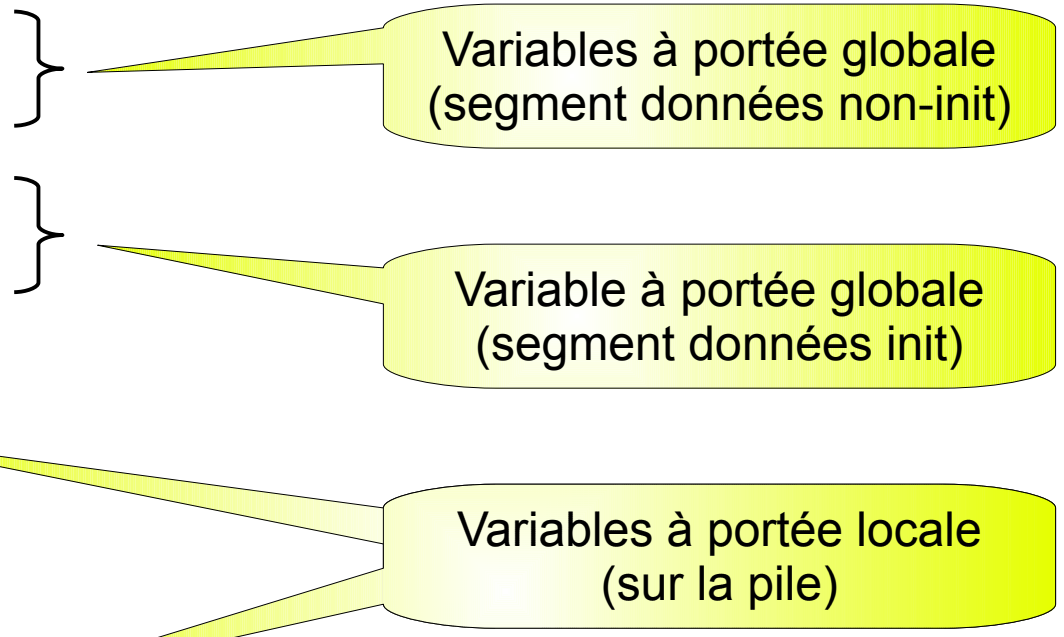
- Storage classes par défaut (`auto`)

```
int var1;
char * tab_str[100];

int var2= 5;
char * str= "Hello";

int fct() {
  int var3;
  /* ... */
}

int main() {
  unsigned int var4;
  /* ... */
  return 0;
}
```



C Storage Classes

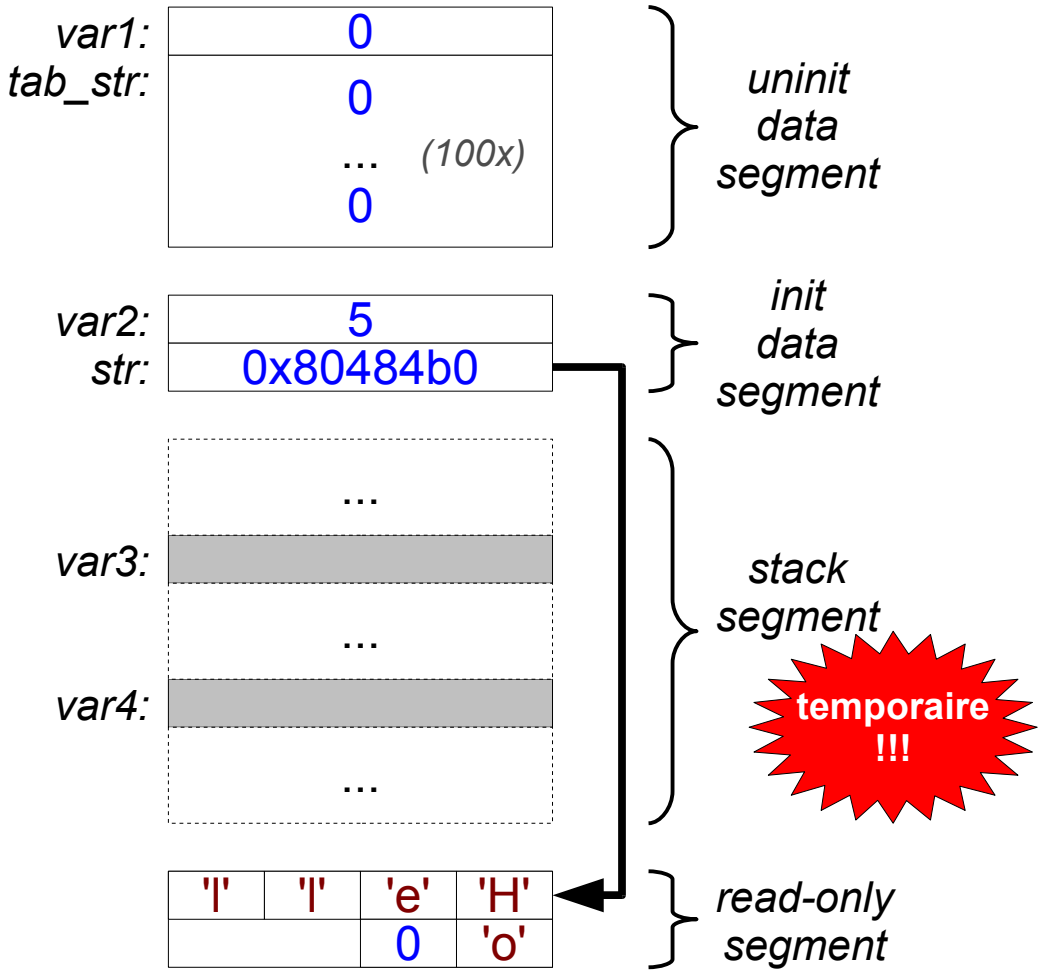
- Storage classes par défaut (`auto`)

```
int var1;
char * tab_str[100];

int var2= 5;
char * str= "Hello";

int fct() {
  int var3;
  /* ... */
}

int main() {
  unsigned int var4;
  /* ... */
  return 0;
}
```



C Storage Classes

- Exemple (`static`)

```c
#include <stdio.h>

int fct() {
  static int var= 0;
  return var++;
}

int main() {
  fct();
  fct();
  printf("Value %d\n", fct());
  return 0;
}
```

Variable à portée locale
à durée globale

```
mortimer bqu$ ./prog
Value 2
mortimer bqu$
```

Note: les variables `static` dans les fonctions sont dangereuses dans les programmes « multi-threadés » (cf. INGI1113 l'année prochaine)

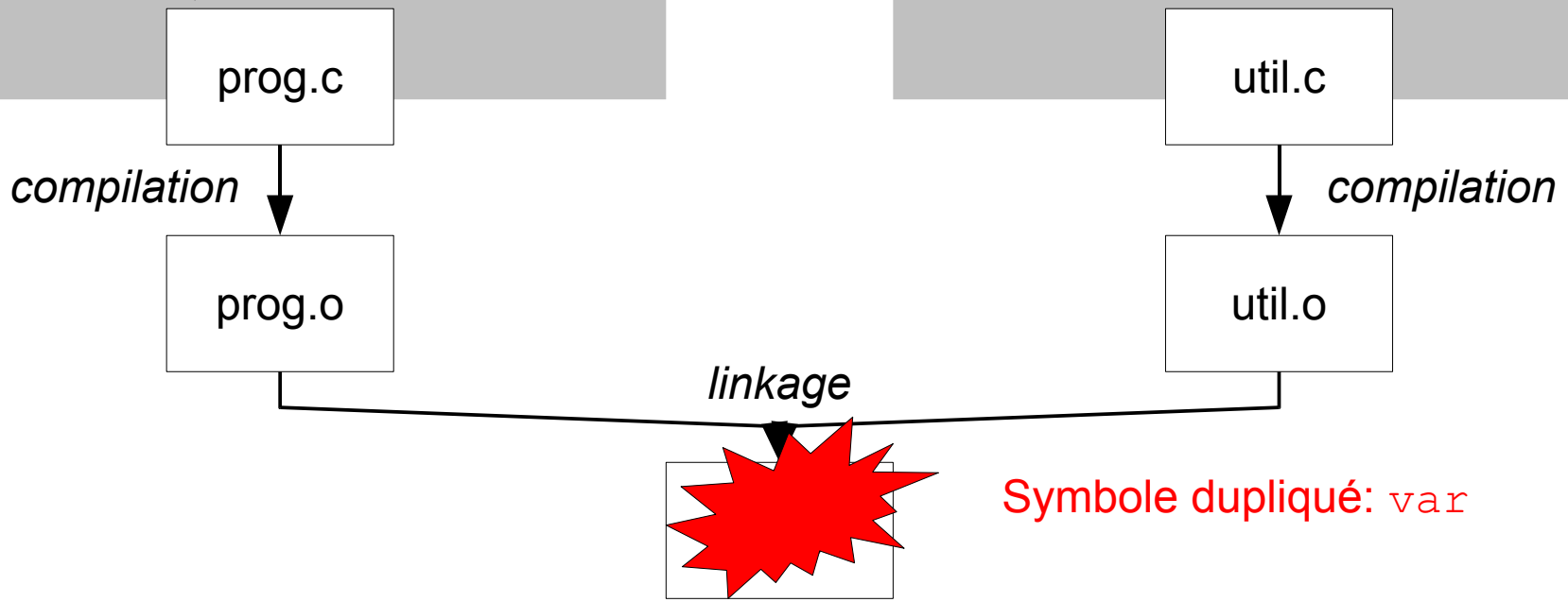
C Storage Classes

- Exemple (`extern`)

```c
#include <stdio.h>
int var= 5;

int main() {
  printf("var=%d\n", var);
  return 0;
}
```

```c
int var= 10;
```



C Storage Classes

- Exemple (`extern`)

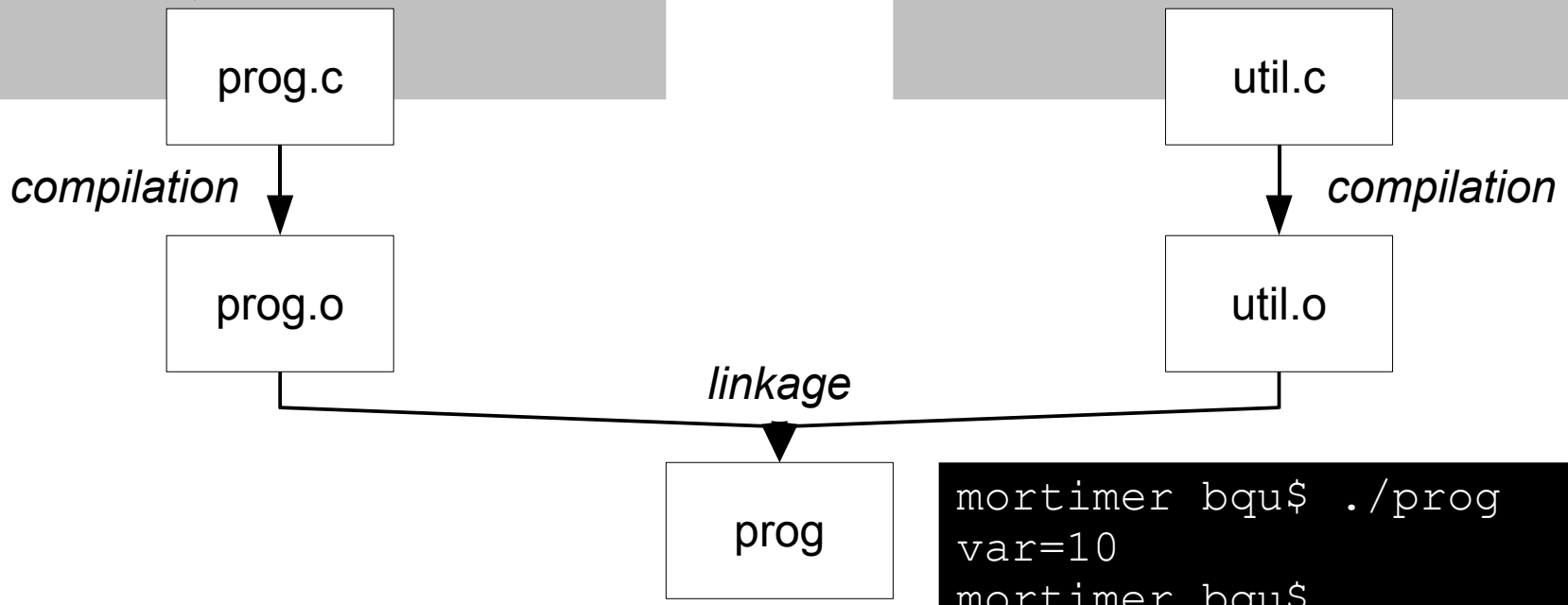
```c
#include <stdio.h>
extern int var;

int main() {
  printf("var=%d\n", var);
  return 0;
}
```

Déclaration

```c
int var= 10;
```

Définition




```
mortimer bqu$ ./prog
var=10
mortimer bqu$
```

C storage classes

- Storage classes des fonctions
  - par défaut: `extern` (i.e. portée globale)
  - peut être changé avec `static`

```c
#include <stdio.h>
#include <util.h>

int main() {
  fct_publique();
  return 0;
}
```

```c
void fct_publique() {
  /* ... */
}

static void fct_privee() {
  /* ... */
}
```

```c
#ifndef __UTIL_H__
#define __UTIL_H__
void fct_publique();
#endif /* __UTIL_H__ */
```

C Type Qualifiers

C Type Qualifiers

- Type qualifiers
  - variable non modifiable (`const`)
  - variable changeant de façon asynchrone (`volatile`)

C Type Qualifiers

- Exemple (`const`)

```
const int a= 10;

a= 5;              KO
```

```
const char * str= strdup("Good morning Vietnam !");

str= "Hello";      OK

str[2]= 'x';       KO
```

pointeur vers zone constante

```
char * const str= strdup("plop");

str[2]= 'x';       OK

str= "re-plop";    KO
```

pointeur constant

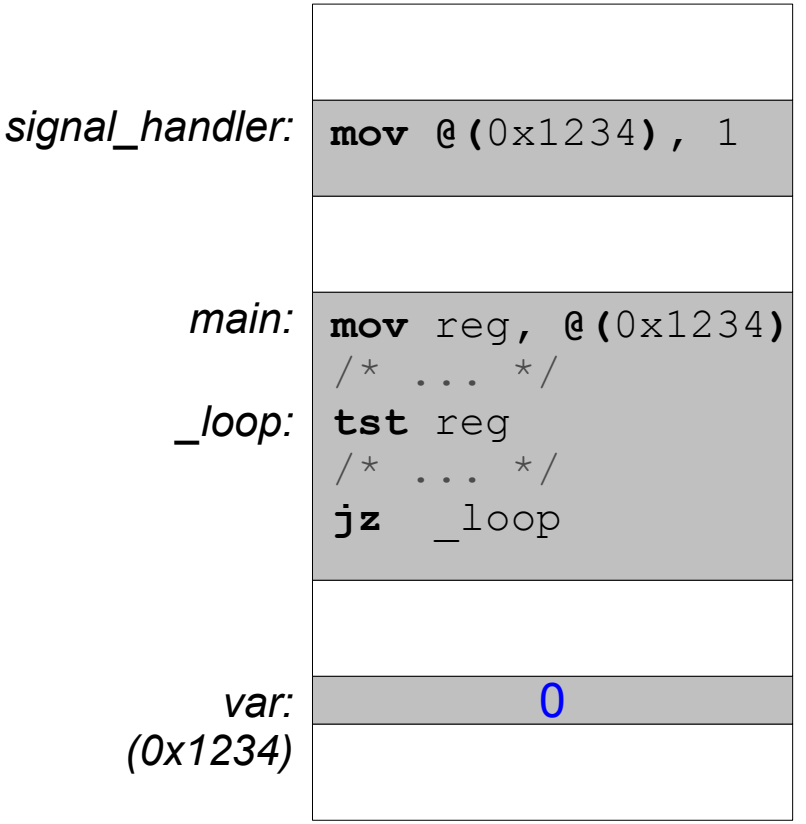
C Type Qualifiers

- Exemple (`volatile`)

```
int flag= 0;

void signal_handler(int signum) {
  flag= 1;
}

int main() {
  /* ... */
  while (!flag) {
    /* ... */
  }
  return 0;
}
```



Note: **`jz`** `_loop` signifie sauter à l'instruction qui est à l'adresse du label `_loop` si le résultat du test (**`tst`** `reg`) vaut 0.

Note: avec gcc, il faut compiler avec l'option -O2 (optimisations) pour rencontrer le problème.

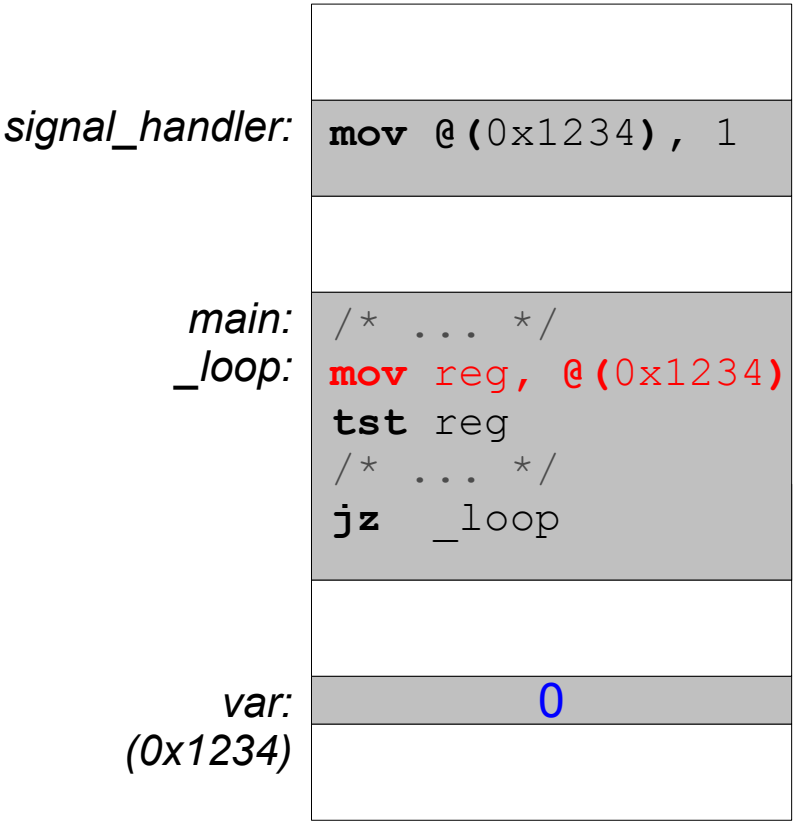
C Type Qualifiers

- Exemple (`volatile`)

```
volatile int flag= 0;

void signal_handler(int signum) {
  flag= 1;
}

int main() {
  /* ... */
  while (!flag) {
    /* ... */
  }
  return 0;
}
```

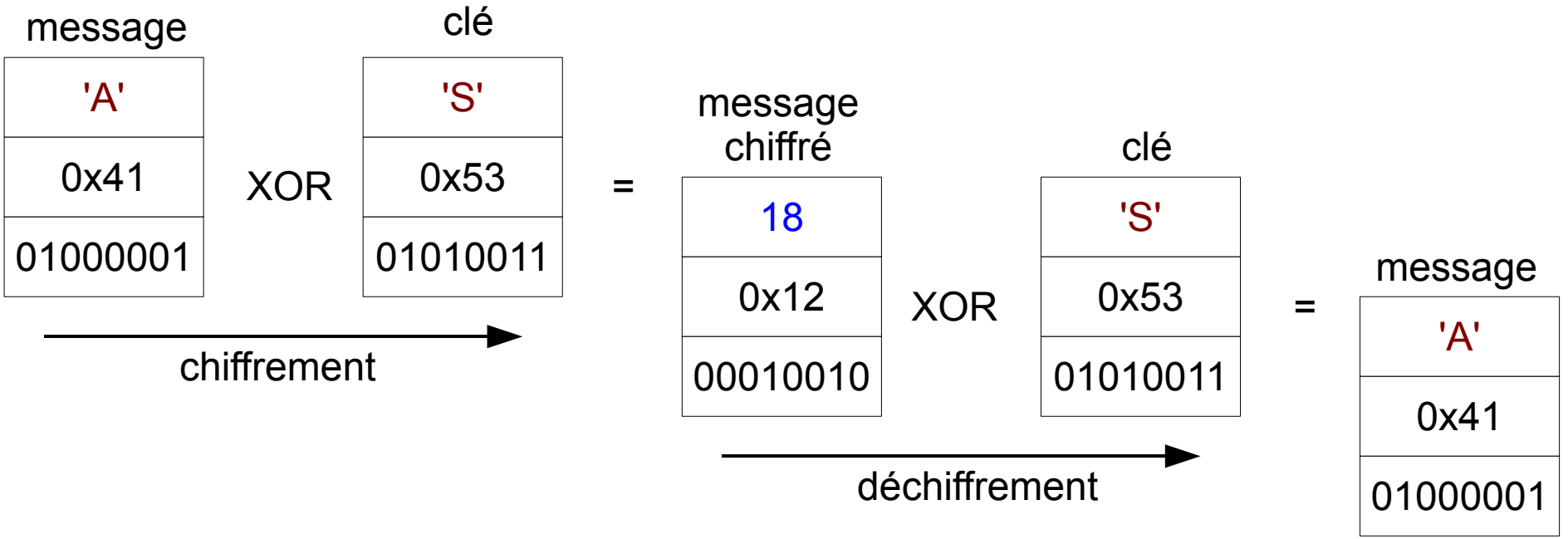


Note: les variables de type `volatile` sont surtout utilisées dans la programmation de systèmes embarqués (gestion d'interruption). A se rappeler dans qques semaines...

Suggestions d'exercices

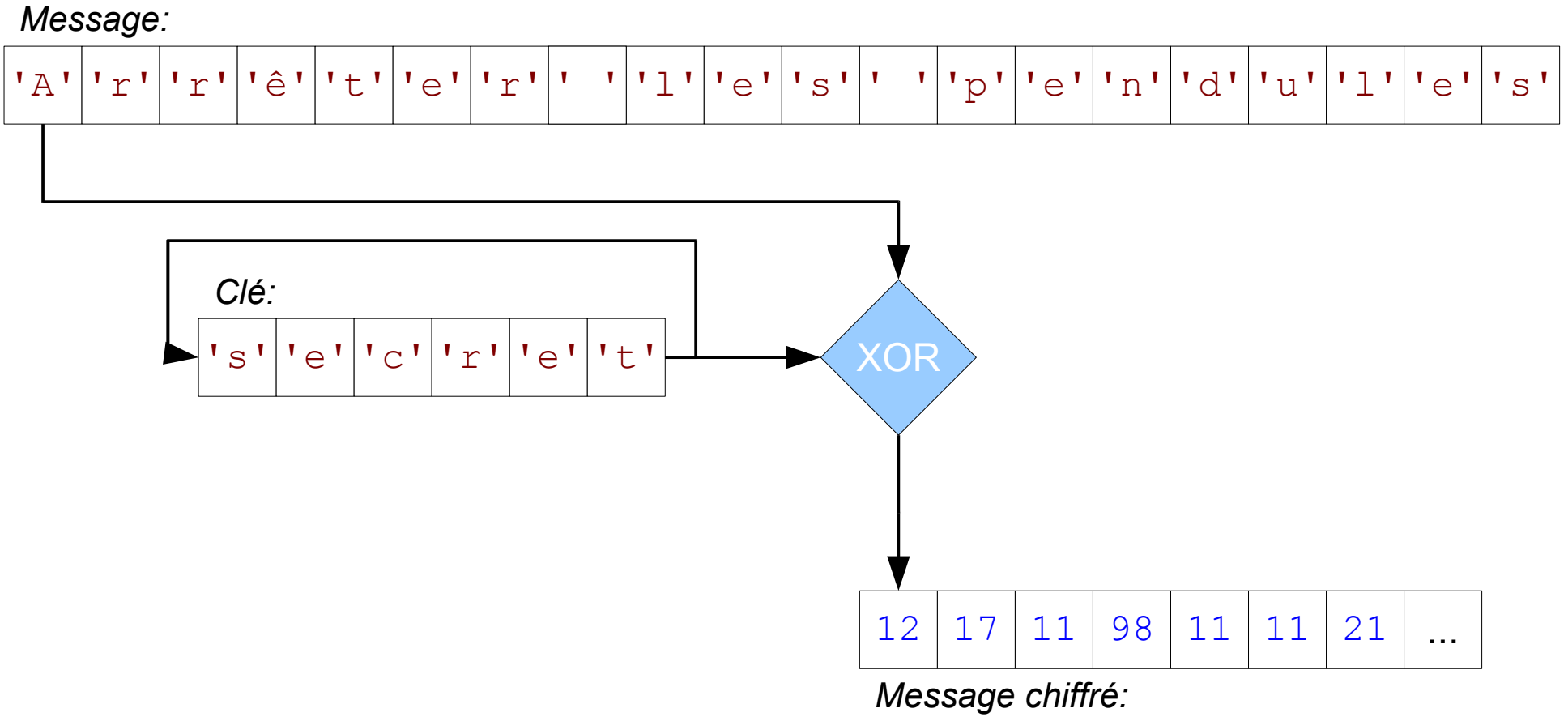
Suggestions d'exercices

- Exercice 1
  - chiffrement avec XOR : (M^k)^k = M
  - chiffrer = déchiffrer (symétrique)
  - utiliser entrée/sortie standards



Suggestions d'exercices

- Exercice 1 (suite)



Suggestions d'exercices

- Exercice 1 (suite)

Enfant ! si j'étais roi, je donnerais l'empire,
Et mon char, et mon sceptre, et mon peuple à genoux
Et ma couronne d'or, et mes bains de porphyre,
Et mes flottes, à qui la mer ne peut suffire,
Pour un regard de vous !

Si j'étais Dieu, la terre et l'air avec les ondes,
Les anges, les démons courbés devant ma loi,
Et le profond chaos aux entrailles fécondes,
L'éternité, l'espace, et les cieux, et les mondes,
Po

```
mortimer bqu$ cat hugo.txt | ./xor secret > hugo.xored
mortimer bqu$
```

0612 4803 0a1b 431a 0250 450c 0017 061e
0136 0916 1107 4e53 1b50 4500 4819 1c99
0c16 0450 160c 0053 0c1e 160c 655f 0d3c
1b36 0550 0b06 0c53 0918 491b 0a53 4804
001e 481e 061a 1f16 1a04 490c 0a53 4804
4316 2d7a 451d 0a1e 4803 090f 1b1c 0d04
1d16 0111 451a 481f 0515 0c19 0a01 625c
4300 8850 1449 0606 0450 4508 0a1e 4802
0a1d 1850 100c 4f07 1d03 030f 1d1a 4415
3f79 1d1f 451b 0106 1a50 020c 1d12 4814
0a17 1e50 1006 4f00 6251 3663 4f1a 4f1a
1b9a 0111 451a 0637 1d15 4545 0e1f 1c50
1d16 0d02 0049 4f07 4f1c 0c08 4f01 1e11
0010 0c1e 160c 655f 4f3c 1180 1d16 011e
4f00 0611 000e 4300 0450 160c 0b53 0599
011c 4803 0a0a 1d06 8112 451a 0a17 0906
1b1d 0550 4508 001f 4419 2063 4f07 0d1c
1f53 0702 0a0f 0b1d 0b50 0401 1c1c 0950
1706 0d50 1107 0e01 0419 0005 4f00 8116
8607 485c 4205 1c16 0900 000a 4f5f 1c15
0353 1b15 0649 0a1a 1005 4545 1b16 0450
1c16 0550 0b06 0a17 4403 3563 1a1c 4802
0106 0a50 0c08 0a00 4802 000d 1b53 011f
4e53 007a

Note: la sortie standard de « `cat` » est redirigée vers l'entrée standard de « `xor` » avec « `|` » (le pipe UNIX). Votre programme ne doit pas se charger d'ouvrir le fichier à chiffrer.

Suggestions d'exercices

- Exercice 2
  - Programme « serveur »
    - SIGUSR1 incrémente valeur
    - SIGUSR2 décrémente valeur
    - SIGALRM affiche la valeur toutes les 5 secondes
  - Programme « client »
    - envoie des signaux SIGUSR1 et SIGUSR2 au « serveur »
  - Utiliser la nouvelle interface POSIX `sigaction()`

FIN

Questions ?

Remerciements: merci à Sébastien Barré et Y. Gillet pour leurs commentaires sur cette partie du cours. Merci aussi à S. Marquet pour ses questions inspirantes.