Sistema Eragileen Oinarriak

8. Praktika:

Bezero/Zerbitzari eredua

BEZERO/ZERBITZARI eredua: Algoritmoa

```
main()
{
  zerbitzariaren_buzoia_sortu_eta_ireki(...);
  while(1)
  {
      eskaera_jaso(...);
      nahiko_baliabide_libre?(...);
            EZ --> itxaron_libratu_arte(...);
      baliabidea_erabili(...);
      erantzun_bezeroari(...);
  }
}
```

B/Z: Kontuan izan beharrekoak

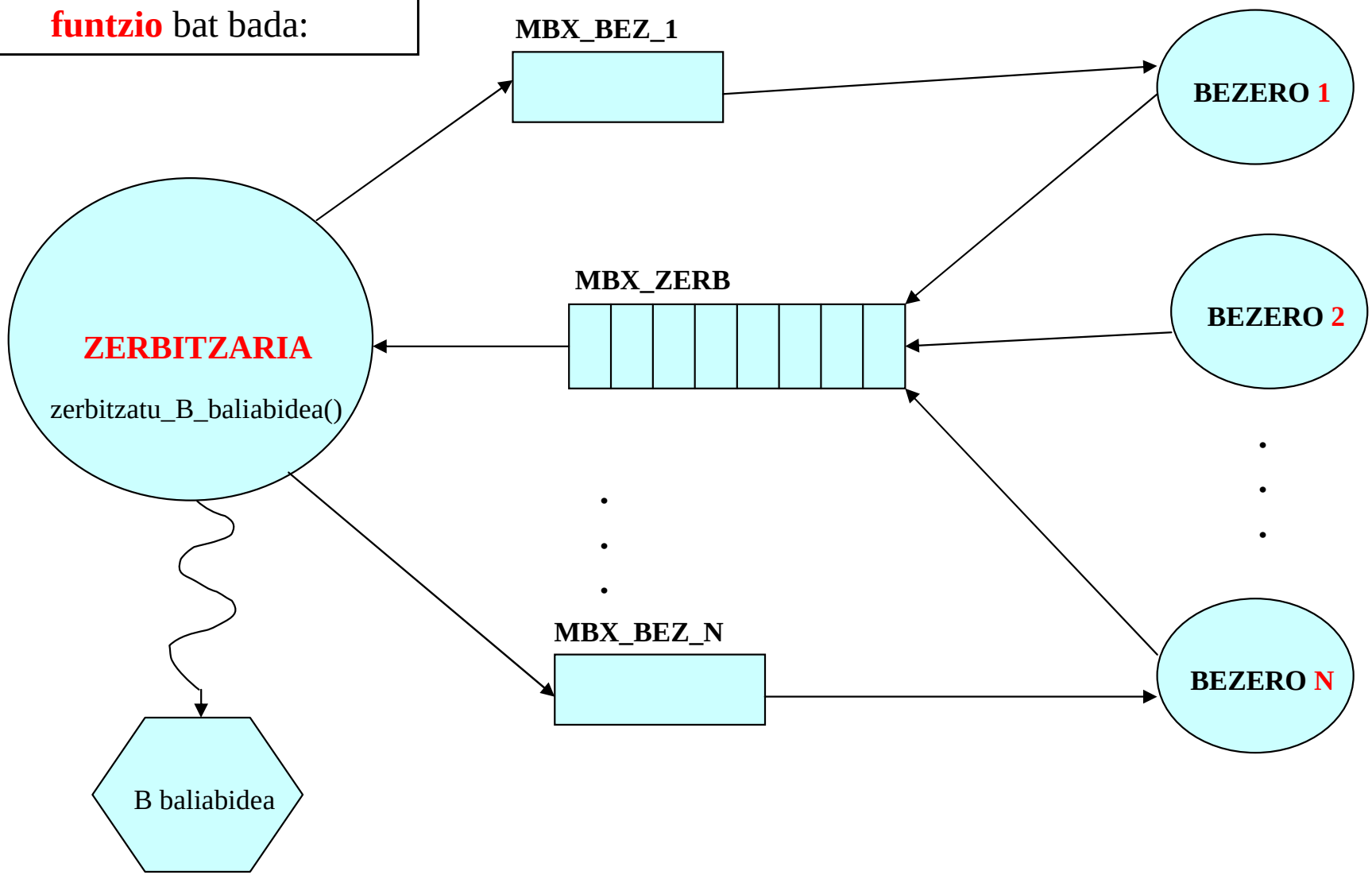
- Baliabide kopurua? **(1 vs N)**
  - **1:** Zerbitzari sekuentziala (eskaerak banaka tratatu)
  - **N:** Zerbitzari konkurrentea
    - Zenbat baliabide eskatu eskaera baten? **(1 vs k, non k <= N)**

- Baliabidea erabiltzeko? **Funtzioa vs Programa**
  - **Funtzioa:** denbora asko edo gutxi?

- Bezeroari noiz (eta nork) erantzun?
  - Eskaera **jasotzean** ***(buzoitik irakurtzerakoan)***
  - Eskaera **zerbitzatzen hastean**
  - Eskaera **zerbitzatzen amaitzean**

zerbitzari1_funtz

zerbitzatu_B_baliabidea **funtzio** bat bada:



zerbitzari1_funtz

```
#define MAX 80
main(int argc, char *argv[])
{

struct eskaera esk;
struct erantzuna eran;
char ema[50];
int fd_buz, fd_bez;

  unlink("MBX_ZERB");
  mkfifo("MBX_ZERB",0666);
  fd_buz = open("MBX_ZERB", O_RDWR);

  while(1) /* eskaera bakoitzeko */
  {
     read(fd_buz, &esk, sizeof(struct eskaera));

     fd_bez = open(esk.bez_buzoia, O_WRONLY);
     eran.bezeroa = esk.bezeroa;
     strcpy(eran.emaitza, "Zerbitzatzen hasi behar gara");
     write(fd_bez, &eran, sizeof(struct erantzuna));
     close(fd_bez);

     ema = zerbitzatu_B_baliabidea(esk.zerbitzatzeko);

  }
}
```

zerbitzatu_B_baliabidea **funtzioa** da

```
struct eskaera {
   int bezeroa;
   char bez_buzoia[20];
   char zerbitzatzeko[40];
 }


struct erantzuna {
   int bezeroa;
   char emaitza[50];
 }
```

Erantzuna zerbitzua hastera doanean = eskaera jasotzean

zerbitzari2_funtz

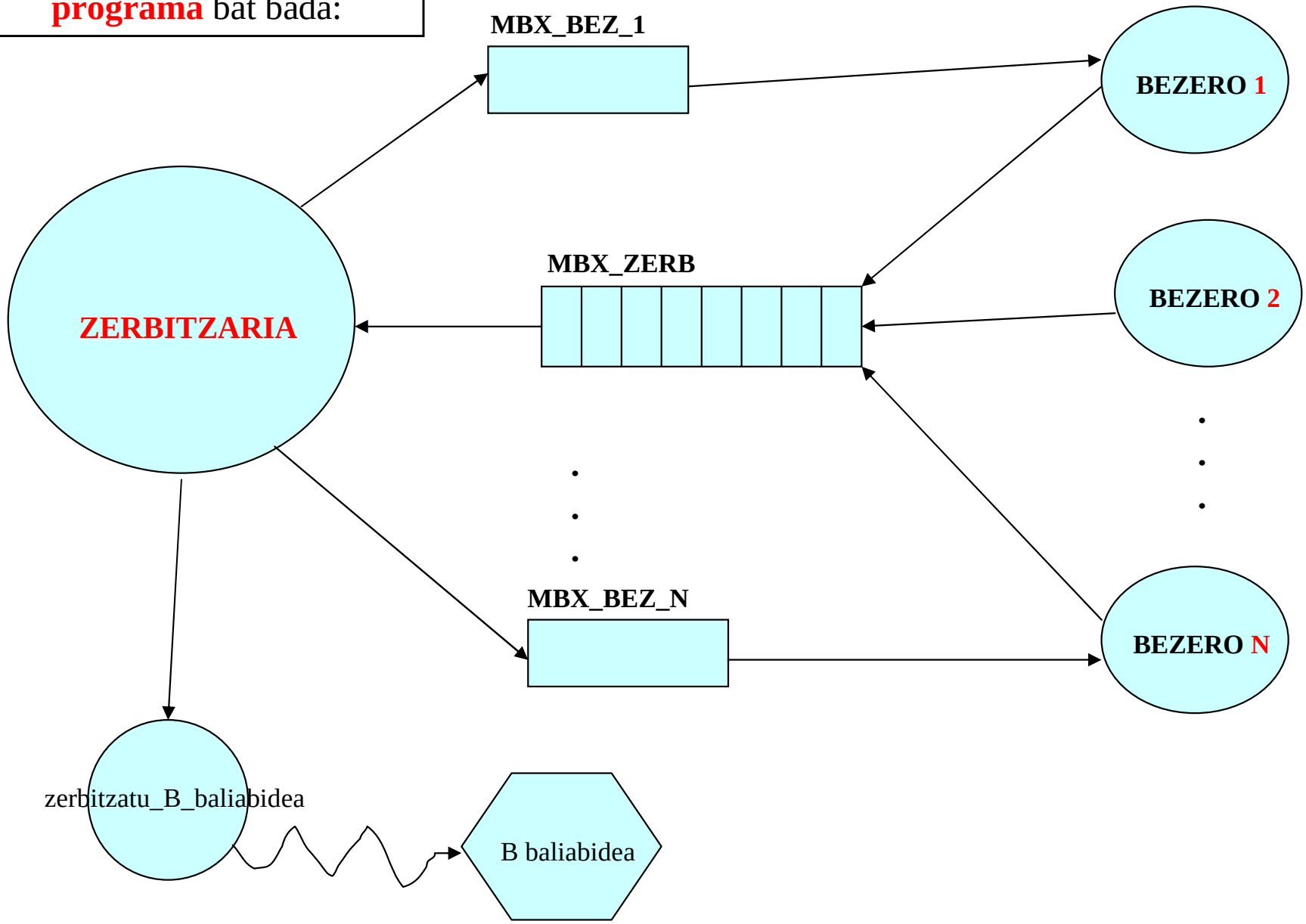
```
#define MAX 80
main(int argc, char *argv[])
{

struct eskaera esk;
struct erantzuna eran;
int fd_buz, fd_bez;

  unlink("MBX_ZERB");
  mkfifo("MBX_ZERB",0666);
  fd_buz = open("MBX_ZERB", O_RDWR);

  while(1) /* eskaera bakoitzeko */
  {
     read(fd_buz, &esk, sizeof(struct eskaera));

     eran.emaitza = zerbitzatu_B_baliabidea(esk.zerbitzatzeko);
     eran.bezeroa = esk.bezeroa;

     fd_bez = open(esk.bez_buzoia, O_WRONLY);
     write(fd_bez, &eran, sizeof(struct erantzuna));
     close(fd_bez);
  }
}
```

zerbitzatu_B_baliabidea **funtzioa** da

Erantzuna zerbitzua bukatu denean

zerbitzari1_prog

zerbitzatu_B_baliabidea **programa** bat bada:



zerbitzari1_prog

```
main(int argc, char *argv[])
{
struct eskaera esk;
struct erantzuna eran;
int fd_buz, fd_bez, pid;

  unlink("MBX_ZERB");
  mkfifo("MBX_ZERB",0666);
  fd_buz = open("MBX_ZERB", O_RDWR);

  while(1) /* eskaera bakoitzeko */
  {
     read(fd_buz, &esk, sizeof(struct eskaera));

     pid = fork();
     if (pid == 0)
     {
        execlp("zerbitzatu_B_baliabidea","zerbitzatu_B_baliabidea",esk.zerbitzatzeko,NULL);
     }
     fd_bez = open(esk.bez_buzoia, O_WRONLY);
     eran.bezeroa = esk.bezeroa;
     strcpy(eran.emaitza, "Zerbitzatzen hasi gara");
     write(fd_bez, &eran, sizeof(struct erantzuna));
     close(fd_bez);

     wait(null);
  }
}
```

zerbitzatu_B_baliabidea **programa** bat bada:

Erantzuna zerbitzua hastera doanean = eskaera jasotzean

zerbitzari2_prog

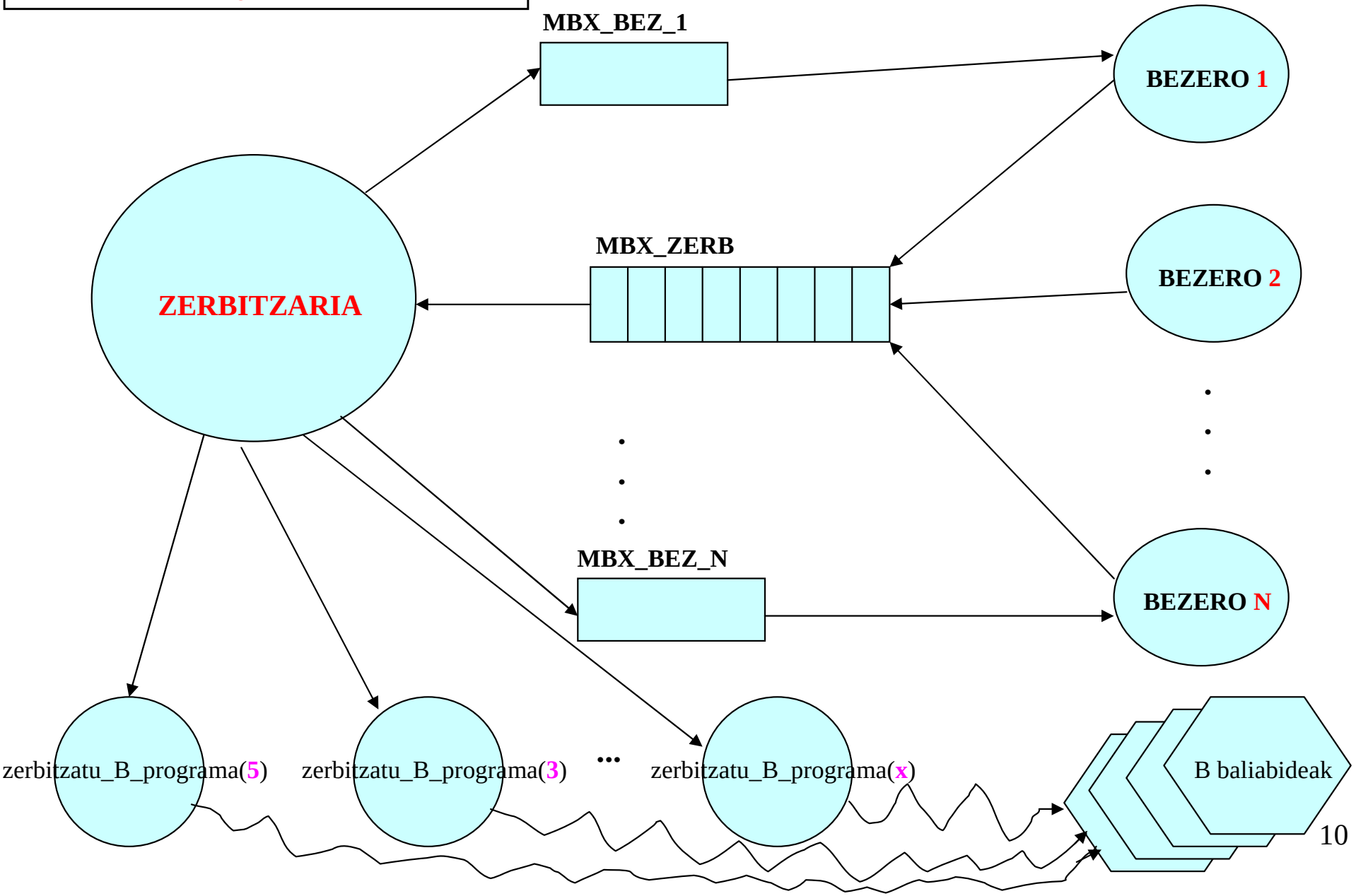
```
main(int argc, char *argv[])
{
struct eskaera esk;
struct erantzuna eran;
int fd_buz, fd_bez, pid, ema;

  unlink("MBX_ZERB");
  mkfifo("MBX_ZERB",0666);
  fd_buz = open("MBX_ZERB", O_RDWR);

  while(1) /* eskaera bakoitzeko */
  {
     read(fd_buz, &esk, sizeof(struct eskaera));

     pid = fork();
     if (pid == 0)
     {
        execlp("zerbitzatu_B_baliabidea","zerbitzatu_B_baliabidea",esk.zerbitzatzeko,NULL);
     }
     wait(&ema);
     sprintf(eran.emaitza, "Emaitza: %d \n", ema);
     eran.bezeroa = esk.bezeroa;

     fd_bez = open(esk.bez_buzoia, O_WRONLY);
     write(fd_bez, &eran, sizeof(struct erantzuna));
     close(fd_bez);
  }
}
```

zerbitzatu_B_baliabidea **programa** bat bada:

Erantzuna zerbitzua bukatu denean

zerbitzari3A

bezeroa esnatu (desblokeatu) behar da, ***eskaera jaso den unean***



zerbitzari3A

```
struct eskaera {
   int bezeroa;
   char bez_buzoia[20];
   char zerbitzatzeko[40];
   int kopurua;
 }
```

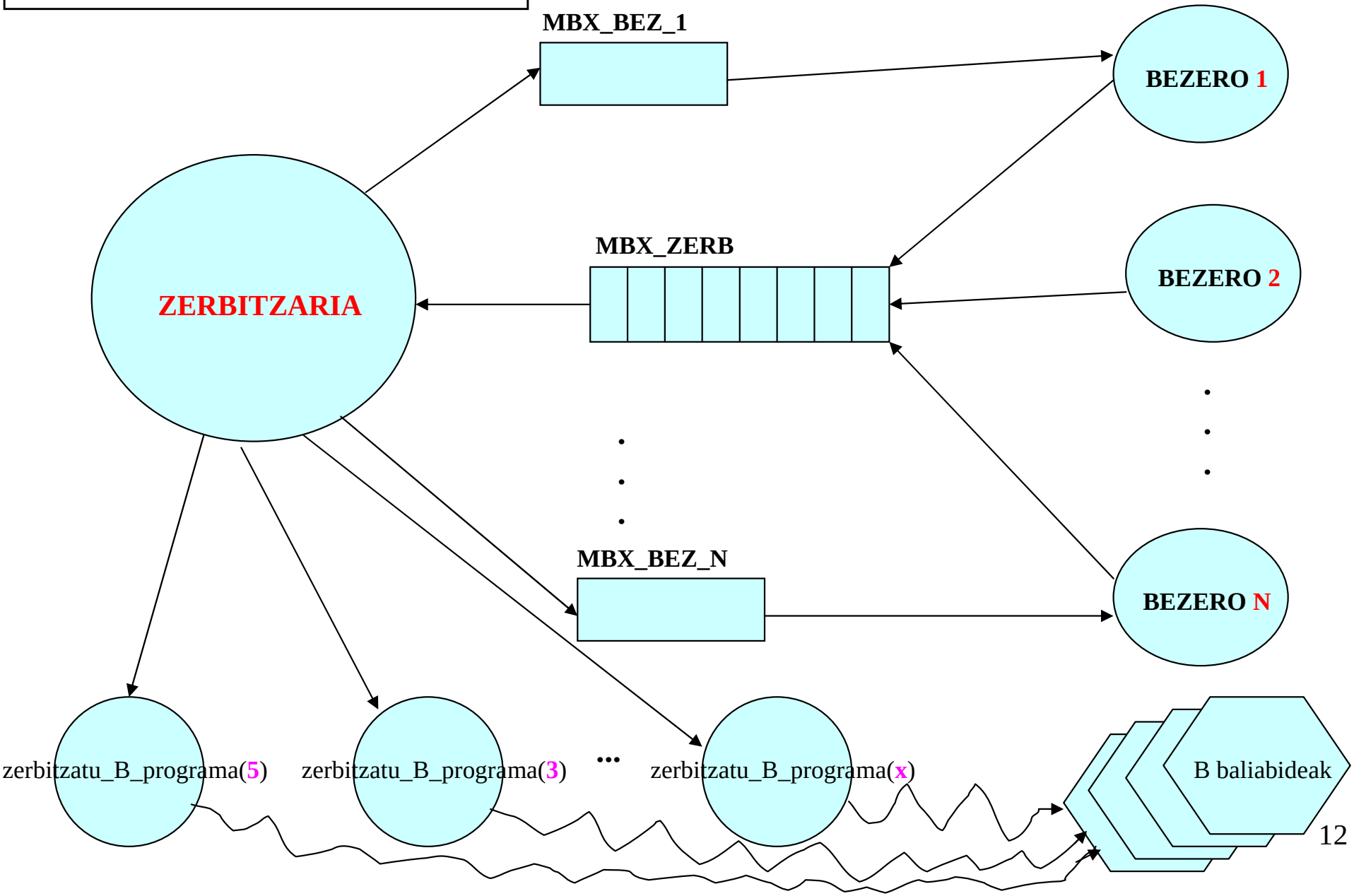
```
#define BAL 10
main(int argc, char *argv[])
{
struct eskaera esk; struct erantzuna eran;
int fd_buz, fd_bez, pid, k, baliabide_okup=0; char kopuru[16];
  unlink("MBX_ZERB");
  mkfifo("MBX_ZERB",0666);
  fd_buz = open("MBX_ZERB", O_RDWR);
  while(1)
  {
     read(fd_buz, &esk, sizeof(struct eskaera));
     fd_bez = open(esk.bez_buzoia, O_WRONLY);
     eran.bezeroa = esk.bezeroa;
     strcpy(eran.emaitza, "Eskaera jaso da!");
     write(fd_bez, &eran, sizeof(struct erantzuna));
     close(fd_bez);

     while (baliabide_okup + esk.kopurua > BAL)
     {
         wait(&k);
         baliabide_okup = baliabide_okup - k;
     }
     baliabide_okup = baliabide_okup + esk.kopurua;
     pid = fork();
     if (pid == 0) { sprintf(kopuru, "%d", esk.kopurua);
           execlp("zerbitzatu_B_programa","zerbitzatu_B_programa",esk.zerbitzatzeko,
                 kopuru,NULL); }
  }
}
```

bezeroa esnatu (desblokeatu) behar da, ***eskaera jaso den unean***

zombie daudenak askatu, edo zain geratu martxan daudenak bukatu arte

zerbitzari3B

bezeroa esnatu (desblokeatu) behar da, ***eskaera zerbitzatzen hastean***



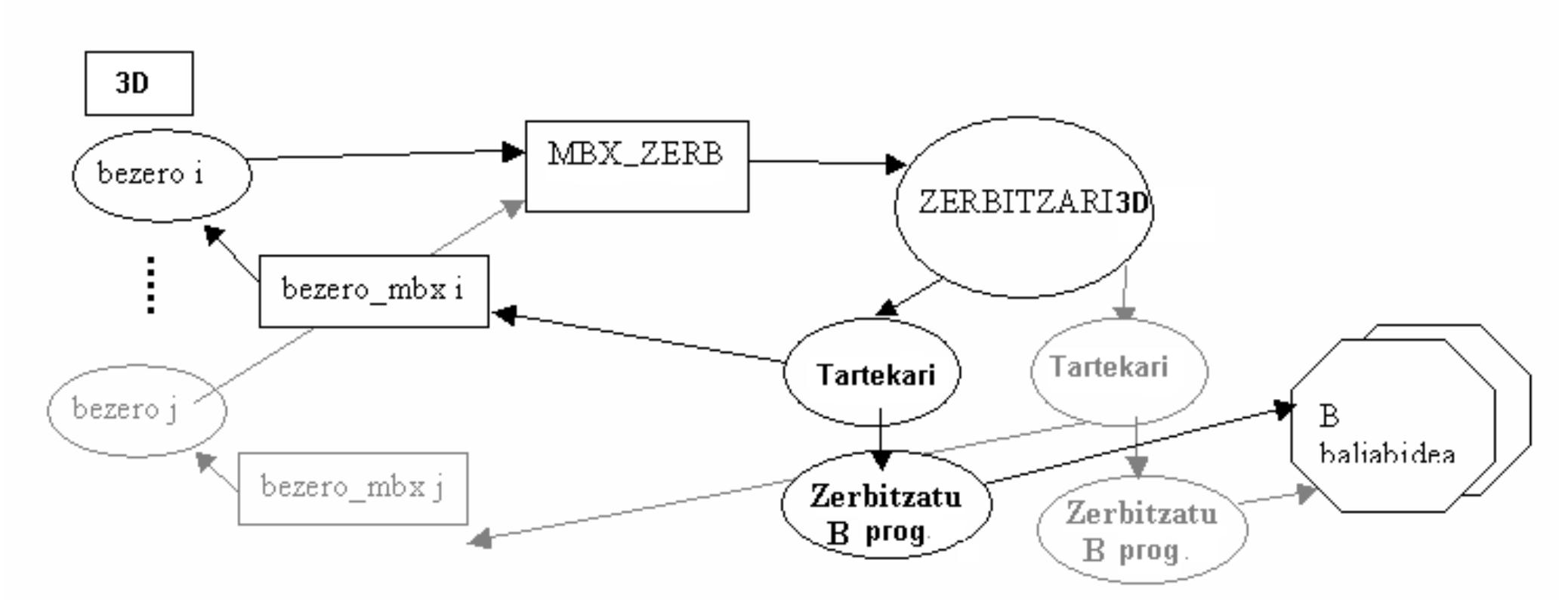
zerbitzari3B

```
#define BAL n;
main(int argc, char *argv[])
{
struct eskaera esk; struct erantzuna eran;
int fd_buz, fd_bez, pid, k, baliabide_okup=0; char kopuru[16];
  unlink("MBX_ZERB");
  mkfifo("MBX_ZERB",0666);
  fd_buz = open("MBX_ZERB", O_RDWR);
  while(1)
  {
     read(fd_buz, &esk, sizeof(struct eskaera));
     while (baliabide_okup + esk.kopurua > BAL)
     {
         wait(&k);
         baliabide_okup = baliabide_okup - k;
     }
     baliabide_okup = baliabide_okup + esk.kopurua;
     pid = fork();
     if (pid == 0)
     {   sprintf(kopuru, "%d", esk.kopurua);
         execlp("zerbitzatu_B_programa","zerbitzatu_B_programa",esk.zerbitzatzeko,
                kopuru,NULL);
     }
     fd_bez = open(esk.bez_buzoia, O_WRONLY);
     eran.bezeroa = esk.bezeroa;
     strcpy(eran.emaitza, "Zerbitzatzen hasi gara");
     write(fd_bez, &eran, sizeof(struct erantzuna));
     close(fd_bez);
  }
}
```

zombie daudenak askatu, edo zain geratu martxan daudenak bukatu arte

zerbitzari3C

bezeroa esnatu (desblokeatu) behar da, ***eskaera zerbitzatzen bukatzean***



zerbitzari3C

```
#define BAL n;
main(int argc, char *argv[])
{
struct eskaera esk;
int fd_buz, fd_bez, pid, k, baliabide_okup=0; char kopuru[16], bezero[16];

  unlink("MBX_ZERB");
  mkfifo("MBX_ZERB",0666);
  fd_buz = open("MBX_ZERB", O_RDWR);
  while(1)
  {
     read(fd_buz, &esk, sizeof(struct eskaera));
     while (baliabide_okup + esk.kopurua > BAL)
     {
         wait(&k);
         baliabide_okup = baliabide_okup - k;
     }
     baliabide_okup = baliabide_okup + esk.kopurua;
     pid = fork();
     if (pid == 0)
     {
         sprintf(kopuru, "%d", esk.kopurua);
         sprintf(bezero, "%d", esk.bezeroa);
         execlp("Tartekari","Tartekari",esk.zerbitzatzeko,kopuru,
                esk.bez_buzoia, bezero,NULL);
     }
  }
}
```

Tartekari

```
main(int argc, char *argv[])
{
char zerbitzua[50], bez_buzoi[50];
int kopurua, fd_bez, bezeroa;
struct erantzuna eran;

  strcpy(zerbitzua, argv[1]);
  kopurua = atoi(argv[2]);
  strcpy(bez_buzoi, argv[3]);
  bezeroa = atoi(argv[4]);

  pid = fork();
  if (pid == 0)
  {
     execlp("zerbitzatu_B_programa","zerbitzatu_B_programa", zerbitzua,
            argv[2], NULL);
  }
  wait(null);

  fd_bez = open(bez_buzoi, O_WRONLY);
  eran.bezeroa = bezeroa;
  strcpy(eran.emaitza, "Zerbitzua amaitu da!");
  write(fd_bez, &eran, sizeof(struct erantzuna));
  close(fd_bez);
  exit(kopurua);
}
```

`zerbitzatu_B_baliabidea zerbitzua kopurua`

programa bat bada:

Bezeroa esnatuko (desblokeatu) da ***baliabidea zerbitzatzen amaitzean***.