Sistemes Operatius II

Lluís Garrido

lluis.garrido@ub.edu

Grau d'Enginyeria Informàtica

Repàs a conceptes de Sistemes Operatius I

- Conceptes a repassar
  - Què és un sistema operatiu ?
  - Què és una crida a sistema i com es realitza ?
  - Què és una interrupció de maquinari i de programari ?
  - Què és, a una CPU, el mode kernel i d'usuari d'execució ?
  - Què és un programa i què és un procés ?
  - Què és la memòria virtual ?
  - Què és un canvi de context i quan es produeix ?
  - Què és un fil d'execució ?

Què és un sistema operatiu ?

- Un sistema operatiu és un programa que s'executa en engegar l'ordinador.
- Aquest programa s'encarrega de gestionar els dispositius d'un ordinador (memòria, discos, teclat, pantalla, xarxa, etc).
- També s'encarrega de gestionar els processos que hi executen.
- El sistema operatiu proporciona al programador d'una interfície comuna per accedir als dispositius i gestionar els processos.

Què és una crida a sistema i com es realitza ?

- Una crida a sistema és una petició perquè el sistema operatiu realitzi una determinada activitat.
- Exemples de crida a sistema: obrir un fitxer, llegir de teclat, enviar dades per la xarxa, etc
- S'assembla molt a una crida usual a una funció, però és molt més costosa. Requereix:
  - Guardar l'estat actual de la CPU
  - Fer que el sistema operatiu prengui el control de la CPU
  - Executar el servei demanat
  - Tornar al punt inicial restaurant l'estat de la CPU

Què és una interrupció de maquinari i de programari ?

- Una interrupció fa que
  - El procés que s'executa a la CPU s'interrompi immediatament.
  - Es guarda l'estat actual de la CPU i el sistema operatiu pren el control d'aquesta.
  - S'executa el servei demanat i en finalitzar es restaura l'estat de la CPU.
- La interrupció de maquinari
  - És asíncrona: es pot produir en qualsevol moment. La pot produir el teclat, el ratolí, el rellotge, el controlador de xarxa, de disc, etc.
  - Es utilitzada pel sistema operatiu per gestionar de forma eficient els dispositius.
- La interrupció de programari
  - La produeix l'aplicació en realitzar una crida a sistema.

Què és, a una CPU, el mode kernel i d'usuari d'execució ?

- Mode usuari d'execució
  - Mode en què la CPU executa els programes d'usuari.
  - Es limiten les instruccions màquina que s'hi poden executar.
- Mode kernel d'execució
  - Mode en què la CPU executa el sistema operatiu.
  - Es pot executar qualsevol instrucció màquina.
- Com es passa de mode usuari a mode kernel ?
  - En produir-se una interrupció de maquinari o de programari es passa de mode usuari a mode kernel.
  - En retornar de la interrupció es passa de mode kernel a mode usuari.

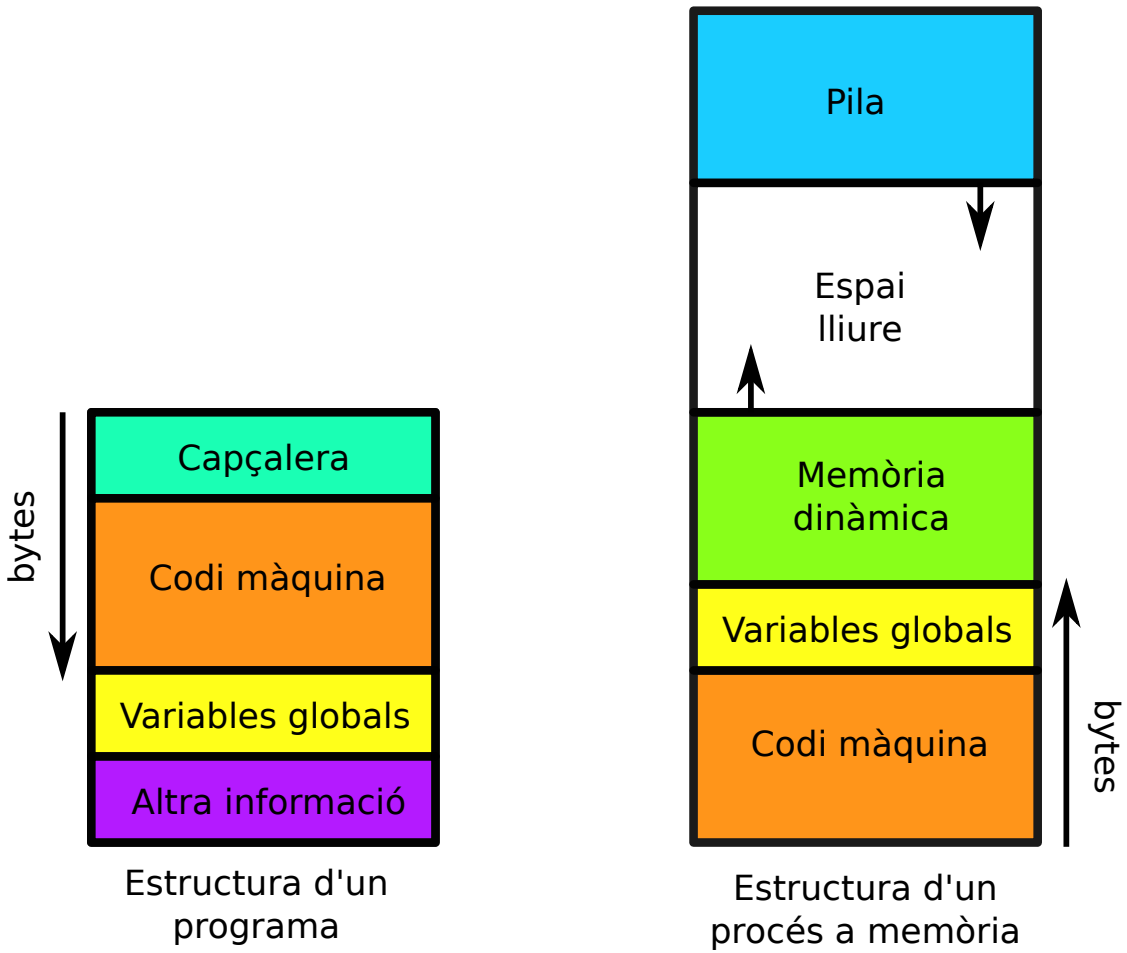
Què és un programa i què és un procés ?

- Un programa és el fitxer (executable) que es troba a disc. El fitxer conté bytes que corresponen a instruccions màquina que la CPU pot executar.
- Un procés és un programa que s'executa a la CPU
  - En fer doble clic a un programa, el sistema operatiu carrega en memòria els bytes del fitxer.
  - Al fitxer hi ha informació sobre quin és el punt d'entrada (main) del programa.
  - En finalitzar la càrrega de bytes, el sistema operatiu pot començar l'execució del programa al punt d'entrada.

Què és un programa i què és un procés ?

Mapa de memòria d'un procés

- La següent figura mostra a grans trets l'estructura d'un programa (a disc) i un procés (a memòria)



- Vegem l'estructura real d'un procés a Linux (codi `exemple1.c`)
  - Compileu el codi
  - Executeu el codi: el programa s'atura durant l'execució i us indica l'identificador de procés (PID).
  - Executeu des d'un altre terminal

    `cat /proc/PID/maps`

    on PID és el número imprès per pantalla.

Què és un programa i què és un procés ?

Mapa de memòria d'un procés

- On s'emmagatzema cada variable en aquest exemple ?
  - Regió global: variables globals
  - Regió de pila: variables locals a la funció i l'històric de les crides a funcions.
  - Regió de memòria dinàmica: la variable "vector" apunta a la zona de memòria dinàmica.

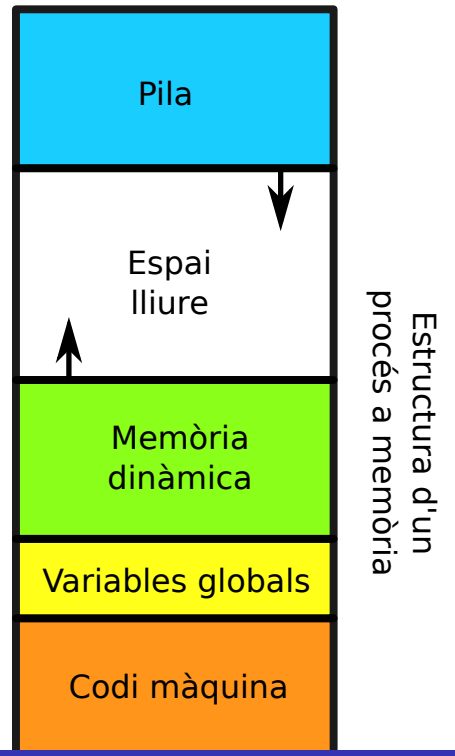
```c
int global;

int func(int a)
{
  int b;
  // Processament
}

void main(void)
{
  char str[10];
  int local, *vector;

  global = local = 0;

  // Processament
  func(2);
  // Reservar memoria
  vector = malloc(1000);
```



Què és un programa i què és un procés ?

Mapa de memòria d'un procés: la realitat és força complexa...

- La memòria dinàmica pot créixer fins el total de memòria (es poden crear regions disjuntes en cas necessari).
- La pila també pot créixer però la seva mida està limitada (i no es poden crear regions disjuntes). Vegeu codi `exemple2.c`.

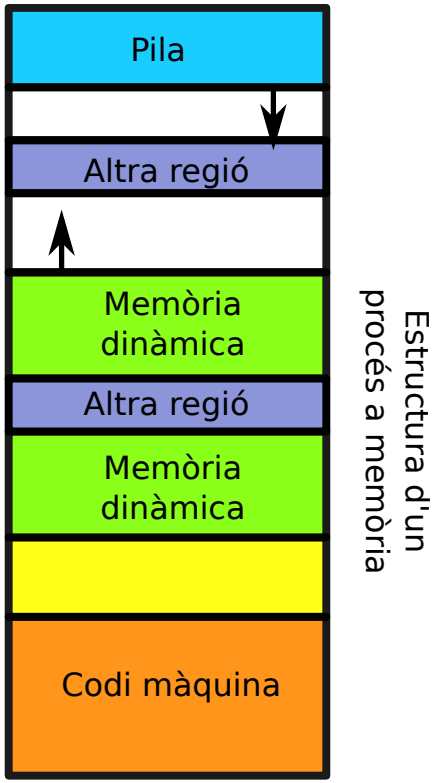
```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
   funcio_recursiva(0);
}

int funcio_recursiva(int n)
{
  int a[1000];

  printf("%d\n", n);

  funcio_recursiva(n+1);
}
```



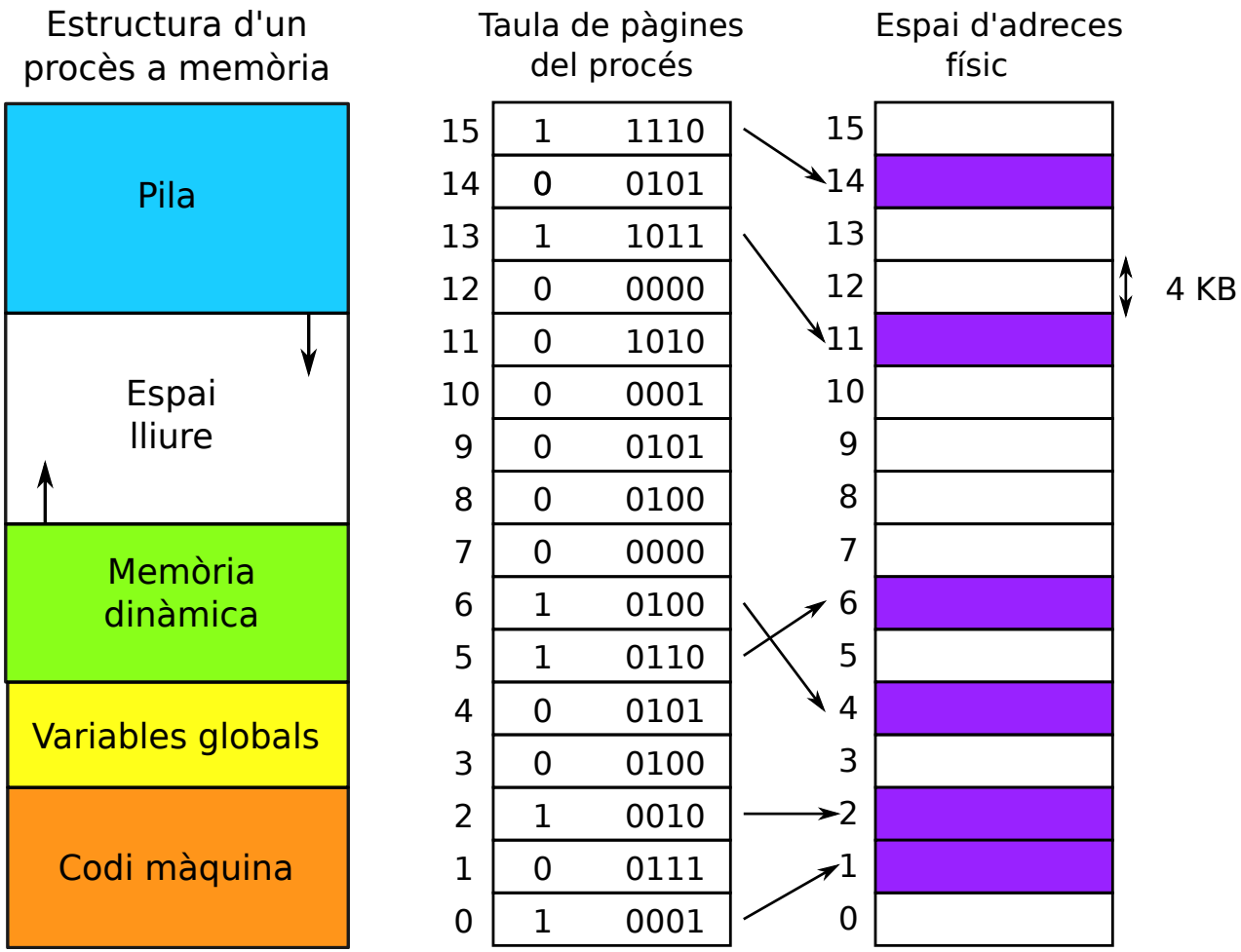
Què és la memòria virtual ?

- Als ordinadors actuals és una tècnica de gestió de la memòria utilitzada per gestionar la memòria dels processos.
- Totes les adreces de memòria generades pels processos (inclòs el comptador de programa) són virtuals i es mapen a adreces físiques mitjançant una taula.
- Aquesta és la sortida per pantalla de `exemple1.c`

```
PID del procés 3687
Funcio main:  40074c
Variable global:  60105c
Variable local:  7fff8ca97a7c
Variable vector:  7fff8ca97a70
Polsa enter per continuar.
```

Què és la memòria virtual ?

- Mapeig d'una adreça virtual a una adreça física (arquitectura de 16 bits)

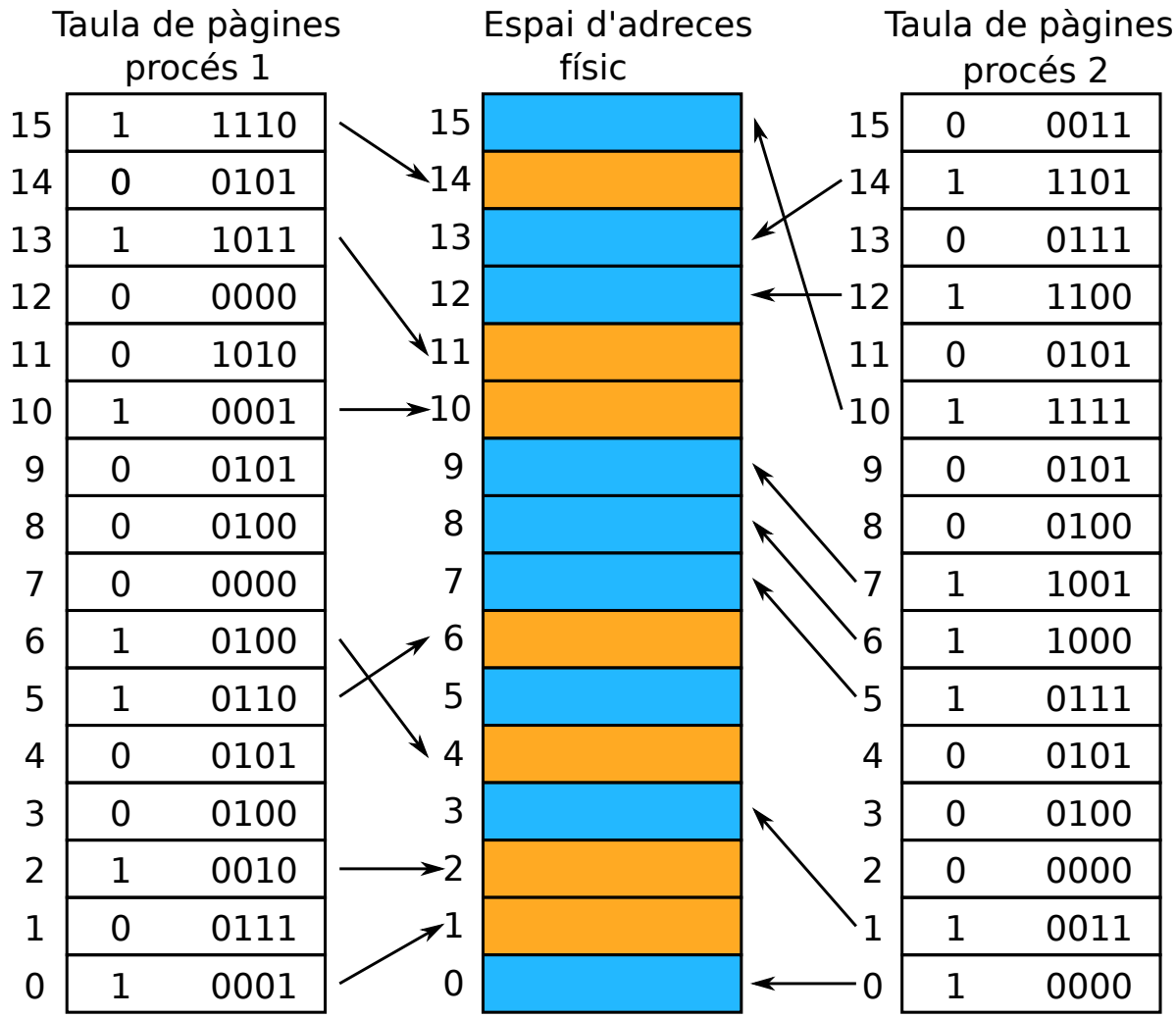


Què és la memòria virtual ?

- A l'exemple anterior
  - L'adreça virtual 0000 1111 0000 1111 es mapa a 0001 1111 0000 1111.
  - L'adreça virtual 0001 1111 0000 1111 produeix una fallada de pàgina.
- La mapeig d'una adreça virtual a adreça física es realitza a nivell de maquinari.
- El sistema operatiu s'encarrega de gestionar les taules dels processos.
- Cada procés té disponible tot l'espai de memòria possible (4GB en sistemes de 32 bits).
- Posicions contigües a memòria virtual no tenen perquè ser contigües a l'espai físic.

Què és la memòria virtual ?

- Exemple amb dos processos: la memòria virtual permet gestionar la protecció de memòria entre aquests.



Què és un canvi de context i quan es produeix ?

- Un canvi de context és un procediment que realitza el sistema operatiu per canviar d'una tasca a una altra.
- En el context de Sistemes Operatius I, una tasca és equivalent a un procés.
- Els canvis de context permeten implementar la multi-tasca: tot i disposar només d'una CPU, sembla que les múltiples tasques s'executin en paral·lel.
- Es poden produir centenars de canvis de context per segon. Ho podeu veure amb la instrucció (des de terminal)

  ```
  $ vmstat
  ```

  (vegeu columna cs)

Què és un canvi de context i quan es produeix ?

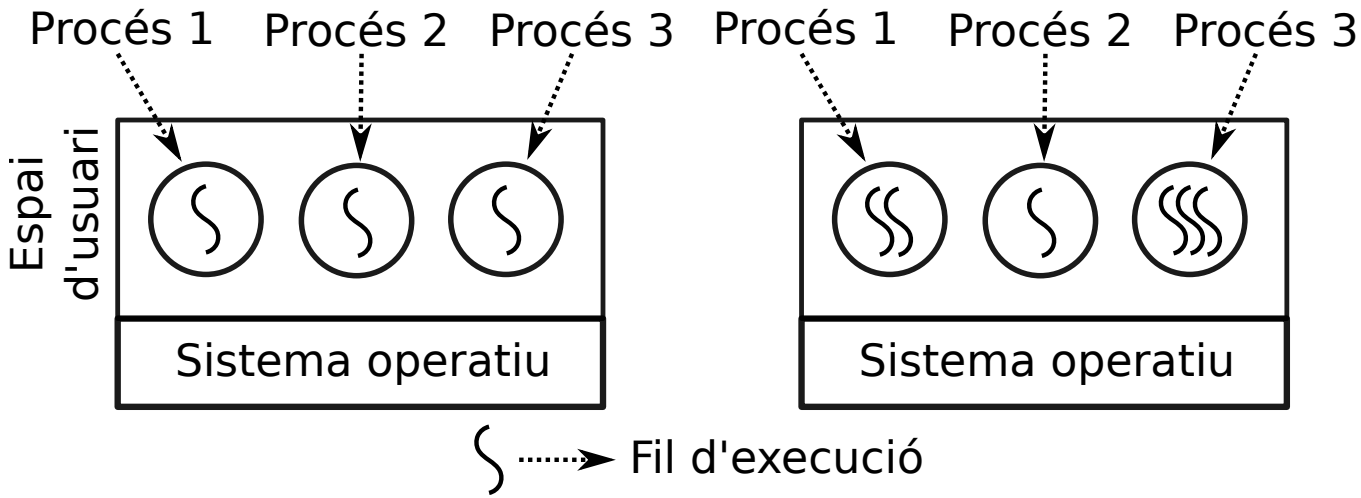
- Un canvi de context es pot produir per diverses raons. Per exemple:
  1. La tasca que executa ha finalitzat la seva llesca de temps.
  2. La tasca que executa realitza una crida a sistema que la fa bloquejar (es posa a dormir, realitza una operació d'entrada-sortida, ...)
  3. Una altra tasca més prioritària requereix l'ús de la CPU.

- Per realitzar un canvi de context cal
  1. Emmagatzemar a memòria els registres de la CPU (comptador de programa, punter a la pila, etc)
  2. Carregar a la CPU els nous registres

  només el sistema operatiu pot fer-ho.

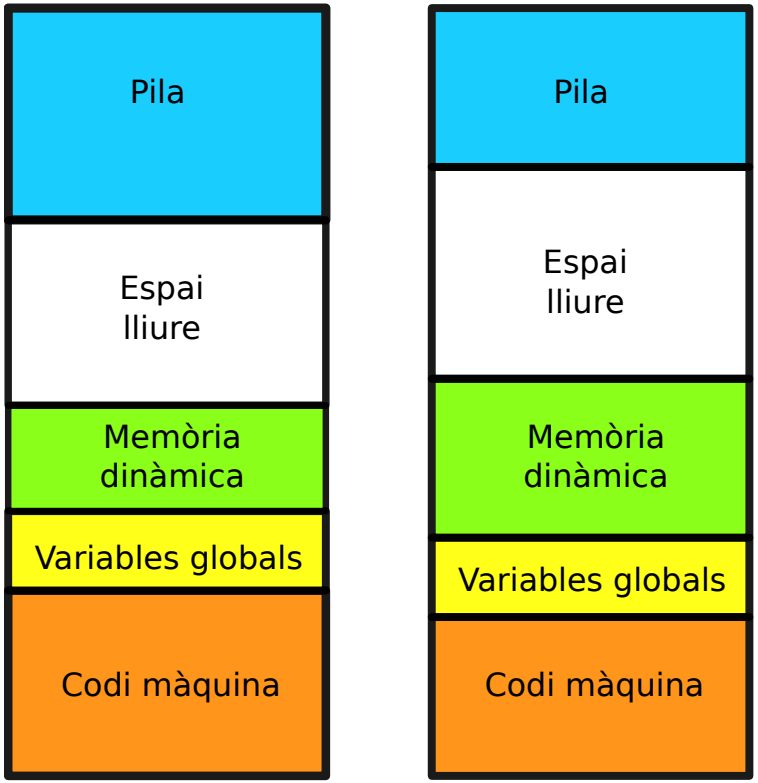
Què és un fil d'execució ?

- El fil és un concepte associat a un procés:
  - Un procés és un programa que s'ha carregat a memòria.
  - Un fil és l'entitat que s'executa a la CPU.
- Tot procés té per defecte (en engegar-se) un fil d'execució.
- El terme multi-fil s'utilitza per denotar que hi ha múltiples fils d'execució en un sol procés.
- En anglès un "fil" és un "*thread*" (no "*threat*", que vol dir "amenaça").



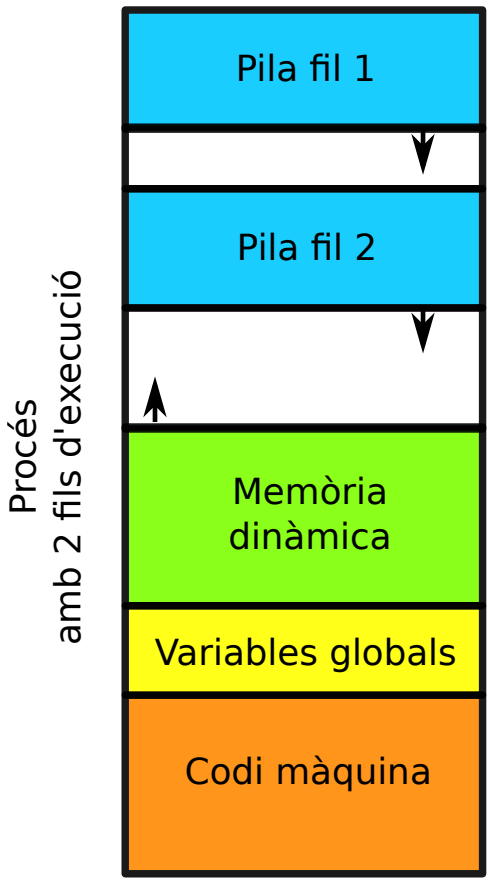
Què és un fil d'execució ?

- Cada procés té un espai de memòria independent.
- Els processos, per defecte, no poden escriure a l'espai de memòria d'un altre procés.



Què és un fil d'execució ?

- Els fils d'un procés comparteixen l'espai de memòria del procés (entre altres coses).
- Cada fil té la seva pròpia pila i registres de la CPU: cada fil pot executar doncs una part (diferent) de l'aplicació.




```c
int global;

void funcio()
{
  int a;

  // Aixo ho executen els dos fils
}

void main(...)
{
   int fil;

   // Aixo ho executa un sol fil, una sola pila

   fil = crear_fil_nou(&funcio); // Retorna id fil creat

   // Aixo ho executa un sol fil, hi ha dues piles

   funcio();

   esperar_que_acabi_fil(fil);
}
```

Què és un fil d'execució ?

- Elements d'un procés
  - Espai d'adreces (codi, variables globals, ...)
  - Fitxers oberts
  - La llista de fils del procés
  - Altres
- Elements propis de cada fil
  - Comptador de programa
  - Registres de la CPU
  - Pila
  - Estat (preparat, bloquejat, execució)

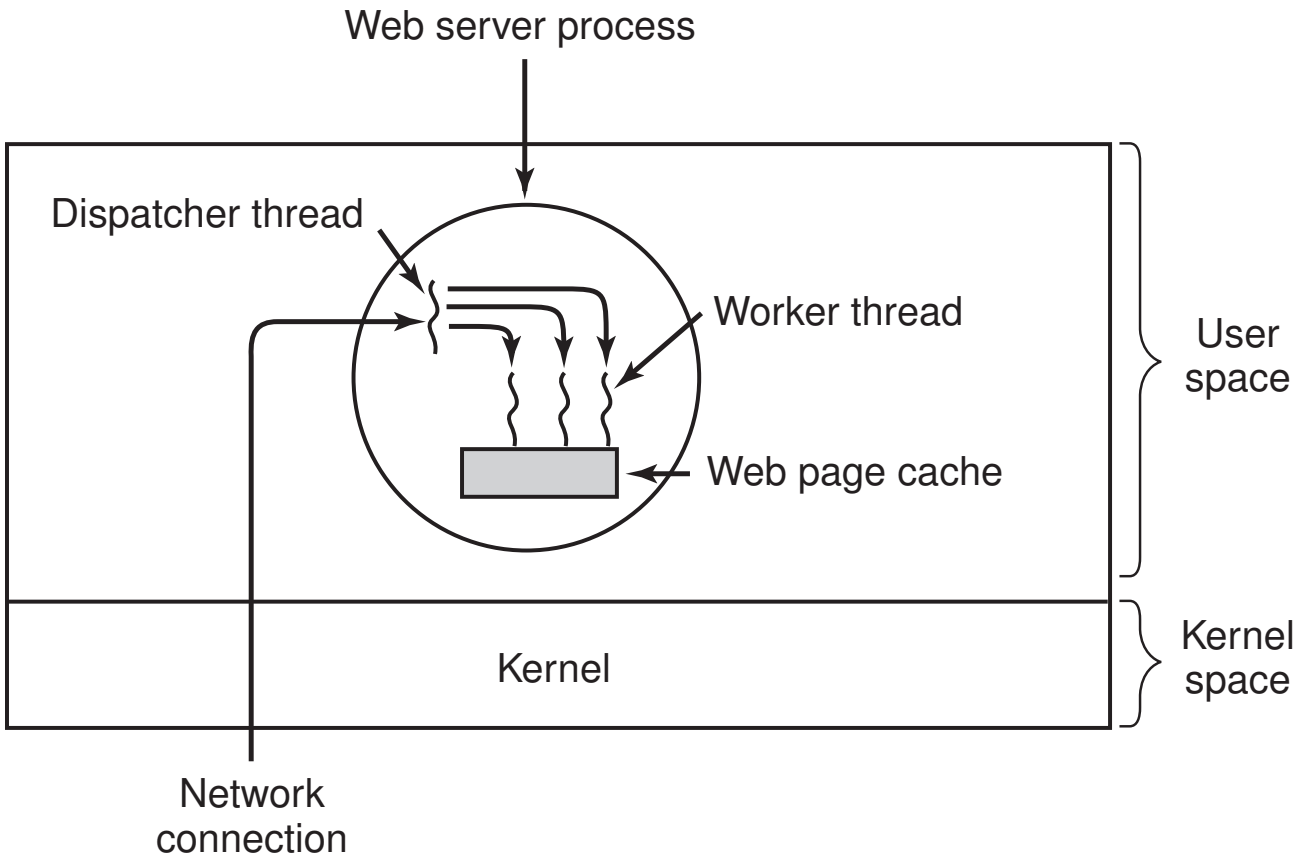
Què és un fil d'execució ?

- Cada fil té el seu propi estat
  - Un fil pot estar preparat, bloquejat o en execució.
  - El sistema operatiu fa els canvis de context a nivell de fil, no de procés.
- El mateix codi (inclús la mateixa línia de codi) pot ser executat per dos o més fils a la vegada.
- Els fils poden utilitzar l'espai d'adreces compartit (variables globals, memòria dinàmica) per intercanviar informació.
- Els fils es poden crear i destruir a nivell d'usuari.

- Quina utilitat tenen doncs, els fils ?

Per a què serveixen els fils ?

- Els fils permeten realitzar programes més eficients.
- Exemple: un servidor web (esquema del llibre de Tanenbaum)
  - 1 fil "dispatcher": escolta peticions del clients
  - 3 fils "worker": processen les peticions dels clients



Per a què serveixen els fils ?

- Moltes de les aplicacions que executen al nostre ordinador són multifil (processador de textos, navegador d'Internet, ...)
- Exemple: processador de textos (esquema del llibre de Tanenbaum)
  - 1 fil que controla l'entrada per teclat
  - 1 fil que que auto-emmagatzema el document a disc
  - 1 fil que dibuixa el text a pantalla
  - i més fils...