Cours INFO 1 – BOUCLES + OPTIMISATIONS 17.XI.2022

For : nbre itérations connu

3 parties :

① Initialisation
② Condition de maintien
③ Evolue du compteur

Bloc d'instructions concerné par la boucle.

```
for ( ① ; ② ; ③ )
{ ..... }
```

Formule de Leibniz pour le calcul de π

$$\pi \simeq 4 \cdot \sum_{n=0}^{N_{max}} \frac{(-1)^n}{2n+1}$$

① $n = 0$
② $n <= N_{max}$
③ $n += 1$

bloc : calculer $\frac{(-1)^n}{2n+1}$

accumuler ce calcul dans une variable "s"

En fin de boucle : afficher 4 x s

TD 20221114

Implémenter cet algorithme

Hyp : Nmax : constante 10'000

Liste des variables

	identificateur	type	valeur initiale
(const)	N_MAX	~~int~~ size_t	10000
	pi	double	0.
	s	double	0.
	n	~~int~~ size_t	0

Résultat pour Nmax = 10'000 : pi = 3.14169

Nmax	π	temps d'exécution	
10'000	3.14169	< 1 s	
10'000'000	"	≃ 1 s	
100'000'000	"	≃ 5 s	(0.25 s)
1'000'000'000	"	≃ 30 s (2086)	(2.34 s)

≃ 4 s

$$\pi \simeq 4 \sum_{n=0}^{N_{MAX}} \frac{(-1)^n}{2n+1}$$

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

int main(void)
{
    const int n_max = 1000000000;
    double pi_sur_4 = 0.;
    double pi = 0.;
    size_t n=0;

    for (size_t n = 0; n <= n_max; n++)
    {
        pi_sur_4 += pow((-1), n) / (2. * n + 1);
    }

    pi = 4. * pi_sur_4;

    printf("pi = %.8lf\n", pi);

    return 0;
}
```

size_t

N_MAX

① ② ③

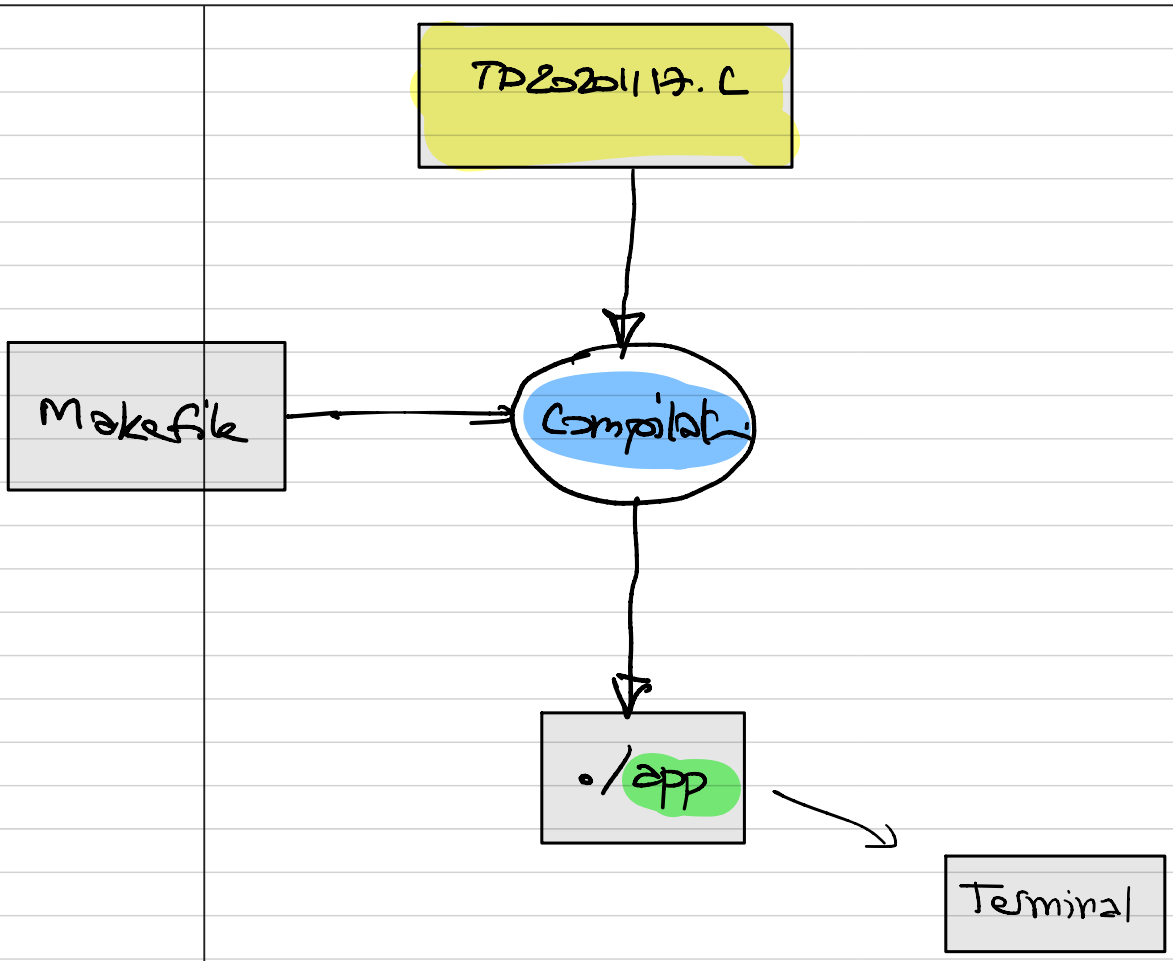
n	$(-1)^n$	2n+1
0	1	1
1	-1	3
2	1	5
3	-1	7
...	...	...

① Modif: optimisez pow(-1, n) 3s → 6s

② Modif : optimisez 2n+1 6s → 25s ?

```c
for (n = 0; n <= k_max; n++) {
    sum += num / den;
    num = -num;
    den += 2;
}
```

③ Modif : optimisez Makefile (-O3) 3.3s



Nom de l'exécutable

Nom du compilateur C

```
# set the executable name
EXEC=app

CC=gcc
CFLAGS+= -std=c99 -Wall -g
CFLAGS+= -Iinclude
CFLAGS+= -D_XOPEN_SOURCE
LDLIBS:= -lm

ODIR:=obj

SRC := $(wildcard *.c)
OBJS = $(patsubst %,$(ODIR)/%,$(SRC:.c=.o))
```

make clean
make app

-03