

# Concurrence

Quentin Fortier

August 22, 2024

## Définition

Un **programme** est une suite d'instructions dans un langage de programmation et stocké dans un fichier appelé **code source**.

# Processus et thread

## Définition

Un **programme** est une suite d'instructions dans un langage de programmation et stocké dans un fichier appelé **code source**.

## Définition

Un **processus** est une instance d'un programme en cours d'exécution. Il est composé d'un espace mémoire, d'un identifiant de processus (PID)...

# Processus et thread

## Définition

Un **programme** est une suite d'instructions dans un langage de programmation et stocké dans un fichier appelé **code source**.

## Définition

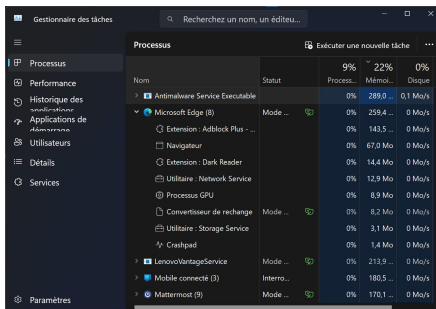
Un **processus** est une instance d'un programme en cours d'exécution. Il est composé d'un espace mémoire, d'un identifiant de processus (PID)...

## Définition

Un **thread** (ou **fil d'exécution**) est une unité d'exécution plus petite qu'un processus. Un processus peut contenir plusieurs threads qui partagent le même espace mémoire.

Les threads d'un même processus partagent la même zone mémoire.

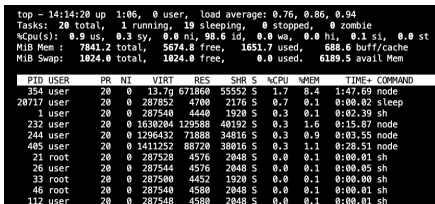
# Processus et thread



The screenshot shows the Windows Task Manager window with the 'Processus' tab selected. The window title is 'Gestionnaire des tâches'. A search bar at the top says 'Recherchez un nom, un éditeur...'. The left sidebar has icons for 'Processus', 'Performance', 'Historique des applications', 'Applications de démarrage', 'Utilisateurs', 'Détails', 'Services', and 'Paramètres'. The main area lists running processes with columns for 'Nom', 'Statut', and resource usage (CPU, Memory, Disk). The processes listed include 'Antimalware Service Executable', 'Microsoft Edge (8)', 'Extension : Adblock Plus - ...', 'Navigateur', 'Extension : Dark Reader', 'Utilitaire : Network Service', 'Processus GPU', 'Convertisseur de rechange', 'Utilitaire : Storage Service', 'Crashpad', 'LenovoVantageService', 'Mobile connecté (3)', and 'Mattermost (9)'.

Nom	Statut	99% Process...	22% Mémoi...	0% Disque
Antimalware Service Executable		0%	289.0 ...	0.1 Mo/s
Microsoft Edge (8)	Mode ...	0%	259.4 ...	0 Mo/s
Extension : Adblock Plus - ...		0%	143.5 ...	0 Mo/s
Navigateur		0%	67.0 Mo	0 Mo/s
Extension : Dark Reader		0%	14.4 Mo	0 Mo/s
Utilitaire : Network Service		0%	12.9 Mo	0 Mo/s
Processus GPU		0%	8.9 Mo	0 Mo/s
Convertisseur de rechange	Mode ...	0%	8.2 Mo	0 Mo/s
Utilitaire : Storage Service		0%	3.1 Mo	0 Mo/s
Crashpad		0%	1.4 Mo	0 Mo/s
LenovoVantageService	Mode ...	0%	213.9 ...	0 Mo/s
Mobile connecté (3)	Interro...	0%	180.5 ...	0 Mo/s
Mattermost (9)	Mode ...	0%	170.1 ...	0 Mo/s

## Gestionnaire des tâches de Windows

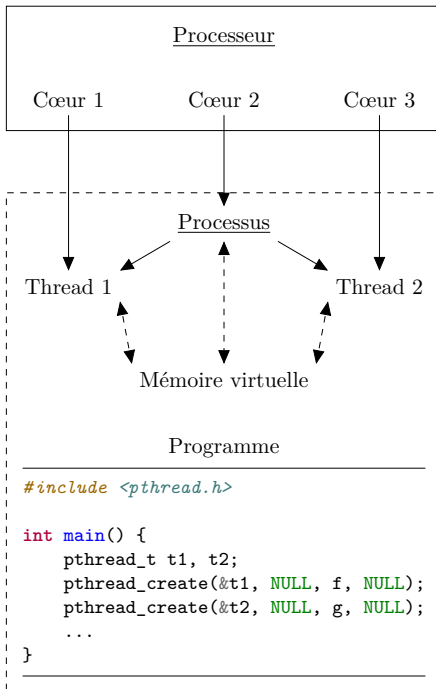


The screenshot shows a Linux terminal window with the output of the 'top' command. The first part shows system statistics: 'top - 14:14:20 up 1:06, 0 user, load average: 0.76, 0.86, 0.94'. The second part shows a table of running processes with columns: PID, USER, PR, NI, VIRT, RES, SHR, S, %CPU, %MEM, TIME+, and COMMAND.

```
top - 14:14:20 up 1:06, 0 user, load average: 0.76, 0.86, 0.94
Tasks: 20 total, 1 running, 19 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.9 us, 0.3 sy, 0.0 ni, 98.6 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st
MiB Mem: 7841.2 total, 5674.8 free, 1651.7 used, 688.6 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 6189.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
354	user	20	0	13.7g	671860	55552	S	1.7	8.4	1:47.69	node
20717	user	20	0	287852	4700	2176	S	0.7	0.1	0:00.02	sleep
1	user	20	0	287540	4440	1920	S	0.3	0.1	0:02.39	sh
232	user	20	0	1630204	129588	40192	S	0.3	1.6	0:15.87	node
244	user	20	0	1296432	71888	34816	S	0.3	0.9	0:03.55	node
405	user	20	0	1411252	88720	38016	S	0.3	1.1	0:28.51	node
21	root	20	0	287528	4576	2048	S	0.0	0.1	0:00.01	sh
26	user	20	0	287544	4576	2048	S	0.0	0.1	0:00.05	sh
33	root	20	0	287500	4452	1920	S	0.0	0.1	0:00.00	sh
46	root	20	0	287540	4580	2048	S	0.0	0.1	0:00.01	sh
112	user	20	0	287548	4580	2048	S	0.0	0.1	0:00.01	sh

## Commande top sous Linux/macOS



## Définition

**Programmation parallèle** : un programme exécute des threads sur plusieurs processeurs (ou cœurs) en même temps.

## Définition

**Programmation parallèle** : un programme exécute des threads sur plusieurs processeurs (ou cœurs) en même temps.

Même avec un seul cœur, on peut simuler le parallélisme en entrelaçant les instructions des différents processus ou threads.

Exemple : gestion des applications sur un système d'exploitation.



# Processus et thread : Concurrency et parallélisme

## Définition

**Programmation parallèle** : un programme exécute des threads sur plusieurs processeurs (ou cœurs) en même temps.

Même avec un seul cœur, on peut simuler le parallélisme en entrelaçant les instructions des différents processus ou threads.

Exemple : gestion des applications sur un système d'exploitation.

## Définition

**Programmation concurrente** : un programme exécute des threads en même temps ou en alternance.

# Processus et thread : Threads en C

---

```
#include <pthread.h> // threads POSIX (standard Linux)

void *f(void *x) {
    int *n = (int *)x; // Conversion du type
    for(int i = 0; i < 100000; i++)
        if(i % 20000 == 0)
            printf("%d %d\n", *n, i);
}

int main() {
    pthread_t t1, t2;
    int n1 = 1, n2 = 2;
    pthread_create(&t1, NULL, f, (void *)&n1);
    pthread_create(&t2, NULL, f, (void *)&n2);
    pthread_join(t1, NULL); // Attendre la fin de t1
    pthread_join(t2, NULL); // Attendre la fin de t2
}
```

---

Compilation : gcc -pthread exemple.c

# Processus et thread : Threads en OCaml

---

```
let f x = ...  
let t = Thread.create f x  
Thread.join t (* Attendre la fin de t *)
```

---

# Processus et thread : Threads en OCaml

## Exemple concret :

---

```
let f x =  
  Printf.printf "Thread %d\n" x;  
  for i = 0 to 2 do  
    Printf.printf "%d %d\n" x i  
  done  
  
let t1 = Thread.create f 1  
let t2 = Thread.create f 2  
Thread.join t1;  
Thread.join t2
```

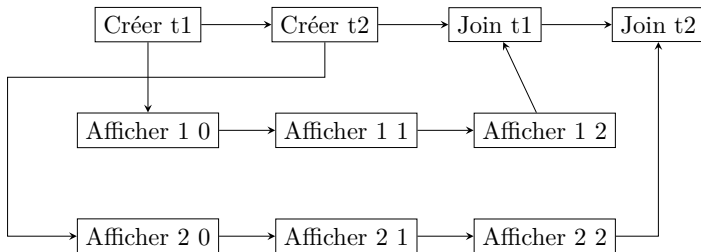
---

## Compilation :

```
ocamlopt -I +unix -I +threads unix.cmxa threads.cmxa  
exemple.ml
```

# Entrelacement : Graphe

On peut représenter les exécutions possibles du programme précédent par un graphe, où un arc  $u \rightarrow v$  signifie que  $v$  est exécuté après  $u$ .



## Définition

Une **trace** d'un programme est l'ordre dans lequel les instructions sont exécutées (qui respecte le graphe précédent).

Une trace possible :

Créer t1	Afficher 1 0	Créer t2	Afficher 2 0	Afficher 2 1	Afficher 1 1	...
----------	--------------	----------	--------------	--------------	--------------	-----

# Entrelacement : Trace

```
int counter;
void *increment(void *arg){
    for (int i = 1; i <= 1000000; i++) {
        counter++;
    }
}
int main(){
    counter = 0;
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
}
```

## Question

Quelle est la valeur de counter à la fin de l'exécution de main ?

# Entrelacement : Trace

```
int counter;
void *increment(void *arg){
    for (int i = 1; i <= 1000000; i++) {
        counter++;
    }
}
int main(){
    counter = 0;
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
}
```

## Question

Quelle est la valeur de counter à la fin de l'exécution de main ?

Pas forcément 2000000...



### Définition

Une opération est **atomique** si elle est exécutée en une seule fois, sans être interrompue.

### Définition

Une opération est **atomique** si elle est exécutée en une seule fois, sans être interrompue.

Une opération élémentaire (lecture ou écriture d'une variable de type `int` par exemple) est atomique.

## Définition

Une opération est **atomique** si elle est exécutée en une seule fois, sans être interrompue.

Une opération élémentaire (lecture ou écriture d'une variable de type `int` par exemple) est atomique.

`counter++` n'est pas atomique, car elle est équivalente à :

---

```
int tmp = counter;  
tmp = tmp + 1;  
counter = tmp;
```

---

# Entrelacement : Trace

---

```
int counter;
void *increment(void *arg){
    for (int i = 1; i <= 1000000; i++) {
        int tmp = counter;
        tmp = tmp + 1;
        counter = tmp;
    }
}
int main(){
    counter = 0;
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
}
```

---

# Entrelacement : Trace

- La valeur maximum de counter est 2000000, s'il n'y a pas d'entrelacement entre les counter++ des deux threads.
- La valeur maximum de counter est 2 avec la trace suivante :

