Fitxategi Sistemak

Fitxategia zenbat karakteristika dituen informazio multzoa da:

Ezaugarriak

Informazioa modu iraunkorrean gordetzen da. Izen baten bidez identifikatzen dira(karaktere katea esaten zaio), non luzapen bat izan dezaken, luzapen honek ze motatako fitxategia den adierazten du. Fitxategiak zuhaitz motako direktorioetan daude banatuta. Izen absolutua (errotik fitxategira heldu arteko direktorio segida da, non fitxategi bakar bati egingo dion erreferentzia). Hainbat direktoriotatik konpartitu daitezke(loturen bidez).

Fitxategien gainean eragiketa desberdinak egin daitezke(irakurri, idatzi, sortu, ezabatu).

Erabiltzaile anitzeko sistemetan, fitxategiak baimenen bidez daude babestuta. Atzipen unitatea, erregistro logikoa da. Modu sekuentzialean edo zuzenean heldu daiteke erregistro honetara. Karakterea da sistemarako deien atzipen unitatea.

Sistema eragileak eta aplikazioek testu fitxategi asko erabiltzen dituzte, konfiguraziorako, eta multimedia fitxategiak beste erabilpen batzuetarako, non byte askotakoak izan daitezke. Aplikazioetako fitxategien bezala ez dira askotan aldatzen.

Fitxategi sistemen karakteristika nagusia modu eraginkorrean gordetzeko aukera da. Gainera, atzipen mugagabea eskaini behar dute, bai irakurketa eta baita idazketarako. Baina, badaude bakarrik behin idazten duten sistemak ere(adb: back-ups).

Fitxategien gaineko eragiketak

Fitxategi bat mota abstraktuko datua da. Sistema eragileak sistema deiak eskaintzen ditu (sortu, idatzi, irakurri, ezabatu).

- **Fitxategiak sortu:** 2 pausutan banatzen da. Lekua bilatzea fitxategi sisteman. Eta direktorioan fitxategi berria ezartzea.
- **Fitxategien idazketa:** Sistema deiak egiten dira fitxategiaren izenarekin eta bertan idatzi nahi den informazioarekin.
- **Fitxategien irakurketa:** Sistema deiak egiten dira fitxategiaren izenarekin eta hurrengo blokea ezarri nahi den memoriako tokiarekin. Sistemak bilatzen du direktorioan fitxategia dagoen tokia. Behin blokea irakurrita punteroa eguneratzen da.
- **Fitxategien birkokapena:** Direktorioan zehaztutako sarrera bilatzen da. Momentuko posizioa aldatu egiten da fitxategiaren hasierara apuntatzen jarriz.
- **Fitxategia ezabatzea:** Fitxategiaren izena bilatzen da direktorioan. Eta honen espazioa libre uzten da eta sarrera ixten da.

Normalean sistema eragileek irekitako fitxategien zerrenda gordetzen duen taula izaten dute. Honek sistema deia egiten du, fitxategia erabili ahal izan aurretik eta taulan txertatuz sarrera. Puntero bat bueltatuko du, non S/I eragiketa guztietan erabiliko den.

Fitxategi sistemen euskarria

Diskoak izan dira informazioaren biltegiratze iraunkorrerako dispositibo ohikoenak. Magnetikoak zein optikoak izan daitezke.

Magnetikoak formatu ezberdinak izan dezakete baina gehien erabiltzen direnak 2,5 eta 3,5 hazbetekoak dira, barrukoak zein kanpokoak.

Disko bat *pistetan* dago antolatuta eta pista bakoitza *sektoretan.* Banaketa honen ondorioz atzerapena sortu izan da diskoetan, eta hauen errendimendua hobetzeko atzerapen horiek minimizatu behar dira. Errendimendua hobetzeko asmoz diskoek *cache* lanak egiten dituzte eta aurretik irakurritakoak *bufferretan* gorde.

Teknologia honek ia mende bat dauka, baina gaur egungo konputagailuen fitxategi sistemaren euskarri izaten jarraitzen dute. Teknologia konplexua da, eta fidagarritasun errendimendu handiak lortu dira kostu txikiarekin(0,05€/Gigabyte inguru).

Mota horretako diskoen alternatiba *flash* memoriak dira. Disko hauek *magnetikoak* alboratzen hari dira. Azkarragoak eta eramangarriagoak baitira. Baina naiz eta, teknologia aldaketa bat gauzatu, fitxategi sistemetan ez da nabari aldaketa hori, sistema eragileek hartutako dispositiboarekiko independenteak diren eskemei esker.

Biltegiratze dispositiboak sistema jakin baten erabilgarria izan dadin, *formatua* eman behar zaio. Fitxategi sistemari dagokion informazioa gordetzean datza. Formatua, fitxategi sistema bakoitzaren ezaugarri propioa da eta dispositiboan informazioa nola gordeko den zehazten du.

Galdera osagarriak

1.- Zuhaitz egitura duen fitxategi-sistema batean fitxategiaren izen absolutuak edo bideak modu unibokoan adierazten al du fitxategia?

Bai, fitxategiaren izen absolutua ezin da errepikatu sistema osoan, erreferentzia bakarra baita fitxategi bakoitzarentzat.

3.- Sistema eragile batek open() eta close() sistema-deiak baztertu daitezke?

Ez, open() sistema-deia exekutatzean fitxategia irekitako fitxategien taulan sartzen da hortik berarekin eragiketak azkarrago egiteko ahalmena izateko eta close() sistema-deia taula horretatik kentzen du, hau ez bagenu egingo taula bete egingo zen fitxategi asko kargatzen baditugu.

4.- Zein da fitxategi-sistema baten funtsezko ezaugarria?

Fitxategi-sistemaren funtsezko ezaugarria biltegiratze iraunkorra da, fitxategi horretan dauden datuak galdu ez daitezen.

5.- Zertan datza biltegiratze gailu bati formatua ematea?

Formatua eman edo formateatzea fitxategi sistemari dagokion informazioa gordetzean datza. Fitxategi sistema bakoitzaren ezaugarri propioa da eta dispositiboan informazioa nola gordeko den zehazten du. Sistema eragile bakoitzak bere formatu propioa du (adibidez, *ext2* edo *ext3* Linux-en eta *ntfs* Windows-en). Hala ere, badaude edozein sistema eragileak ulertzen dituen formatu batzuk, horietatik *fat32* da ohikoena, flash memorientzat.

6.- Gaur egun non erabiltzen dira SSD diskoak?

Eramangarritasuna beharrezkoa den euskarrietan, adibidez: sakeleko telefonoak, tabletak eta arinagoak izan behar dien gailuetan.

Direktorioak Kudeatzen

Aurreko atazetan fitxategiak kudeatzeko utilitate edo komandoekin lan egin dugu. Direktorioak ere fitxategiak dira, baina mota berezi batekoak. Nola inplementatuko dugu ls bezalako komando bat? edo nola sor dezakegu direktorio bat? Utilitate mota hau inplementatu ahal izateko, beharrezkoa da ulertzea nola den barnetik direktorio bat eta nola lor daitezkeen direktorio bateko fitxategien ezaugarriak. Ondorioz, direktorioak kudeatzeko funtzioak eta fitxategien ezaugarriak lortzeko funtzioak aztertuko ditugu, batez ere Linux-en sistema-deien interfazea eta C-ko liburutegi estandarra eskaintzen dituztenak aztertuko ditugu.

ls komandoa sortzen

nire_ls utilitatearen lehen bertsio bat sortuko dugu, honek direktorio baten edukia erakutsi beharko digu. Programa honek ezaguna dugun ls komandoaren antzeko konportamendua edukiko du. Berau exekutatzeko terminalean "**ls**" jartzea besterik ez dugu behar, baina baditu bere aukerak ere:

Programa garatzeko erabilitako c-liburutegia:

- ***opendir(3):***

  Erabiltzeko <sys/types.h> eta <dirent.h> liburutegiak erabili behar ditugu.

  opendir(const char ***name**)

  Funtzio honek emandako direktorioa irekitzen du, direktorioari begira dagoen erakuslea bueltatuz.

- ***readdir(3):***

  <dirent.h> liburutegia erabili behar da, funtzio hau erabili nahi bada.

  readdir(DIR ***dirp**)

  Dirent egitura bati begira dagoen erakuslea bueltatzen du, betiere “dirp” direktorioaren hurrengo sarrera izanik.

Zer egitura erabiltzen dira?

Horrela definituko da dirent egitura:

```
struct dirent {
       ino_t          d_ino;       /* inode number */
       off_t          d_off;       /* hurrengo dirent egiturarainoko distantzia*/
       unsigned short   d_reclen;    /* egituraren luzera */
       unsigned char    d_type;      /* fitxategi-mota */
       char           d_name[256]; /* fitxategiaren izena */
};
```

Horretaz gain beste funtzio batzuk gehitzeko nire_ls komandoari stat egitura erabili beharko da:

```
struct stat {
        dev_t     st_dev;     /* ID of device containing file */
        ino_t     st_ino;     /* inode number */
        mode_t    st_mode;    /* protection */
        nlink_t   st_nlink;   /* number of hard links */
        uid_t     st_uid;     /* user ID of owner */
        gid_t     st_gid;     /* group ID of owner */
        dev_t     st_rdev;    /* device ID (if special file) */
        off_t     st_size;    /* total size, in bytes */
        blksize_t st_blksize; /* blocksize for file system I/O */
        blkcnt_t  st_blocks;  /* number of 512B blocks allocated */
        time_t    st_atime;   /* time of last access */
        time_t    st_mtime;   /* time of last modification */
        time_t    st_ctime;   /* time of last status change */
};
```

Beste hainbat liburutegi funtzio erabiltzen dira nire_ls programa egiten:

- ***closedir(3):***

  opendir(3)-ren oso antzekoa da, baina kasu honetan pasatako direktorioa ireki beharrean itxi egingo da.

  closedir(const char *name)

- ***strcat(3):***

  Bigarren tokian pasatutako String-a, lehenengo tokian pasatako String-aren atzetik jarriko du eta String berrirako punteroa bueltatuko du.

  char * **strcat**(char *s1, const char *s2)

- ***strcmp(3):***
  Bi String konparatuko ditu eta 0 itzuli baldin eta berdinak badira.

  int **strcmp** (const char *s1, const char *s2)

- ***strcpy(3):***
  Bigarren String-a, lehenengo String-an kopiatuko du. Non lehenengo String horretarako punteroa itzuliko duen.

  char ***strcpy*** (char *s1, const char *s2)

- ***strlen(3):***
  String-aren karaktere kopurua itzuliko du.

  int ***strlen*** (const char *s)

Fitxategien babes-ezaugarriak

Fitxategien ezaugarriak ikusteko i_node-ra jo behar da, non fitxero bat modu bakarrean definitzen duen informazioa bertan aurkitzen baita.

Fitxategien propietateak:

Tamaina
Lotura Kopurura
Data
Mota
Tamaina (bytetan)
Baimenak

Hainbat fitxategi mota ditugu:

d direktorioak
'-' fitxategi normalak
l loturak

Gailuak(2 motakoak):

Karaktereei atzipena(c): adb: terminala
Blokeetarako atzipena(b): almazenatzeko gailuak.

/dev: gordetzen dira sistema guztiko gailuen izenak.

Sistema eragile multierabiltzailea: Sistema eragilea non erabiltzaile desberdinak erazagutzen dituen, eta babes mekanismoak sartzen ditu erabiltzaile bakoitzaren informazioari.

Zerk egiten du fitxategi bat exekutagarri?

Sistema eragile guztietan daude exekutatu daitezkeen fitxategiak eta exekutatu ezin direnak. Linux-en ordea luzapena ez da ezaugarri bat, izena besterik ez da. Fitxategi bat exekutagarri bihurtzeko bere erabiltzaile baimenak aldatu behar dira exekutagarri baimena aktibatuz.

ls-l programaren bidez ikusi daiteke ze motako baimenak dituzten erabiltzaileek fitxategia atzitzeko.

ADB: guraso.sh fitxategi exekutatzen saiatuko gara, ./guraso.sh eginez. Baina baimenik ez dugula esaten digu. Hau gertatzen da ez delako exekutagarria baizik eta testu fitxategi bat.
Exekutagarri baten baimenak emateko ondorengo komandoa erabiliko dugu.

chmod +x guraso.sh

+x erabilita gehitzen zaio exekutatzeko baimena, eta baimen hori kendu nahi bada, -x jarri beharko zaio komandoari.

- ***chmod(1):***
  Fitxategi baten baimenak aldatzeko komandoa da, erabiltzaileak zehazten du ezarri ero kendu nahi diren baimenak(modu sinbolikoan edo zenbaki oktal baten bitartez ema daiteke).

  chmod <Aukera><Modua><Fitxategia>
  chmod <Aukera><Modu oktaleko fitxategia>

Fitxategien baimenak:

Hiru oinarrizko baimen existitzen dira fitxategientzat: irakurketa, idazketa eta exekuzioa:

- Irakurketa-baimena(Read):
  Fitxategi barruko edukiak irakurri ditzake erabiltzaileak. r bidez adierazten da.

- Idazketa-baimena(write):
  Bertako edukian aldaketak egiteko baimena ematen du, idatzi, ezabatu, edo edukiak gehitu. w bidez adierazten da.

- Exekuzio-baimena(execute):
  Baimen honen bidez, konputagailuari adierazten zaio fitxategi hori programa bat bezala exekutatzeko. Programa edo script bat bada exekutatuko du. x karakterea erabiltzen da baimen hau ezarri edo kentzeko.

Direktorioen baimenak:

Aurretik fitxategien baimenekin gertatu den bezala 3 oinarrizko baimen daude: irakurketa, idazketa eta exekuzioa:

- Irakurketa:

  Direktorio barruan dauden fitxategien izenak, baimenak... ikusi daitezke, ls komandoa erabili daiteke. Baina horrek ez du esan nahi gero bertako fitxategien edukia ikus daitekeenik, gerta daiteke erabiltzaileak horretarako beharrezko baimenik ez izatea.

- Idazketa:

  Fitxategiak gehitu, ezabatu edo mugitu daitezke.

- Exekuzioa:

  Bere barruko fitxategien arteko eragiketak egiteko gaitasuna ematen du.

Baimenen banaketa

Ikusi daitekeen bezala, 9 espazio daude, bertan letrak jartzeko(baimen bat adieraziko du letra bakoitzak). 9 espazio horietatik, erabiltzaile mota bakoitzarentzako 3 daude, Administratzailea, Taldea eta Bestelakoak:

Baimenak modu oktalean adierazita:

rwx	7
rw-	6
r-x	5
r--	4
-wx	3
-w-	2
--x	1
---	0

Nola oparitu fitxategi bat lagun bati

Horretarako oinarrizko direktorioari exekutatzeko baimena eman behar zaio, hori ondorengo komandoaren bidez egiten da:

chmod g+x <Oinarrizko direktorioa>

Horrela beste erabiltzaileek direktorio hori exekutatzeko aukera izango dute, eta irakurtzeko baimena daukaten direktorio eta fitxategiak ikusiko dituzte soilik. Baimen horretaz aparte beste hainbat baimen eman diezaiokegu, eta hauen bidez gauza bat edo bestea egiteko ahalmena izango dute.

Hala ere kontuan izan behar da, partekatzeko modu honekin baimen egokiak dituzten erabiltzaileek edozein gauza egin ditzaketela. Ondorioz arriskutsua izan daiteke edozein baimen ematea.

Nola aldatu daiteke /etc/passwd baimenik izan gabe?

ls –l /etc/passwd egiten bada, ondorengoa agertuko da terminalean:

ls-l /etc/passwd
-rw-r—r—1 root

Ondorioz, bakarrik root-ek izango ditu idazketa baimenak. Horrela izanda nola da posible passwd aldatzea?
Programa bat exekutatzen ari denean 2 identifikatzaile posible ditu:

Erabiltzaile erreala (sistemarena)
Erabiltzaile eraginkorra

Sistema gehienetan bi erabiltzaile hauek berdinak izaten dira. Erabiltzaile erreala ezin da aldatu baina eraginkorra bai.

Erabiltzaile eraginkorra eta erreala berdinak ez izateak dakarren ondorioa da, programa bat exekutatzen ari denean ez dela erabiltzaile erreala baizik eta eraginkorra. Erabiltzaile eraginkorra aldatzeko setuid erabiltzen da.

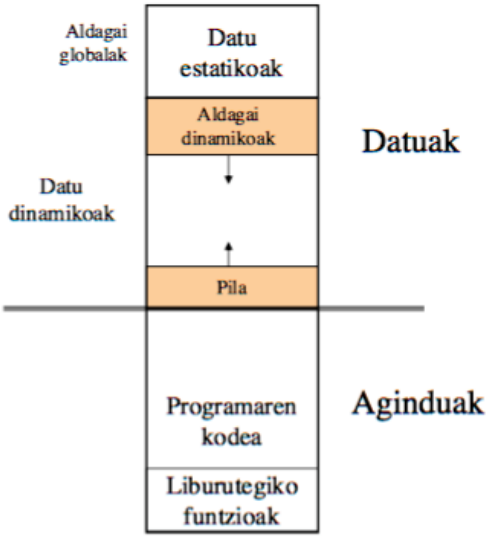
setuid: sistemarako deia da. Fitxategi baten erabiltzaile nagusiari, erabiltzaile eraginkor bat gehitzen zaio. Fitxategi exekutagarri batean, programa exekutatuko duen erabiltzaileak, erabiltzaile nagusia izango du erabiltzaile eraginkor bezala. Ondorioz, aldaketak egin ahal izango ditu.

setgid: direktorio baten gainean egiten dira eragiketa. Direktorio horren barruan sortzen den edozein fitxategik talde modura direktorio horretako taldea izango du. Honen bidez, fitxategi bat sortzen bada, taldea ez da izango erabiltzaile nagusiaren talde jabea. Eta direktorio horretarako talde espezifiko bat sortu beharko litzateke.

sticky: direktorioetako erabiltzen da soilik. Honek, direktorio barruko edozein fitxategi ezabatzeko baimena ematen dio, direktorioaren jabeari eta fitxategiaren jabeari.

Memoriaren kudeaketa

Memorian programa bat sartzen denean ondorengo egitura hartuko du. Egitura horretan datuak eta aginduak bereizten dira.



Gainera datuak estatikoak edo dinamikoak izan daitezke, hauen arteko desberdintasuna ondoren azalduko da.

Programa bat konpilatzerako orduan hiru modutara egin daiteke:

- ***Iturburu-kode bakarra:*** Iturburu-kode bakarra izan ezkero, konpilatzerako orduan konpiladoreak Objektu-kode bakarra sortuko du eta ondoren, honen exekutagarria eraiki.
- ***Hainbat iturburu-kodetik, objektu-kode bakarra:*** Hainbat iturburu-kode izanda, konpiladoreari esan dakiogu, Objektu-kode bakarra sortzeko, eta horrela kode honetatik exekutagarria eraikiko du.
- ***Hainbat iturburu-kode izanda, bakoitzetik objektu-kode bat sortzea:*** Kasu honetan aurrekoan bezala, hainbat iturburu-kode daude, baina oraingoan bakoitzetik objektu-kode bat sortuko da eta ondoren liburutegiekin estekatu guztiak exekutagarri bat eraikiz.

Konpilazioa gauzatu oztean fitxategi exekutagarri bat edukiko da. Linuxen exekutagarri horrek hainbat formatu ezberdin izan ditzake:

- ***ELF(Executable and Linking Format)*** gehien erabiltzen den formatua da eta Unix System Laboratories-ek garatutakoa da. Hainbat sistema eragiletan formatu estandarra da.
- ***a.out(Assembler OUTput format)*** Linuxeko bertsio zaharrek erabiltzen dute, gaur egun ez da asko erabiltzen.

MS-DOSek ere bere formatu propioa du, kasu honetan ordea bakarra da. ***EXE*** formatua.

Estekatze motak

Aurretik aipatu bezala bi eskatze mota daude: estekatze dinamikoa eta estatikoa:

- ***Estekatze estatikoa:*** Liburutegi estatikoak erabiltzen dira, kasu honetan, sortzen den fitxategi exekutagarriak, modulu guztiak barneratzen ditu. Ondorioz hainbat desabantaila ditu:
    - Liburutegiko funtzioen kodea errepikatu egiten da exekutagarri desberdinetan.
    - Fitxategi exekutagarri handiak sortzen dira.
    - Memorian ere liburutegiko funtzioen kopia anitz gordetzen dira.
    - Liburutegiak eguneratzen direnean, exekutagarri guztiak berriro linkatu behar dira.

Nahiz eta hainbat desabantaila izan, askotan konpilatu behar diren kode txikiko programetan oso erabilgarria da, konpilazioa eta estekatzea oso azkarra izaten baita.
Estekatze estatikoaren gabeziei aurre egiteko estekatze dinamikoa sortu zen.

- ***Estekatze dinamikoa:*** Estekatze dinamikoko liburutegiek gutxi okupatzen dute, bai memorian eta bai diskoan. Non estekatzaileak programa exekutagarrian errutinak zuzenean sartu beharrean sistema-dei berezi bat ipintzen du, honi, *estekatze-errutina* deritzo. Estekatze dinamikoaren hobekuntzak:
    - Funtzio erabilienen kodea ez da errepikatzen.
    - Estekatze dinamikoko liburutegiko funtzioen kodea ez dago fitxategi exekutagarrietan.
    - Fitxategi exekutagarri txikiagoak.
    - Programak aldatu/eguneratu daitezke birkonpilatu gabe.
    - Memoria fisikoa baino handiagoak diren programak exekutatu daitezke.

Baina estekatze dinamikoak ere zenbait eragozpen ditu:

- Dei mekanismo motelagoa(Iturburu-kodea handiak behin bakarrik exekutatzeko ez da komeni estekatze mota hau erabiltzea).
- Liburutegi kudeaketa konplexuagoa da.
- Aplikazioak instalatzerakoan, estekatze dinamikoko liburutegien menpekotasunak kontuan hartu behar dira.

Programak memorian kokatzeko moduak

Kasu askotan memorian programa bat baino gehiago sartu behar dira. Eta beti ez da lortzen lekua programa osorik edo jarraian sartzeko. Ondorioz, ezinezkoa da programa guztiak egoiliar mantentzea.

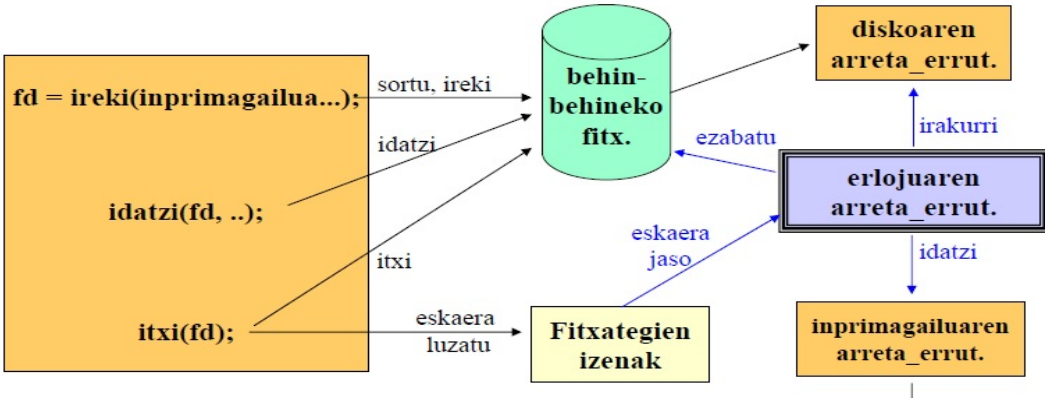
- ***Ez jarraian:*** Programa zatitan banatu behar da, memorian sartu ahal izateko. Zatikatze hori gauzatzeko bi teknika daude:
    - ***Segmentazioa:*** Tamaina desberdineko zatiak egiten dira, eta memoriako helbide desberdinetan sartu bakoitza. Zatiketa hau konpiladorearen bidez egiten da.
    - ***Orrikapena:*** Tamaina berdineko zatiak egiten dira. Eta hau egiteaz Sistema Eragilea arduratzen da.
- ***Ez osorik:*** Alegiazko memorian egiten da, orrikapena edo segmentazioarekin konbinatuz. Arazoak gertatu daitezke, atzipen-hutsegitea, itzulpena egiteko unean, alegiazko helbide bati dagokion helbide fisikoa ez existitzea da. Kasu honetan, falta den orria diskotik memoriara pasako da.

Linux-eko shell-a

Prozesadorearen geldiuneak aprobetxatzeko hainbat mekanismo desberdin daude:

- ***S/I asinkronoak:*** S/I-ren zain dagoen bitartean beste zerbait egiten da, horretarako:
    - Aginduak berrordenatzen dira.
    - Programak zatitzen dira.
    - Sinkronizazio esplizitua behar du, jakin beharra dago ea S/I noiz amaitzen de.
- ***Buffering:*** S/I paraleloan exekutatzen da programarekin:
    - S/I-ren geldotasunaren emaitzak hobetzeko, buffer-a jartzen da prozesadore eta S/I artean.
    - Buffer-ek memoria okupatzen dute.
- ***Spooling:*** Hardwarea azkartzen du:
    - Tartean hardware azkarragoa jartzen da.
    - Irakurketan: front-end.
    - Idazketetan: back-end edota Sistema Eragilea.

Spooling-aren bilakaera inprimagailu baten bidez azalduta:



Spooling-a erabiltzeak hainbat abantaila dauzka:

- Fitxategien kopiak egitea errazagoa da.
- Programa eta S/I paraleloan exekutatzen dira.

Baina hainbat desabantaila handi dauzka:

- Diskoan espazioa okupatzen du: behin behineko fitxategia dela eta.
- Tratamendu-denbora luzeagoa da.

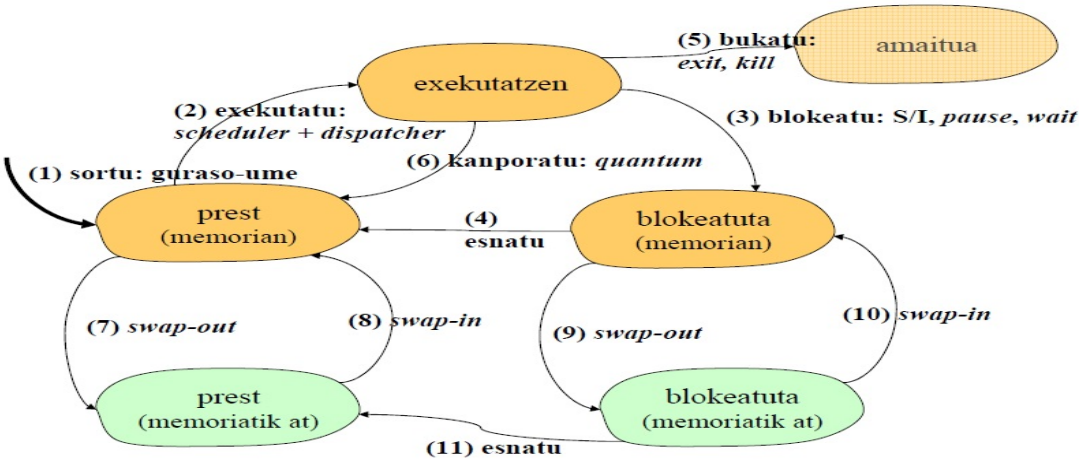
Desabantaila horien soluzioa hainbat programa batera exekutatzean datza. Honako prozesuaren 2 modu desberdin aurki daitezke:

- ***Multiprogramazioa:*** Hainbat programa exekutatzen dira prozesadore-denbora txandakatuz.
- ***Multifluxua:*** programa bakarra exekutatzen da, hainbat fluxurekin, eta hauek beraien artean prozesadore-denbora txandakatzen dute.

Prozesu hori gauzatzeko CPU-ak programa guztien informazioa edo **testuingurua** jakin behar du. Hori beharrezkoa da Multiprogramazioa gauzatzeko CPU-ak bere testuingurua aldatu behar duelako.

Prozesuen kontrola

CPU-ak prozesuak exekutatzen ditu, programen instantziak. Prozesu horiek bi datu gordeko dituzte: testuingurua eta programa bera.
Ondorengo eskeman ikusi daiteke prozesuen bilakaera nolakoa den:



OHARRA: Guraso-prozesua ume-prozesua baino lehenago bukatuko balitz, umezurtz geratuko litzateke eta init prozesuak adoptatuko luke.

Multiprogramazioaren eta sistema monoprogramatuen arteko desberdintasunak

Multiprogramazioaren kasuan prozesuen kudeaketa zailagoa da eta sinkronizazioa beharrezkoa bihurtzen da. Oso garrantzitsua da memoria asko izatea, bestela ezin direlako prozesu guztiak batera memorian eduki. Hori saihesteko programak ez dira osorik gordetzen eta estekatze dinamikoa erabiltzen da. Sistema deiak erabiltzen dira sinkronizazioa era prozesuen arteko komunikaziorako. Zenbait programa atzeko planoan exekutatzen dira. Askotan prozesuen artean informazioa trukatzen da, horretarako pipe-ak erabiltzen dira.

Prozesuen kontrolerako sistema-deiak

- ***getpid(2):*** prozesuaren identifikadorea bueltatzen du.

  int **getpid()**
- ***getppid(2):*** Prozesu gurasoaren identifikadorea bueltatzen du

  int **getppid()**
- ***getuid(2):*** Prozesuaren jabea den erabiltzailearen identifikazioa bueltatzen du:

  int **getuid()**

Ondoren prozesuen sorrerarako erabiltzen diren hainbat sistema dei ikusiko dira:

- ***fork(2):*** Prozesu-ume bat sortzen du. Non fork-aren hurrengo agindutik hasiko duen exekuzioa. Kontuan izan behar da, prozesu umeak dena heredatzen duela prozesu-gurasotik, baina beraien pid-a desberdina dela. fork-ek umeari 0 pid-a emango dio, eta gurasoari umearen pid-a.

  int **fork()**
- ***execlp(2):*** Argumentu bat edo gehiago eta fitxategi bat sarturik, argumentu horiek exekutatuko ditu. Bukaeran beti NULL balio jarri behar zaio-

  int **execlp(**char ***file**, char ***arg0**, ..., NULL**)**
- ***execvp(2):*** Beste programa bati deitzeko eta honi argumentuak sartzeko erabiltzen da.

  int **execvp(**char ***file**, char ***arg[])**

Hainbat sistema-dei oso erabilgarri daude prozesuen bukaerarako erabiltzen direnak:

- ***exit(2):*** Prozesua modu kontrolatuan bukatzeko erabiltzen da.

  exit(int **egoera)**

- ***wait(2):*** Deitzaileak itxarongo du bere prozesu umeren batek bukatu arte. Umerik ez badu, -1 bueltatuko du deitzailea blokeatu gabe. Gainera, bukatu duen umearen pid-a bueltatzen du.

  int **wait(**int ***egoera)**
- ***getuid(2):*** pid hori duen prozesua amaitzen da.

  int **kill(**int **pid**, int **SIGKILL)**