Programmation en C

John Samuel

CPE Lyon

Année: 2017-2018

Courriel: john(dot)samuel(at)cpe(dot)fr

Objectifs

- Syntaxe et bases de la programmation
- Manipulation de buffer
- Initiation au concept de chaine de compilation

Ce module est divisé en deux parties:

- Cours: 12h
- Travaux pratiques: 24h

Programmation en C

Module:

- Système d'exploitation: Linux
- Compilateur: gcc
- Éditeur: Vim, Emacs (ou selon votre choix)

Cours:

- Interactifs
- Slides: disponibles en Français, Anglais
- Les questions: chaque 20-30 mins
- Devoir surveillé: 60%

Travaux pratiques:

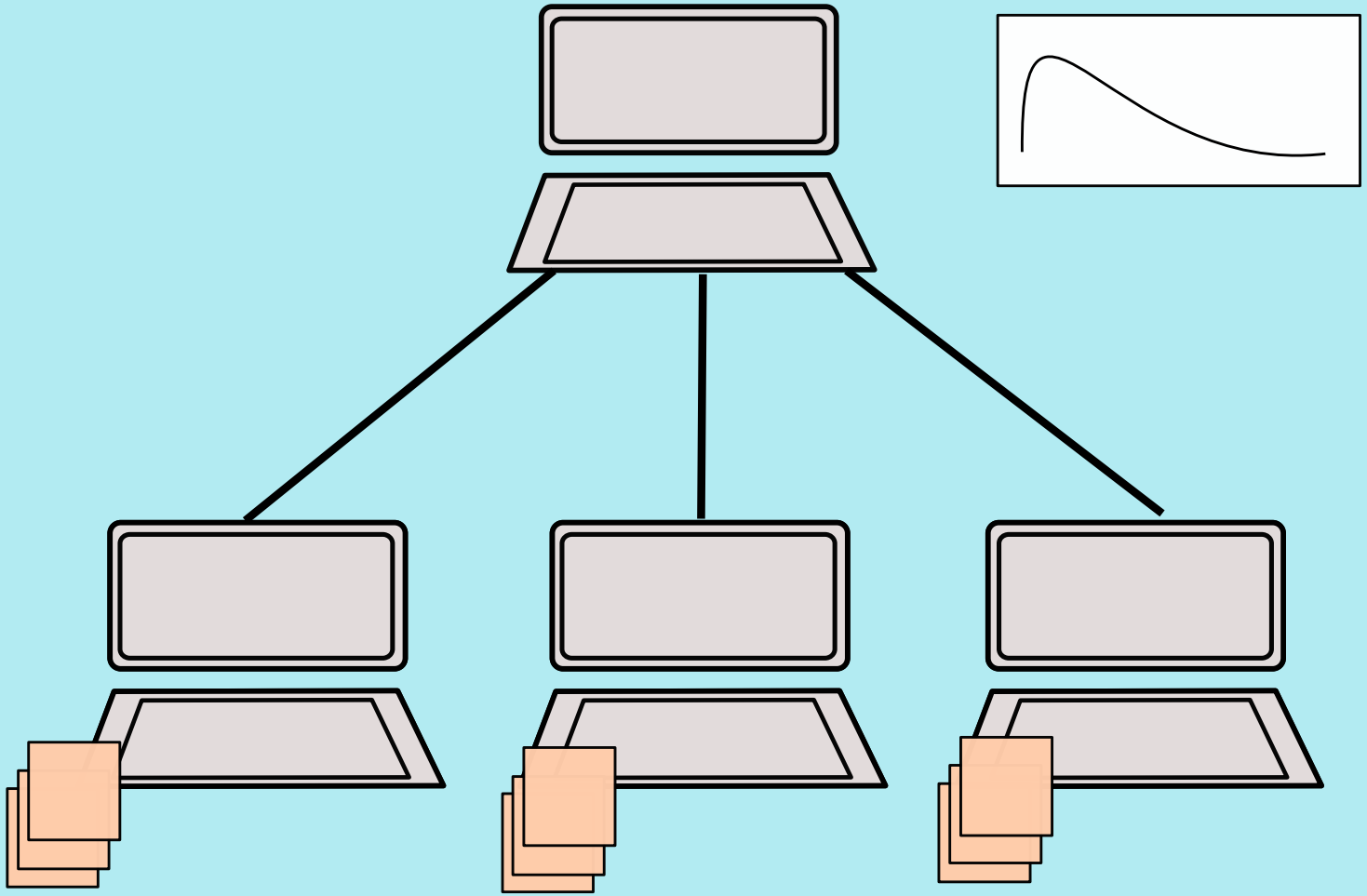
- TP: 40%
- Projet divisé en 7 exercises

Cours	Dates
Cours 1	12 septembre
Cours 2	13 septembre
Cours 3	19 septembre
Cours 4	20 septembre
Cours 5	26 septembre
Cours 6	3 octobre

Travaux pratiques

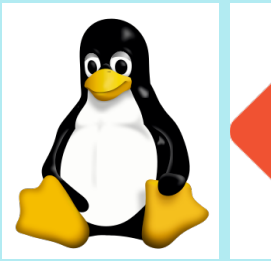
- Groupes: A, B, C, D
- Programmation en binôme

Travaux pratiques



Donnez des exemples de logiciels écrits en C

C est un langage de programmation



- impératif (procédural)
- structuré
- compilée

Remarque: Pas de classes (Ce n'est pas un langage de programmation orienté-objet!!!)

C

- **Auteur** Dennis Ritchie, Bell Labs
- **Date de première version** en 1972

Bonjour le Monde!!

```c
/* Fichier: bonjour1.c
* affiche 'Bonjour le Monde!!!' à l'écran.
* auteur: John Samuel
* Ceci est un commentaire sur plusieurs lignes
*/

#include <stdio.h> // headers

// Ceci est un commentaire sur une ligne
int main() {
  printf("Bonjour le Monde !!!");
  return 0;
}
```

Bonjour le Monde!!

```c
/* Fichier: bonjour2.c
* affiche un message à l'écran en utilisant un variable
* auteur: John Samuel
* Ceci est un commentaire sur plusieurs lignes
*/

#include <stdio.h> // headers

int main() {
  int year = 2017; //déclaration d'un variable
  printf("Bonjour le Monde!!! C'est l'annee %d", year);
  return 0;
}
```

La compilation

```
$ gcc bonjour1.c
```

L'exécution

```
$./a.out
Bonjour le Monde!!!
```

La compilation

```
$ gcc -o bonjour bonjour2.c
```

L'exécution

```
$./bonjour
Bonjour le Monde!!! C'est l'annee 2017
```

Commentaires sur une ou plusieurs lignes

```
// Ceci est un commentaire sur une ligne


/* Ceci est un
 * commentaire sur
 * quatre lignes
 */
```

Les types de base

Types	Mots clés	Exemples
caractères	char	'h', 'a', ...
entiers	short, int, long, long long	...,-1,0,1,...
nombres en flottant	float, double, long double	3.14, 3.14e23
énumérations	enum	ETUDIANT, STAGIAIRE

Les types caractères et entiers

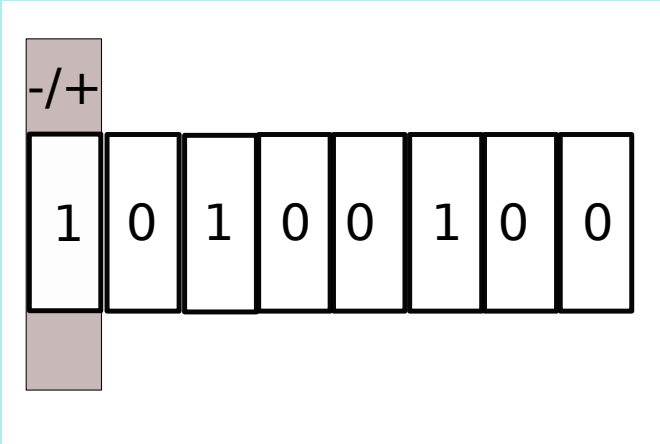
Types	Mots-clés
caractères	signed char, unsigned char
entiers	signed short, signed int, signed long, signed long long, unsigned short, unsigned int, unsigned long, unsigned long long

La taille des types de base n'est pas standardisée!

Declaration des variables

```
char my_char_var1 = 'a';
char my_char_var2 = -125;
unsigned char my_char_var3 = 225;
```

Remarque: Remarquez-bien l'utilisation de sous-tiret en nommant les variables



Declaration des variables

```
char my_char_var = 'a';
unsigned char my_uchar_var = 234;
short my_short_var = -12;
unsigned short my_ushort_var = 65535;
int my_int_var = 12;
unsigned int my_uint_var = 3456;
long my_long_var = -1234553L;
unsigned long my_ulong_var = 234556UL;
long long my_llong_var = 1123345LL;
unsigned long long my_ullong_var = 1234567ULL;
float my_int_var = 3.14;
double my_uint_var = 3.14E-12;
long double my_long_var = 3.14E-22;
```

énumrérations

```
enum status {ETUDIANT, STAGIAIRE};
enum status s = ETUDIANT;
enum status {ETUDIANT=1, STAGIAIRE};
enum boolean {FAUX=0, VRAI};
```

Remarque: enum: unsigned int

L'intervalle minimum et maximum de types de base en utilisant limits.h

Mots clés	Intervalle
signed char	[SCHAR_MIN, SCHAR_MAX]
unsigned char	[UCHAR_MIN, UCHAR_MAX]

L'intervalle minimum et maximum

Mots clés	Intervalle
(signed) short int	[SHRT_MIN, SHRT_MAX]
unsigned short int	[0, USHRT_MAX]
(signed) int	[INT_MIN, INT_MAX]
unsigned int	[0, UINT_MAX]
(signed) long	[LONG_MIN, LONG_MAX]
unsigned long	[0, ULONG_MAX]
(signed) long long	[LLONG_MIN, LLONG_MAX]
unsigned long long	[0, ULLONG_MAX]

L'intervalle minimum et maximum de types flottant en utilisant float.h

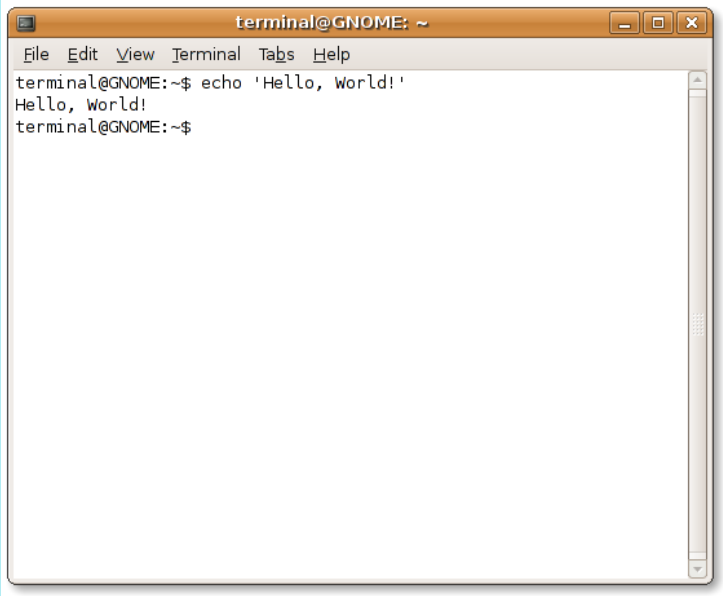
Mots clés	Intervalle
float	[FLT_MIN, FLT_MAX]
double	[DBL_MIN, DBL_MAX]
long double	[LDBL_MIN, LDBL_MAX]

sizeof

```
sizeof (char) //type
sizeof (my_uchar_var) //variable
```

Afficher à l'écran

```
printf("%d", my_int_var);
printf("%f", my_float_var);
```



printf: Afficher à l'écran

Mots clés	Code de conversion
char	c
unsigned char	hhu
short	hd
unsigned short	hu
int	d, i
unsigned int	u
long int	ld
unsigned long int	lu

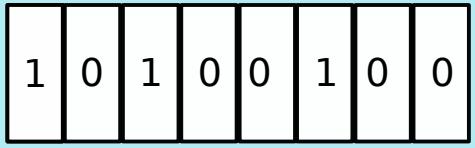
printf: Afficher à l'écran

Mots clés	Code de conversion
long long int	lld
unsigned long long int	llu
float	f, F
double	g, G
long double	Lf
string of characters	s

printf: Afficher à l'écran

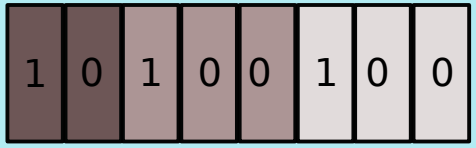
Character	Code de conversion
Retour-chariot	\n
Tabulation	\t

Notation binaire



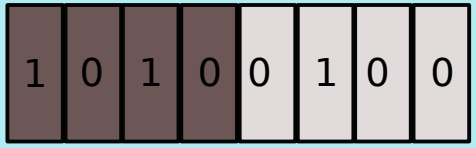

```
int valeur = 0b10100100;
```

Notation octale




```
int valeur = 0b10100100;
printf("notation octale: %o\n", valeur);
```

Notation héxadecimale




```
int valeur = 0b10100100;
printf("notation hexadecimale: %x\n", valeur);
```

Les opérateurs arithmétiques

Opérateur	Objectif
+	addition
-	soustraction
*	multiplication
/	division
%	modulus

Les opérateurs arithmétiques

```
int a = 20, b = 10;
```

Opérateur	Exemple	Résultat
+	a + b	30
-	a - b	10
*	a * b	200
/	a / b	2
%	a % b	0

Les opérateurs relationnels

Opérateur	Objectif
<	inférieur
<=	inférieur ou égale
>	supérieur
>=	supérieur ou égale
==	égale
!=	différent

Les opérateurs relationnels

```
int a = 20, b = 10;
```

Opérateur	Exemple	Résultat
<	a < b	0
<=	a <= b	0
>	a > b	1
>=	a >= b	1
==	a == b	0
!=	a != b	1

Les opérateurs logiques

```
int a = 20, b = 0;
```

Opérateur	Objectif	Exemple	Résultat
!	Négation	!a	0
&&	Et	a && b	0
||	Ou	a || b	1

Les opérateurs d'incrementation

```
int a = 20, b = 0;
```

Opérateur	Exemple	Résultat
a++	b = a++	a = 21, b = 20
++a	b = ++a	a = 21, b = 21
a--	b = a--	a = 19, b = 20
--a	b = --a	a = 19, b = 19

Les opérateurs de manipulation de bits

```
int a = 0x01000100;
```

Opérateur	Objectif	Exemple	Résultat
~	Négation	~a	0xfffffbb
&	ET	a & 0x4	0x4
|	OR	a | 0x2	0x46
^	XOR	a ^ 0x4	0x40
<<	décalage à gauche	a << 1	0x88
>>	décalage à droite	a >> 1	0x22

Les opérateurs d'affectation

```
int a = 20, b = 0;
```

Opérateur	Objectif	Exemple
=	equal	a = b
+=	addition	a += b
-=	substraction	a -= b
*=	multiplication	a *= b
/=	division	a /= b
%=	modulo	a %= b

Remarque: a op = b ::- a = a op b

Les opérateurs d'affectation

Opérateur	Objectif	Exemple
&=	ET	a &= b
|=	OU	a |= b
^=	XOR	a ^= b
<<=	décalage à gauche	a <<= b
>>=	décalage à droite	a >>= b

Remarque: a op = b ::- a = a op b

if

```
if (condition) {
 ...
}
```

if

```
int a = 20, b = 0;
if (a > b) {
 printf("a est supérieur à b");
}
```

Structures de contrôle: les branchements conditionnels

if

```
if (condition1) {
 ...
} else if (condition2) {
 ...
} else {
 ...
}
```

Remarque 1: Toutes les nombres non nulles sont ont considérées comme vrai ('TRUE')
Remarque 2: else est optionnel

if

```
int a = 20, b = 0;
if (a > b) {
 printf("a est supérieur à b");
} else if (a < b) {
 printf("a est inférieur à b");
} else {
 printf("a égale b");
}
```

switch

```
switch (expression) {
 case valeur1 : statements1
 case valeur2 : statements2
 ...
 default : statementsn
}
```

Remarque: expression doit être soit: char, short, int ou long

switch

```
int a = 20;
switch (a) {
 case 10 : statement1
  break;
 case 20 : statement2
 case 30 : statement3
  break;
 ...
 default : statementn
}
```

Remarque: Les deux instructions statement2 et statement3 seront exécutées.

Que sera-t-il affiché à l'écran suite à l'exécution de ce programme?

```
if (1) {
  printf("Hi");
} else {
  printf("Bonjour");
}
```

for

```
for(initialisation;condition;actualisation){
 ...
}
```

for

```
int a = 0;
for( a = 0; a > 10; a++){
 ...
}
```

for

```
int a = 0;
for(; a > 10; ){
 ...
}
```

Remarque: Une ou toutes les instructions d'initialisation, de condition ou d'actualisation peuvent être manquant.

for

```
int a = 0;
for( a = 0; a > 10; a++){
 ...
  a += 2 ;
 ...
}
```

while

```
while(condition){
  ...
}
```

while

```
int a = 20;
while(a > 0){
 ...
 a--;
 ...
}
```

while

```
int a = 0;
while(a < 20){
 ...
 a++;
 ...
}
```

do..while

```
do{
 ...
} while(condition);
```

do..while

```
int a = 20;
do{
 ...
  a --;
 ...
} while(a > 0);
```

do..while

```
int a = 0;
do{
 ...
  a ++;
 ...
} while(a < 20);
```

break

```
do{
 ...
 if (condition1) {
  ...
  break;
 }
 ...
} while(condition);
```

continue

```
do{
 ...
 if (condition1) {
  ...
  continue;
 }
 ...
} while(condition);
```

break

```
while(condition){
 ...
 if (condition1) {
  ...
  break;
 }
 ...
};
```

continue

```
while(condition){
 ...
 if (condition1) {
  ...
  continue;
 }
 ...
};
```

break

```
for(initialisation;condition;actualisation){
 ...
 if (condition1) {
  ...
  break;
 }
 ...
};
```

continue

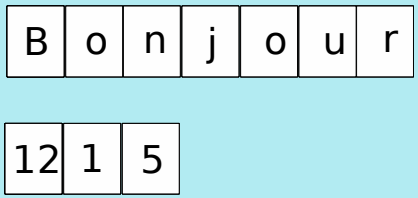
```
for(initialisation;condition;actualisation){
 ...
 if (condition1) {
  ...
  continue;
 }
 ...
};
```

Écrivez un programme qui affiche 0..20 en ordre croissant et décroissant utilisant for, while and do..while.

Écrivez un boucle infini en utilisant for, while, do..while.

Tableau

est un ensemble d'éléments homogènes



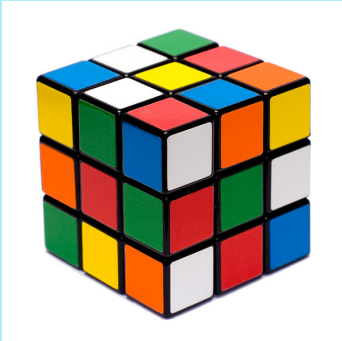
Tableaux de tableaux

Tableaux à deux indices

12	1	5
3	8	9

Tableaux de tableaux

Tableaux à plusieurs indices



Déclaration

```
char name[20];
```

Remarque: Langage C n'a pas un type nommé 'string'.

Tableaux d'entiers

Déclaration

```
int iarray[20];
float farray[20];
double darray[20];
```

Initialisation

```
int i;
int array[20];
for ( i = 0; i < 20; i++) {
 array[i] = i;
}
```

Initialisation

```
int prices[5] = { 11, 12, 13, 14, 15 };
int rooms[] = { 301, 302, 303 };
char message[] = "Bonjour Le Monde!!";
```

Remarque: Nous n'avons pas écrit la taille de rooms, message

Initialisation

```
int prices[2][2] = {
  {11, 12},
  {13, 14}
};
int rooms[][] = {
  {201, 202},
  {301, 302}
};
char message[2][8] = {"Bonjour", "Le Monde!!"};
```

Références

- Langage C, Claude Delannoy
- C
- C data types

Crédits d'images

- Wikimedia Commons