

### 3. GAIA: Komando-interpretatzailea (KI)

1. Programen kontrola
2. Komando-interpretatzailearen ezaugarriak
3. Komando-interpretatzailearen kodea

# Programen kontrola

Beharrak:

- ⇒ Programen exekuzioak automatikoki kateatu
- ⇒ Programen exekuzioen arteko denborak galduak ekidin
- ⇒ Lan periodikoen automatizazioa
- ⇒ Konputagailua erabili programatu behar izan gabe (komandoak, ez sistema-deiak)

- Soluzioa: errutina-deien kontzeptua programetara zabaltzea
- Programa hori = **komando-interpretatzailea (KI)**
- KI-rekin, programak bukatzean beste **programa** bat deitu edo **abiarazi** daiteke
- Programa batek beste bati dei egiteko aukera: **exekutatu\_programa** sistema-deia

# Programen kontrola (2)

- Programen kontrolerako lengoaien sententziak = **komandoak**
- Hasieran, KI = SEaren helbide-espazioan egoiliarra.
- Gaur egun, KI = erabiltzailearen mailako beste programa bat: *shell* (aukerak: sh, csh, tcsh, bash, perl, Gnome, KDE...; Windows)

## *batch edo lotekakoa (off-line)*

operadoreari eman → txartel irakurgailua → komando multzoa

## *elkarreragile (on-line): terminalak*

programak pantailan **editatzeko** aukera.

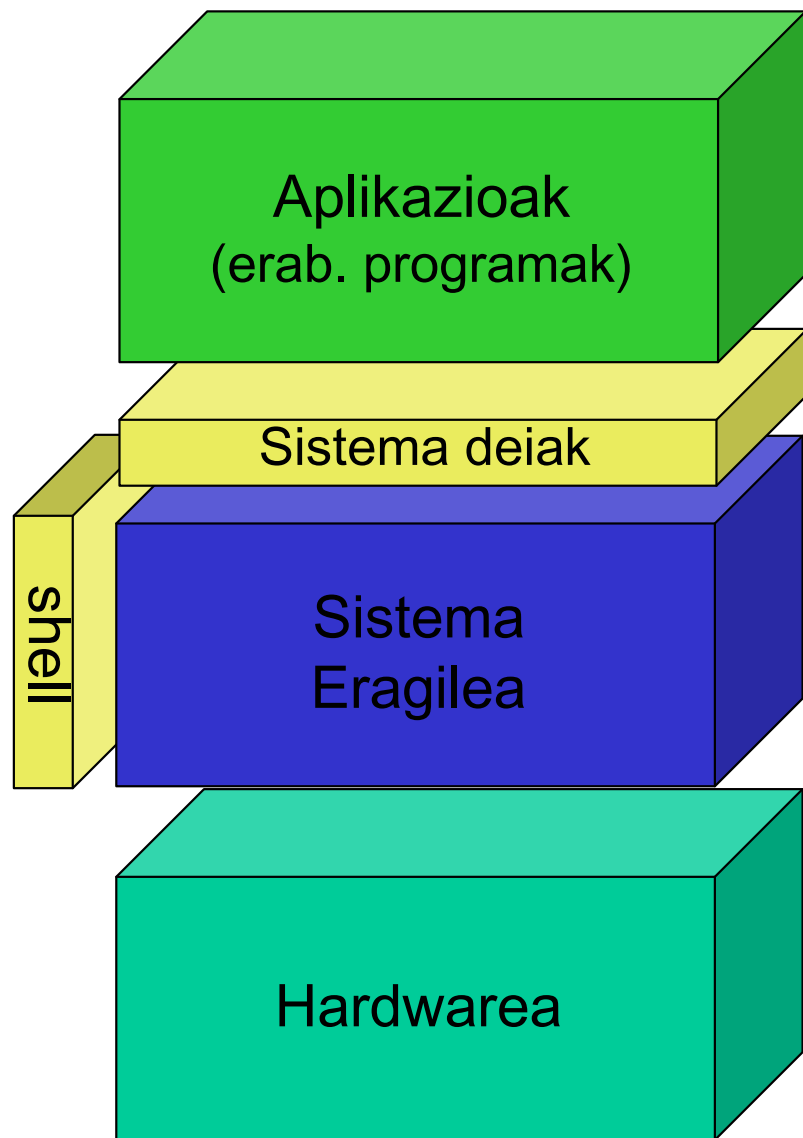
Teklatua → Pantaila: oihartzuna (*echo*) ← teklatuaren arreta-errutina

# Programen kontrola (3)

- KI baten **komando arruntenak**: kopianatu, berrizendatu, ezabatu, ikusi eta inprimatu fitxategiak; sortu, listatu eta ezabatu katalogoak...
  
- Komando-lengoiaren eboluzioa → Programazioa: *script*
  - komando gutxi eta laburrak → **komando-lengoaian** idatzitako programa konplexuak  
(baldintzazko egiturak, egitura errepikakorak, aldagaien erabilpena...)
  
- KI alfanumerikoak / Pantaila grafikoak + sagua  
→ ingurune atseginagoak: **GUI**
  
- ⇒ informatikari buruz gutxi dakiten erabiltzaileentzako erraztasuna
- ⇒ S/I konplexuagoa ⇒ liburutegi estandarrak: X arau estandarra<sub>4</sub>

# Komando interpretatzailea eta SEa

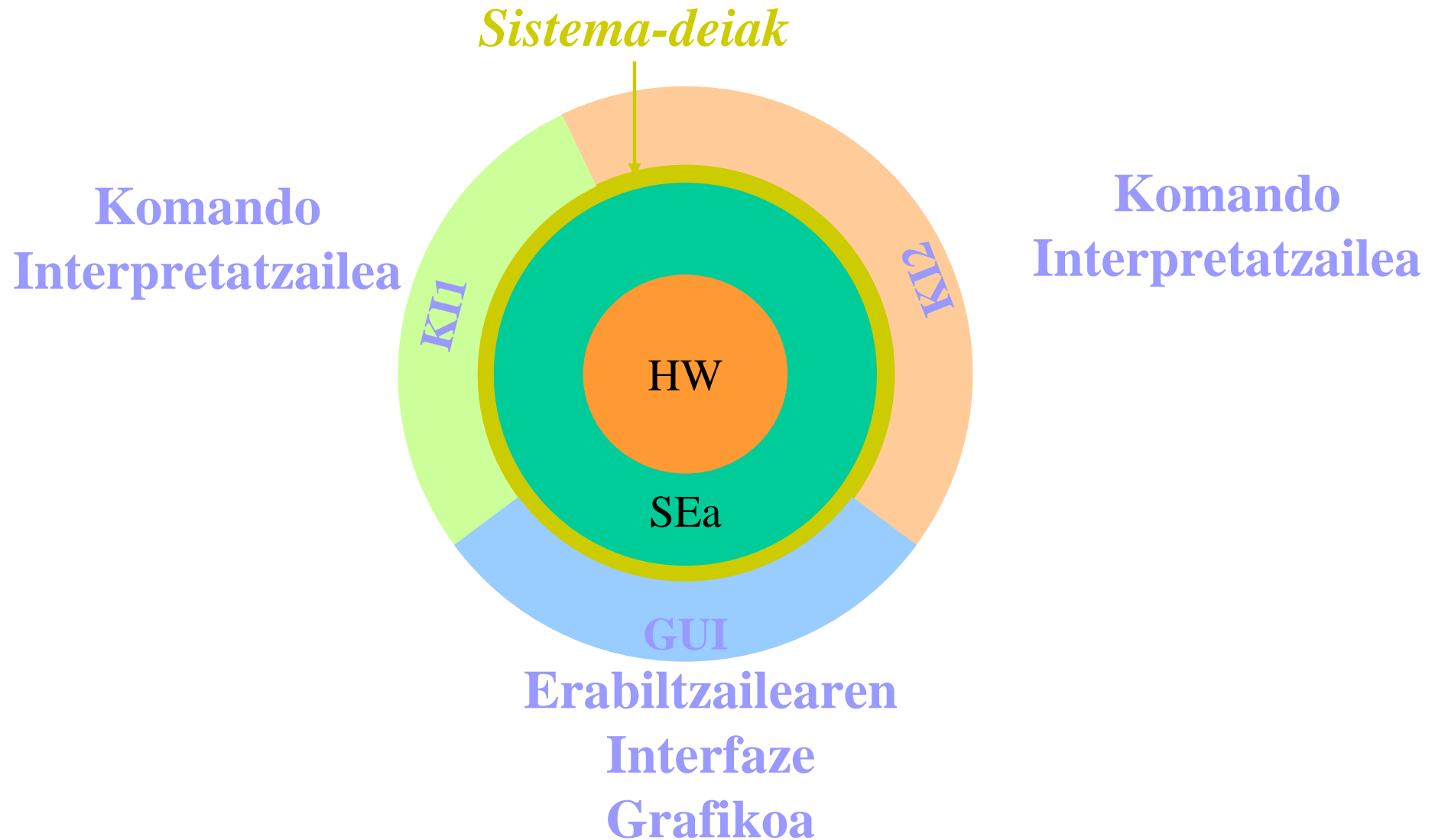
bi interfaze  
- sistema deiak  
- shell



ATAZAK:  
Koordinatu ekintzak  
eta baliabideen erabilpena  
aplikazioetan

Izkutatu hardware  
berezitasunak

# Komando interpretatzailea eta SEa



# Komandoen implementazioa

- **ls:**
  - opendir(...)
  - readdir(...)
  - closedir(...)
- **cat:**
  - read(...)
  - write(...)

clear komandoa:

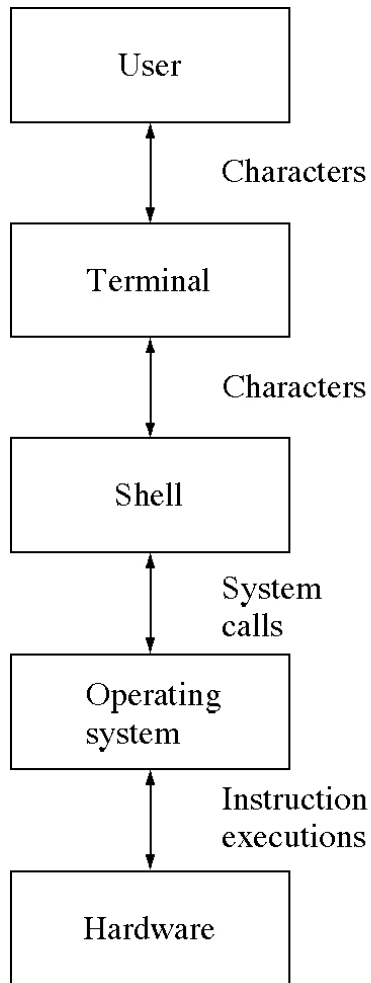
```
//clear.c
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("\033[H\033[J");
    return(1);
}
```

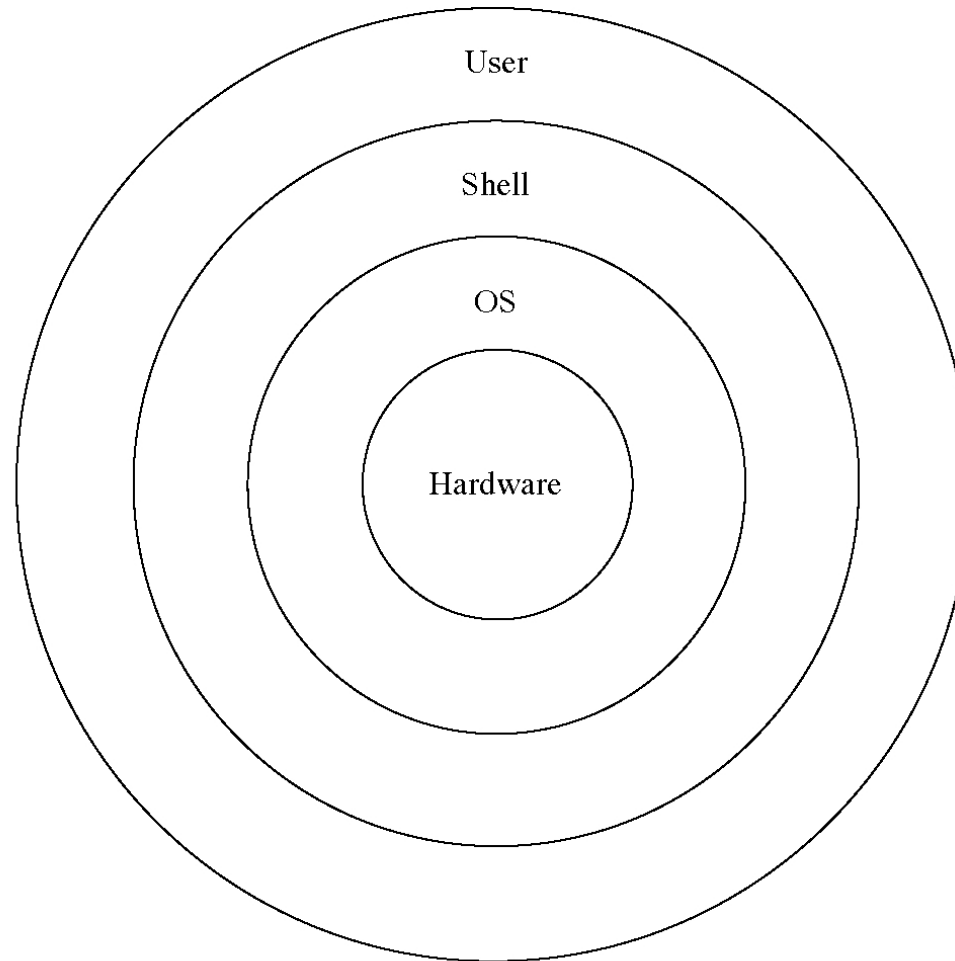
# KI-a sistema eragilearen barne-atala al da?

- Ez: shell-a SE-aren kerneletik kanpoko elementu bat bezala uler daiteke.  
Kernela soilik dago hardwarearen menpe.
- Malgutasun gehiago. Aparteko programa bat izanik, erabiltzaile bakoitzak shell desberdin bat erabili dezake.
- Oro har, kode gehiena SE-tik kanpo uztea komeni da.

# KI-aren bi ikuspegi



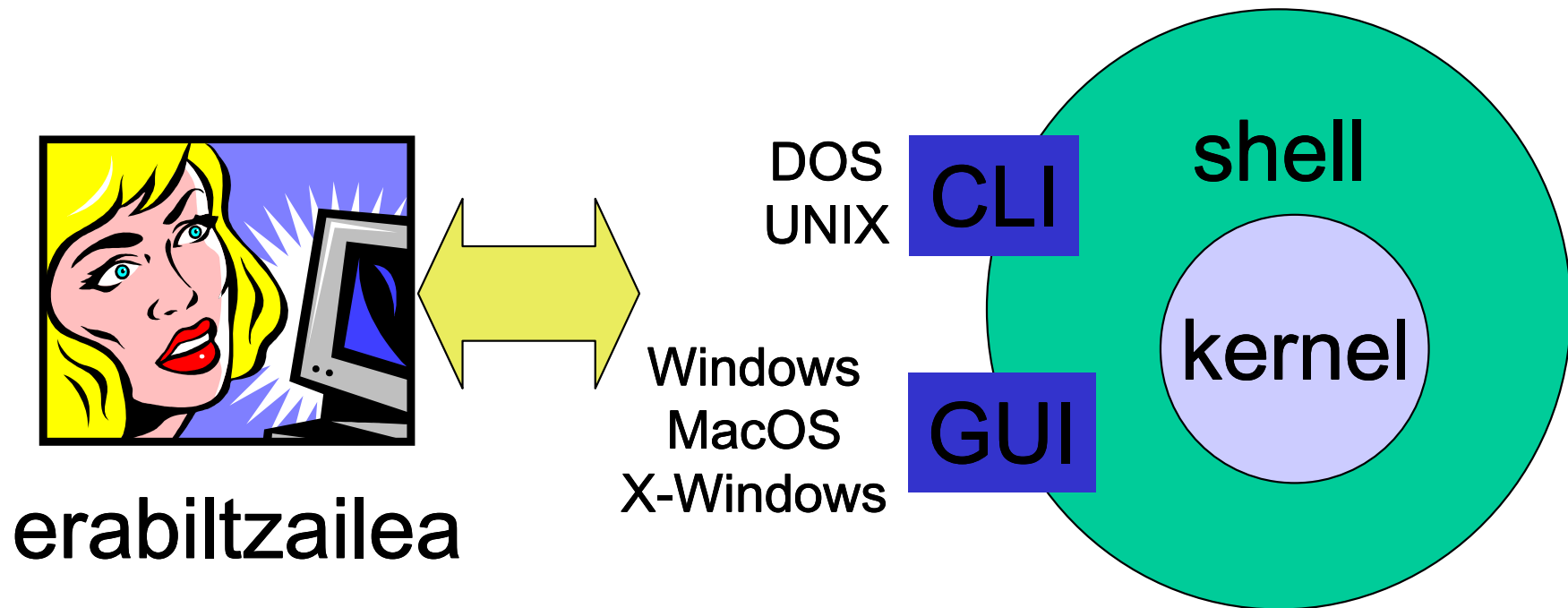
(a) Shell as a level



(b) Shell as a covering

# Komando interpretatzailea

- Erabiltzaile eta SEaren arteko interfazea



CLI = command line interface  
GUI = graphical user interface

# KI-aren aukeraketa

- Shell-aren aurretik:  
JCL, CL SEak (komando lengoaiak), ...
- Zenbait sistema eragilek (adib. UNIX) erabiltzaileari aukera ematen diote KI desberdinak erabili ahal izateko, interfaze egokiena erabili ahal izateko
- Windows-eko hasierako bertsioak berez MS-DOS-erako KI-aren ordezkapen grafikoak besterik ez ziren

# CLI vs GUI

- Unix-ek ez du behar GUI bat martxan jartzeko. Windows NT-k bai behar du. Grafikoek disko eta memoria asko behar dute.
- Unix-ekin, posiblea da GUI desberdinak erabiltzea X- Windows sistemaren bitartez. Gaur egun utilitate askok interfaze grafikoa dute Unix-en, batzuk Java-rekin inplementatutakoak.

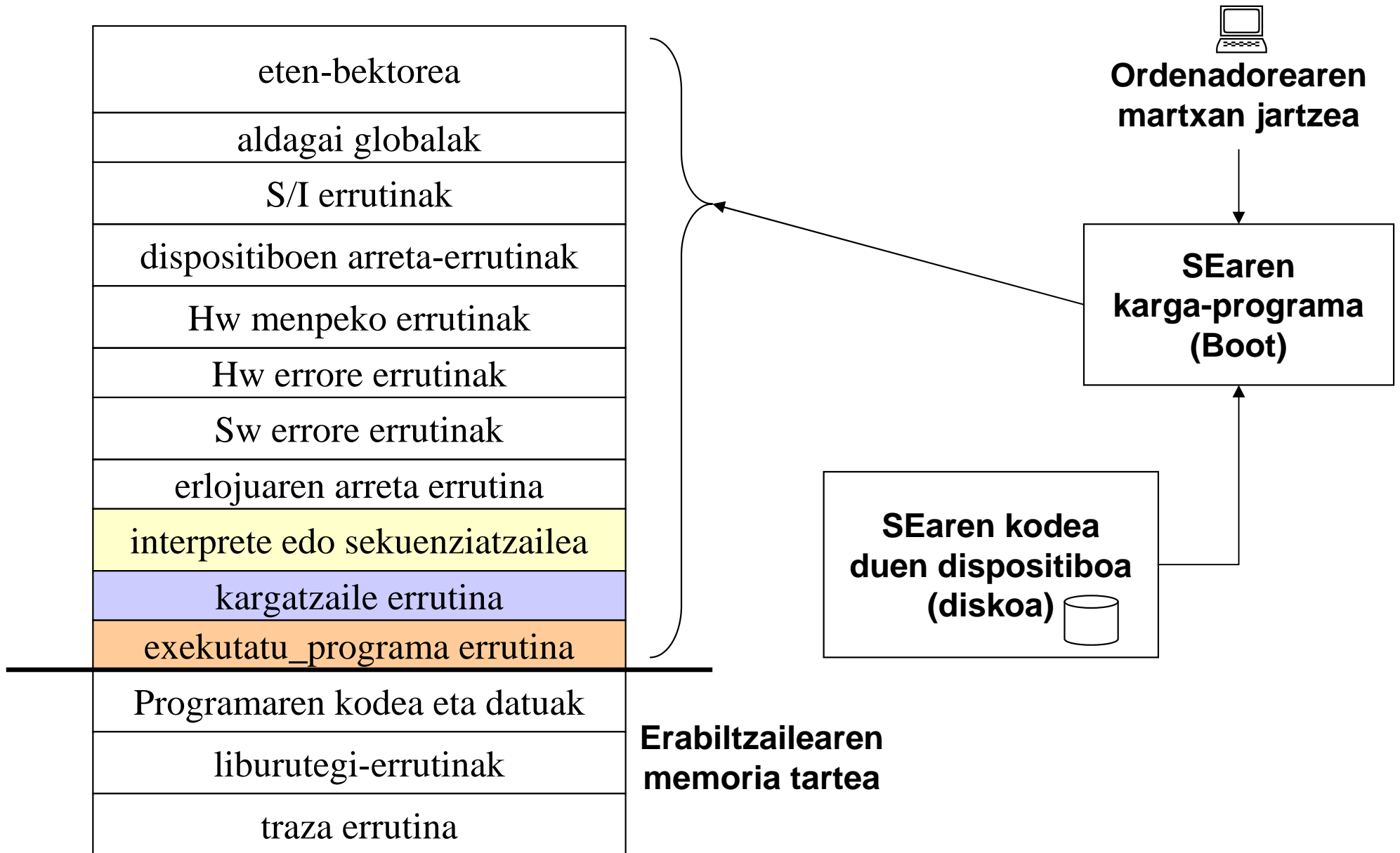
# GUI zerbitzaria: leihoen kudeaketa

- GUI shell-ak pantaila kontrolatzen du
  - Memoria erreserbatzen du leihoentzako
  - Leiho bakoitzaren jabea den aplikazioa identifikatzen du
  - Aplikazioei leiho anitz irekitzen uzten die...
  - Leihoak marraztuko diren ordena erabakitzen du.
  - Xaguaren klik-ak kontrolatu eta erantzuna ematen die

# KI-aren ezaugarriak

- Oinarrizko komandoak:
  - Programen kudeaketarako:
    - Programak abiatu (Run, Spawn, pipes,...)
    - Sarrera eta irteera estandarrak berbideratu
    - Prozesuak zerrendatu
    - Prozesuak bukatu
  - Fitxategi sistemaren kudeaketarako:
    - Fitxategiak: edukia erakutsi, kopiatu, izena aldatu, ezabatu, inprimatu, aldatu (editatu), ...
    - Katalogoak: edukia erakutsi, uneko katalogoa jakin/aldatu, katalogoak sortu, ezabatu, bilatu, ...

# SE baten lan egiteko modu orokorra



# SE monoprogramatu baten lan egiteko modua

## 1. SEa hasieratu

⇒ errutina egoiliarrak memorian kargatzen ditu, KI barne

⇒ KI-k hartzen du prozesadorearen kontrola

## 2. KI-k komandoak irakurtzen ditu aldez aurretik finkatutako gailu batetik, adibidez teklatutik

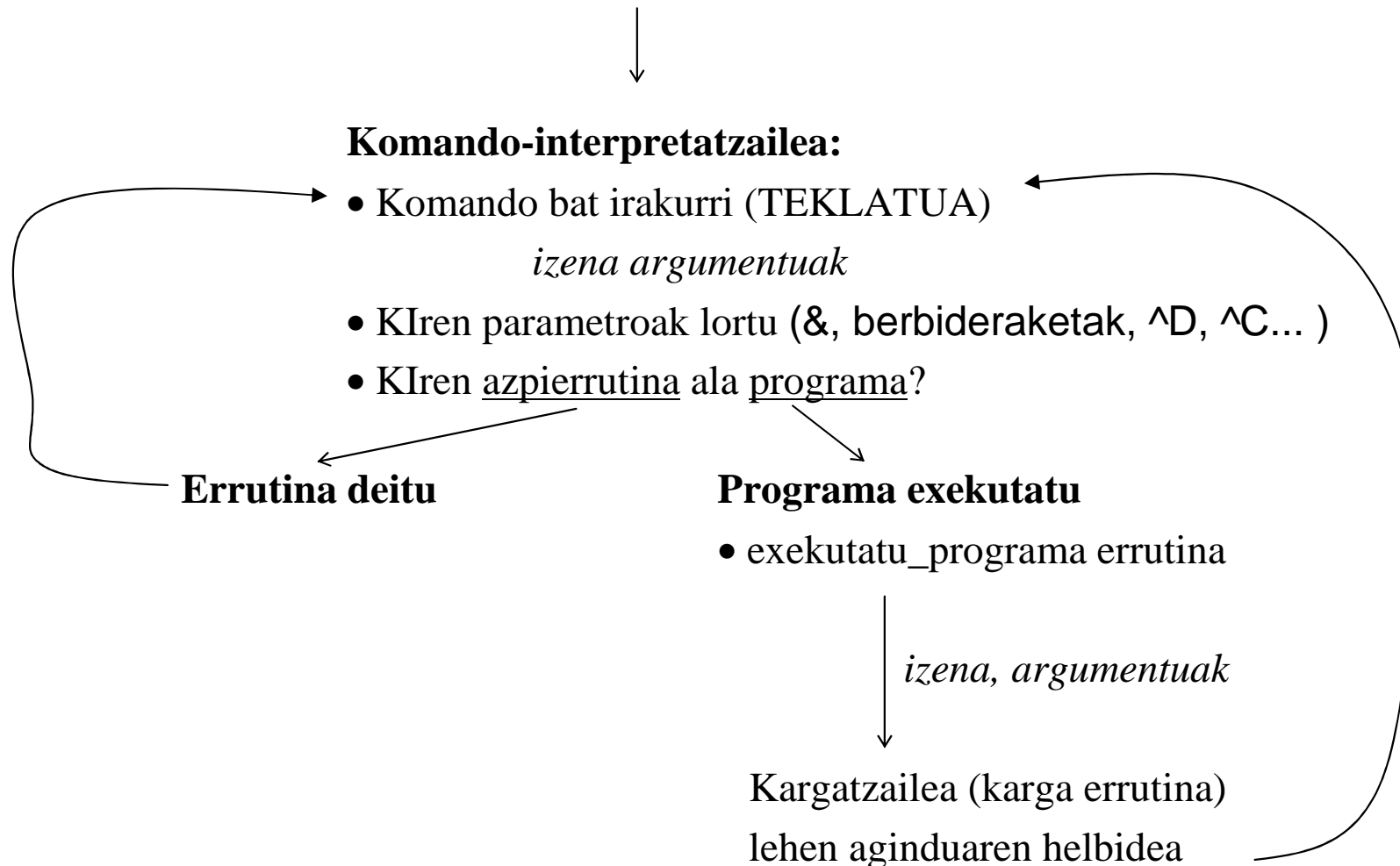
## 3. Komandoa programa bat ala komando-interpretatzailearen azpirrutina bat den egiaztatzen da:

- **Azpierrutina:** zuzenean exekuta daiteke errutina-dei baten bidez
- **Kanpo-programa:** kargatzailea ⇒ kontrola pasa

## 4. 2. Pausura bueltatu

# SE monoprogramatu baten lan egiteko modua (2)

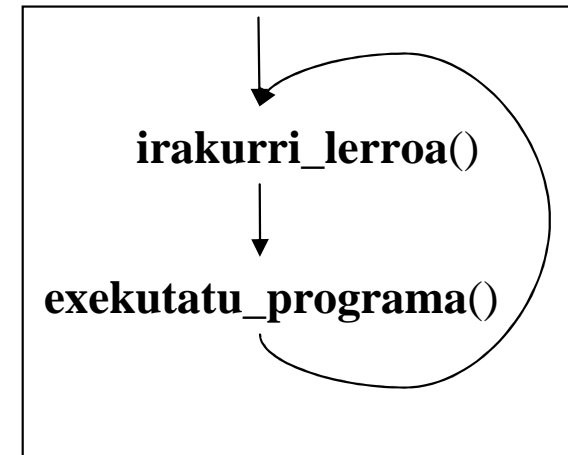
- 🖥️ Konputagailua piztu → SEaren errutinak memorian kargatzen duen programa (**Boot**)  
→ SEak behar dituen datu-egiturak kargatu eta hasieratu



# Komando-interpretatzailearen kodea

## komando interpretatzailea

```
Komando_interpretatzailea ( )  
{  
  char komando_lerroa[80];  
  for ( ; ; )  
  { lib_errut_sarrera_1(komando_lerroa, 80, SINKRO);  
    errore = lortu_parametroak(komando_lerroa, izena, argv);  
    if (!erroreak (KOMANDOEA, errore)) {  
      if (barne_errutina_da(izena))  
        deitu_errutina(izena, argv);  
      else  
        sistema_deia(EXEKUTATU_PROGRAMA);  
    }  
  }  
}
```



## aldagai globalak

```
char izena[20]; //komandoaren izena  
char *argv[MAX_ARGUMENTUAK];  
int argc;  
int (*helbidea)();
```

## exekutatu\_programa errutina

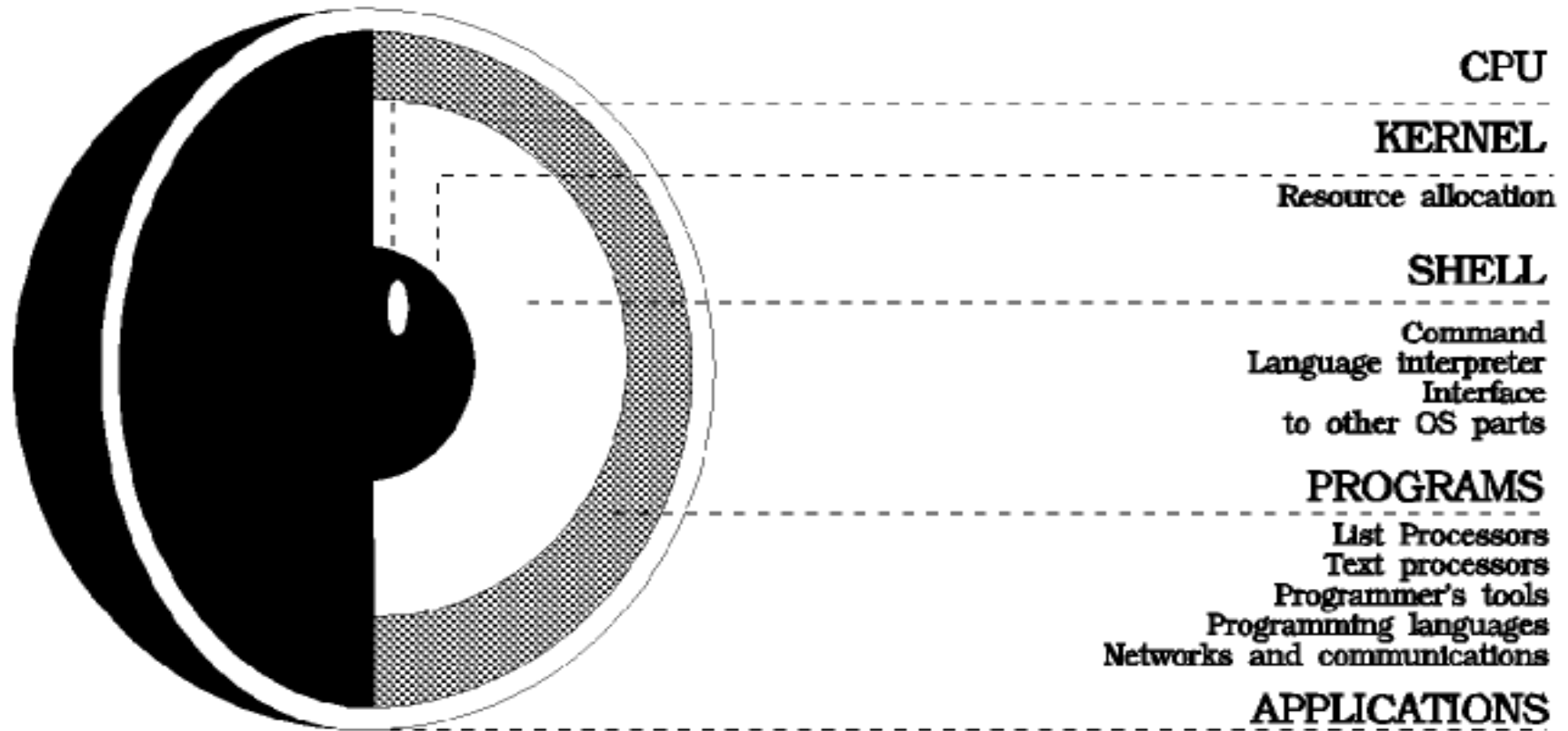
```
exekutatu_programa ( )  
{ if ((errore=kargatzailea(izena, &helbidea)) != OK)  
  erroreak(KARGATZAILEA, errore);  
  else  
    (*helbidea)();  
}
```

# UNIX

## ■ Hainbat KI (shell) dira posible UNIX instalazio arrunt batean:

- 1) *Bourne shell (\$)* – UNIX instalazio arrunten KI lehenetsia.  
Bourne shell garatu zen AT&T System V.2 UNIX ingurunerako.  
Sistemaren administratzaileek erabili ohi dute
- 2) *Korn shell (\$)* – Bourne shell-aren hobekuntza.  
Bourne shell-aren ezaugarri gehienak ditu, gehi berri batzuk ere.  
Industriaren estandarra izan ohi da erabiltzaile arruntentzat
- 3) *C shell (%)* – C programazio lengoaiaren oinarritutako KIa.  
Korn shell bezala, ezaugarri aurreratuak ditu, hots aliasing eta history. Sun etxeak programatzaileentzat garatu zuen C shell, baina geroz eta gehiago erabiltzaile arruntek erabili ohi dute gaur egun

# UNIX



# Windows-eko Klak

- *Run... command* hasierako menuan
  - WinWord
  - calc
  - cmd
- MS/DOS prompt-a
  - cd ..
  - cd PROGRA~1
  - cd MICROS~2
  - cd OFFICE
  - WinWord
- Baina ingurune grafikoa askoz ere errazagoa da