4. Gaia: Erabiltzaileen kudeaketa eta segurtasuna

1. Erabiltzaile anitzeko sistemak
2. Babeserako mekanismoak
3. Babesa eta segurtasunerako sistema-deiak

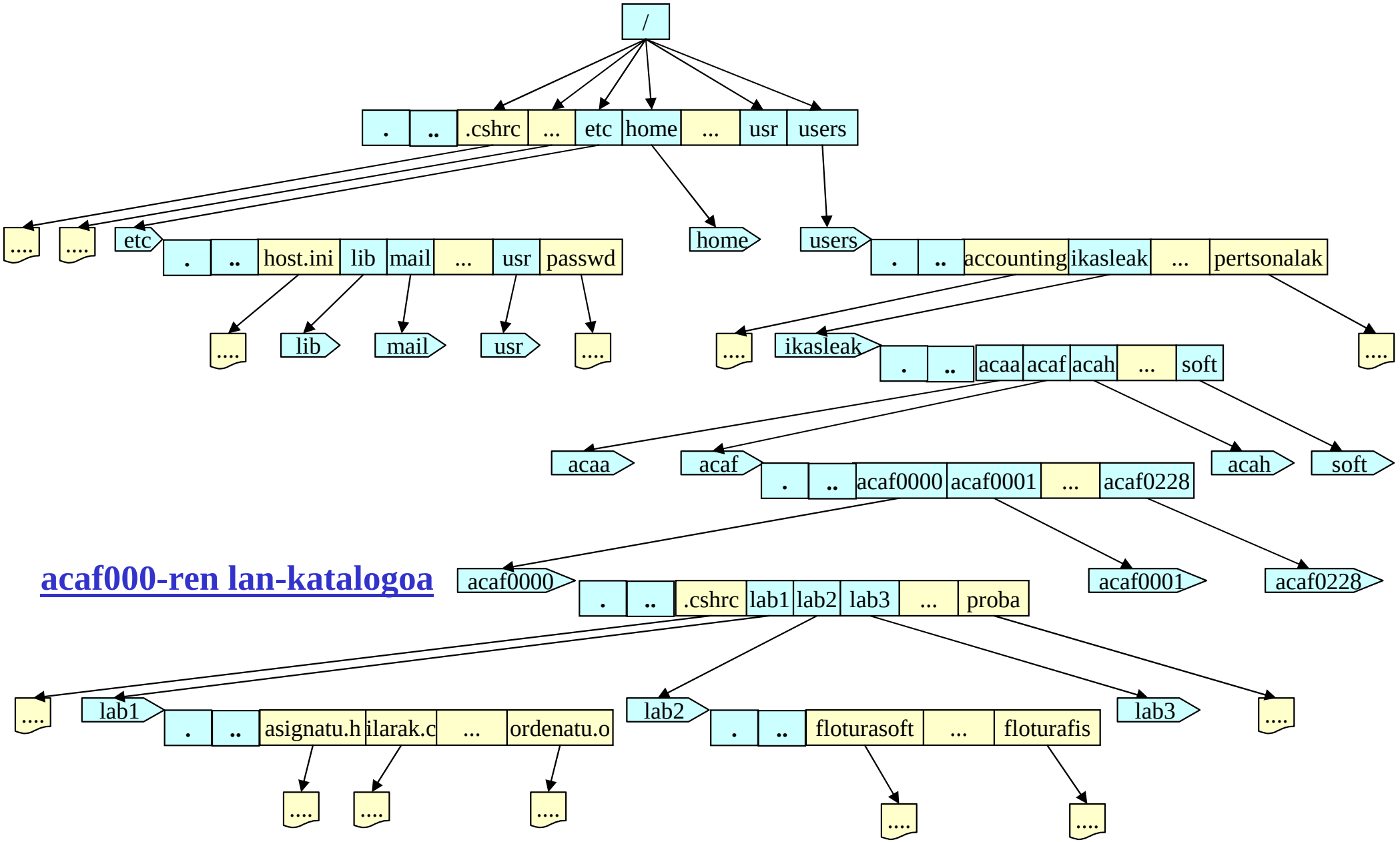
Gaitasunak:

- *Erabiltzaileen kudeaketa eta babes mekanismoen beharra azaltzea*
- *Sistema eragilearen komandoak eta programazio interfazea erabiltzea tresnak garatzeko*

Sarrera

- Erabiltzaile desberdinen arteko konputagailuaren erabilera
  - Aldiberekoa edota txandaka
- Sistema eragileak erabiltzaileak ezagutu behar ditu
  - Atzipen kontrolaren beharra (login / logout), erabiltzaile izena eta pasahitza bidez
- Sistema eragileak, erabiltzaileen informazioen atzipenean, segurtasun eta babeserako mekanismoak eskaini behar ditu
  - Fitxategi-sisteman gordetako informazioaren atzipen kontrolatua

Erabiltzaile anitzeko fitxategi-sistema



acaf000-ren lan-katalogoa

Erabiltzaile anitzeko sistemak

- *Accounting*: Kontabilitatea
- Informazioaren konfidentzialtasuna
- Informazioaren segurtasuna/fidagarritasuna
- Baimenen kudeaketa
- Atzipenen kontrola/konprobaketa
- Baliabideen erabilpenaren kontabilitatea
- Baliabideen atzipen murrizpenak (kuotak)
- Erabiltzaileen taldeak, jabea, …
- Erabiltzaile berezia(k): administraria (*root*)

Baimenak UNIXen:
- Irakurketa
- Idazketa
- Exekuzioa

jabea	taldea	besteak
RWX	RWX	RWX

i-node baten egitura (UNIX)

Fitxategi-mota: arrunta, katalogoa, karaktere-gailua, bloke-gailua, pipe, lotura, socket

RWX baimenak jabea, taldea eta besteentzako

Jabea eta taldearen identifikadoreak

Fitxategiaren tamaina bytetan

Hainbat data: azken atzipena, azken aldaketa…

Fitxategiaren lotura kopurua (izen desberdinak fitxategi-sisteman)

Beste informazio batzuk: gailu zenbakia, stiky-bit…

UNIX-eko baimenak: `ls -l`

Fitxategi mota:

-	**fitxategi arruntak**
d	**katalogoak (*directory*)**
l	**loturak (*link*)**
c	**karakteretako fitxategi bereziak (gailu motelak)**
b	**bloketako fitxategi bereziak (gailu azkarrak)**
p	**izena duten *pipe*-ak edo FIFOak (komunikaziorako)**

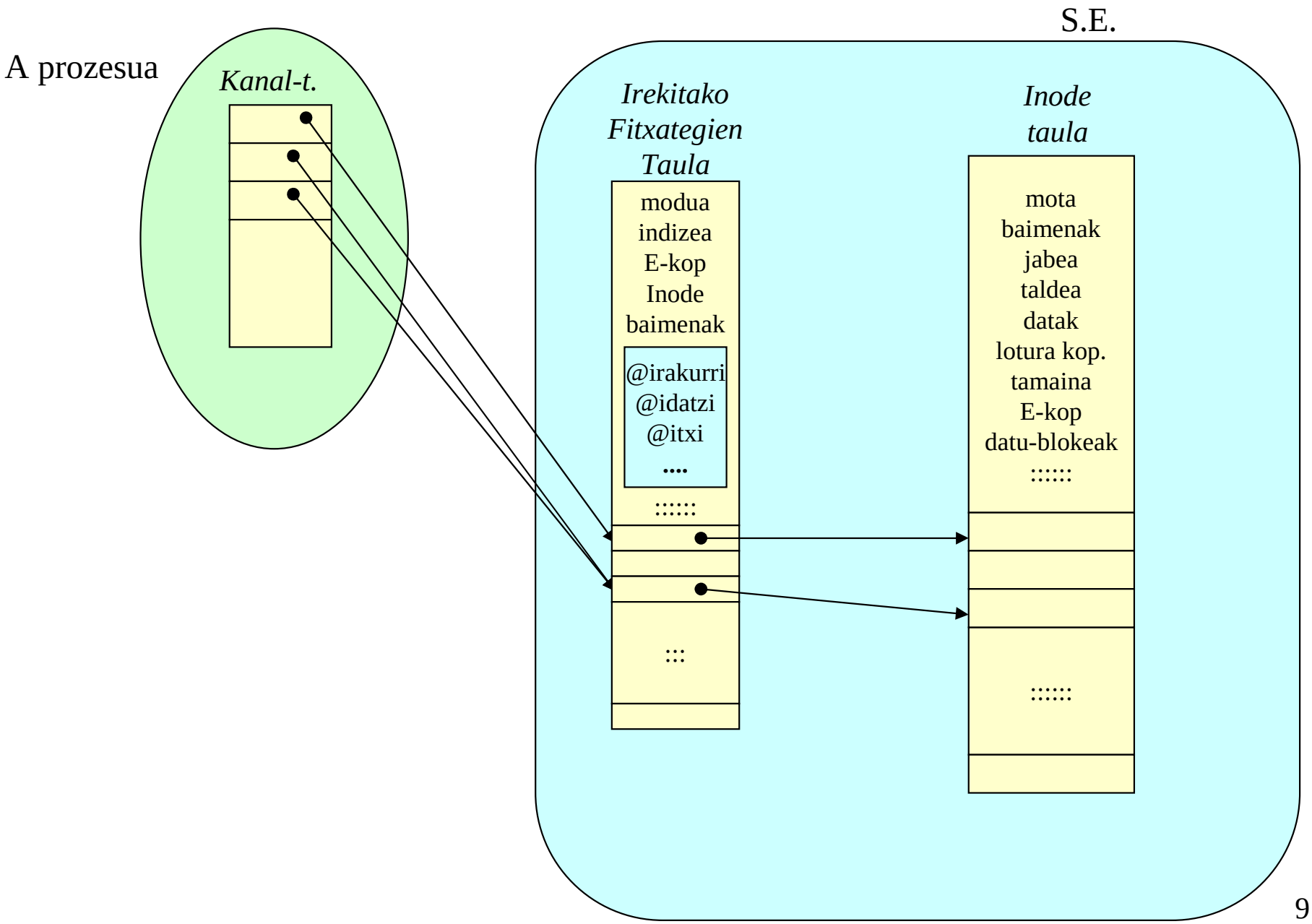
Unix-eko baimenak

Baimenak	Binary	Octal
`- - -`	`0 0 0`	`0`
`- - x`	`0 0 1`	`1`
`- w -`	`0 1 0`	`2`
`- w x`	`0 1 1`	`3`
`r - -`	`1 0 0`	`4`
`r – x`	`1 0 1`	`5`
`r w -`	`1 1 0`	`6`
`r w x`	`1 1 1`	`7`

chmod komandoa

- Aurreko zenbaki zortzitarren adierazpena erabiliz, `chmod` komandoaren bitartez fitxategien baimenak alda daitezke
- Sintaxia:
  `chmod BaimenenMaskara fitxategia`
- Adibidea:
  `chmod 755 public_html`
- Oro har, katalogo eta exekutagarriek “755” baimenak izan ohi dituzte, aldiz, beste fitxategiek “644” baimenak

Baimenen egiaztapena fitxategien atzipenean



Sistema-deiak

- **Irekiera / sortzea**
  - `int open (char *path, int flags, int baim);`
  - `int open (char *path, int flags);`
  - `int creat (char *path, int baim);`

Flags (aukerak):

O_RDONLY	**O_WRONLY**	**O_RDWR**	O_NDELAY
O_APPEND	O_DSYNC	O_RSYNC	O_SYNC
O_NOCTTY	**O_CREAT**	O_EXCL	O_TRUNC

- **Katalogoen kontrola**
  - `int mkdir (char *path, int baim);`

Sistema-deiak (2)

- **Fitxategien kontrola**
  - `int stat (char *path, struct stat *sbuf);`
  - `int fstat (int fd, struct stat *sbuf);`

stat datu-egitura

- **st_dev**: fitxategiaren euskarri den unitatearen zenbakia (short)
- **st_ino**: inode zenbakia (ushort)
- **st_mode**: modua (short) eta baimen bitak
- **st_nlink**: lotura kopurua (short)
- **st_uid**: jabearen identifikatzailea (ushort)
- **st_gid**: taldearen identifikatzailea (ushort)
- **st_rdev**: fitxategi berezientzako unitate zenbakia (short)
- **st_size**: tamaina (long) (0 fitxategi berezientzako)
- **st_atime**: azken atzipenaren ordua (long)
- **st_mtime**: azken aldaketaren ordua (long)
- **st_ctme**: sorrera ordua (long)
- …

Sistema-deiak (3)

- **Erabiltzaileanitza**
  - `int chmod (char *path, int modua);`
  - `int chown (char *path, int jabea, int taldea);`
  - `int access (char *path, int modua);`

    `modua: R_OK, W_OK, X_OK`
  - `int umask (int modua);`
  - `uid_t getuid(void);`
  - `gid_t getgid(void);`

Sistema-deiak (4)

- **Erabiltzaileanitza**

```
#include <pwd.h>
#include <sys/types.h>
```

- `struct passwd *getpwnam (const char *izena);`
- `struct passwd *getpwuid (uid_t uid);`

```
struct passwd {
    char    *pw_name;       /* erabiltzaile izena */
    char    *pw_passwd;     /* pasahitza zifratua */
    uid_t   pw_uid;         /* erabiltzailea */
    gid_t   pw_gid;         /* taldea */
    char    *pw_gecos;      /* izen osoa */
    char    *pw_dir;        /* hasierako katalogoa */
    char    *pw_shell;      /* lehenetsiko shella */
};
```

Sistema-deiak (5)

- **Erabiltzaileanitza**

```
#include <grp.h>
#include <sys/types.h>
```

- `struct group *getgrnam (const char *izena);`
- `struct group *getgrgid (gid_t gid);`

```
struct group {
    char    *gr_name;       /* taldearen izena */
    char    *gr_passwd;     /* pasahitza zifratua */
    gid_t   gr_gid;         /* taldea */
    char    **gr_mem;       /* taldekideak */
};
```

Adibidea: baimenak.c

```
/* ============================================================================
 Name        : baimenak.c
……

#define MENSAJE_ERROR1 "No argumentos erroneo\n“

int main(int argc, char *argv[])
{
  int j;
  struct stat infoinode;
  mode_t  modo;
  char Linea[256];
  struct passwd *ppas;
  struct group  *pgrp;

  if (argc < 2 ) {
    write(2, MENSAJE_ERROR1, strlen(MENSAJE_ERROR1));
    exit(1);
  }
  for (j = 1; j < argc; j++)  {
    stat(argv[j], &infoinode);
    modo = infoinode.st_mode;
    sprintf(Linea, "%12s : Mode(%06o) ", argv[j], modo);
    ppas = getpwuid(infoinode.st_uid);
    pgrp = getgrgid(infoinode.st_gid);
    sprintf(Linea, "%s %s %s ", Linea, ppas->pw_name, pgrp->gr_name);
    traza_modo(modo, Linea+strlen(Linea));
    sprintf(Linea, "%s \n", Linea);
    write(1, Linea, strlen(Linea));
  }
  return EXIT_SUCCESS;
}
```

Adibidea: baimenak.c

```
/*
============================================================================
 Name        : baimenak.c
 Author      : G.A. & M.L.
 Version     : 1.0
 Copyright   : ATC-KAT - Fac. Informatica - UPV/EHU
 Description : Ejemplo de mostrar los permisos de un fichero
 ============================================================================
 */
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include <grp.h>
```

```
const mode_t vmodos[] = {
        S_IRUSR, S_IWUSR, S_IXUSR, S_IRGRP, S_IWGRP, S_IXGRP,
        S_IROTH, S_IWOTH, S_IXOTH
};

const char mode_chars[3]  = "rwx";

void traza_modo(mode_t modo, char *Linea)
{
  char car, *pLinea;
  int i;
  pLinea = Linea;
  for (i = 0; i < 9; i++)  {
    if (vmodos[i] & modo) car = mode_chars[i%3];
    else car = '-';
    *pLinea = car;
    pLinea++;
  }
  *pLinea = '\0';
}
```

Adibidea: proba.c

```
#define errore(a) {perror(a); exit(1);};

main (int argc, char *argv[])
{
   int fd, fd2, n; char buf[80];

   if (argc != 2) errore("argumentuak gaizki");

   unlink(argv[1]);
   if ((fd = open(argv[1], O_WRONLY|O_CREAT|O_TRUNC, 0666)) == -1)
      errore("open 1");
   if (write(fd, "123456789\n", 10) != 10) errore("write 1");
   close(fd);

   if ((fd = open(argv[1], O_RDONLY)) == -1) errore("open 2");

   if (chmod(argv[1], 0266) == -1) errore("chmod");

   if ((fd2 = open(argv[1], O_RDONLY)) == -1) perror("open 2");

   if ((n = read(fd, buf, 4)) != 4) errore("read");
   if (write(1, buf, n) != n) errore("write 2");
   if (write(1, "\n", 1) != 1) errore("write 3");
   close(fd);

   if ((fd = open(argv[1], O_RDWR)) == -1) errore("open 3");
}
```

Adibidea: remove.c

```
main(int argc, char *argv[])
{
   if (argc != 2) {
      printf("Usage:  %s filename\n", argv[0]);
      exit(1);
   }

   if (access(argv[1], F_OK) == -1) {  /* check that file exists */
      perror(argv[1]);
      exit(1);
   }

   if (access(argv[1], W_OK) == -1) {  /* check for write permission */
      printf("File:  %s  is write protected!\n", argv[1]);
      exit(1);
   }

   if (unlink(argv[1]) == -1) {
      perror(argv[1]);
      exit(1);
   }
}
```

Adibidea: maskara_irakurri.c

```
mode_t read_umask()
{
    mode_t mask;

    mask = umask(0);
    umask(mask);

    return mask;
}
```