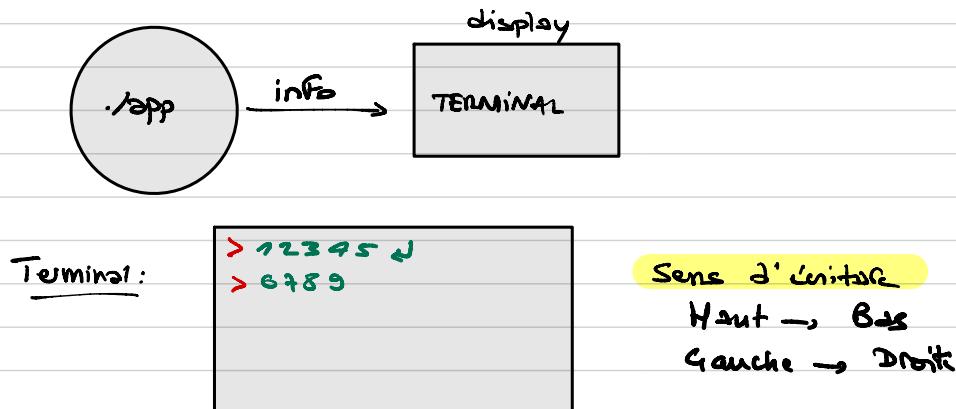


cons

17.8.22

Envoyer des informations à l'écran
 printf puts putchar



Éléments (ou informations) qui sont envoyés au terminal :
 que des caractères format ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Chr	Dec	Hx	Oct	Chr	Dec	Hx	Oct	Chr
0	0	000	NUL (null)	32	20	040	Space	64	40	100	Ø	96	60	140	'
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT (end of transmission)	36	24	044	\$	68	44	104	D	100	64	144	d
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS (backspace)	40	28	050	(72	48	110	H	104	68	150	h
9	9	011	TAB (horizontal tab)	41	29	051)	73	49	111	I	105	69	151	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE (data link escape)	48	30	060	Ø	80	50	120	P	112	70	160	p
17	11	021	DCL (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135]	125	7D	175	}
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

TD 20221017

juste le main.

Envoyer le texte
en utilisant la
fonction "Hello"
vers le terminal
printf

```
#include <stdio.h>

int main(int argc, char const *argv[])
{
    printf("Hello");
    return 0;
}
```

Ø : valeur numérique
= fin de chaîne

'\Ø'

il n'est pas
envoyé au terminal.

'H'	72
'e'	101
'l'	108
'l'	108
'o'	111
'\Ø'	

Hello

```
root@f81b319f0f31:/workspaces/Info1-TIN-A/TD/TD20221017# ./app
Hello
root@f81b319f0f31:/workspaces/Info1-TIN-A/TD/TD20221017#
```

printf ("Hello"); → printf ("Hello\n");

\n

saut de ligne (vers le bas)

```
root@f81b319f0f31:/workspaces/Info1-TIN-A/TD/TD20221017# ./app
Hello
root@f81b319f0f31:/workspaces/Info1-TIN-A/TD/TD20221017#
```

puts ("Hello"); ajoute systématiquement
le \n

```
printf("Hello");
printf("Hello\n");
puts("Hello");
```

printf

print

+ format

format → présentation des informations.

Quels types d'info peut-on afficher ?

- chaîne de caractères "Hello"
- valeur réelle 0.250000
- valeur entier. 42

ENTIER

printf(chaîne , valeur entier);

 / |
 "%d \n"

soit une variable x
une constante y
une valeur littérale 42

(TD)

1. créez une variable a de type int initialisée à la valeur de votre choix (42)

2. Affichez cette valeur à l'écran en utilisant %d

"[%d] \n"

[42]

"[%8d] \n"

[42]

← → 8 caractères

"[%8d] \n"

[42]

3. nouvelle variable b int -42
affichez b avec les formats %8d et %+8d.

4. Forcez l'affichage du signe pour b

%+8d

- + force l'affichage du signe

printf("[%8d]\n", a); → [+42]

printf("[%+8d]\n", b); → [-42]

[+42uuuuu] [-42uuuuu]

printf("%-+8d\n", a);
printf("%-+8d\n", b);

5. Modifiez le format pour afficher en un seul printf :

a = +42, b = -42

printf("a=%+d, b=%+d\n", a, b);

Formats pour les entiers :

%d
%u
%x or %X
%o

Sign' base 10
non sign' base 10
base 16 hexadecimal
base 8 octal.

6. Affichez la valeur de a. En 3 lignes

] en base 10 a = 42
] en base 8 a = 52
] en base 16 a = 2A

printf("a=%d\n", a);
printf("a=%o\n", a);
printf("a=%X\n", a);

7. Créez 3 variables entiers non signés

h = 2
m = 5
s = 8

Afficher h, m, s pour obtenir l'affichage suivant

heure : **02:05:08**

printf("heure : %02u : %02u : %02u\n", h, m, s);

Affichage des valeurs réelles

type	float	%f
	double	%ff

8. crée un valeur réelle de type double

$x = \text{valeur de } \pi$

affichez x au format [%f]

valeur de π : M_PI <math.h>

accessible si on ajoute dans le fichier Makefile

EXEC=app

CC=gcc

CFLAGS+= -std=c99 -Wall -g

CFLAGS+= -Iinclude

CFLAGS+= -D_XOPEN_SOURCE

LDLIBS+= -lm

math.h :

M_PI

3.14159265358979323846

[%f] → [x=3.141593]

6 chiffres après la virgule.

%f force l'affichage du signe

Format spécifiant la largeur totale et
le nombre de chiffres après la virgule

[%.8f] [x=3.14159265]

[%.14.8f] [x=3.14159265]

← 14 →

9. Créez une variable y de type double initialisée à la valeur 8.3 affichez y avec 20 chiffres après la virgule.

```
double y = 8.3;  
printf("y=%.*lf\n", y);
```

???

y=8.30000000000000071054

20 chiffres après
la virgule

```
float yf = 8.3;  
printf("y=%.*f\n", y);
```

yf=8.3000019073486328125

~~~~~