

Sisteme eragile klasikoak

Batch Sistemak

txartel zolatuek

Ez eragiltorre S/I-ren zain

Mono programazioa

Soluzioa

Multitask/Multiprogramazioa

S/Iren zain beste atzede egin Andren

Programazio exekuzio tartelak

Programazio Memoriak gara

Elkarrekin-tza (Denbora-partekatutako sisteme)

Quantum (denbora-tarte txikiak)

Erantzen denbora txikiak

Multi-programazio sententzia
(Unix - ez jabeak)

Eragiltzaile aritzeko sisteme

-Segurtasuna

Interfaze grafikoa

Sozializazioa → Linux

Administrazio-atzede erraztu

Sareak (TCP/IP)

Berezio-zerbitzari-eskema

Sisteme autonomak

Gailu Murgillorrek

Sisteme eragileko murgillorrek (Android)

S/I-gailu berrak eta interakzio berrak

Sarek konexioak berrak.

Sisteme deiekt Bufferen betetean egiten dira, haxe sisteme
beteen egon daitezkeen abiadura diferentziala elidituko. Ondorioz
unitateen bufferen eta putcher-aren ez dira berdineak eta
ez dute modo beteen idatziko nonenb

Apunteak 500

Prozesuak eta Egoneak

Exekutatzen 5 modu edo egoera

↳ Running → martxan.

Sleeping → lo edo zain

Stuck → trabatuta.

Zombie → zombi

Stopped → blokeatuta geldituta.

Programa guztiak egin memoria egoan, baina eragileak
hau banpartzen.

1. Prozesuak sortu → fork() → id berrituta

2. Exekutatu → hasi prozesu
Zain

↳ programa berridatzen eta
testuinguru ezarri.

- Scheduler → politika bat erabiltzen, ze prozesu aurreratu
erabaki.

- Dispatcher → Exekutatu edo den programa exekutatu.

3. Blokeatu

↳ Sistema dierren bidez

4. Etena

↳ Sistema erabiltzen ez denean

↳ blokeatutako prozesuak - parte

egonean jartzen; ~~erabiltzen, dena erabiltzen.~~

~~5. Berridatzen~~

↳

↳ Stopped egoan
utzi

↳ Lokatutako
programa berridatzen
exekutatu.

5. Berridatzen → exit(0) sistema dena erabiltzen den bakoitzak,
eta berridatzen prozesuak erabiltzen ari zenean berridatzen guztiak
erabiltzen diren. Horrela, hurrengo prozesuak exekutatuak utzi.

2a. gortetik de gureta unue batua baten bultzea da?

↳ Unue unesuta gortetik de eta init
prozesuak herikito du beregarik.

Prozesuak

↳ Dependientiak edo independentiak

Prozesu independentiak → ~~bat~~ bi prozesu desberdinak
argumentu berdinekin puzten bidekizko, emaitza berdineko izango
da, lehen artea eta baitago dependentsia eta ondorioz exekuzioa
beheratzen bera obetitu egongo du, emaitza bera lortuz.

Prozesu dependientiak → Kasu konatan, bi prozesuak artean
dependentsia izango da, ondorioz emaitza desberdinekin bideko
gortituko. Bate, baina baina gortituko exekuzioa ezker.

Bi kasu du unue desberdin bereizten dira.

Elasiko-kontakuntza eredu. Kasu da, elasoak
tankean jartzea buffer baten artean ditzu denak eta kontakuntza
tzeirekin bidekizko lortu.

Berezko-zerbitzari eredu: Elementuak artean dira
eta periodikoki beste prozesu baten eta elementuak b. al. dagoen
konprobaturak da.

Komunikazioa

↳ prozesu arteko informazio trukeak

Sinkronizazioa

↳ Entzutea erabiliz ⇒ Esplizituki:

↳ Implizituki: ⇒ komunikazio-mekanismoa bidez

Selezio kritika

↳ CPU bakoari doli bere lan egiteko deiala selezio

~~korrektu~~ komunikazioa exekutatzen denean gertatzen da,
CPU bakoitako i helio desbordinatzen ondorioz, esan ditzake bi helio
exekutatzen ibili.

↳ Hainbat prozesuak abiatzen duten ~~pro~~ kode-zatiari deritzen

Fitoa komunikazioaren erabiltzea den fita-ategia da, PPE elin
alderatuta, ez dela aurrinua eta bi inplementazio abizagarrak dela
ere desatxatu. Beraz, idetzi ekel izateko derrigorrezkoak da bi
bakoak muntzatu egotea. Eta, narrait bakoari idetziari orri kode
bakoak ibazten egiten ~~egiten~~ behar da, bakoak aurrin ~~beste~~

Mez - trukaren bidezko komunikazioa esateko zero, adarago
metodak daude.

~~na~~ Buzoiak FIFO (First in First Out): Buzoi mota hau
hainbat funtzioaren bidez sortu daitezke:

uInode(): PPA heliozko fita-ategia bat sortuko da,
~~beraz~~ eta puntu bako ~~da~~ bidez bakoari begiratu da.

int uInode (char *path, int mode, int dev)

Fifo bat sortu nda: kode erabilera ondorengoa da.

fd = uInode (str, O_RDONLY | O_CREAT, 0);

Beste funktion habe ich mit ~~figo~~ $\text{figo}()$ besu

becareea surselor de putere-jermitelor izvoarelor, "noile" surse
energetice noi care trebuie definite din nou.

Isentropic, adiabatic and isothermal, Pipe derivate:

int pipe() \rightarrow pipe bat sartuko du \rightarrow non fitxategi batera bidartuko den puntuera, eta bertan idetzi edo irakurri egia ditzakeen, normalean pdf hori o balioa sartze bidez irakurketa egiten eta bat balioa idazteko, eragiketa egia ondoren informazioa garbitzea da.

PAGE - c

PPG-a
La Noranda balneariile de pi prozeșure ertelie
pipe baten, pipe-c sortu duc prozeșure ~~baterile~~ ~~epitig~~
sincronizatu chel izorgo de kate prozeșurelori baine obiectuiz
cizurele izorgo de Baldu ete, ca kate sortu baine pipe bat
sitratigi bari begindze.

Prozesse kontaktierendes System-objekt

Identifikasjon

getpid(): Prozesoren identifizieren, beibehalten do

getppid(): Prozessor, Prozess garantieren identifikation beibehalten

getuides: Prozesoren jobs, Erhältlichkeit und Identifikation beibehalten

P

Procedural Sottare

Sorte U: Sistema-dei lavelle prozesu ~~berri~~ bat sortze du, hau prozesu-umen deritza. Prozesu umeak gurazoren ezaugarri guztiak berak ditu, ~~bera~~ identifikatzea ize, aita, lavelle O beldioa lortzeko du eta gurazola umeak.

examp: parametro beside pasatzen daizen `##` "programa, konduko..."
exekutatuko da, Bultzeren null balioa jartze baten zati bati. Arren
da jartze baten dala pasatuko daizkion argumentu kopur
Zain den.

exemp: lisa konstan bi ~~egomente~~ ^{parameter} soilih pastulu disliogo.

erfolgte nicht der programm etc. konnte dieser argumentiert
Adoniz, bei, herrschere Alter über, es die bekanntesten jenseits
Zitat den originale Kopie zahlreich.

Prozess der Erinnerung

Politi klasifikasi algoritmoda dard koratda, kor shodlari
definitza diti. Korat adingta bati eksistat diti ordene.
Korbat inzipide dard alkoratada ordene.

CPU-charge ist keine beste Praxis:

First-Come First-Serve \rightarrow FiFo baze bidez implementazioa da.
~~prozesu~~ denbora gehien daraman prozesu exekutza de labeirik ez du
 Gogorikorra, baina ez bade ~~egokiorra~~. gogorien prozesu exekutzeaz
 startatzen da.

Shortest Job First (SJF): Lan guthia gin belar dia proses
eksekusi da lakaing posisi, so omisiune de gende lilelele
pisu gelier dia proses inoi ez eksekusi.

Parabola kado lixavole kado prosesuei

Round Robin \rightarrow o processo baloitate de bora tarte letidago do bora exeluziva guezatela de "quantum", dentao de bora-tarte horri, de bora tarte horrefen buletzen ez bade exeluzioa, prozesua, prozesu-iberen buleera jatan de eta lurrego prozesu erebutzen hasiko da.

Prozesse lebendigeren oder asiguelen

Prozess hat sorten davon, schiedlich. Zahl hat
ganz die lebendigeren oder, die noch charakter d.
Prozess behaltenden pisen.

lebendigeren estetik: Zahl oder es ist absetzen

lebendigeren dynamik: absetz gegen deutsche prozess.
exekutivene da einen.

Adoriz com de jahn ze prozess exekutivene der lebendiger
es beideltige ze politike erhaltend oder schiedlich.

Fitxategi sistemen ezaugarriak

Ikuspuntu funtzional baten arabera, fitxategi bat, ezaugarri batzuk dituen informazio multzo bat dela esan daiteke. Fitxategien ezaugarriak hauek dira:

- Informazioa era iraunkorean gordetzen da, programek erabiltzen dituzten beste objektu batzuk ez bezala (adibidez aldagaiak).
- Karaktere kate batez irudikatzen den izen baten bidez identifikatzen dira. Izenak esanahi berezia duen zati bat izan dezake (luzapen bat) edo ez, Linux-en bezala. Luzapena fitxategiaren mota zehazteko erabil daiteke.
- Fitxategiak direktorioetan multzokatzen dira. Direktorioek antolaketa egitura hierarkikoa dute (zuhaitza). Horrela Fitxategi Sistema osatzen dute. Fitxategi baten izen absolutuak (edo zuhaitzaren errotiko bidea) fitxategi sistemako fitxategi bakar bati egiten dio erreferentzia.
- Batzuetan, fitxategi bat direktorio bat baino gehiagotik elkarbana daiteke (loturen bidez).
- Fitxategien gainean eragiketa anitz egin daiteke (sortu, irakurri, idatzi, ezabatu, aldatu, ezaugarriak aldatu ...).
- Erabiltzaile anitzeko sistemetan atzipen baimenen bidez babesten dira, erabiltzaile bakoitzak atzipena duen edo ez adieraziz, eta atzipena izanez gero, egin ditzakeen eragiketak zehaztuz.
- Fitxategi baten informaziorako atzipen unitatea erregistro logikoa da. Sistema eragiletan normalena, sistema deietarako atzipen unitate gisa karakterea izatea da. Erregistro logikoak gehienetan sekuentzialki atzitzen dira, baina era zuzenean atzitzeko aukera ere izan ohi da.

Gure konputagailuan gordetako fitxategiei erreparatuz gero, eduki, tamaina eta erabilpen oso anitzak dituzten fitxategiak aurkituko ditugu. Adibidez, sistema eragileak eta aplikazioek konfiguraziorako byte gutxi batzuetako testu fitxategi asko erabiltzen dituzte. Bestalde, multimedia fitxategiak oso handiak izan ohi dira. Azken hauek normalean aldaketa gutxi jasaten dute, aplikazioetako kode fitxategiek bezala, halere, badira aplikazioak (testu editoreak edo *gcc*) biziraupen laburreko aldi baterako fitxategiak erabiltzen dituztenak.

Fitxategi sistema batek ezaugarri zehatz batzuk dituen euskarri-gailu baten beharra du. Funtsean, gailuak biltegitratze iraunkorra (ez-hegazkor) eskaini behar du. Gainera, atzipen kopuru mugaezina baimendu behar du, irakurketan zein idazketan. Halere, badaude sistemak non irakurri bakarrik egin daitekeen, eta sistemak non nahikoa den, baita hobe ere, behin baino ez idaztea (adibidez back-up-etan).

¹ Bai, izen absolutuak fitxategi bati soilik egiten diotako erreferentzia.

² Funtsean biltegitratze iraunkorra eskaini behar du.

2 Konputagailu pertsonalen sistema eragileak

1970ko hamarkadaren amaieran hardwarea merkatu zen eta mikroprozesadoreak agertu ziren, ondorioz posible zen erabiltzaile bakar bati sistema dedikatu bat eskaintzea, ezaugarri hau funtsezkoa da sistema pertsonal batean. Hasiera batean, konputagailu pertsonalen sistema eragileak, adibidez MS-DOS (1981), **erabiltzaile bakarrekoak** ziren, (ez zuten babes-mekanismorik) eta **ataza bakarrekoak** ziren; hau da, ez ziren 30 urte lehenagoko jatorrizko monitorean oinarritutako sistemen oso desberdinak, desberdintasun nagusia terminal baten bidez zuzenean jardun daitekeela da. Baina 1990ko hamarkadatik aurrera, konputagailu pertsonal merkeetan erabilgarri dagoen hardwareak multi-ataza sistemak onartzen dituzte (Mac OS, Windows, LINUX) eta **erabiltzailearen interfaze grafiko** aurrerratuak ere onartzen dituzte. Gainera, gaur egungo konputagailu pertsonalak erabiltzaile anitzekoak dira. Konputagailu pertsonalen ezaugarrietako bat erabiltzaileak sistemaren administratzailea izan behar duela da, ondorioz beharrezkoa da administrazio-atazak ahalik eta gehien erraztea.

Informatika hedatzeaz gain, konputagailu pertsonalen sorrerak eragin gehigarri bat du. Ordura arte, enpresa handien multiprogramatutako sistema zentralak urrunetik, terminal "tontoetatik", atzi zitezkeen telefono-modemen bidez. Konputagailu pertsonalek terminal hauek ordezkatu dituzte eta orain sistema autonomoak dira, hauek konputazio-ataza batzuk burutu ditzaketen sistema autonomoak dira eta sistema zentralera deskarga dezakete. Bereziki edozein komunikazio-protokolo exekuta dezakete era autonomoan. Protokolo estandarra adostu ondoren (adibidez TCP/IP) konputagailu pertsonalak beraien artean ere zuzenean komunika daitezke. Konputagailu zentralaren kontzeptua desagertzen doa; gaur egun beraien artean sare-azpiegitura baten bidez komunikatzen den konputagailu-multzo bat ematen da. Baliabide zehatz batera sarbidea eskaintzen eta kudeatzen duen sareko makina izango da baliabide horren **zerbitzaria**. **Bezeroek** baliabidea atzitzen dute **bezero-zerbitzari eskema** baten bidez. Konputagailu-sareen agerpen eta hedapenak asko korapilatu du ez bakarrik sistema eragilea, baita ere bertan inplementatzen diren zerbitzuak, (*middleware* deritzona). Ondorioz Interneteko esparruan hedatzen diren **sistema banatuak** sortu dira eta zerbitzuen eskema eta kontzeptu oso elaboratuak sortu dituzte, adibidez *web-zerbitzuak* eta *cloud computing* zerbitzuak. Ikastaro honetan sistema zentralizatuak aztertzea mugatuko gara, baina ez dugu ahaztu behar gaur egungo egoera askoz ere konplexuagoa dela.

LINUX (1991) sistema eragilea UNIX sistema-familiako kidea da eta konputagailu pertsonaletan erabiltzera zuzenduta dago, baino beste ingurunetan ere erabil daiteke. LINUX nabarmena da **software libreko**¹ produktu bat delako, fenomeno honek softwarearen merkatua irauli du, bereziki sistema eragileena. Software librearen ezaugarri nagusia da ez duela kopia eta banaketa mugarik, **jabedun softwareak** ez bezala. Internet beharrezko baliabide bat izan da ideiak eta kodea komunitate batean partekatu eta trukatzeko software librea garatzen duten programatzaileen artean. Ondorioz eta frogatu den bezala, softwarearen egokitzea eta produktu berrien garapena errazten da, gainera erroreak zuzentzen dira eta bertsioak erraztasunez doitzen dira. GNU² bezalako erakundeek software librearen kopia, eraldaketa eta birbanaketa lizentziak ematen dituzte baldintza bakarrarekin, distribuzio berriak iturburu-kodea barneratu behar du³. LINUX da gaur egun eta askogatik UNIX familiako gehien erabiltzen den sistema.

¹Free software. Ez nahastu *freeware*-arekin. Ez da ezta ere doako software bezala ulertu behar.

²<http://www.gnu.org>

³Lizentzi honi *Copyleft* deitzen zaio.

Hasieran

↳ Erabiltzaile talerra eta ataza berrak

90ko hamarkadan

↳ Multi-ataza eta interfaze grafikoak

Gaur egun → Erabiltzaile auitz

↳ Administrazio-ataza → erraztu.

Komputagailu - pertsonalek → terminalek ordeztu

↳ Sistema autonomoak

Ez danda zerbitzari batetara lotuta

zerbitzari.

↳ Zerbitzari eta kudeaketa eskumen baliabide

Zehatz bidera

↳ Bezeroak → Bezero zerbitzari eskumenak
bidera atartu.

91 → linux sortu → Software libre

GNU

↳ Bereziki erabiltzaile

* Kopie eta berraketa
inguratu

↳ baldintza

↳ Distribuzio berrak iturburu

Kode berraketa


```

/*-----*/
temporizadoreak.c
/*-----*/

#include <nds.h>
#include <stdio.h>
#include "definizioak.h"

void DenBieBaimendu10()
{
    //Denboragailu baten etenak baimendu (Timer0)
    //Horretarako lehenengo eten guztiak galarazi eta bukaeran berriro baimendu

    void DenBieGalarazi10()
    {
        //Denboragailu baten etenak galarazi (Timer0)
        //Horretarako lehenengo eten guztiak galarazi eta bukaeran berriro baimendu

        void konfiguratTemporizadore10()
        {
            //Temporizadorearen konfigurazioa bere SI/ erregistroak aldatuz

            void DenBieEn10()
            {
                //Denboragailuaren (Timer0) etenaren tratamendurako zerbitzu errutina
            }
        }
    }
}

/*-----*/
etenkudeaketa.c
/*-----*/

// a adir los includes que sean necesarios
#include <nds.h>
#include <stdio.h>
#include "teklaketa01.h"
#include "temporizadore01.h"
#include "definizioak.h"

/* Prozedura honek eten taula hasieratu egiten du. Horrela, eten kudeatzailerak jakin ahal izango du
eten eskiera bat inisten zaitonean zein den exekutatu behar duen zerbitzu errutina.*/

void etenak010()
{
    //...
    irqSet(Irq_TIMER0, DenBieEn10);
}

```

```

/*-----*/
definizioak.h
/*-----*/

#include <nds.h>
#include <stdio.h>

//Etenen kudeaketarako erregistroak

#define IME          (*(vuin32*)0x04000208) //Interrupt Master Enable -- Eten guztiak
baimendu-galarazi
#define IE           (*(vuin32*)0x04000210) //Interrupt Enable -- Etenak banaka baimendu-
galarazi
#define IF           (*(vuin32*)0x04000214) //Interrupt Flag -- Eten eskieren erregistroa

//Etenen tratamendurako funtzioak
#define EtenakBaimendu0 REG_IME=1 //Eten guztiak baimendu
#define EtenakGalarazi0 REG_IME=0 //Eten guztiak galarazi

//teklatuaren erregistroak
#define TEKLAK_DAT  (*(vu16*)0x4000130) //teklatuaren datu erregistroa
#define TEKLAK_KNT  (*(vu16*)0x4000132) //teklatuaren kontrol erregistroa

//denboragailuaren (Timer0) erregistroak
#define DENB0_KNT  (*(vuin16*)0x04000102) //kontrol erregistroa
#define DENB0_DAT  (*(vuin16*)0x04000100) //datu-erregistroa

// Teklei izena eman diegu zenbakikiak nahasgarriak izan daitezkeelako

#define A          0
#define B          1
#define SELECT     2
#define START      3
#define ESKUBI     4
#define EZKER      5
#define GORA       6
#define BEHERA     7
#define R          8
#define L          9

// Teklekin egindako gauza bera egin daiteke egoerekin
#define HASIERA    0
//...

// aldagai orokorrak
extern int EGOERA;

```

```

/*-----
teklatua.c
-----*/

#include <nds.h>
#include <stdio.h>
#include "definizioak.h"
#include "grafikoak.h"
#include "spriteak01.h" //Teklatuarekin egiten den zer baiten ondorioz spriteekin lan egin behar
izanez gero.

int tekla01; //sakatuako tekla gordetzeko aldagai

int TeklaDetektatu01()
{
    //TRUE itzultzen du tekla bat sakatu dela detektatzen badu
}

int SakatutakoTekla01()
{
    //Sakatutako tekla balioa itzultzen du: A=0;B=1;Select=2;Start=3;Esk=4;Ezk=5;
    //Gora=6;Behera=7;R=8;L=9;
}

int IrakurriTeklatuInketa01()
{
    //inketa tekla bat sakatu dela detektatu arte (gora, behera, ezkerra, eskuinera)
    //tekla balioa itzuli
}

void TekEtenBaimendu01()
{
    //Teklatuaren etenak baimendu
    //Lan hau burutzeko lehenengo eten guztiak galarazi behar dira eta bukaera baimendu
}

void TekEtenGalarazi01()
{
    //Teklatuaren etenak galarazi
    //Lan hau burutzeko lehenengo eten guztiak galarazi behar dira eta bukaera baimendu
}

void konfiguratuTeklatua01()
{
    //Teklatuaren konfigurazioa bere S/I erregistroak aldatuz
}

void TekEten01()
{
    //Teklatuaren zerbitzu errutina, teklatuaren etenei arreta egiteko
}

```

```

/*-----
jokoa.c - Programa Nagusia
-----*/

/*
Kode hau garatu da dovotoren "Simple sprite demo" adibidean eta Jaeden Ameronen beste
adibide batean oinarrituta.
-----*/

#include <nds.h>
#include <stdio.h>
#include <stdlib.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDL_ttf.h>
#include <unistd.h>

//Geuk garatutako fitxategiak

#include "grafikoak.h"
#include "fondoak01.h"
#include "spriteak01.h"
#include "definizioak.h"
#include "jokoa01.h"

touchPosition pos_pantaila;

void jokoa01()
{
    erakutsiAtea01(); //fondoak.c fitxategian definituta

    initSpriteMem01();
    SpriteakMemorianGorde01();
    PaletaNagusiaEzarri01();

    while(1)
    {
        touchRead(&pos_pantaila);
        while(pos_pantaila.px==0 && pos_pantaila.py==0)
            touchRead(&pos_pantaila);
        PaletaNagusiaEzarri01();
        ErakutsiErronboak01(0, 100, 100);
        erakutsiAteaIrekitu01(); //fondoak.c fitxategian definituta
    }
}

```


- SISTEMA ERAGILE KLASIKOAK -

- 50 hamarkadan Batch sistematik (lehenengo SEak)
- SE : programaren exekuzioa kontrolatzea.
- Ashton lan gabe. SI gailuak motelagoak
- Eraginera gisa: multiprogramazioa (60h)
- terminalak, komunikatzea. Olan patuta jeregn bat eptan jeregn bitartean.
- 60h. → denb. partekatuta ko sist. (UNIX)
- berindatzi → eraman garia (73)
- Garatida → erabilgarri antzekoak

90. Hamarkadan, Multi atzeko sortu eta interakzio grafikoko hobeki

Gaur egun, erabilgarri antzeko eta administrazio atzeko erazte, konputagailu partekatu sortu, sisteme autonomoak.

Zerbitzario, Sarbide eta Audeaketa eskaintzen du baliabide zehatz batere. Ondoren, bezeroak Bezero zerbitzari eskaintzen bidez atzitu.

91. linux sortu, unix-en oinarrituta, eta software librea, ondoren ez du kopie eta bantlete mugerik.

GNU lizentziak antzeko erazte dira, beldintza bilerak, distribuzio berririk iturburu kodea berruntzeki izaten da.

- urteak pasa ahala gero eta txikiago eta eramangarriago eta eraginkorrago.
- Konputagailu pertsonaletarako sistema eragileak murriztu \rightarrow telefono adimenduetarako eta tabletetarako (Mac, windows)
- Bestela, sistema eragile espezifikoak (Android).
- Sarrera / Irteera gailu berriak (kamera, sentzoreak...), interakzio berriak (utimen-pantaila).
- Gailu hauek sareaz konektatu daitezke gure portatuetarako gabe, norahiko sistema.
- Denbora errealeko sistemak (autoaren orekaren sistema kontrola), epemuga bat bete behar.
- Erabilera orokorreko S/I, baina denbora errealeko atazekin bateragarritasun ~~eta~~ ^{bateragarritasun} baina epemuga garrantzi dezaketenean (multimedia aplikazioak)
- Sistema eragileen ibilbidea ziklikoa \rightarrow software eta hardwarearen diferentzia nabarmena eta ez ziren guztia bateragarriak. Orain mugikorrak berdina gertatzen da.

Aitzol Elu
 Oihan Arroyo
 Ander Mancosidor

Batch sistemuak txartel gutxi

Ez eragiltzen S/I-ren zain

Monoprogramatze.

Soluzioa

• Multitasking / multiprogramatze → Effektivitate

- S/Iren zain laster atzera gertatu

- Prog. exekuzio tartelak

- Memoriaren partekatzea

- Quantitate (denbora-tarte txikiak)

↳ Erantzun-denbora txikiak izan daitezke

↓
multiprogramazio sententzia eratu.

Hori, denbora partekatutako sistemei esaten zaio.

(Unix-en jabeak)

↓
erabiltzaile anitzeko sistemuak

- Segurtasunaren txikiak

- Interfaze grafikoa

- Sektoreak, Linux

- administrazio-eremuak

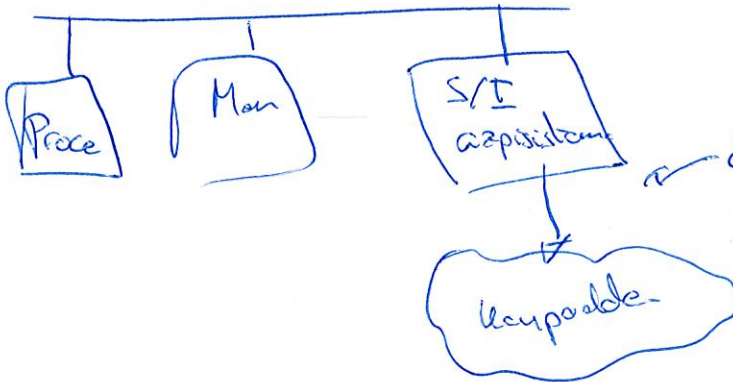
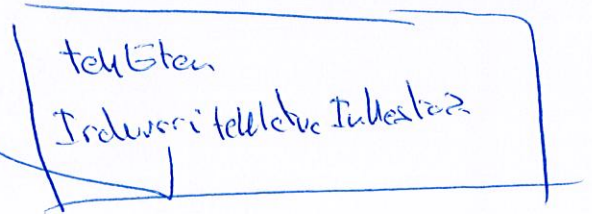
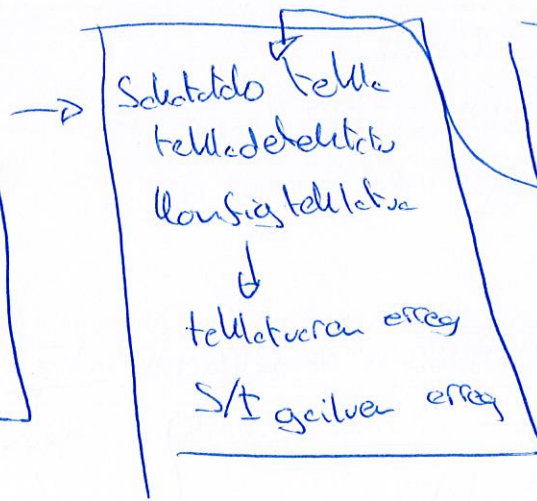
Sareak

↓
TCP/IP

Bezero-zerbitari eskuak
Sistemei buruz

↓
Gailu mugikorretara

↓
Sistemei eragile espezifikoak



Erregistred konfigerliche konposide
 konposide

