GIE/ICE

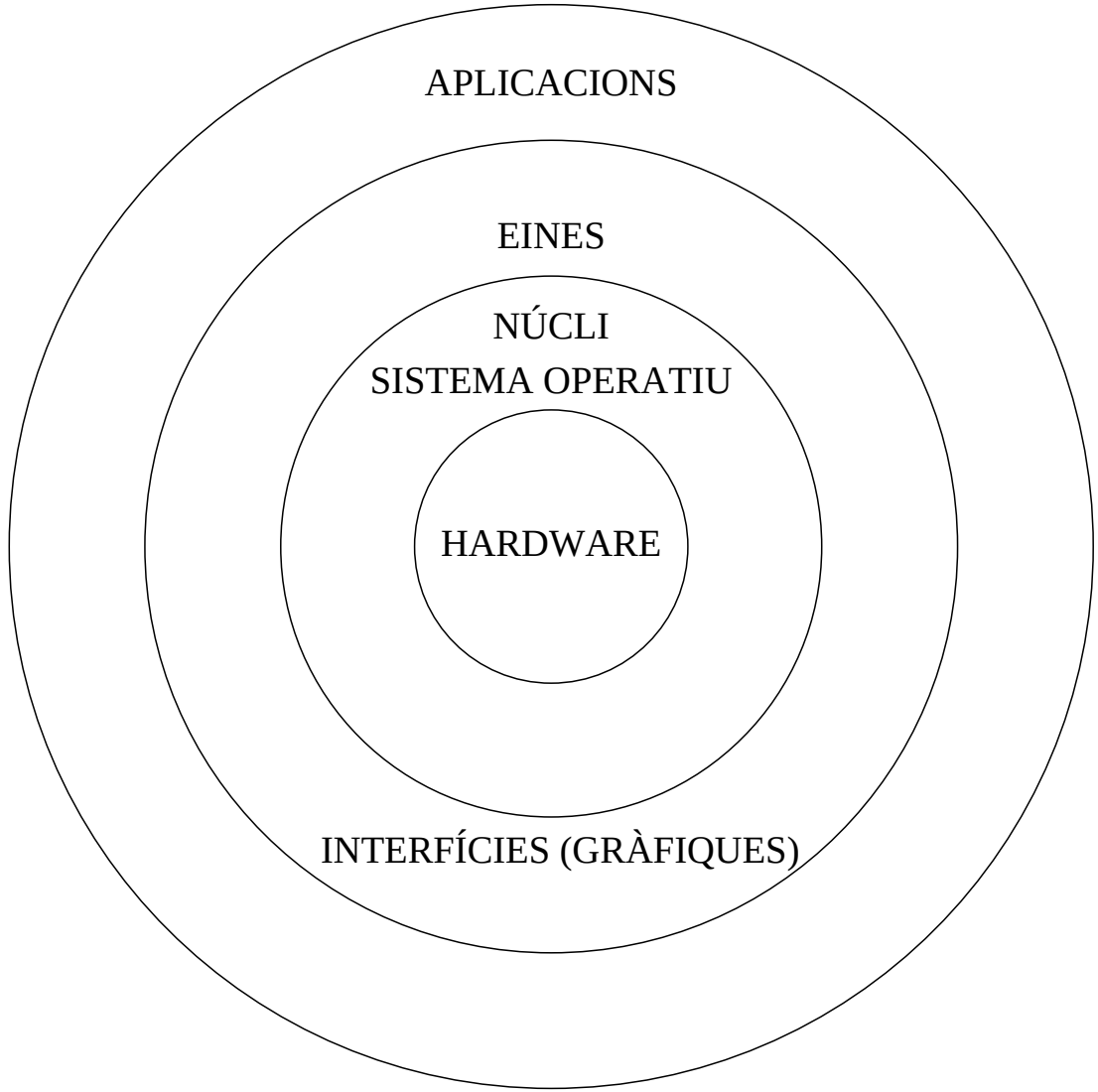
Introducció a Linux

Filosofia

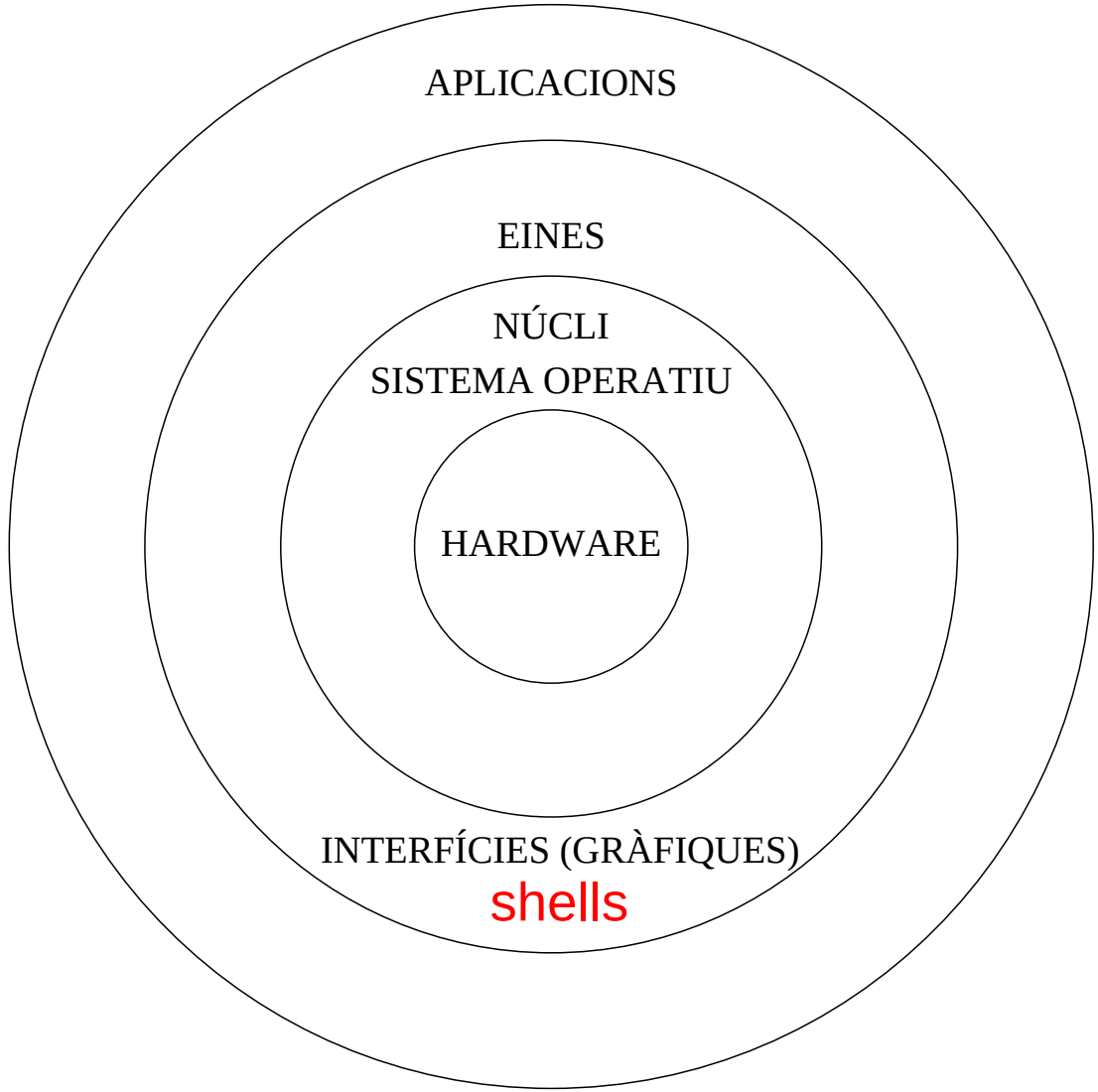
Josep Vilaplana

UPC/GIE

Nivells del sistema operatiu



Nivells del sistema operatiu



- Maximitzar els recursos disponibles de l'ordinador:
  - Processador: Utilitzant processos (programes en execució) de forma concurrent. Evita que cada programa usi el processador per esperar els recursos que necessita. El processador atén a altres programes que necessiten feines més útils. És un sistema operatiu **multi-tasca**.
  - Memòria: Evita repeticions de fragments de codi, l'organitza per ser compartida per la resta de processos i/o usuaris existents, i quan s'exhaureix la memòria principal, usa la la memòria secundària (**memòria virtual**).
  - Perifèrics: Maximitza la gestió d'entrades i sortides entre perifèrics.

- Maximitzar els recursos disponibles de l'ordinador:
  - Processador: Utilitzant processos (programes en execució) de forma concurrent. Evita que cada programa usi el processador per esperar els recursos que necessita. El processador atén a altres programes que necessiten feines més útils. És un sistema operatiu **multi-tasca**.
  - Memòria: Evita repeticions de fragments de codi, l'organitza per ser compartida per la resta de processos i/o usuaris existents, i quan s'exhaureix la memòria principal, usa la la memòria secundària (**memòria virtual**).
  - Perifèrics: Maximitza la gestió d'entrades i sortides entre perifèrics.
- Gestionar que el sistema sigui compartit per més d'un usuari (**multi-usuari**).

- Maximitzar els recursos disponibles de l'ordinador:
  - Processador: Utilitzant processos (programes en execució) de forma concurrent. Evita que cada programa usi el processador per esperar els recursos que necessita. El processador atén a altres programes que necessiten feines més útils. És un sistema operatiu **multi-tasca**.
  - Memòria: Evita repeticions de fragments de codi, l'organitza per ser compartida per la resta de processos i/o usuaris existents, i quan s'exhaureix la memòria principal, usa la la memòria secundària (**memòria virtual**).
  - Perifèrics: Maximitza la gestió d'entrades i sortides entre perifèrics.
- Gestionar que el sistema sigui compartit per més d'un usuari (**multi-usuari**).
- Gestionar aspectes de seguretat entre usuaris aportant mecanismes de protecció.

- Maximitzar els recursos disponibles de l'ordinador:
  - Processador: Utilitzant processos (programes en execució) de forma concurrent. Evita que cada programa usi el processador per esperar els recursos que necessita. El processador atén a altres programes que necessiten feines més útils. És un sistema operatiu **multi-tasca**.
  - Memòria: Evita repeticions de fragments de codi, l'organitza per ser compartida per la resta de processos i/o usuaris existents, i quan s'exhaureix la memòria principal, usa la la memòria secundària (**memòria virtual**).
  - Perifèrics: Maximitza la gestió d'entrades i sortides entre perifèrics.
- Gestionar que el sistema sigui compartit per més d'un usuari (**multi-usuari**).
- Gestionar aspectes de seguretat entre usuaris aportant mecanismes de protecció.
- **Independència del hardware** sobre el que hagi de recòrrer. El disseny d'Unix/Linux ha demostrat estar preparat per assolir les possibles innovacions que puguin sorgir.

- Maximitzar els recursos disponibles de l'ordinador:
  - Processador: Utilitzant processos (programes en execució) de forma concurrent. Evita que cada programa usi el processador per esperar els recursos que necessita. El processador atén a altres programes que necessiten feines més útils. És un sistema operatiu **multi-tasca**.
  - Memòria: Evita repeticions de fragments de codi, l'organitza per ser compartida per la resta de processos i/o usuaris existents, i quan s'exhaureix la memòria principal, usa la la memòria secundària (**memòria virtual**).
  - Perifèrics: Maximitza la gestió d'entrades i sortides entre perifèrics.
- Gestionar que el sistema sigui compartit per més d'un usuari (**multi-usuari**).
- Gestionar aspectes de seguretat entre usuaris aportant mecanismes de protecció.
- **Independència del hardware** sobre el que hagi de recòrrer. El disseny d'Unix/Linux ha demostrat estar preparat per assolir les possibles innovacions que puguin sorgir.

- Maximitzar els recursos disponibles de l'ordinador:
  - Processador: Utilitzant processos (programes en execució) de forma concurrent. Evita que cada programa usi el processador per esperar els recursos que necessita. El processador atén a altres programes que necessiten feines més útils. És un sistema operatiu **multi-tasca**.
  - Memòria: Evita repeticions de fragments de codi, l'organitza per ser compartida per la resta de processos i/o usuaris existents, i quan s'exhaureix la memòria principal, usa la la memòria secundària (**memòria virtual**).
  - Perifèrics: Maximitza la gestió d'entrades i sortides entre perifèrics.
- Gestionar que el sistema sigui compartit per més d'un usuari (**multi-usuari**).
- Gestionar aspectes de seguretat entre usuaris aportant mecanismes de protecció.
- **Independència del hardware** sobre el que hagi de recòrrer. El disseny d'Unix/Linux ha demostrat estar preparat per assolir les possibles innovacions que puguin sorgir.

- Maximitzar els recursos disponibles de l'ordinador:
  - Processador: Utilitzant processos (programes en execució) de forma concurrent. Evita que cada programa usi el processador per esperar els recursos que necessita. El processador atén a altres programes que necessiten feines més útils. És un sistema operatiu **multi-tasca**.
  - Memòria: Evita repeticions de fragments de codi, l'organitza per ser compartida per la resta de processos i/o usuaris existents, i quan s'exhaureix la memòria principal, usa la la memòria secundària (**memòria virtual**).
  - Perifèrics: Maximitza la gestió d'entrades i sortides entre perifèrics.
- Gestionar que el sistema sigui compartit per més d'un usuari (**multi-usuari**).
- Gestionar aspectes de seguretat entre usuaris aportant mecanismes de protecció.
- **Independència del hardware** sobre el que hagi de recòrrer. El disseny d'Unix/Linux ha demostrat estar preparat per assolir les possibles innovacions que puguin sorgir.

Unix/Linux promou simplicitat i "elegància" en:

- Disseny de programes amb objectius senzills ben definits, exactes i clars.

Unix/Linux promou simplicitat i "elegància" en:

- Disseny de programes amb objectius senzills ben definits, exactes i clars.
- Projectes senzills requereixen menys temps de desenvolupament i verificacions (tests) més senzilles.

Unix/Linux promou simplicitat i "elegància" en:

- Disseny de programes amb objectius senzills ben definits, exactes i clars.
- Projectes senzills requereixen menys temps de desenvolupament i verificacions (tests) més senzilles.
- Independència del projecte a requisits més complexes com pot ser el llenguatge de programació, API's i altres vinculacions pròpies d'un desenvolupament.

Unix/Linux promou simplicitat i "elegància" en:
Per cada context d'aplicació, disposar de

- Biblioteques de subprogrames pel llenguatge de programació decidit.

Unix/Linux promou simplicitat i "elegància" en:
Per cada context d'aplicació, disposar de

- Biblioteques de subprogrames pel llenguatge de programació decidit.
- Llenguatges de guions que permetin fer sistemes més complexes a partir d'altres programes o guions més senzills i així aprofitar el que ja està fet.

Unix/Linux promou simplicitat i "elegància" en:
Per cada context d'aplicació, disposar de

- Biblioteques de subprogrames pel llenguatge de programació decidit.
- Llenguatges de guions que permetin fer sistemes més complexes a partir d'altres programes o guions més senzills i així aprofitar el que ja està fet.
- Ús obert i flexible dels programes.

Unix/Linux promou simplicitat i "elegància" en:
Per cada context d'aplicació, disposar de

- Biblioteques de subprogrames pel llenguatge de programació decidit.
- Llenguatges de guions que permetin fer sistemes més complexes a partir d'altres programes o guions més senzills i així aprofitar el que ja està fet.
- Ús obert i flexible dels programes.
- Màxim ús de les possibilitats del sistema operatiu.

Unix/Linux promou simplicitat i "elegància" en:
Per cada context d'aplicació, disposar de

- Biblioteques de subprogrames pel llenguatge de programació decidit.
- Llenguatges de guions que permetin fer sistemes més complexes a partir d'altres programes o guions més senzills i així aprofitar el que ja està fet.
- Ús obert i flexible dels programes.
- Màxim ús de les possibilitats del sistema operatiu.
- Unificació i Màxim aprofitament d'una eina. La funcionalitat d'una comanda/programa serveix per qualsevol dispositiu de hardware.

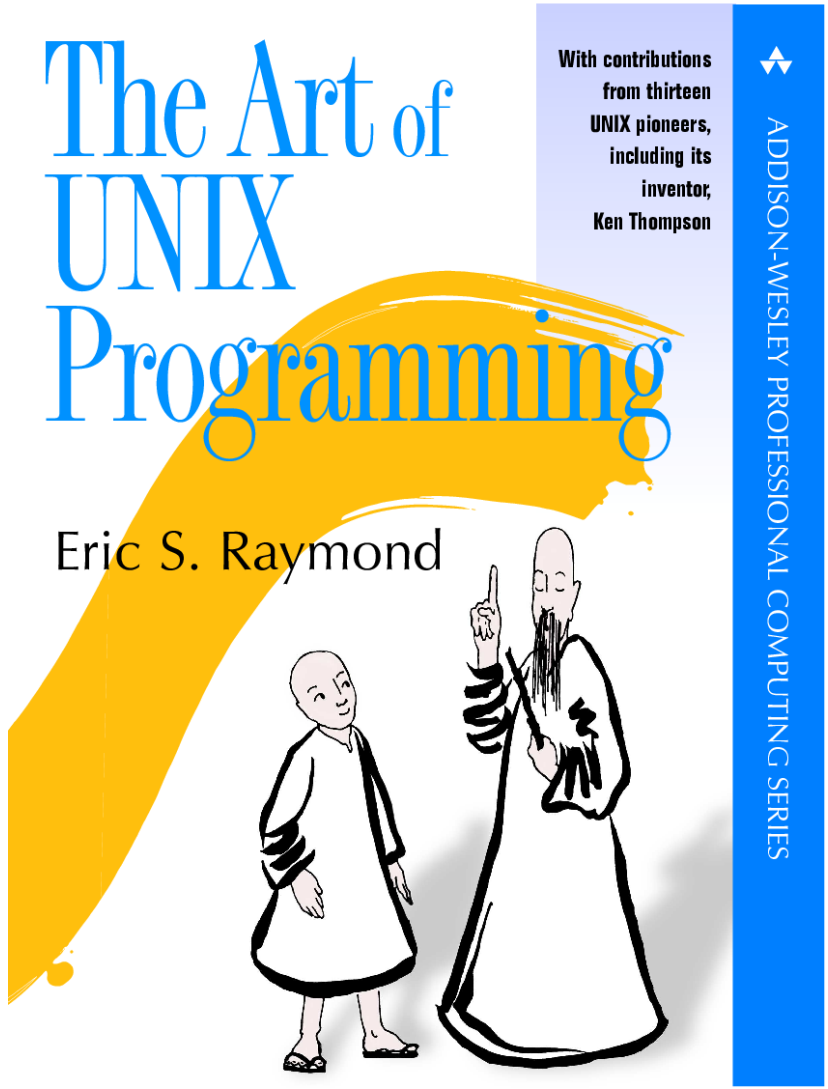
Unix/Linux promou simplicitat i "elegància" en:
Per cada context d'aplicació, disposar de

- Biblioteques de subprogrames pel llenguatge de programació decidit.
- Llenguatges de guions que permetin fer sistemes més complexes a partir d'altres programes o guions més senzills i així aprofitar el que ja està fet.
- Ús obert i flexible dels programes.
- Màxim ús de les possibilitats del sistema operatiu.
- Unificació i Màxim aprofitament d'una eina. La funcionalitat d'una comanda/programa serveix per qualsevol dispositiu de hardware.
- Competitivitat i consens

Unix/Linux promou simplicitat i "elegància" en:
Per cada context d'aplicació, disposar de

- Biblioteques de subprogrames pel llenguatge de programació decidit.
- Llenguatges de guions que permetin fer sistemes més complexes a partir d'altres programes o guions més senzills i així aprofitar el que ja està fet.
- Ús obert i flexible dels programes.
- Màxim ús de les possibilitats del sistema operatiu.
- Unificació i Màxim aprofitament d'una eina. La funcionalitat d'una comanda/programa serveix per qualsevol dispositiu de hardware.
- Competitivitat i consens
- Inconvenient agradable? pel programador/usuari: moltes eines a decidir per qualsevol objectiu.

Filosofia UNIX.



- X aporta mecanismes no pas principis o normes: Desplaça les decisions sobre les eines i la presentació (normes) a nivell d'aplicació. Els programes d'Unix aporten moltes opcions de comportament i preferències.

Filosofia UNIX.

- X aporta mecanismes no pas principis o normes: Desplaça les decisions sobre les eines i la presentació (normes) a nivell d'aplicació. Els programes d'Unix aporten moltes opcions de comportament i preferències.
- "it is better to solve the right problem the wrong way than the wrong problem the right way" Dick Hamming

Filosofia UNIX.

- X aporta mecanismes no pas principis o normes: Desplaça les decisions sobre les eines i la presentació (normes) a nivell d'aplicació. Els programes d'Unix aporten moltes opcions de comportament i preferències.
- "it is better to solve the right problem the wrong way than the wrong problem the right way" Dick Hamming
- Usuaris no tècnics s'atabalen amb tantes opcions i estils, que prefereixen aquells sistemes que com a mínim ofereixen simplicitat.

Filosofia UNIX.

- Doug McIlroy 78:
  Fes que cada programa faci una cosa bé. Per fer un nou treball, construeix quelcom fresc en comptes de complicar un programa vell afegint-li noves característiques.

Filosofia UNIX.

- Doug McIlroy 78:
  Fes que cada programa faci una cosa bé. Per fer un nou treball, construeix quelcom fresc en comptes de complicar un programa vell afegint-li noves característiques.
- Compta amb que la sortida d'un programa pot esdevenir l'entrada a un altre programa encara per conèixer. No facis sortides confoses amb informació estranya. Evita formats d'entrada binaris. No insisteixis sobre entrades interactives.

Filosofia UNIX.

- Doug McIlroy 78:
  Fes que cada programa faci una cosa bé. Per fer un nou treball, construeix quelcom fresc en comptes de complicar un programa vell afegint-li noves característiques.
- Compta amb que la sortida d'un programa pot esdevenir l'entrada a un altre programa encara per conèixer. No facis sortides confoses amb informació estranya. Evita formats d'entrada binaris. No insisteixis sobre entrades interactives.
- Dissenya i construeix programari, inclòs sistemes operatius, per ser provats aviat, idealment en setmanes. No t'amoïnis per llençar parts conflictives i reconstruir-les.

Filosofia UNIX.

- Usa preferentment eines que ajudin a alleugerir la tasca de programació, inclòs si cal, construir-les per després llençar-les un cop s'ha acabat.

Filosofia UNIX.

- Usa preferentment eines que ajudin a alleugerir la tasca de programació, inclòs si cal, construir-les per després llençar-les un cop s'ha acabat.
- La filosofia d'Unix és: Escriu programes que facin una cosa i la facin bé. Escriu programes que puguin treballar junts. Escriu programes que gestionin fluxos de text, ja que és una interfície universal.

Filosofia UNIX.

- Regla de la Modularitat: Escriu parts senzilles connectades per interfícies clares.

Filosofia UNIX.

- Regla de la Modularitat: Escriu parts senzilles connectades per interfícies clares.
- Regla de la Claredat: Millor ser clar que enginyós.

Filosofia UNIX.

- Regla de la Modularitat: Escriu parts senzilles connectades per interfícies clares.
- Regla de la Claredat: Millor ser clar que enginyós.
- Regla de la Composició: Dissenya programes per ser connectats a altres programes.

Filosofia UNIX.

- Regla de la Modularitat: Escriu parts senzilles connectades per interfícies clares.
- Regla de la Claredat: Millor ser clar que enginyós.
- Regla de la Composició: Dissenya programes per ser connectats a altres programes.
- Regla de la Separació: Separa normes del mecanisme; separa interfícies dels motors.

Filosofia UNIX.

- Regla de la Simplicitat: Dissenya per a la simplicitat; afegeix complexitat només on calgui.

Filosofia UNIX.

- Regla de la Simplicitat: Dissenya per a la simplicitat; afegeix complexitat només on calgui.
- Regla de la Parsimònia: Escriu un programa gran només quan està ben demostrat que no hi cap altra manera de fer-ho.

Filosofia UNIX.

- Regla de la Simplicitat: Dissenya per a la simplicitat; afegeix complexitat només on calgui.
- Regla de la Parsimònia: Escriu un programa gran només quan està ben demostrat que no hi cap altra manera de fer-ho.
- Regla de la Transparència: Dissenya per a la visibilitat per fer la inspecció i seguiment i trobada d'errors més fàcil.

Filosofia UNIX.

- Regla de la Simplicitat: Dissenya per a la simplicitat; afegeix complexitat només on calgui.
- Regla de la Parsimònia: Escriu un programa gran només quan està ben demostrat que no hi cap altra manera de fer-ho.
- Regla de la Transparència: Dissenya per a la visibilitat per fer la inspecció i seguiment i trobada d'errors més fàcil.
- Regla de la Robustesa: La robustesa és el fill de la transparència i la simplicitat.

- Regla de la Representació: Recull el coneixement en dades de forma que el programa sigui estúpid i robust.

Filosofia UNIX.

- Regla de la Representació: Recull el coneixement en dades de forma que el programa sigui estúpid i robust.
- Regla de la Mínima Sorpresa: En disseny d'interfícies, sempre fer la cosa menys sorprenent.

Filosofia UNIX.

- Regla de la Representació: Recull el coneixement en dades de forma que el programa sigui estúpid i robust.
- Regla de la Mínima Sorpresa: En disseny d'interfícies, sempre fer la cosa menys sorprenent.
- Regla del Silenci: Quan un programa sorprenentement no té res a dir, no hauria de dir res.

Filosofia UNIX.

- Regla de la Representació: Recull el coneixement en dades de forma que el programa sigui estúpid i robust.
- Regla de la Mínima Sorpresa: En disseny d'interfícies, sempre fer la cosa menys sorprenent.
- Regla del Silenci: Quan un programa sorprenentement no té res a dir, no hauria de dir res.
- Regla de Reparació: Quan hagi de fallar, que falli sorollosament, i tant aviat com sigui possible.

Filosofia UNIX.

- Regla d'Economia: El temps del programador és car; conserva'l en preferència al temps de màquina.

Filosofia UNIX.

- Regla d'Economia: El temps del programador és car; conserva'l en preferència al temps de màquina.
- Regla de Generació: Evita hand-hacking; quan puguis escriu programes per escriure programes.

Filosofia UNIX.

- Regla d'Economia: El temps del programador és car; conserva'l en preferència al temps de màquina.
- Regla de Generació: Evita hand-hacking; quan puguis escriu programes per escriure programes.
- Regla de Optimització: Prototipa abans de polir. Fes que treballi abans d'optimitzar.

Filosofia UNIX.

- Regla d'Economia: El temps del programador és car; conserva'l en preferència al temps de màquina.
- Regla de Generació: Evita hand-hacking; quan puguis escriu programes per escriure programes.
- Regla de Optimització: Prototipa abans de polir. Fes que treballi abans d'optimitzar.
- Regla de Diversitat: Desconfia de totes les pretensions d"'un únic camí veritable".

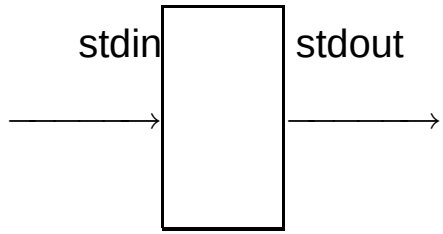
- Regla d'Extensibilitat: Dissenya pel futur, ja que esdevindrà més aviat del que penses.

En una sola frase:

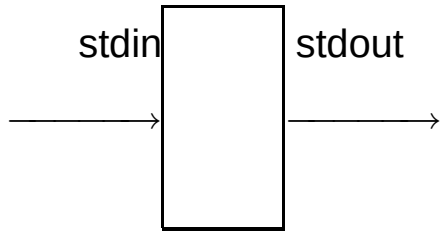
K.I.S.S.

Keep it Simple Stupid

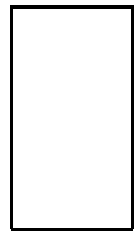
- El Filtre *(grep, tr, ...)*
  Prescripció de Postel: Sigues generós amb el que acceptes i rigorós amb el que emets.
  Quan filtris no llencis la informació que no necessites.
  Quan filtris no afegeixis soroll.



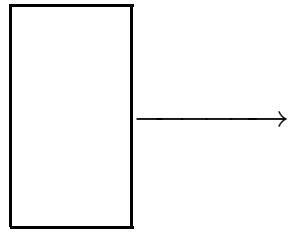
- El Filtre *(grep, tr, ...)*
  Prescripció de Postel: Sigues generós amb el que acceptes i rigorós amb el que emets.
  Quan filtris no llencis la informació que no necessites.
  Quan filtris no afegeixis soroll.



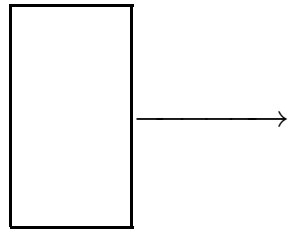
- El "cantrip" *(rm,. touch, startx)*
  cap entrada, cap sortida. Nomes un status numèric.



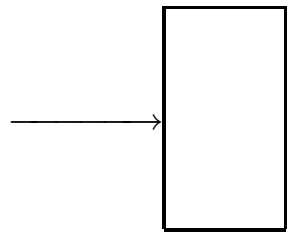
- La font *(ls, ps, who)*
  cap entrada, emet sortida controlada per les condicions d'inici.



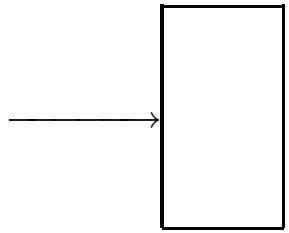
- La font *(ls, ps, who)*
  cap entrada, emet sortida controlada per les condicions d'inici.



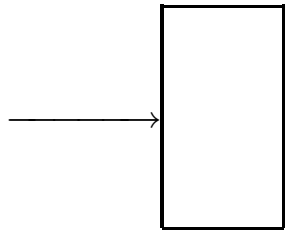
- El Clavegueró *(lpr, mail)*
  consumeix entrada, cap sortida.



- Esponja consumeix tota l'entrada, abans de fer la sortida (sort)

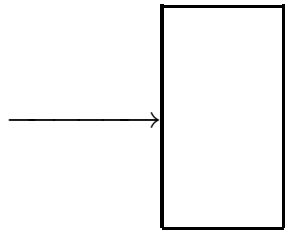


- Esponja consumeix tota l'entrada, abans de fer la sortida (sort)



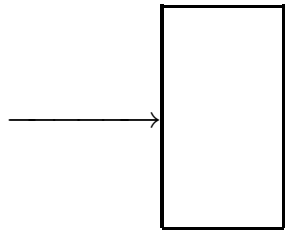
- Compilador (gcc, gif2png, gzip)
  cap entrada, cap sortida. Emet en el canal d'errors els errors. Transforma fitxers

- Esponja consumeix tota l'entrada, abans de fer la sortida (sort)



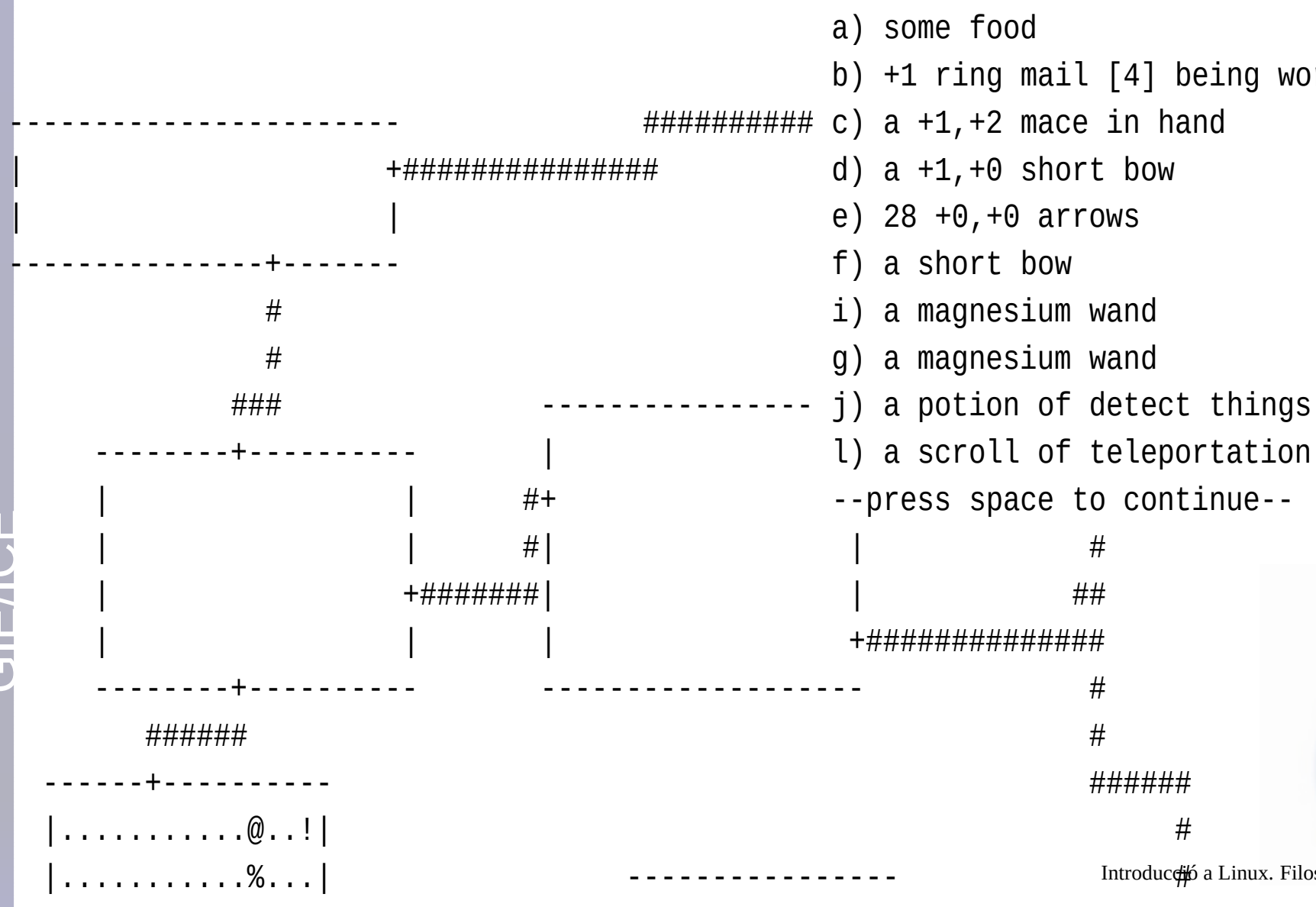
- Compilador (gcc, gif2png, gzip)
  cap entrada, cap sortida. Emet en el canal d'errors els errors. Transforma fitxers
- ed

- Esponja consumeix tota l'entrada, abans de fer la sortida (sort)



- Compilador (gcc, gif2png, gzip)
  cap entrada, cap sortida. Emet en el canal d'errors els errors. Transforma fitxers
- ed
- roguelike visualització a traves dels caràcters del terminal, interfície gràfica.

Joc original de Rogue:

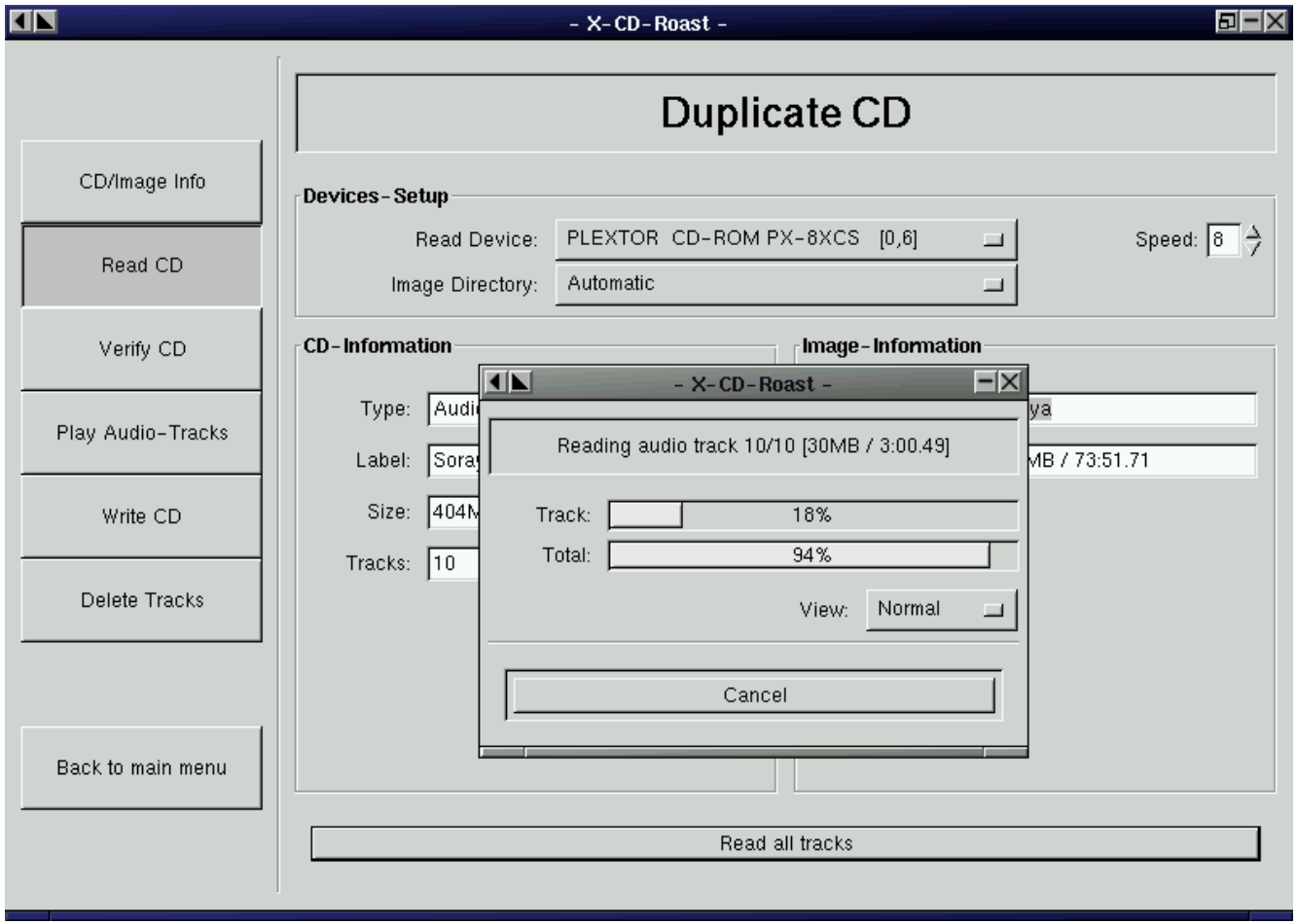


- motor i interfície separat *model-view-controller*

- motor i interfície separat *model-view-controller*
- parella configurador/actor *fetchmail/fetchmailconf*

- motor i interfície separat *model-view-controller*
- parella configurador/actor *fetchmail/fetchmailconf*
- una variant de parella configurador/actor: parella spooler/dimoni accés serialitzat a un recurs compartit mitjançant lots. *lpr/lpd at/atd sendmail i qmail*

- parella controlador (conductor)/motor
  per ex. gv o ghostscript (interfícies gr.) i gs (interpret postscript) xcdroast amb mkisofs i cdrecord



- parella client/servidor és com la parella controlador (conductor)/motor però la part del motor és un dimoni sense interfície d'usuari, normalment es per accedir a un recurs compartit (base de dades, flux de transaccions, hardware com pla placa de so.) Essent un dimoni s'estalvia costos d'inicialització.
  Ec: ftp/ftpd navegador/portal psql,

- parella client/servidor és com la parella controlador (conductor)/motor però la part del motor és un dimoni sense interfície d'usuari, normalment es per accedir a un recurs compartit (base de dades, flux de transaccions, hardware com pla placa de so.) Essent un dimoni s'estalvia costos d'inicialització.
  Ec: ftp/ftpd navegador/portal psql,
- servidor CLI (Command line interpreter)
  interfície basada en (mini)llenguatge de comandes (shells, bc, dc)
  llenguatges de guió encastrats (emacs, gimp (script-fu))