T1-Introducció

Índex

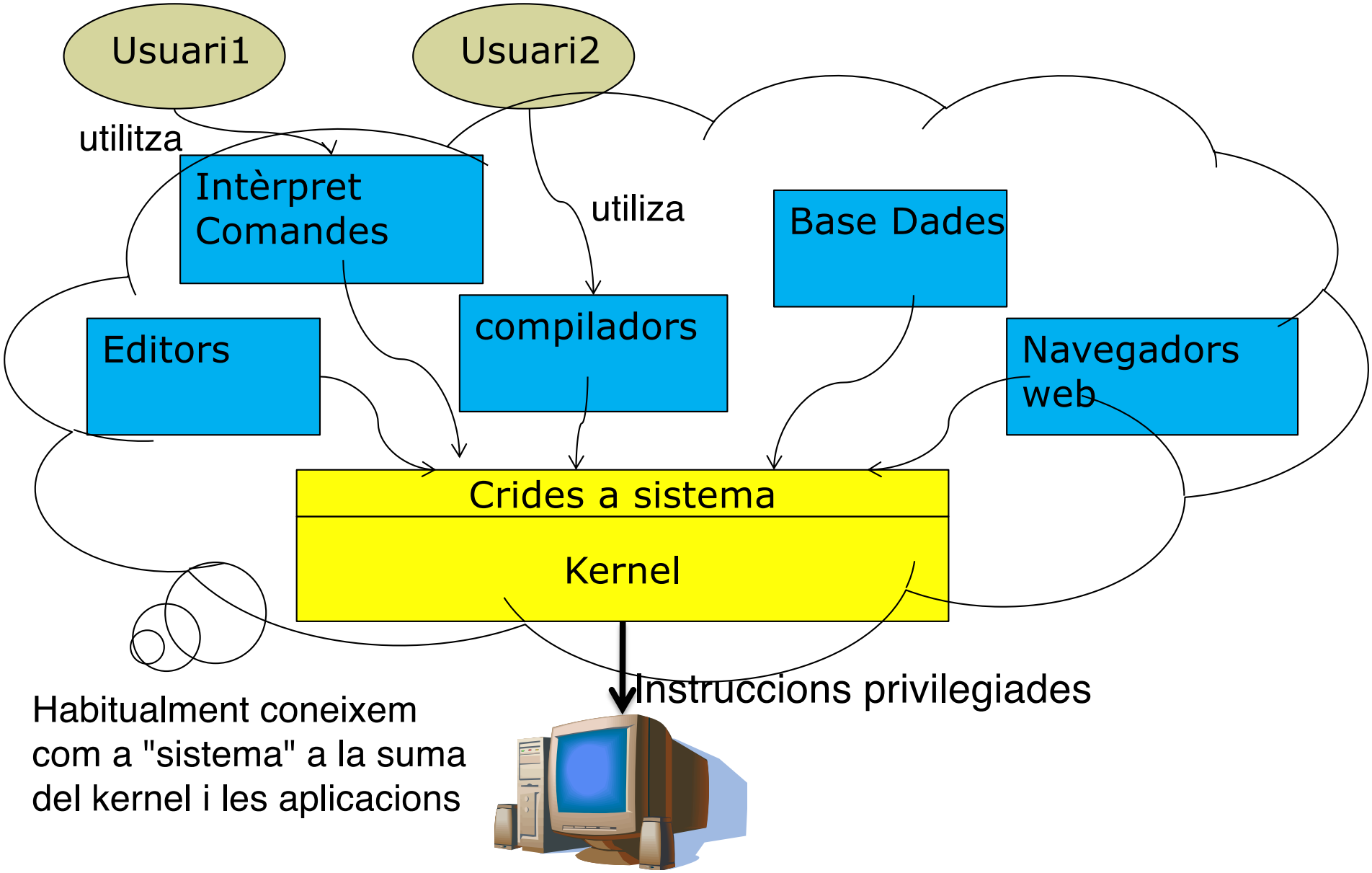
- El paper del S.O.
- Serveis que ofereix el S.O.
- Formes d'accedir al kernel (Tema 8 EC)
  - Modes d' execució
  - Interrupcions, excepcions i crides a sistema
- Crides a sistema (Tema 8 EC)
  - Generació d'executables
  - Requeriments de les crides a sistema
  - Llibreries

EL PAPER DEL S.O.

Què és un SO?

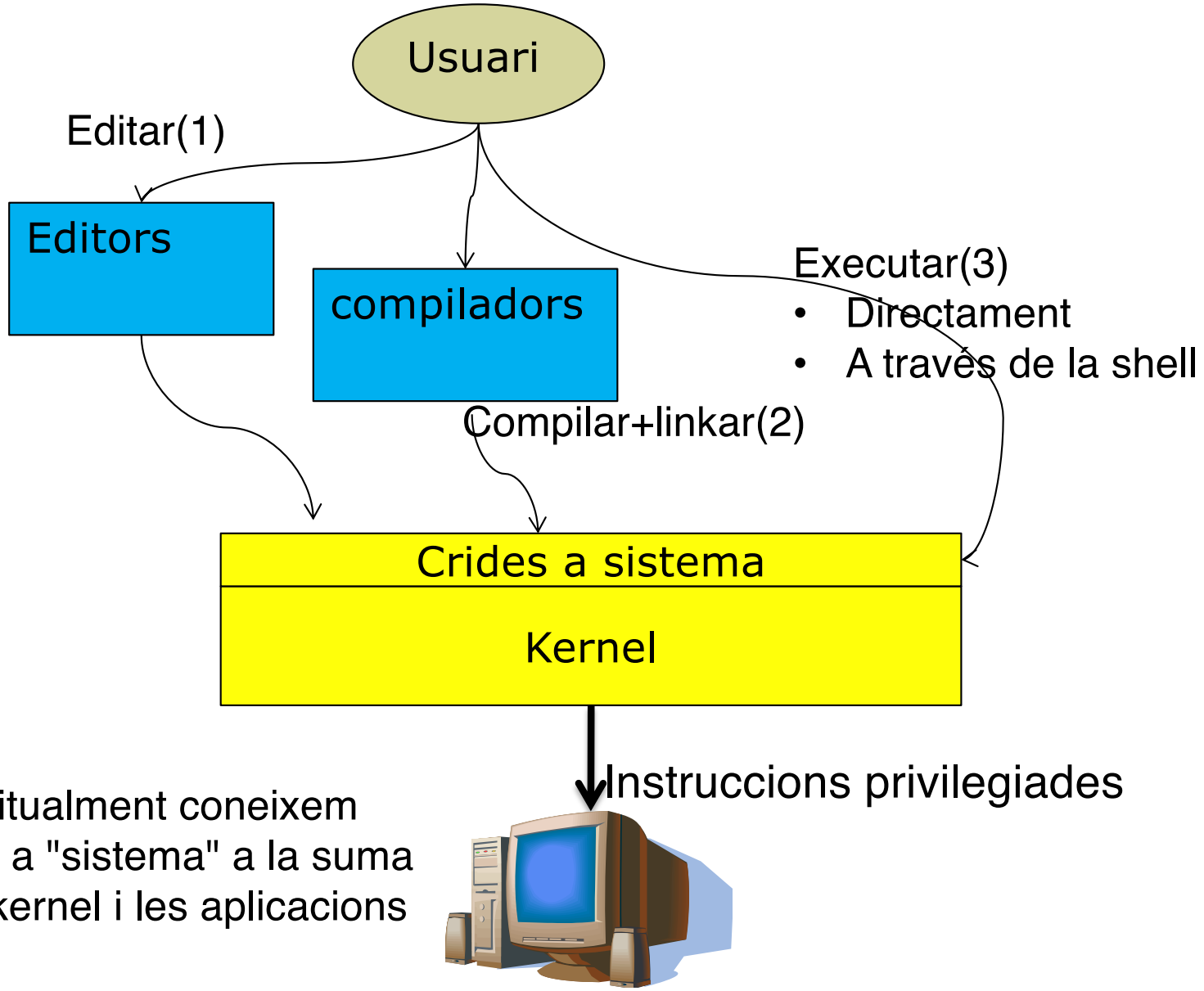
- El Sistema Operatiu és un programari (software) que controla els recursos disponibles del sistema maquinari (hardware) que volem utilitzar i que actua d'intermediari entre les aplicacions i el maquinari
  - **Internament** defineix estructures de dades per gestionar el HW i algorismes per decidir com utilitzar-lo
  - **Externament** ofereix un conjunt de funcions per accedir a la seva funcionalitat (o serveis) de gestió de recursos

Components del sistema



Habitualment coneixem com a "sistema" a la suma del kernel i les aplicacions

Utilització del sistema: Desenvolupament i execució de programes



Habitualment coneixem com a "sistema" a la suma del kernel i les aplicacions

Utilització del sistema

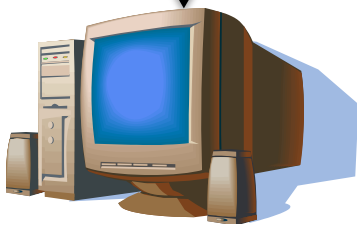
```
# gedit p1.c
# gcc –o p1 p1.c
# p1
```

Normalment utilitzem un **entorn de treball**, l'anomenarem shell o intèrpret de coamndes.

Crides a sistema

Kernel

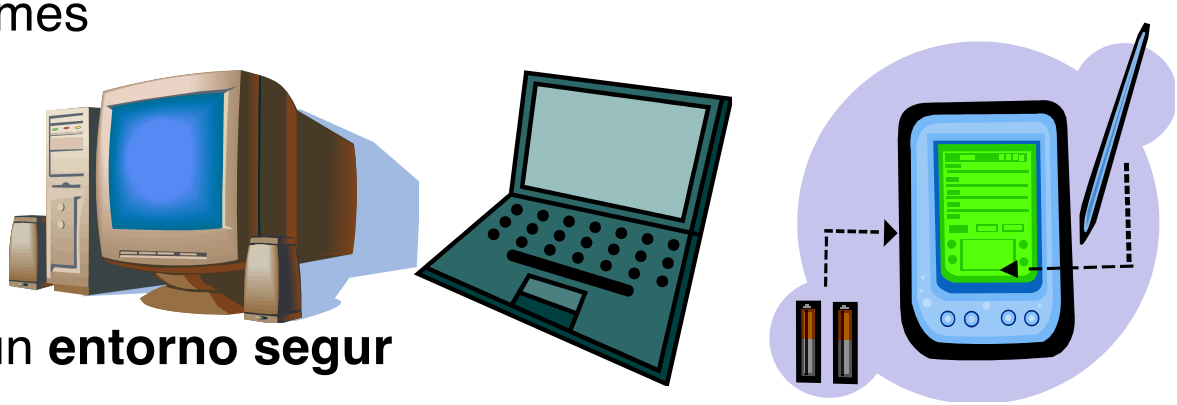
Instruccions privilegiades



Habitualment coneixem com a "sistema" a la suma del kernel i les aplicacions

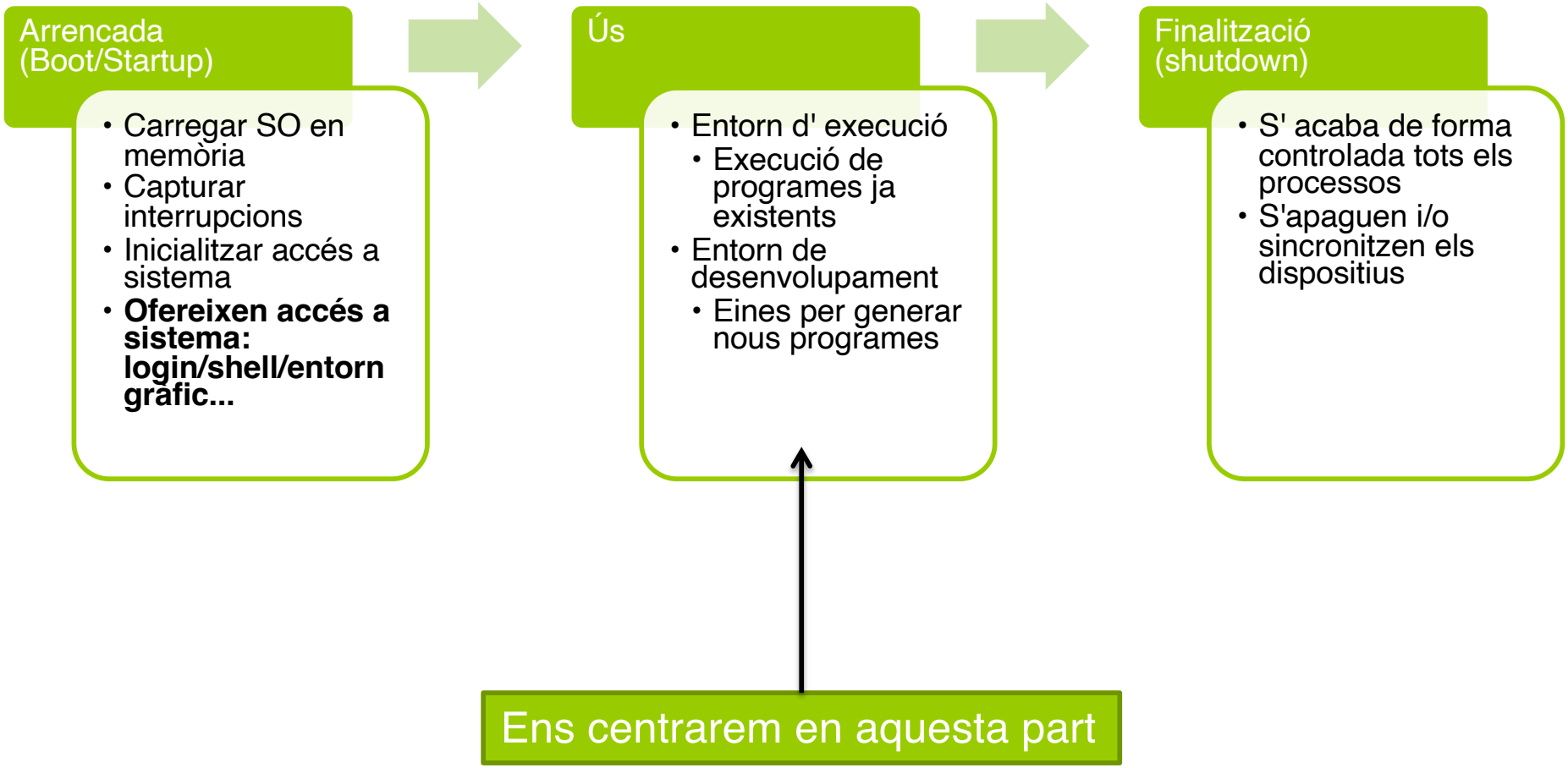
Què esperem del S.O?

- Ofereix un **entorn“usable”**
  - Abstreu els usuaris de les diferències que hi ha entre els diferents sistemes



- Ofereix un **entorno segur**
  - Protegeix el HW d' accessos incorrectes i a uns usuaris d' un altres
- Ofereix un **entorn eficient**
  - Proporciona un ús eficient dels recursos del sistema
  - Múltiples usuaris accedeixen a un mateix sistema i tenen una sensació d'accés en "exclusiva”

Que ofereix el SO?

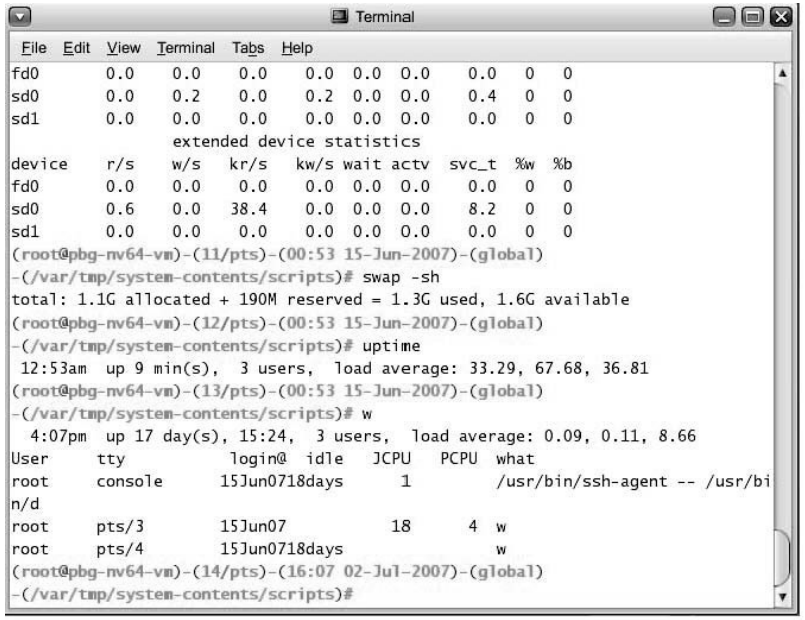


Arrencada del sistema

- El maquinari carrega el SO (o una part) en arrencar. Es coneix com a *boot*
  - **El sistema pot tenir més d'un SO instal·lat (al disc) però només un executant-se**
  - El SO es copia en memòria (tot o part d'ell)
  - Inicialitza les estructures de dades necessàries (maquinari/software) per controlar la màquina i oferir els serveis
  - Les interrupcions maquinari són capturades pel SO
  - Al final el sistema posa en marxa un programa que permet accedir al sistema
    - Login (username/password)
    - Shell
    - Entorn gràfic

Entorn de desenvolupament

- El SO ofereix, com a part dels seus serveis, un entorn de treball **interactiu**. Fonamentalment pot ser de dos tipus: intèrpret de comandaments o entorn gràfic
- Depenent del sistema, aquest entorn pot formar part del kernel o pot ser un programa a part. En qualsevol cas el trobem com a part del sistema per poder utilitzar-lo.



Entorn "usable", segur i eficient

- Durant el funcionament del sistema, aquests són els seus tres objectius bàsicsUsabilidad
  - Seguretat
  - Eficiència
- També ho hauran de ser dels vostres programes
  - Usables:
    - Si l' usuari no l' executa correctament, caldrà donar un missatge indicant-lo que sigui clar i útil
    - Si alguna funció d' usuari o trucada a sistema falla, el programa haurà de donar un missatge clar i útil
  - Segur:
    - El programa no pot (o no hauria de) "fallar" si l'usuari l'utilitza de forma incorrecta, ha de controlar totes les fonts d'error.
    - Molt menys fer fallar el sistema
  - Eficient:
    - Cal plantejar-se la millor manera d'aprofitar els recursos de la màquina, pensant que els recursos que té l'usuari són limitats (execució en entorn compartit)

(Explicat en EC)

FORMES D' ACCEDIR AL KERNEL

Com oferir un entorn segur: "Modes" d'execució

- El SO necessita una forma de garantir la seva seguretat, la del maquinari i la de la resta de processos
- Necessitem instruccions de llenguatge màquina privilegiades que només pot executar el kernel
- El HW coneix quan s'està executant el kernel i quan una aplicació d'usuari. Hi ha una instrucció de LM per passar d'una manera a l'altra.
- El SO s'executa en un mode d'execució privilegiat.
  - **Mínim 2 modes (poden haver-n'hi més)**
    - **Mode d'execució NO-privilegiat , user mode**
    - **Mode d' execució privilegiat, kernel mode**
  - **Hi ha parts de la memòria només accessibles en mode privilegiat i determinades instruccions de llenguatge màquina només es poden executar en mode privilegiat**
- Objectiu: Entendre que són els modes d' execució i perquè els necessitem

Quan s' executa codi de kernel?

- Quan una aplicació executa una trucada a sistema
- Quan una aplicació provoca una excepció
- Quan un dispositiu provoca una interrupció

- Aquests esdeveniments podrien no tenir lloc, i el SO no s'executaria El SO perdria el control del sistema.
- **El SO configura periòdicament la interrupció de rellotge per evitar perdre el control i que un usuari pugui acaparar tots els recursos**
  - **Cada 10 ms per exemple**
  - **Típicament s' executa la planificació del sistema**

Accés al codi del kernel

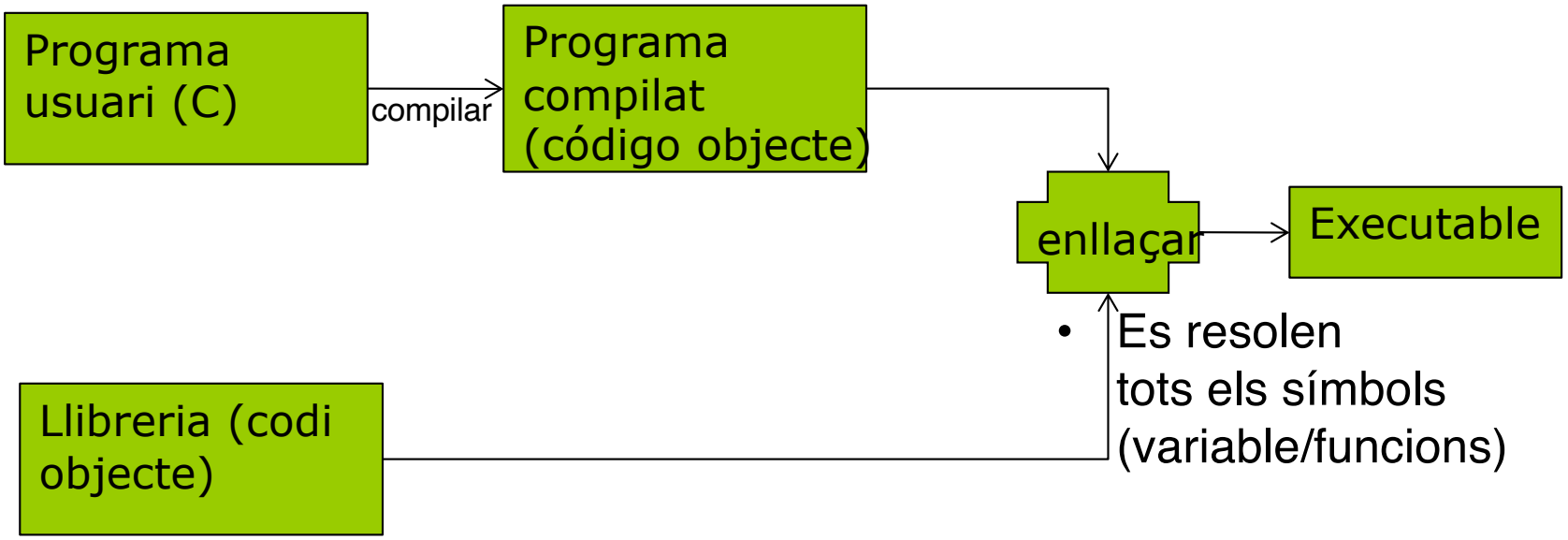
- El kernel és un codi guiat per esdeveniments
  - Interrupció del flux actual d' usuari per realitzar una tasca del SO
- Tres formes d'accedir al codi del SO (Vist a EC):
  - **Interrupcions** generades pel maquinari (teclat, rellotge, DMA, ...)
    - asíncrones (entre dues instruccions de llenguatge màquina)
  - Els errors de programari generen **excepcions** (divisió per zero, fallada de pàgina,...)
    - síncrones
    - provocades per l' execució d' una instrucció de llenguatge màquina
    - es resolen (si es pot) dins de la instrucción
  - Peticions de servei de programes: **Crides a sistema**
    - Síncron
    - provocats per una instrucció explícitament (de llenguatge màquina)
    - per demanar un servei al SO (crida al sistema)
  - Des del punt de vista maquinari són molt semblants (gairebé iguals)

- Generació d' executables
- Requeriments de les trucades a sistema
- Llibreries

CRIDES A SISTEMA

Generació executables

- Es comprova la sintaxi, tipus de dades, etc.
- Es fa una primera generació de codi



- Llibreries: Rutines/Funcions ja compilades (és codi objecte), que s'enllacen amb el programa i que el programador només necessita cridar
- Poden ser a nivell de llenguatge (libc) o de sistema (libso)

Crides a Sistema

- Conjunt de **FUNCIONS** que ofereix el kernel per accedir als seus serveis
- **Des del punt de vista del programador és igual a l'interfície de qualsevol llibreria del llenguatge(C,C++, etc)**
- Normalment, els llenguatges ofereixen un API de més alt nivell que és més còmoda d'utilitzar i ofereix més funcionalitats
  - Exemple: Llibreria de C: printf en lloc de write
    - Nota: La llibreria s'executa en **mode usuari** i no pot accedir al dispositiu directament
- Exemples
  - Win32 API per a Windows
  - POSIX API per a sistemes POSIX (UNIX, Linux, Max OS X)
  - Java API per a la Java Virtual Machine

Requeriments crides a sistema

- Requeriments
  - Des del punt de vista del programador
    - Ha de ser tan senzill com una trucada a funció
      - <tipus> nom_funció(<tipus1> arg1, <tipus2> argc2..);
    - No es pot modificar el seu context (igual que en una crida a funció)
      - S'han de salvar/restaurar els registres modificats
  - Des del punt de vista del kernel necessita:
    - Execució en mode privilegiat suport HW
    - Pas de paràmetres i retorn de resultats entre modes d'execució diferents → depèn HW
    - Les adreces que ocupen les crides a sistema han de poder ser variables per suportar diferents versions de kernel i diferents S.O.→ per portabilitat

Solució: Llibreria "de sistema" amb suportHW

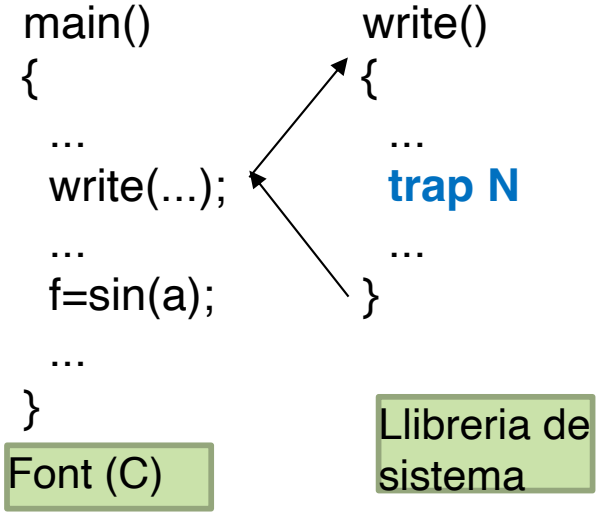
- La llibreria de sistema s'encarrega de traduir de la funció que veu l'usuari a la petició de servei explícit al sistema
  - Passa paràmetres al kernel
  - Invoca el kernel TRAP
  - Recull resultats del kernel
  - Homogeneïtzens (totes les crides a sistema en linux retornen -1 en cas d'error
- ¿Com aconseguim la portabilitat entre diferents versions del SO?
  - La crida a sistema no s'identifica amb una direcció, sinó amb un identificador (un número), que fem servir per indexar una taula de crides a sistema (que s'ha de conservar constant entre versions)

Crides a sistema

- La llibreria de sistema aïlla l'usuari dels detalls de l'arquitectura
- El mode d'execució privilegiat ofereix seguretat: només el kernel s'executa en mode privilegiat
- La taula de crides a sistema ofereix compatibilitat entre versions del SO
- Tenim una solució que només depèn de l'arquitectura: la qual cosa és inevitable ja que és un binari amb un llenguatge màquina concret!!

Llibreries “de sistema”

- El SO ofereix llibreries del sistema per aïllar els programes d'usuari de tots els passos que cal fer per
  1. Passar els paràmetres
  2. Invocar el codi del kernel
  3. Recollir i processar els resultats



- S' anomenen llibreria de sistema però s' executa en **mode usuari, només serveixen per facilitar la invocació de la crida a sistema**

Codi genèric en entrar/sortir del kernel

- Hi ha passos comuns a interrupcions, excepcions i crides a sistema
- En el cas d'interrupcions i excepcions, no s'invoca explícitament ja que la invocació la realitza la CPU, la resta de passos si s' apliquen.

	Funció “normal”	Funció kernel
Passem els paràmetres	Push paràmetres	**DEPÈN**
Per invocar-la	call	sysenter, int o similar
A l' inici	Salvar els registres que vam fer servir(push)	Salvam tots els registres(push)
Accés a paràmetres	A través de la pila: Ex: mov 8(ebp),eax	**DEPÈN**
Abans de tornar	Recuperar els registres salvats en entrar(pop)	Recuperar els registres salvats (tots) en entrar(pop)
Retorn resultats	eax (o registre equivalent)	**DEPÈN**
Per tornar al codi que la va invocar	ret	sysexit, iret o similar