Conceptes Avançats de Sistemes Operatius

Facultat d'Informàtica de Barcelona
Dept. d'Arquitectura de Computadors

Curs 2013/14 Q1

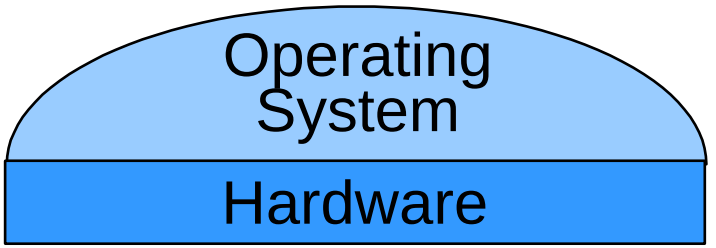
Abstraccions del Sistema Operatiu

Índex

- Què és un Sistema Operatiu?
- Abstraccions del sistema operatiu
- Fluxos de sistema
  - Mach
  - Linux
- Fluxos d'usuari: Pthreads
- Gestió de memòria

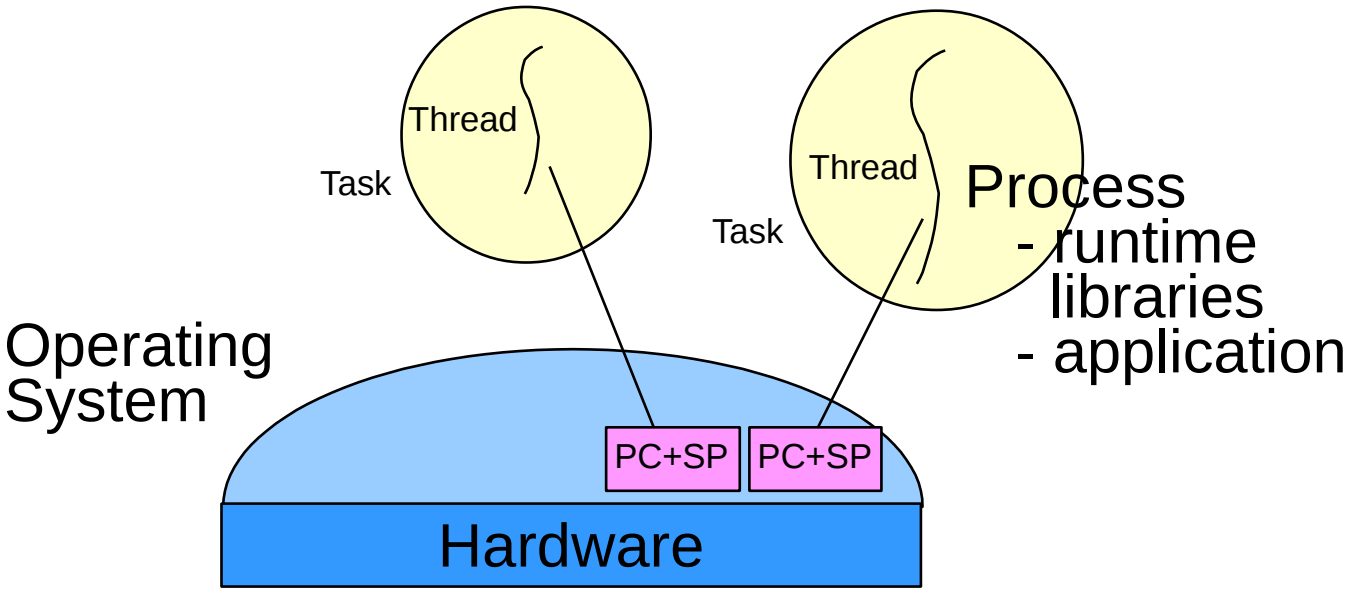
Sistema Operatiu

- Intermediari entre l'usuari i la màquina
- Proporciona un entorn d'execució entre convenient i eficient per executar programes
  - Servidors / sobre-taula / portàtils / mòbils
- Gestiona la màquina
- Ofereix protecció entre usuaris



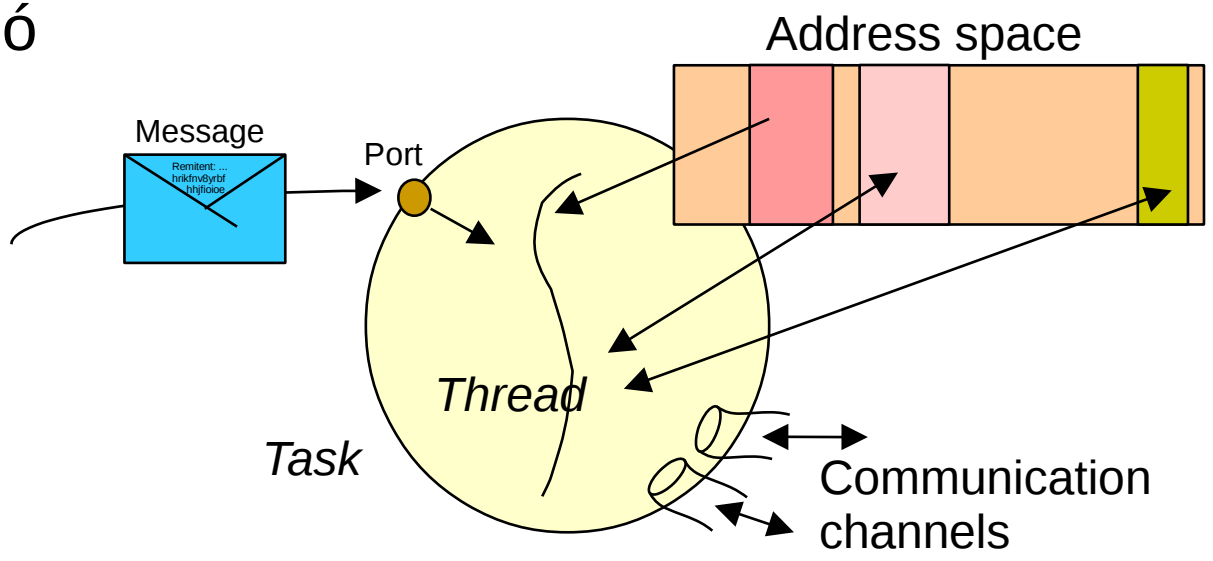
Abstraccions del Sistema Operatiu

- Entorn d'execució
  - Hardware
  - Sistema operatiu
  - Llibreries de suport
  - Aplicacions



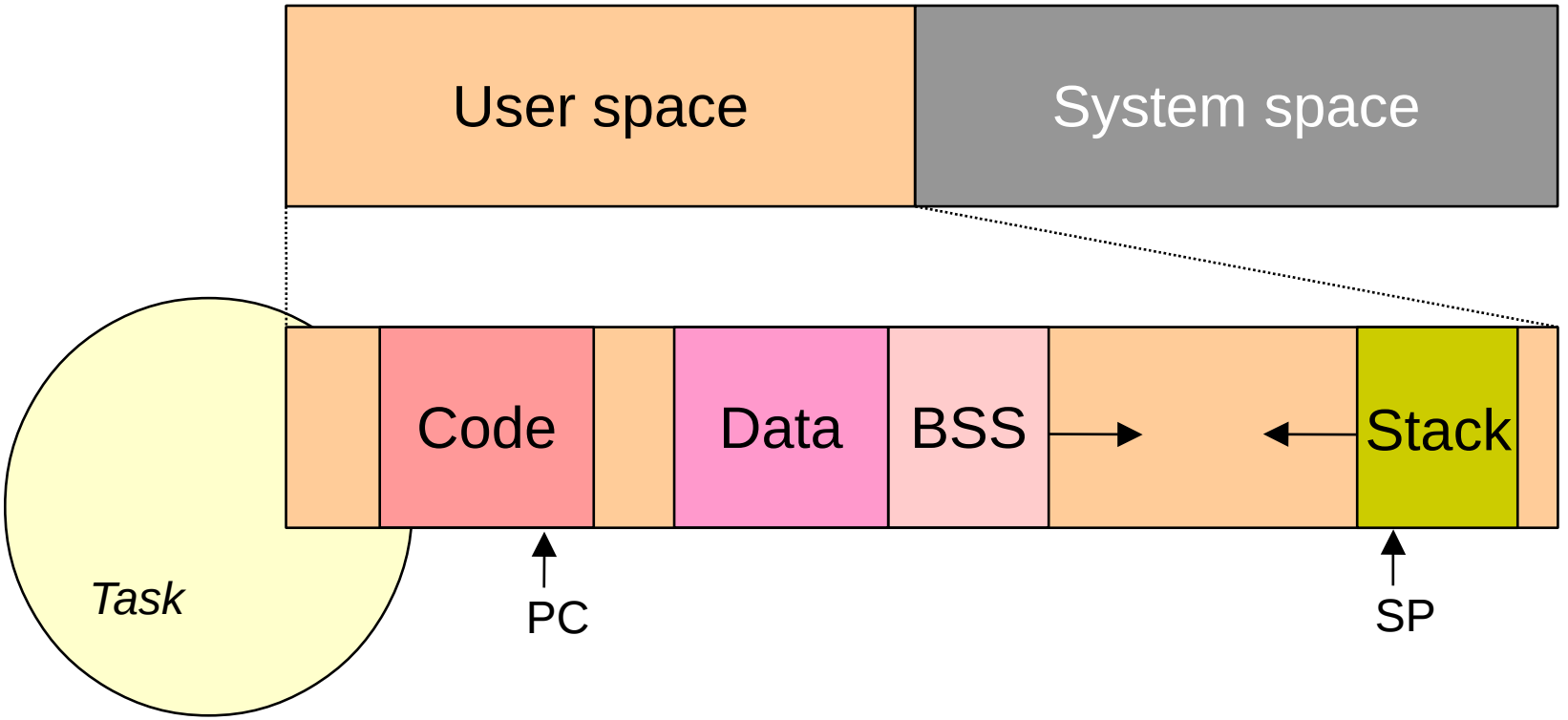
Abstraccions del Sistema Operatiu

- Definició de procés
  - Unitat d'assignació de recursos
  - Entorn d'execució, del qual formen part...
    - Un (o més) espais d'adreces
    - Canals de comunicació / ports
    - Fluxos d'execució



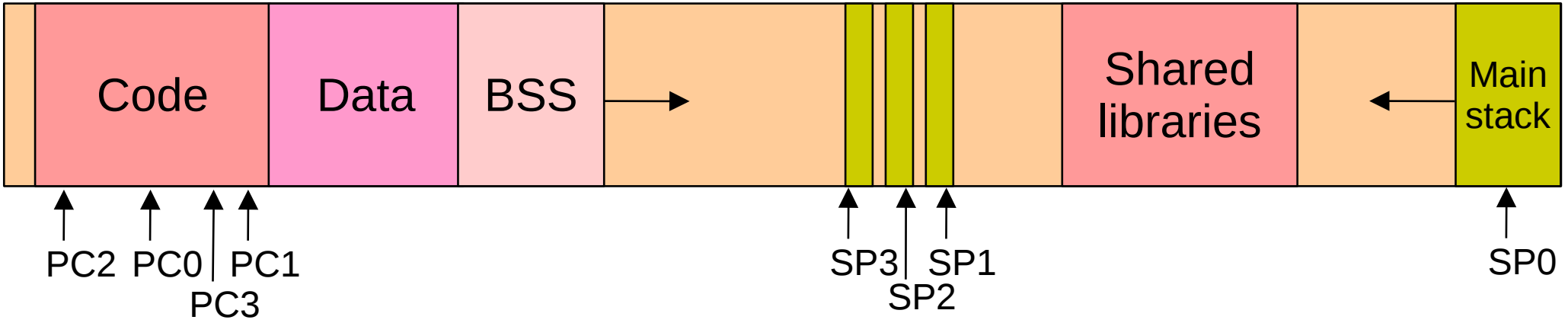
Abstraccions del Sistema Operatiu

- Espai d'adreces
  - Espai virtual adreçable, conté codi, dades (data i BSS) i pila



Abstraccions del Sistema Operatiu

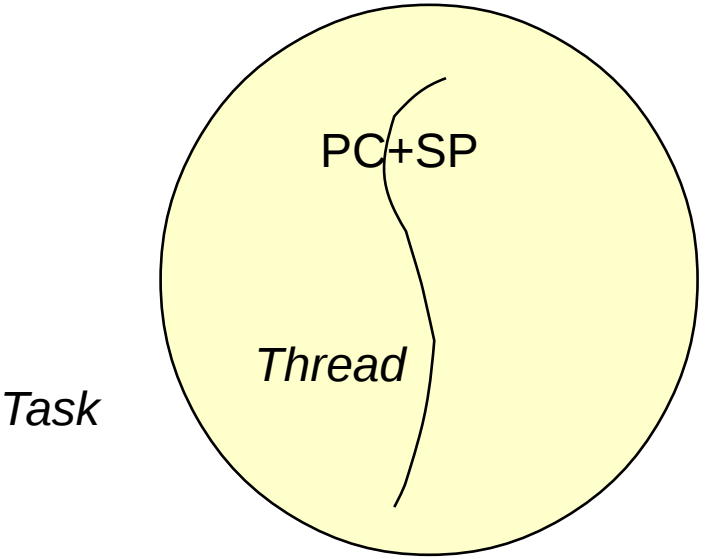
- Amb fluxos, l'estructura genèrica d'un espai d'adreces és:



  - La pila “principal” creix automàticament, fins a:
    - ulimit -s (exemple: 8Mb)
  - Les piles dels altres fluxos no creixen
    - Es podrien fer crèixer atenent al signal SIGSEGV

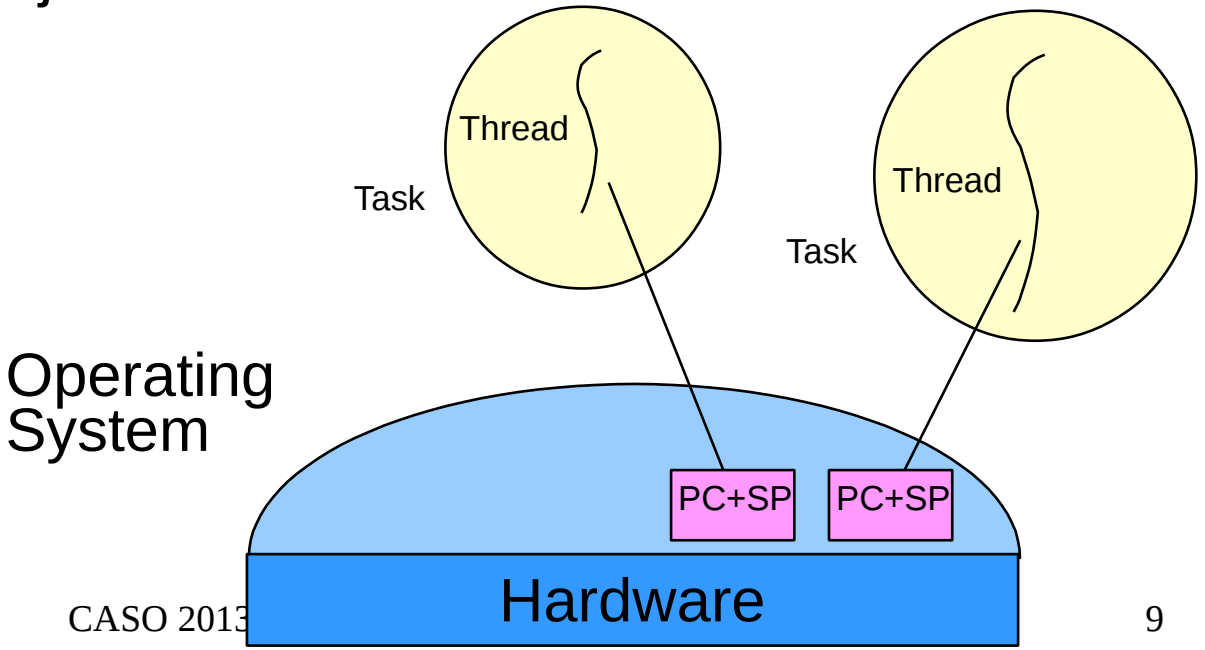
Abstraccions del Sistema Operatiu

- Flux d'execució
  - Mínim: PC + SP
  - S'afegeixen: els registres del processador
    - Registres de control, flags
    - Enters
    - Coma flotant
    - Extensions multimèdia
    - Extensions SIMD



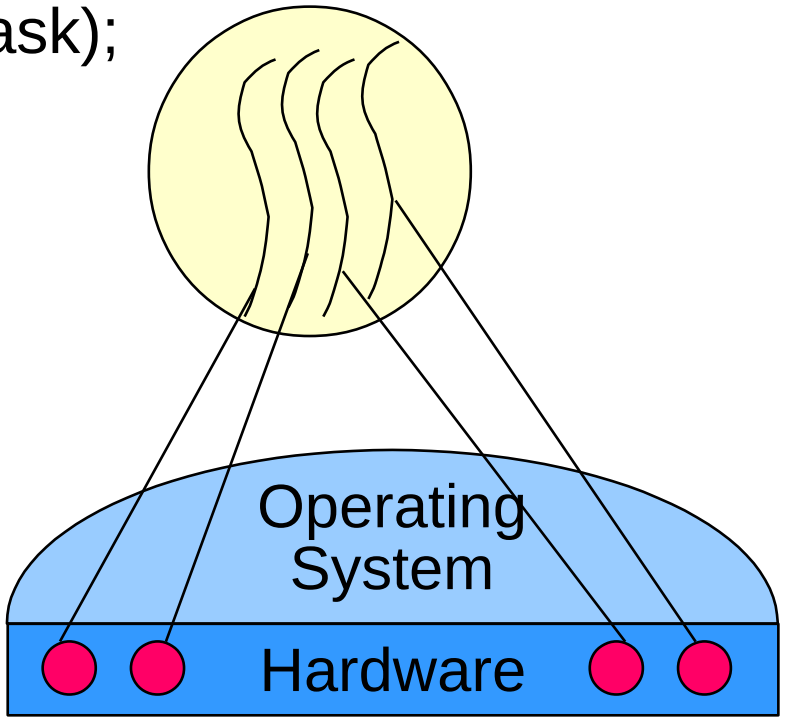
Abstraccions del Sistema Operatiu

- Propietats dels fluxos
  - De sistema i d'usuari
    - Prioritat: importància
    - Quantum: quantitat màxima de temps que haurà de passar mentre el flux s'estigui executant, abans que el sistema operatiu es planteji canviar de context



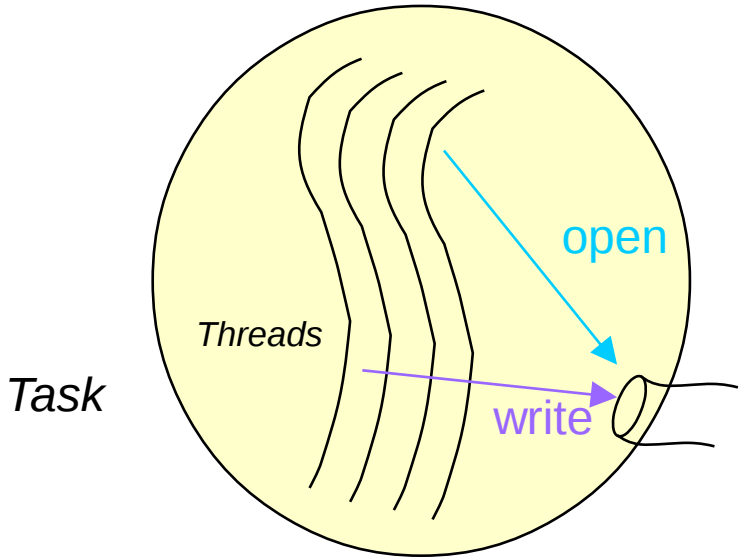
Fluxos de sistema

- Oferts per la interfície del sistema operatiu
  - Preempció: interrupció de rellotge
  - Assignables als processadors (cores) de la màquina
    - sched_setaffinity(pid, size, mask);



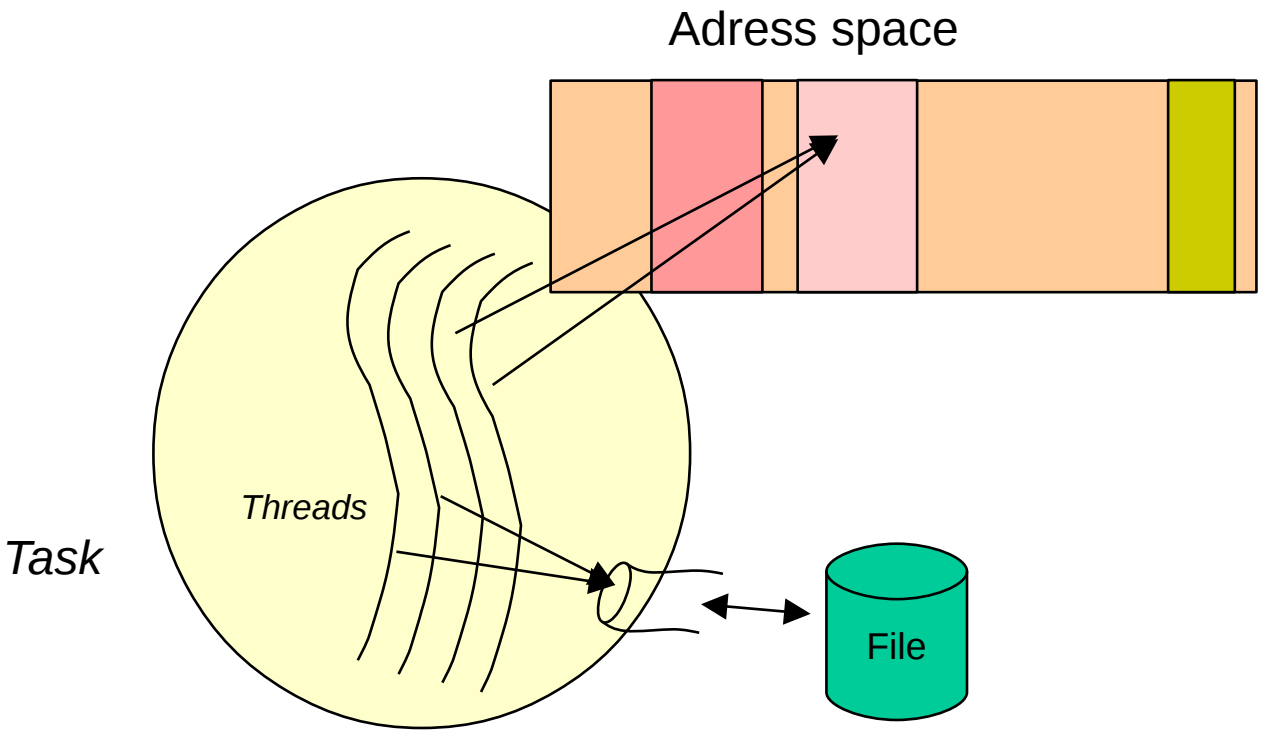
Fluxos de sistema

- Comparteixen tots els recursos del procés
- Poden demanar nous recursos pel procés
- Poden alliberar recursos



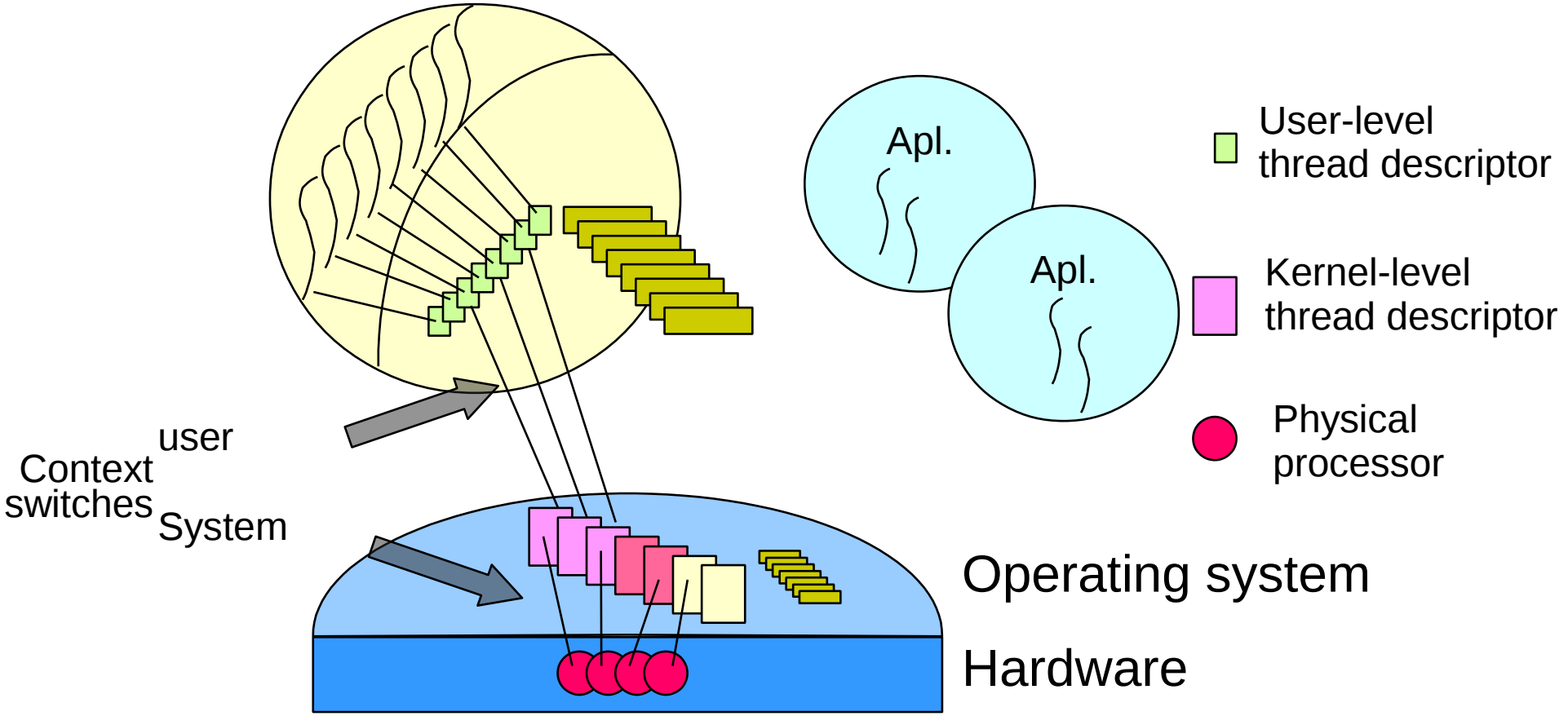
Fluxos de sistema

- Sincronització necessària per:
  - Accedir a dades compartides (usuari/sistema)
  - Accedir a recursos compartits (sistema)



Nivells de fluxos

- Entorn híbrid (N:M)
  - Gestionat per una llibreria de suport (Pthreads)
  - Processadors <= fluxos de sistema <= fluxos d'usuari

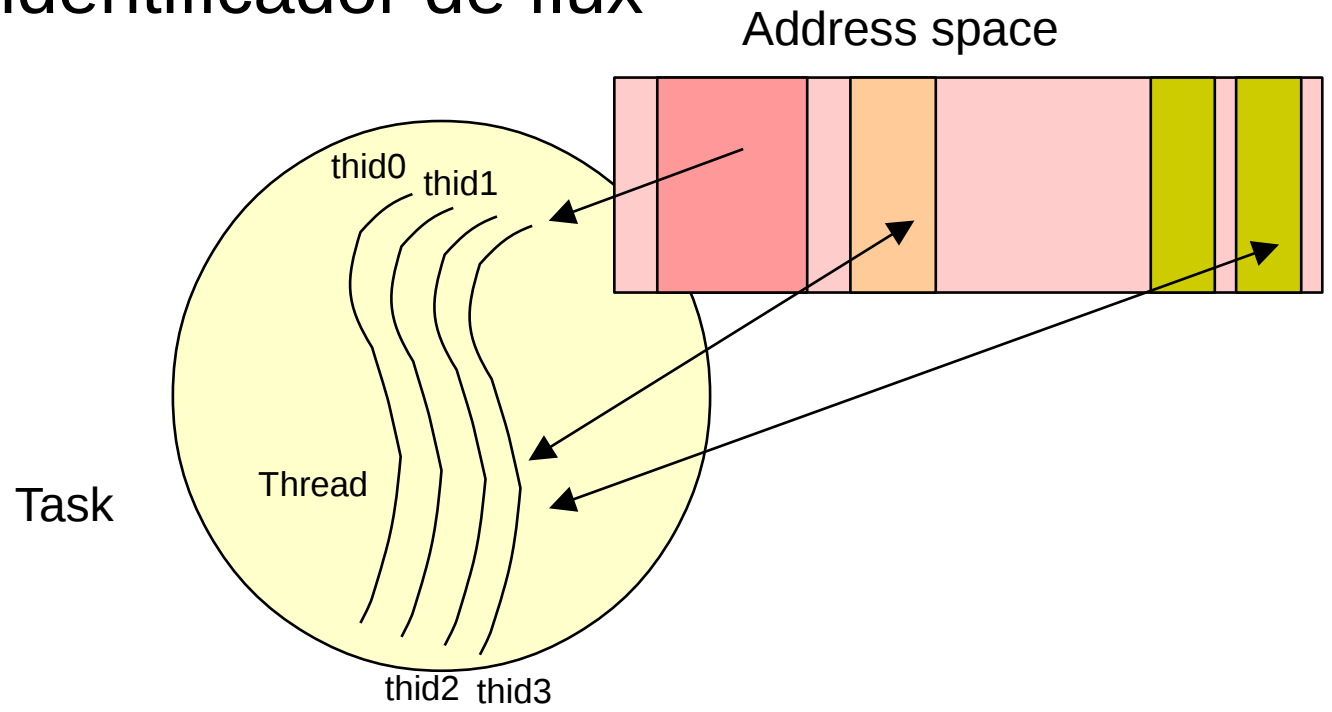


Fluxos de sistema

- Avantatges (comparats amb els processos)
  - Poden explotar paral·lelisme dins de les aplicacions
  - Lleugers
    - Canvi de context més eficient que entre processos
      - No cal invalidar tot el TLB
    - Reutilitzen dades portades per altres fluxos
      - Similaritat amb el “hyperthreading”
  - Estalvien recursos
    - Compartits amb els altres fluxos del procés

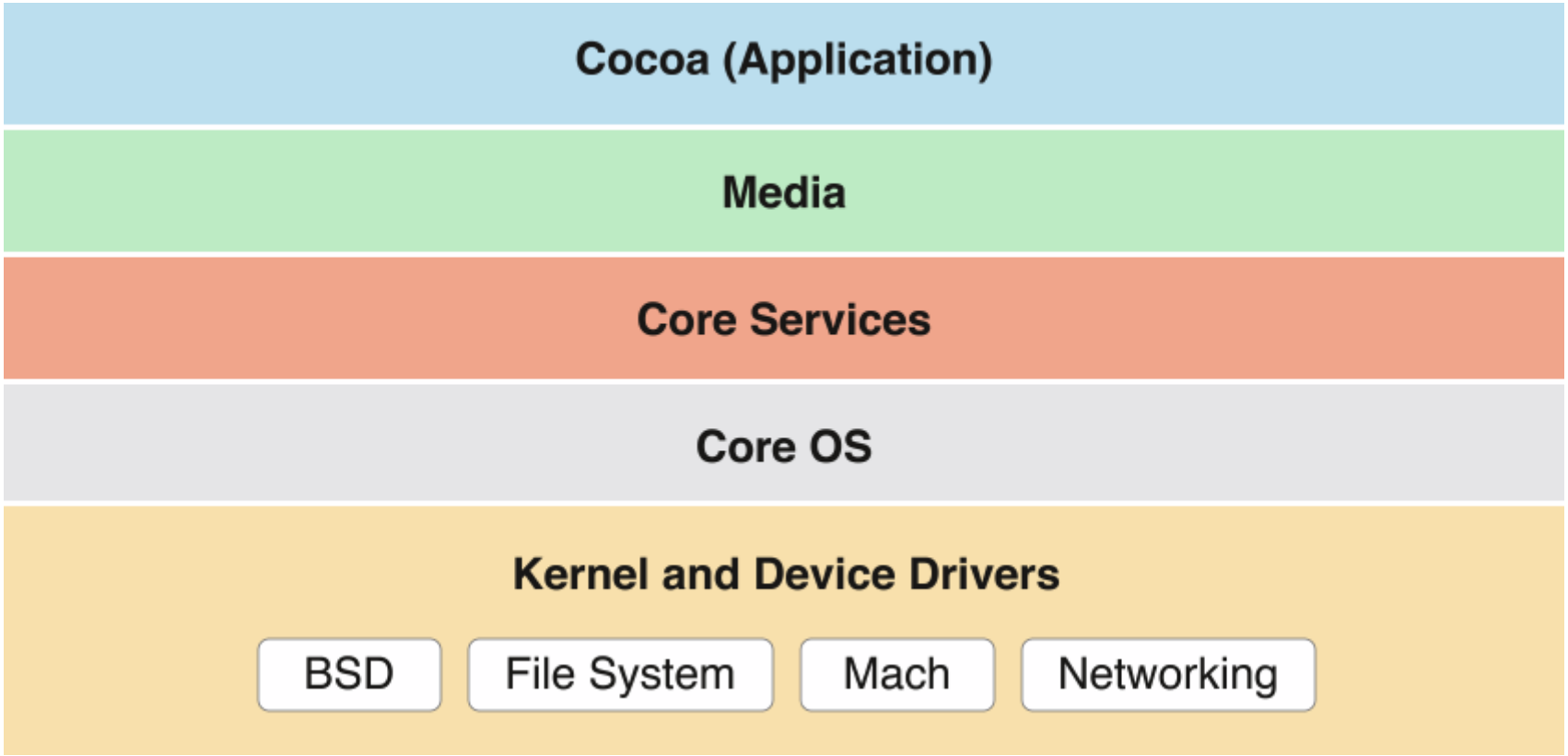
Fluxos de sistema

- Opció 1 d'implementació
  - Abstracció independent
  - Amb identificador de flux



Fluxos de sistema

- Exemples: Mach, Tru64 UNIX (AlphaAXP)*, Solaris, HP-UX, Mac OS-X*, GNU Hurd*, Windows

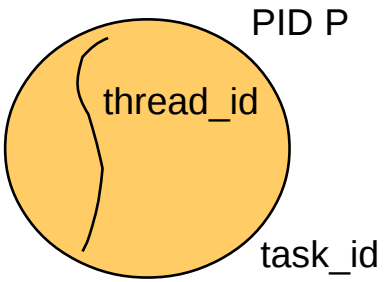


https://developer.apple.com

* Mach-based

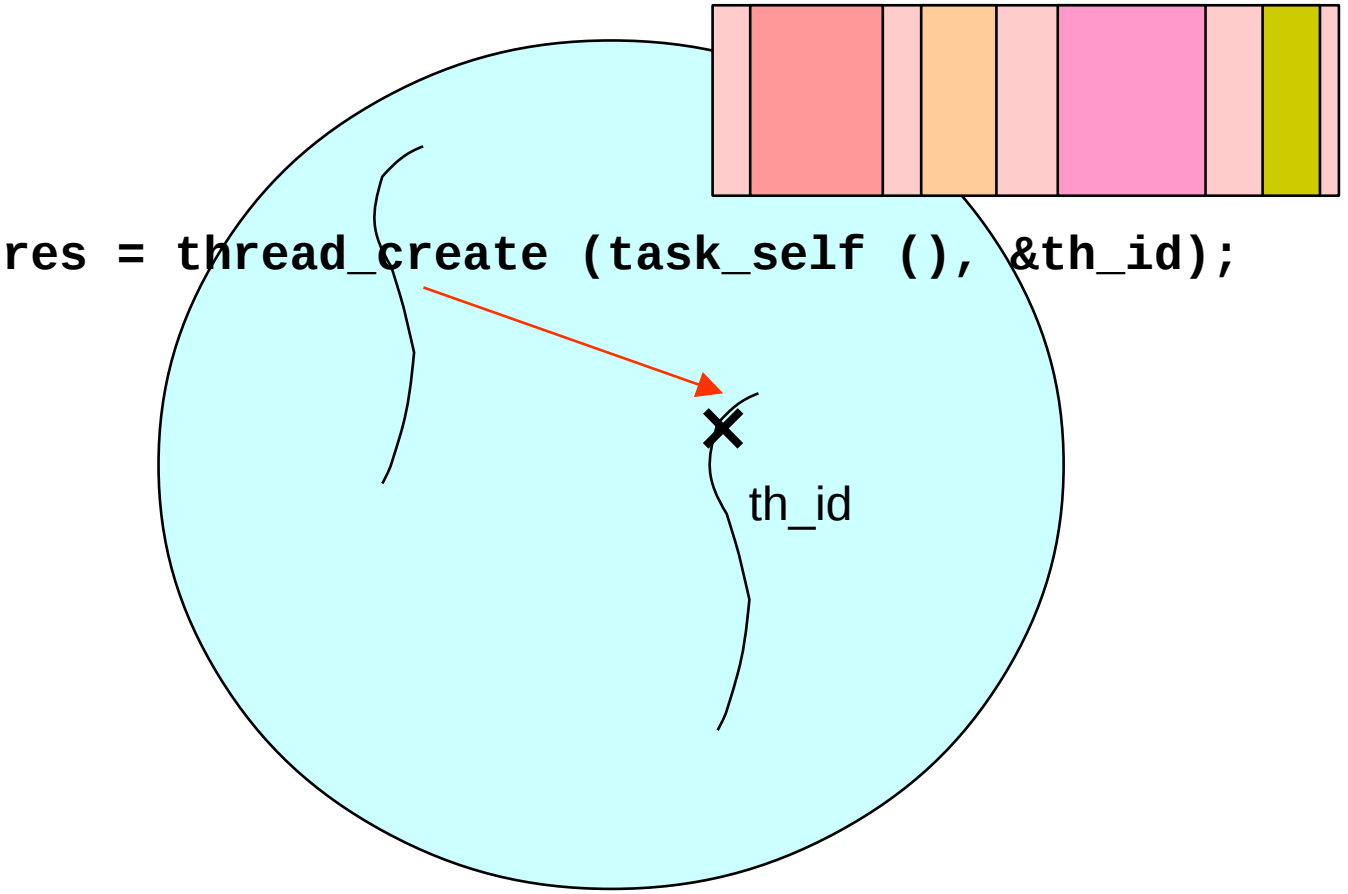
Fluxos de sistema

- Mach
  - Tasks identificades amb task_id
    - task_id = task_self();
  - Exemple de creació d'un flux
    - res = thread_create (task_id, &thread_id);
    - res = thread_get_state (thread_id, &context, size);
    - // Initialize PC, SP and other registers needed
    - res = thread_set_state (thread_id, &context, size);
    - res = thread_resume (thread_id);



Fluxos de sistema

- thread_create(...)



Fluxos de sistema

- thread_create(...)

Example

```
res = thread_create (task_self (), &thid);
if (res!=KERN_SUCCESS) {
   mach_error (“thread_create”, res);
   exit (1);
}
res = thread_get_state (thid, &context, sizeof (context));
context.pc = func;
context.sp = malloc (stack_size) + stack_size;
context.a0 = argument;
res = thread_set_state (thid, &context, sizeof (context));
res = thread_resume (thid);
printf (“New thread id %d\n”, thid);
```

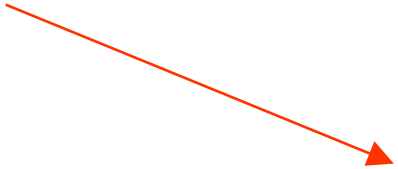
(continua)

Fluxos de sistema

- thread_create(...)

Example

```
res = thread_create (task_self (), &thid);
...
res = thread_resume (thid);
```



```
void func (int argument)
{
   printf (“thread %d, argument %d\n”,
           thread_self (), argument);
   thread_terminate (thread_self ());
}
```

Fluxos de sistema

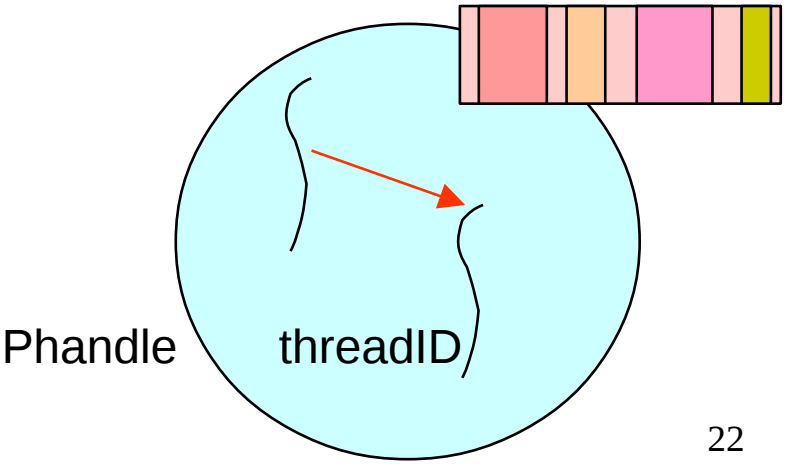
- Conversió d'identificadors
  - pid_for_task(task_id, &pid)
    - Retorna el pid de la task
  - task_for_pid(task_self(), pid, &task);
    - Retorna el task_id del procés indicat per pid

Fluxos de sistema

- Windows – crear fluxos en altres processos
  - CreateRemoteThread(...);
  - CreateRemoteThreadEx (Phandle, attr, stackSize, function, argument, creationFlags, attrList, &threadID);

    [CREATE_SUSPENDED]
  - CreateThread (... sense handle, ni attrList...)
    - Crea un flux en el propi procés

http://msdn.microsoft.com/en-us/library/
→ Windows Development
→ System Services
→ Processes and Threads



Fluxos de sistema

- Opció 2 d'implementació: Linux
  - Crides menys estructurades
    - fork() / clone()
    - pause()
    - signal() / kill(), SIGCONT, SIGSTOP
    - nice()
    - exit()
  - Funcionalitat reduïda
  - No és prou versàtil, en general no es poden assignar recursos a altres processos
    - I de vegades tampoc se'ls poden canviar algunes característiques
  - Solució: clone(...)

Suport a Linux

- Processos compartits (Linux clones)
  - pid = clone(funció, stack, flags, argument);

Example

```
pid = clone (func, usp, flags, argument);
if (pid<0) {
   perror (“clone”);
   exit (1);
}
printf (“New clone pid %d\n”, pid);
res = wait (&status);
...
```

```
void func (void * arg)
{
   int argument = (int) arg;
   printf (“clone %d, arg %d\n”,
           getpid (), argument);
   exit (0);
}
```

Suport a Linux

- Linux clones, flags:
  - CLONE_VM, compartir l'espai d'adreces
  - CLONE_FILES, compartir els descriptors de fitxer
  - CLONE_FS, compartir directoris arrel I actual
  - CLONE_SIGHAND, compartir programació de signals
  - … altres
  - Signal a enviar al pare, quan el fill acabi
- UNIX fork() és un cas especial de clone
  - I així s'implementa
    - clone (NULL, NULL, SIGCHLD, NULL);

Pthreads

- Llibreria de suport (de nivell usuari)
  - Defineix la interfície per una gestió fàcil dels fluxos
    - Creació, destrucció
    - Característiques
    - Prioritat
    - Política de planificació
  - Principal objectiu: portabilitat

Pthreads

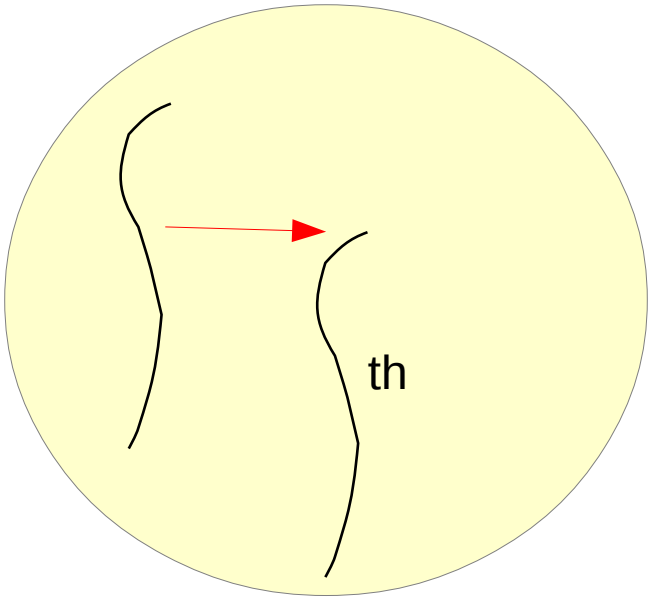
- Creació d'un pthread
  - Demana una estructura “descriptor de flux”
  - Demana una pila i construeix un “stack frame”
    - Funció, argument
  - Crea un clone (Linux) o un flux de sistema (Mach)
    - Compartint tots els recursos, inclòs el “pid”

```
#include <pthread.h>
int pthread_create(
     pthread_t       * thread ,
     pthread_attr_t  * attr ,
     void *         (* start_routine) (void *),
     void            * arg );
```

Pthreads

- Exemple de creació d'un flux

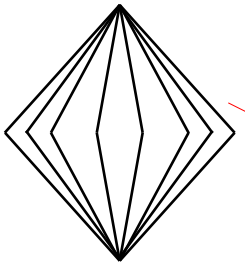
```
void main ()
{
   int res;
   pthread_t th;
   ...
   res = pthread_create (&th, NULL, func, argument);
   ...
}

void * func (void * argument)
{
   printf (“argument %d\n”, (int) argument);
}
```



Pthreads

- Suporta diferents tipus de paral·lelisme
  - Join (estructurat) / detach (no estructurat)

Fork / join

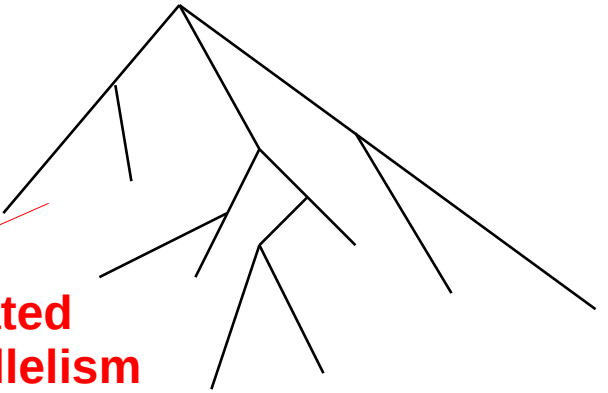


Like Loop/Task Parallelism

Join

Gets the termination code of the pthreads

Unstructured



Like Unrelated Tasks Parallelism

Detach

The application does not need to get the pthreads termination code

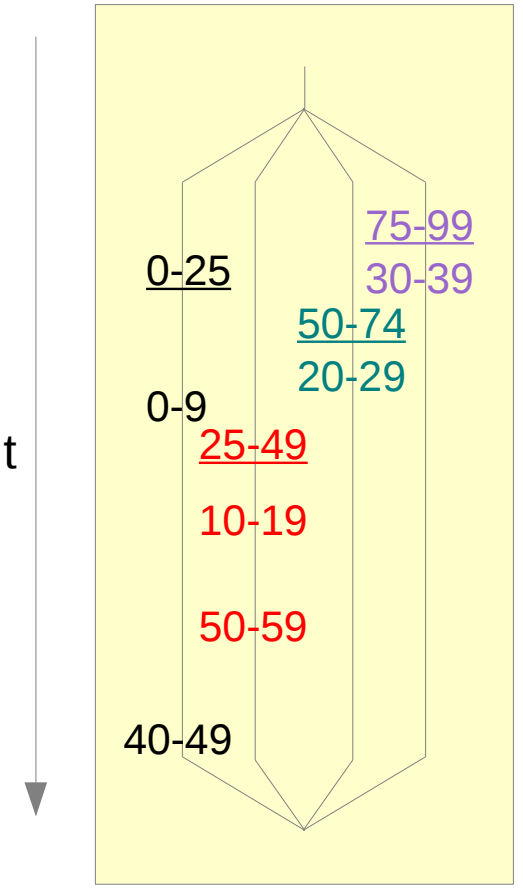
Pthreads

- Join / detach
  - pthread_exit(estat) salva l'estat de finalització en l'estructura del pthread
  - pthread_join(pth, &status) espera que el pthread “pth” acabi I recupera el codi de finalització
  - pthread_detach(pth) marca l'estructura del pthread indicant que no s'esperarà per ell
  - El descriptor del pthread es pot alliberar després de:
    - Haver fet un pthread_exit (el thread) i un pthread_join()
    - Haver fet un pthread_detach i un pthread_exit (el thread)

Pthreads

- Habitualment s'usa com a base per implementar altres models
  - OpenMP / OmpSs...

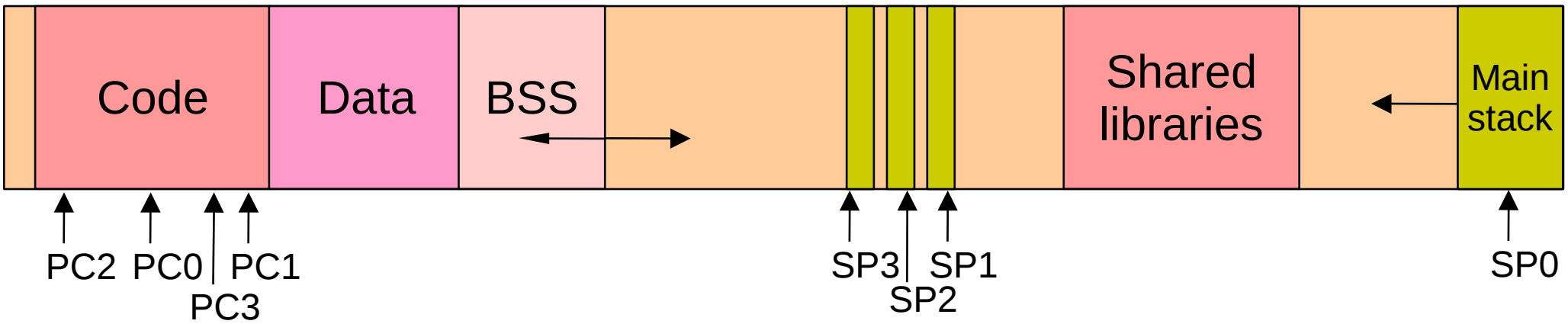
```
#pragma omp parallel
{
#pragma omp for schedule (STATIC) nowait
   for (i=0; i<100; i++)
      A[i] =    B[i] + x*B[i-1] + y*B[i+1];

#pragma omp for schedule (STATIC,10)
   for (i=0; i<60; i++)
     A[i] =    C[i] + y*C[i+1];
}
```



Gestió de memòria

- brk() / sbrk(): break point
  - Increment (+/-)
  - Set

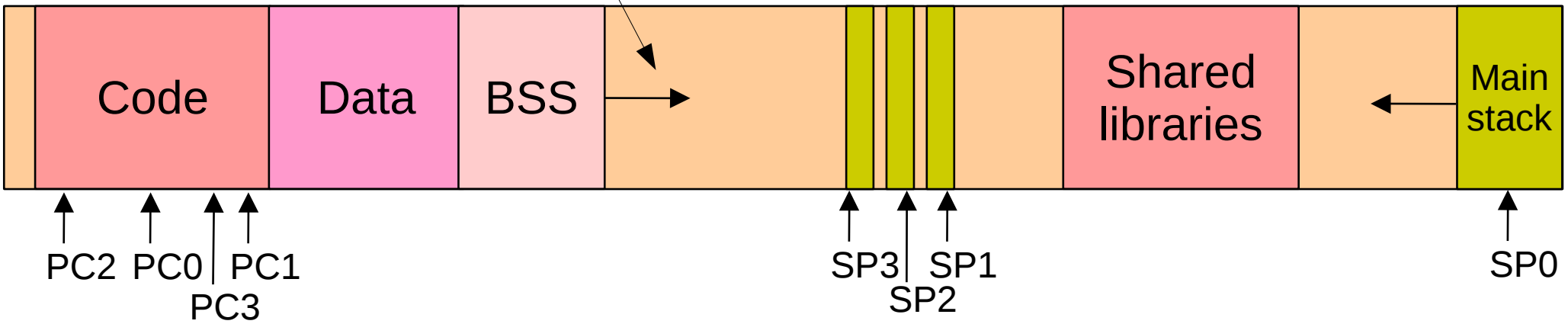
Interfície de Sistema



Gestió de memòria

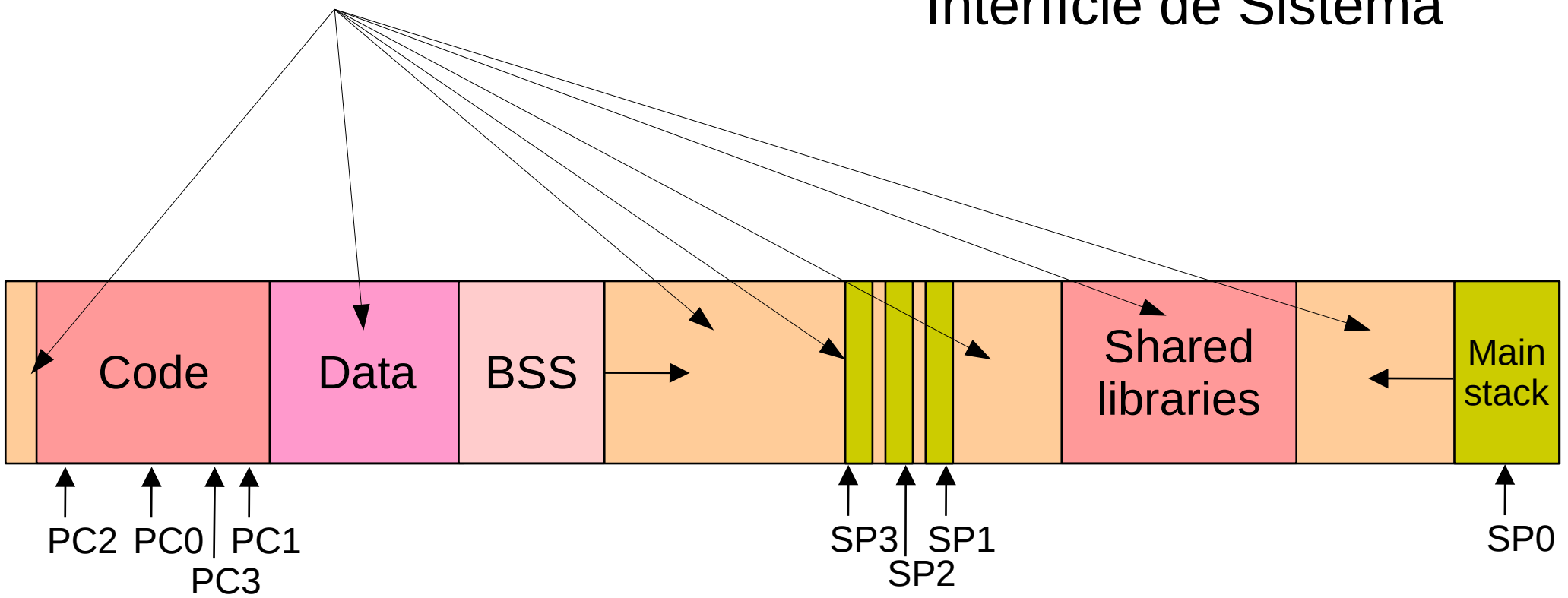
- malloc()/free()
- memalign()
  - sobre brk

Interfície d'Usuari



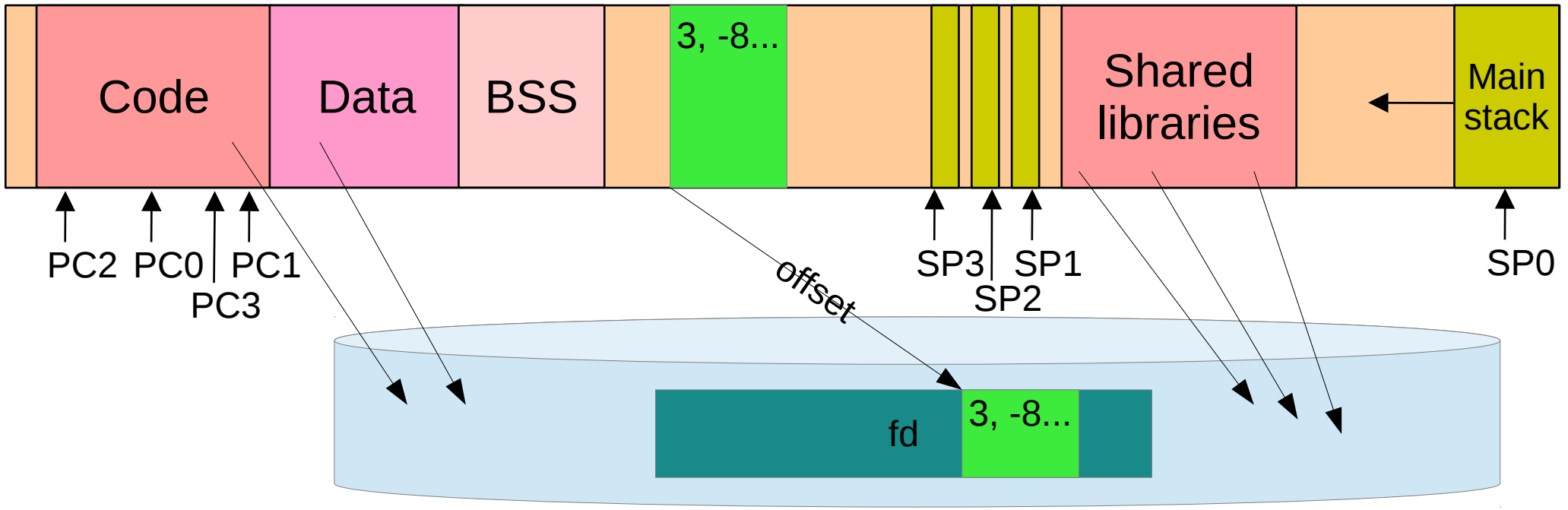
Gestió de memòria

- mmap()/munmap()

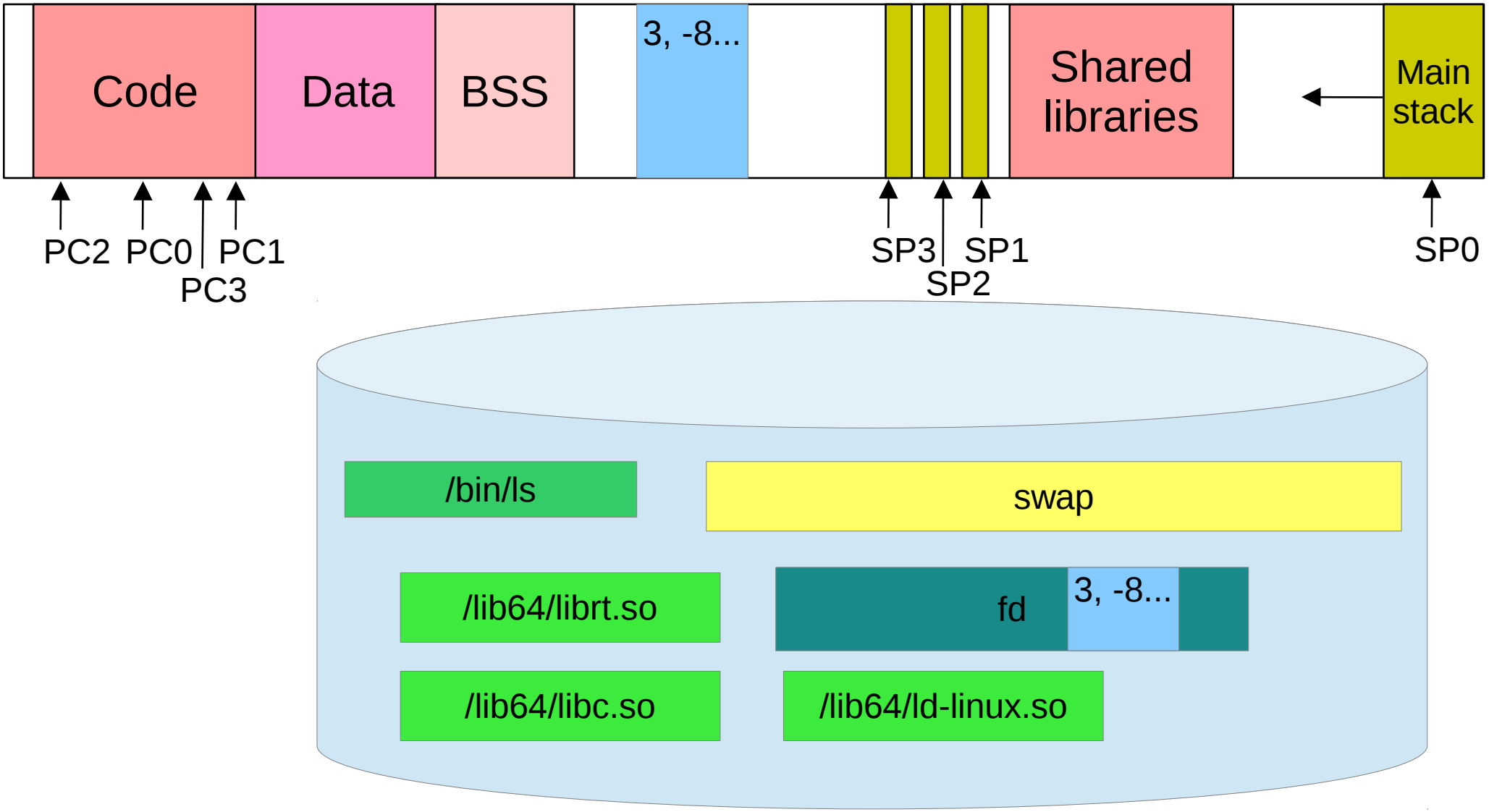


Gestió de memòria

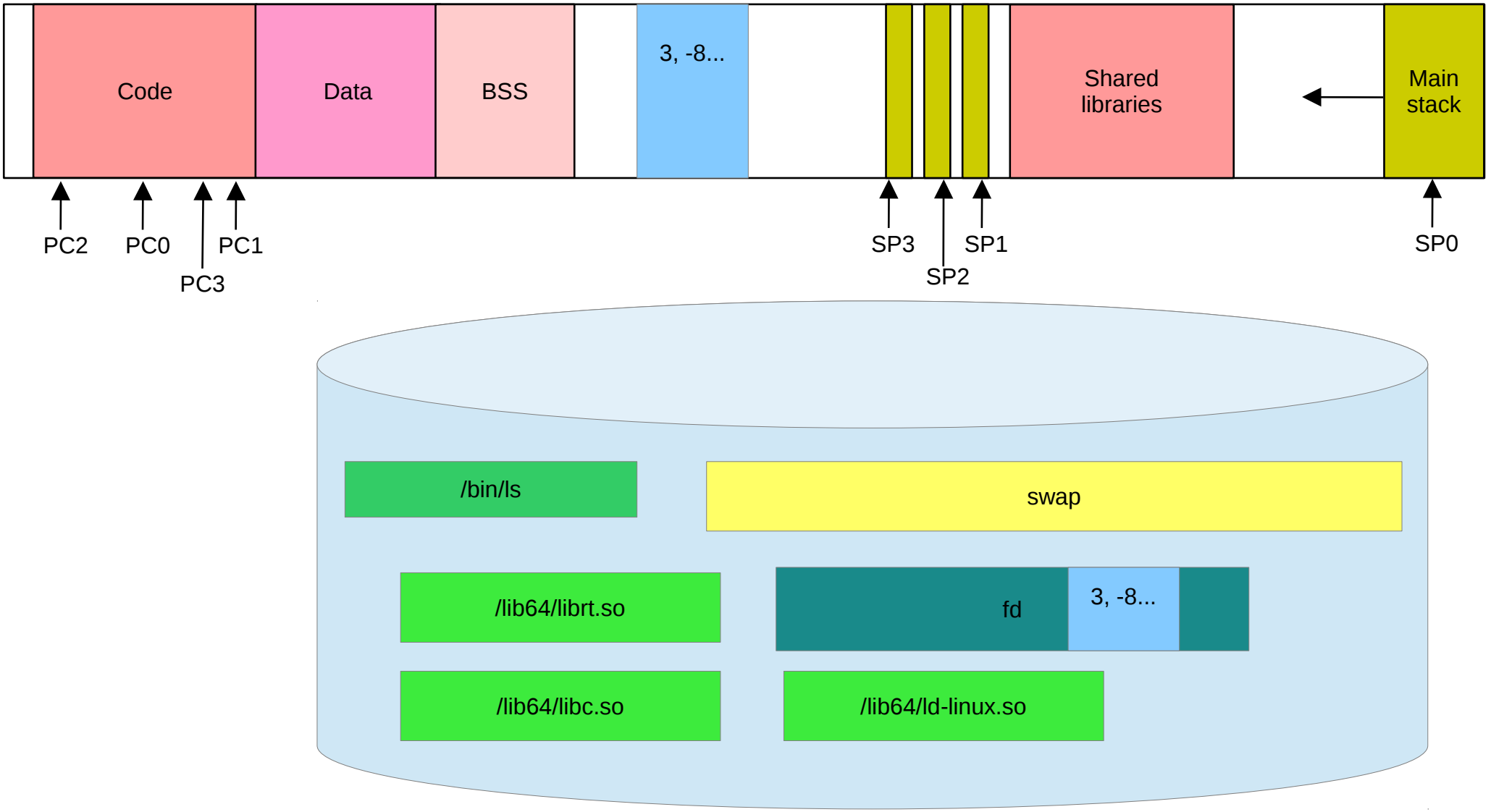
- mmap (addr, size, permissions, flags, fd, offset);
  - Permet demanar:
    - Memòria anònima: virtualitzada a l'àrea de swap
    - Memòria mapejada en un fitxer



Activitat

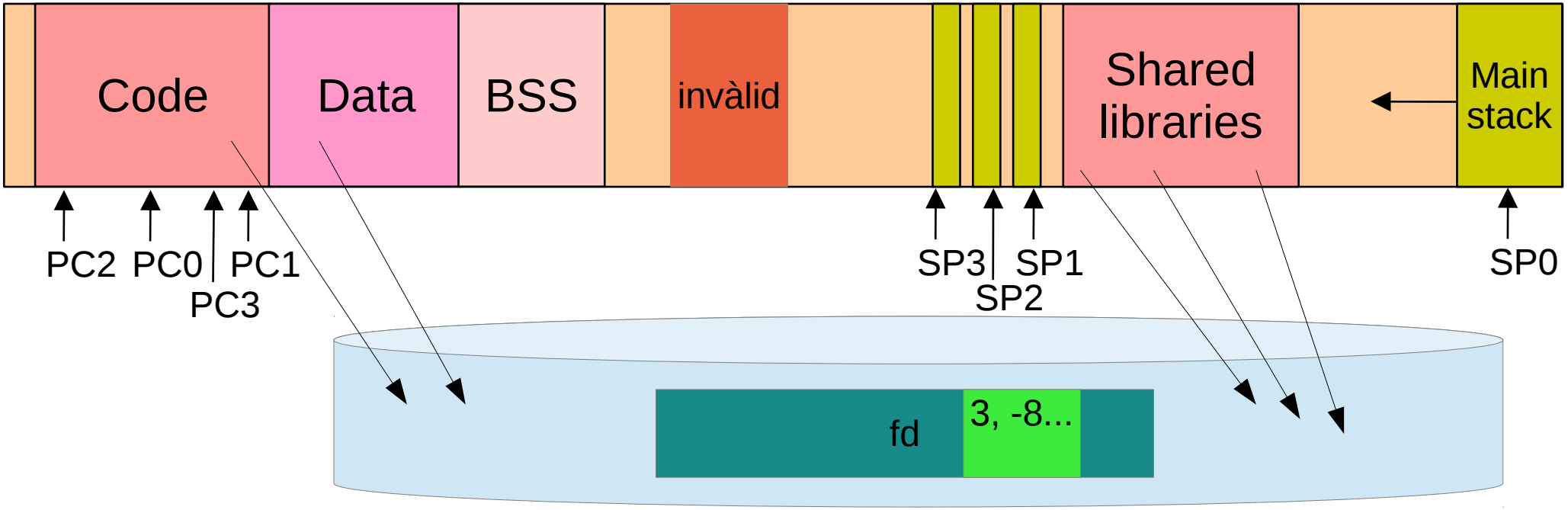

Code
Data
BSS
3, -8...
Shared libraries
Main stack
PC2
PC0
PC1
PC3
SP3
SP2
SP1
SP0
/bin/ls
swap
/lib64/librt.so
fd
3, -8...
/lib64/libc.so
/lib64/ld-linux.so

Activitat


Code
Data
BSS
3, -8...
Shared libraries
Main stack
PC2
PC0
PC3
PC1
SP3
SP2
SP1
SP0
/bin/ls
swap
/lib64/librt.so
fd
3, -8...
/lib64/libc.so
/lib64/ld-linux.so

Gestió de memòria

- munmap (addr, size);
  - Allibera la memòria, de forma que la regió queda invàlida
    - accessos causaran segmentation fault



Per a la setmana vinent

- Documentar-se
  - com compilar el sistema operatiu (Linux)
  - Suport hardware necessari per a la virtualització
- Entregueu: un paràgraf descrivint com configurar i compilar el kernel de linux