ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

[DΣIM]

Fonaments de Sistemes Operatius
Pràctica 2

curs 2022-23

Estudiants:

Rubén López Martínez

Pedro Mendoza Fernandez

Data de lliurament: 22/05/2021

1. Introducció 3

2. Decisions de disseny.....4

2.1 Cocos1 4

2.2 Cocos2 6

2.3 Cocos3 7

2.4 Cocos4 9

3. Implementació 14

3.1 Cocos1 14

3.2 Cocos2 32

3.3 Cocos3 49

3.4 Cocos4 71

4. Joc de proves 100

4.1 Cocos1 100

4.2 Cocos2 101

4.3 Cocos3 102

4.4 Cocos4 103

5. Autoavaluació i conclusions 105

1. Introducció

Aquesta pràctica tracta d'implementar el joc del "Menjacocos" amb caràcters ASCII en llenguatge C, millorant la seva execució mitjançant threads i processos amb sincronització.

El jugador assumeix el control d'un personatge anomenat "Menjacocos", el qual ha de menjar mentre evita els enemics, coneguts com a "fantasmes", en un laberint. El joc es desenvolupa en una pantalla de laberint, on el jugador ha de guiar "Menjacocos" per menjar tots els cocos del laberint abans de que els fantasmes aconsegueixen atrapar-lo, ja que sinó perd la vida i s'acaba la partida.

El moviment de "Menjacocos" es controla mitjançant les tecles de direcció del teclat. Per a la implementació del videojoc, es dibuixa un laberint de joc amb dimensions triades. També es decideix la quantitat de fantasmes que hi haurà al tauler segons uns paràmetres determinats. Aquests paràmetres es troben en un fitxer el qual es llegeix a partir del programa principal i es carrega el joc amb aquests determinats paràmetres.

Amb l'objectiu d'aprendre programació multithreading, es proposa modificar el joc base que ja se'ns proporciona, de manera que l'usuari pugui moure "Menjacocos" mentre s'executen de forma independent i simultània el moviment d'un o més fantasmes. Per tant, la idea principal és programar una rutina per al control de "Menjacocos" i després una o diverses per als fantasmes. Això s'aconseguirà mitjançant l'ús de fils (threads) o processos independents que es crearan en el programa principal.

2. Decisions de disseny

2.1 Cocos1

Inicialment tenim el cocos0.c versió inicial amb un sol fantasma. Aquesta versió es proporciona dins de l'espai moodle, partint de l' anterior, hem de modificar les funcions que controlen el moviment del menjacocos i del fantasma perquè puguin actuar com a fils d'execució independents. Així el programa principal crearà un fil pel menjacocos i un fil per a cada un dels fantasmes.

Per a aquesta fase, el primer que hem fet va ser canviar les funcions que volíem executar de manera independent aplicant un bucle i la condició d'execució d'aquest; en aquest cas mou_menjacocos i mou_fantasma.

En cas de que un fantasma toqui amb el menjacocos aquest pararà el programa.

En les dues funcions, el bucle de cadascuna comprovarà que qualsevol d'aquestes dues funcions sigui diferent de 1, ja que si alguna d'aquestes variables es 1, el bucle es trencarà i, per tant, finalitzarà la partida.

```c
void * mou_fantasma(void * index)
{
  int ind = (intptr_t) index;
  objecte seg;
  int resultat;
  int k, vk, nd, vd[3];
  int retardFantasma;
  int nMostraFantasma=ind;
  char buffer[10];

  do {  // Bucle propio de la función
    resultat = 0; nd = 0;
    retardFantasma=fl[ind].r;  // Retardo en función del índice
    for (k=-1; k<=1; k++) {   /* provar direccio actual i dir. veines */
      vk = (fl[ind].d + k) % 4; /* direccio veina */
      if (vk < 0) vk += 4;  /* corregeix negatius */
      seg.f = fl[ind].f + df[vk]; /* calcular posicio en la nova dir.*/
      seg.c = fl[ind].c + dc[vk];
      seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */
      if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0')) {
        vd[nd] = vk;  /* memoritza com a direccio possible */
        nd++;
      }
    }
    if (nd == 0) {  /* si no pot continuar, */
      fl[ind].d = (fl[ind].d + 2) % 4;  /* canvia totalment de sentit */
    } else {
      if (nd == 1) {  /* si nomes pot en una direccio */
        fl[ind].d = vd[0];  /* li assigna aquesta */
      } else {  /* altrament */
        fl[ind].d = vd[rand() % nd];  /* segueix una dir. aleatoria */
      }

      seg.f = fl[ind].f + df[fl[ind].d];  /* calcular seguent posicio final */
      seg.c = fl[ind].c + dc[fl[ind].d];
      seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */
      win_retard(retardFantasma*retard);  // Retardo
      win_escricar(fl[ind].f,fl[ind].c,fl[ind].a,NO_INV);  /* esborra posicio anteri
      fl[ind].f = seg.f; fl[ind].c = seg.c; fl[ind].a = seg.a;  /* actualitza posici

      nMostraFantasma=ind+1;
      sprintf(buffer, "%d", nMostraFantasma);
      win_escricar(fl[ind].f,fl[ind].c,buffer[0],NO_INV);  /* redibuixa fantasma */
      if (fl[ind].a == '0') resultat = 1; /* ha capturat menjacocos */
    }
    if (resultat) {
      fi2=resultat;

      break;  // Sale del bucle si ha capturado al personaje principal
    }

  } while (!fi1 && !fi2);

  return (NULL);
}
```

```c
/* funcio per moure el menjacocos una posicio, en funcio de la direccio de   */
/* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots  */
/* els cocos, i 0 altrament */
void * mou_menjacocos(void * null)
{
  char strin[12];
  objecte seg;
  int tecla, resultat;
  int retardMenjacocos;

  do{
    retardMenjacocos=mc.r;
    resultat=0;
    tecla = win_gettec();
    if (tecla != 0)
    switch (tecla)    /* modificar direccio menjacocos segons tecla */
    {
      case TEC_AMUNT:   mc.d = 0; break;
      case TEC_ESQUER:  mc.d = 1; break;
      case TEC_AVALL:   mc.d = 2; break;
      case TEC_DRETA:   mc.d = 3; break;
      case TEC_RETURN:  resultat = -1; break;
    }
    seg.f = mc.f + df[mc.d];  /* calcular seguent posicio */
    seg.c = mc.c + dc[mc.d];
    seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */
    if ((seg.a == ' ') || (seg.a == '.'))
    {
      win_retard(retardMenjacocos*retard);  // retard
      win_escricar(mc.f,mc.c,' ',NO_INV);   /* esborra posicio anterior */
      mc.f = seg.f; mc.c = seg.c;     /* actualitza posicio */
      win_escricar(mc.f,mc.c,'0',NO_INV);   /* redibuixa menjacocos */
      if (seg.a == '.')
      {
        cocos--;
        sprintf(strin,"Cocos: %d", cocos); win_escristr(strin);
        if (cocos == 0) resultat = 1; //guanya l'usuari (1)
      }
    }
    fi1=resultat;
  }while(fi1==0 && fi2==0);
  return (NULL);
}
```

El segon que vam fer, es crear les funcions pthread_create() per crear els fils d'execució (del menjacocos i els fantasmes) i, posteriorment els pthread_join() per fer que el procés esperi la finalització dels diferents threads creats.

```
fi1 = 0; fi2 = 0;

    /********** bucle principal del joc **********/

pthread_create(&tid[1], NULL, mou_menjacocos, (void *)(intptr_t)0);

for (int i = 0; i < numFantasmes; i++) {
  pthread_create(&tid[i], NULL, mou_fantasma, (void *)(intptr_t)i);
}

pthread_create(&tid[10], NULL, cronometre, (void *)(intptr_t)0);

pthread_create(&tid[10], NULL, info, (void *)(intptr_t)0);
//p++; if ((p%2)==0)    /* ralentitza fantasma a 2*retard */
```

```
for(int j = 0; j < MAX_THREADS; j++)
{
  pthread_join(tid[j], NULL);
}
```

Per últim, vam modificar les variables globals respecte a l'execució dels fantasmes, ja que en la pràctica ens demanaven que havíem de crear tants fantasmes existents com estiguin al fitxer per crear el joc, per tant aquestes variables globals respectives al fantasma les hem modificat a vectors de 9 posicions, que es el número màxim de fantasmes (threads) que podrem tenir al mateix temps en execució.
A més, també hem hagut d'afegit noves variables pel control del joc com el id_fantasma (per saber el identificador del thread (en quant al vector tid) que s'ha de crear, nFantasma (número de fantasmes existents al fitxer llegit) i numFantames (número de fantasmes que estaran executant-se en partida).

```
/* variables globals */
int n_fil1, n_col;    /* dimensions del camp de joc */
char tauler[70];    /* nom del fitxer amb el laberint de joc */
char c_req;         /* caracter de pared del laberint */

objecte mc;         /* informacio del menjacocos */
objecte f1[9];          /* informacio del fantasma 1 */

int df[] = {-1, 0, 1, 0}; /* moviments de les 4 direccions possibles */
int dc[] = {0, -1, 0, 1}; /* dalt, esquerra, baix, dreta */

int cocos;      /* numero restant de cocos per menjar */
int retard;       /* valor del retard de moviment, en mil.lisegons */
int fi1=0, fi2=0;
int nFantasmes=-1; //número de fantasma per guardar-ho en el vector a pa
int numFantasmes=0; //número de fantasmes existents
int idFantasmes=1; //identificador thread fantasma
pthread_t tid[MAX_THREADS];

char infoTemps[64];
char infoCocos[12];
int minuts=0,segons=0,hora=0;
```

Per últim, vam afegir un comptador de joc en el que ens imprimeix el temps transcorregut en hores, minuts i segons que portem en partida i es pausarà quan es compleixin les condicions de finalització de joc. Aixo ho hem fet amb una funció el qual actualitza els valors del contador i una altra funció per mostrar-ho per pantalla. Aquestes dues funcions les hem elaborat aplicant un thread per a cadascuna per a que s'executi de manera independent.

```
void * cronometre(void * index)
{

  bool acaba=false;
  do{
    if(segons==60){
      segons=0;
      minuts++;
      if(minuts==60){
        minuts=0;
        hora++;
      }
    }


    //win_escristr(infoTemps);
    win_retard(1000);
    segons++;
    if(((fi1)==1)||((fi2)==1)){
      acaba=true;
      fprintf(stderr, "fi1:%i  \t fi2:%i",fi1,fi2);
    }
  }while(acaba==false);
  return (NULL);
}



void* info(void* nul)
{
  char info[50];
  int auxCocos=0;
  int auxHores=0;
  int auxMinuts=0;
  int auxSegons=0;
  while(!fi1 && !fi2)
  {
      if((auxCocos!=cocos) || (auxHores!=hora) || (auxMinuts!=minuts) || (auxSegons!=segons)){
          info[0]='\0';
          sprintf(infoTemps,"Temps %i:%i:%i",hora,minuts,segons);
          sprintf(infoCocos,"Cocos: %d", cocos);
          strcat(info, infoTemps);
          strcat(info, " - ");
          strcat(info, infoCocos);

          win_escristr(info);
          auxCocos=cocos;
          auxHores=hora;
          auxMinuts=minuts;
          auxSegons=segons;
      }

  }
  return (NULL);
}
```

2. Decisions de disseny

2.2 Cocos2

En aquest apartat el que fem és sincronitzar els fils d'execució perquè a l'hora d'executar els seus respectius codis i facin ús de les funcions del winsuport no hagin problemes alhora de fer actualitzacions de la pantalla. D'aquesta manera el que vam fer, va ser definir les diferents seccions crítiques usant mutex. D'aquesta manera declarem el mutex, ho creem al programa principal mitjançant l'ús de la funció pthread_mutex_init() i una vegada acabat el programa, ho eliminem mitjançant pthread_mutex_destroy() per destruir-lo.

```
pthread_mutex_t fantasmamutex= PTHREAD_MUTEX_INITIALIZER;    /* crea un sem. Global*/
```

```
pthread_mutex_destroy(&fantasmamutex);
```

Les seccions crítiques les definim a través de pthread_mutex_lock() i pthread_mutex_unlock() en les funcions principals del joc (mou_fantasma i mou_menjacocos). D'aquesta manera, quan cada fantasma comprova i calcula la següent posició, ho fa de manera atòmica (sense cap interrupció per part de qualsevol element possible del joc). Per últim, vam declarar secció crítica al modificar en pantalla la nova posició del cocos per tal de fer les actualitzacions de la pantalla correctament.

```
void * mou_menjacocos(void * null)
{

  objecte seg;
  int tecla, resultat;
  int retardMenjacocos;

  do{
    retardMenjacocos=mc.r;
    resultat=0;
    tecla = win_gettec();
    if (tecla != 0)
    switch (tecla)    /* modificar direccio menjacocos segons tecla */
    {
      case TEC_AMUNT:   mc.d = 0; break;
      case TEC_ESQUER:  mc.d = 1; break;
      case TEC_AVALL:   mc.d = 2; break;
      case TEC_DRETA:   mc.d = 3; break;
      case TEC_RETURN:  resultat = -1; break;
    }
    seg.f = mc.f + df[mc.d];  /* calcular seguent posicio */
    seg.c = mc.c + dc[mc.d];
    seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

    if ((seg.a == ' ') || (seg.a == '.'))
    {
      pthread_mutex_lock(&fantasmamutex);
      win_escricar(mc.f,mc.c,' ',NO_INV);   /* esborra posicio anterior */
      mc.f = seg.f; mc.c = seg.c;     /* actualitza posicio */
      win_escricar(mc.f,mc.c,'0',NO_INV);   /* redibuixa menjacocos */
      pthread_mutex_unlock(&fantasmamutex);
      if (seg.a == '.')
      {
        cocos--;
        //win_escristr(strin);
        if (cocos == 0) resultat = 1; //guanya l'usuari (1)
      }
    }

    fi1=resultat;
    win_retard(retardMenjacocos*retard);  // retard

  }while(fi1==0 && fi2==0);
  return (NULL);
}
```

```
void * mou_fantasma(void * index)
{
  int ind = (intptr_t) index;
  objecte seg;
  int resultat;
  int k, vk, nd, vd[3];
  int retardFantasma;
  int nMostraFantasma=ind;
  char buffer[10];

  do {  // Bucle propio de la función
    resultat = 0; nd = 0;
    retardFantasma=fl[ind].r;  // Retardo en función del índice
    pthread_mutex_lock(&fantasmamutex);
    for (k=-1; k<=1; k++) {   /* provar direccio actual i dir. veines */
      vk = (fl[ind].d + k) % 4; /* direccio veina */
      if (vk < 0) vk += 4;  /* corregeix negatius */
      seg.f = fl[ind].f + df[vk]; /* calcular posicio en la nova dir.*/
      seg.c = fl[ind].c + dc[vk];

      seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

      if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0')) {
        vd[nd] = vk;  /* memoritza com a direccio possible */
        nd++;
      }
    }
    if (nd == 0) {  /* si no pot continuar, */
      fl[ind].d = (fl[ind].d + 2) % 4;  /* canvia totalment de sentit */
    } else {
      if (nd == 1) {  /* si nomes pot en una direccio */
        fl[ind].d = vd[0];  /* li assigna aquesta */
      } else {  /* altrament */
        fl[ind].d = vd[rand() % nd];  /* segueix una dir. aleatoria */
      }

      seg.f = fl[ind].f + df[fl[ind].d];  /* calcular seguent posicio final */
      seg.c = fl[ind].c + dc[fl[ind].d];

      seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

      win_escricar(fl[ind].f,fl[ind].c,fl[ind].a,NO_INV);  /* esborra posicio anterior */


      fl[ind].f = seg.f; fl[ind].c = seg.c; fl[ind].a = seg.a;  /* actualitza posicio */

      nMostraFantasma=ind+1;
      sprintf(buffer, "%d", nMostraFantasma);

      win_escricar(fl[ind].f,fl[ind].c,buffer[0],NO_INV);  /* redibuixa fantasma */


      if (fl[ind].a == '0') resultat = 1; /* ha capturat menjacocos */
    }

    pthread_mutex_unlock(&fantasmamutex);

    if (resultat) {
      fi2=resultat;

      break;  // Sale del bucle si ha capturado al personaje principal
    }
    win_retard(retardFantasma*retard);  // Retardo
  } while (!fi1 && !fi2);

  return (NULL);
}
```

2. Decisions de disseny

2.3 Cocos3

En aquesta fase, ens demanaven fer el mateix que vam fer a la fase cocos1 però, en comptes de treballar ara amb threads, treballarem amb processos (multiprocés). Per tant, en aquest cas, cada fantasma serà controlat per un procés fill. El codi d'aquests processos fill haurà de residir en un fitxer executable diferent del fitxer executable del programa principal (programa pare). Per tant, vam crear un fitxer anomenat fantasma.3 en el qual s'aniran creant (utilitzant la funció fork()) i executant en forma de processos fills els diferents fantasmes.

Per resoldre aquest fet, el que vam fer va ser decidir quines variables estarien en memòria compartida per tal de canviar els seus valors tant als processos fills com al procés pare; aquestes variables son el valor del taulell de joc (del qual obtenim el seu identificador del IPC), juntament amb els valors de fi1 i fi2 per tal de saber la finalització del joc.

Aquestes variables en memòria compartida, primer de tot les inicialitzem amb la funció ini_mem(), i després obtenim l'adreça en memòria compartida (a partir de la funció map_mem()) de cada variable en qüestió que necessitarà els processos fills (fantasmes) per tal de modificar o treballar amb aquestes variables compartides; per tal d'obtenir les adreces de memòria i per tant, els valors d'aquestes variables compartides en els processos fills, el fitxer fantasma3 s'encarregarà d'obtenir totes les variables que las passarà a enters (ja que el procés pare se las passa en Strings), i farà un map_mem en cada variable en memòria compartida per tal de recuperar les adreces d'aquestes variables.

```
//TAULER
id_tauler = ini_mem(rc);   //crear zona de memòria compartida amb els bytes que ocupa el mapa del joc
p_tauler = map_mem(id_tauler); //estableix un identificador
//set_filcol(p_tauler,n_fil1, n_col);
win_set(p_tauler,n_fil1,n_col);

inicialitza_joc();

char id_taulerx[10];
char n_fil1x[10];
char n_colx[10];
sprintf(id_taulerx, "%d", id_tauler);
sprintf(n_fil1x, "%d", n_fil1);
sprintf(n_colx, "%d", n_col);


//FI1 i FI2//
char id_fil1c[10];
char id_fi2c[10];
id_fi1 = ini_mem(sizeof(int));  /* crear zona mem. compartida */
fi1 = map_mem(id_fi1);    /* obtenir adres. de mem. compartida */
*fi1 = 0;       /* inicialitza variable compartida */
sprintf(id_fil1c,"%i",id_fi1);

id_fi2 = ini_mem(sizeof(int));  /* crear zona mem. compartida */
fi2 = map_mem(id_fi2);    /* obtenir adres. de mem. compartida */
*fi2 = 0;
```

Fitxer cocos3 (pare)

```
p_tauler = map_mem(id_tauler);  /* obtenir adres. de mem. compartida */
if (p_tauler == (int*) -1)
{
    fprintf(stderr, "proces (%d): error en identificador de memoria \n",(int)getpid());
    exit(0);
}
win_set(p_tauler,n_fil1,n_col); /* crea acces a finestra oberta pel proces pare */


fi1 = map_mem(id_fi1);      /* obtenir adres. de mem. compartida */
fi2 = map_mem(id_fi2);      /* obtenir adres. de mem. compartida */
```

Fitxer fantasma3 (fill)

Finalment, aquestes variables (juntament amb les variables que no son a memòria compartida però que necessiten els processos fills per consultar els seus valors) les passem a String i les guardem en vectors de caràcters necessaris pel funcionament de la funció execlp() ubicat al bucle del funcionament de joc del programa principal, funció que substitueix el codi, les dades i la pila del procés que la invoca per un nou entorn que es carregarà a partir del fitxer executable necessari pel funcionament dels processos fills.

```
for(int i=0; i<numFantasmes; i++)
{
  tpid[numProcesos]=fork();   /* crea un nou proces (fill)*/
  if(tpid[numProcesos]==(pid_t)0)
  {
    sprintf(id_fantasmax, "%d", i);
    sprintf(fl_fx, "%d", fl[i].f);
    sprintf(fl_cx, "%d", fl[i].c);
    sprintf(fl_dx, "%d", fl[i].d);
    sprintf(fl_rx, "%f", fl[i].r);

    execlp("./fantasma3", "fantasma3", id_taulerx, n_fillx, n_colx, id_fantasmax, fl_fx, fl_cx, fl_dx, fl_rx, id_filc, id_fi2c, retardx,(char *)0);
    fprintf(stderr, "error: no puc executar el process fill");
    exit(0);
  }
  else if (tpid[numProcesos] > 0) numProcesos++;    /* branca del pare */
}
```

Fitxer cocos3 (pare)

També afegir, que igual que com va fer a cocos1, tenim la mateixa estructura en quant a la implementació del cronòmetre del joc.

2. Decisions de disseny

2.4 Cocos4

En aquesta fase ens demanen completar la fase anterior de manera que l'execució dels processos es sincronitzi a l'hora d'accedir als recursos compartits (per exemple, l'entorn de dibuix, rutines de curses i variables globals necessàries). D'aquesta manera s'han de solucionar els problemes de visualització que presenta la fase anterior, els quals es fan patents quan s'intenta executar el programa sobre el servidor remot bsd.deim.urv.cat. En aquesta fase volem realitzar una sincronització en la sortida de fantasmes i també s'ha una interacció entre els fantasmes per tal de localitzar i atrapar al menjacocos. També afegir comunicació entre processos mitjançant comunicació indirecta (a partir de pas de missatges mitjançant bústies).

SEMÀFORS

Per tant, per resoldre aquesta fase, vam intentar aplicar semàfors, on el joc actualitza la pantalla o modifica variables globals en les que poden accedir diferents processos al mateix temps.

A diferència del cocos2, on també vam aplicar semàfors pels threads, en processos les funcions son diferents:
-Primer cridem a la funció ini_sem(1) per tal d'inicialitzar els semàfors. Aquesta funció genera un identificador que ho guardem en una variable i la passem a String com vam fer amb les variables de la fase anterior, guardant aquest String en un vector de caràcters.

```
id_sem = ini_sem(1);    /* crear semafor IPC inicialment obert */
sprintf(a5,"%i",id_sem);    /* convertir identificador sem. en string */
```

Fitxer cocos4 (pare)

-En aquest cas, vam intentar aplicar els waits i signals (passant-li el semàfor per memòria compartida a cada fantasma) , però quan intentàvem aplicar la mateixa lògica que al cocos2 (el qual aplicàvem mutex lock i unlock on es calculava i es modificava la posició de cada fantasma), em aquest cas no ens funcionava i no ens printava els fantasmes; per aquesta raó no hem pogut aplicar semàfors a les seccions crítiques adients.

PAS DE MISSATGES

Pel que fa al pas de missatges (per tal de crear un fantasma i passar la seva informació cada cop que es produeixi dos xocs a les parets per part del menjacocos), els passos que vam seguir son:

1. Comprovem al procés pare, a la funció moure_mejacocos, si el menjacocos s'ha xocat dos cops amb un bloc; si és el cas, s'envia per una bústia compartida independent de cada fantasma, la informació del proper fantasma que es crearà degut a aquesta condició dels xocs.

```
}
else if (xocs<=2)
{
  xocs++;
}
if ((xocs==2) && (numProcesos<numFantasmes))
{
  tpid[numProcesos]=fork();
  if(tpid[numProcesos]==(pid_t)0)
  {
    sprintf(id_fil1c,"%i",id_fil1);
    sprintf(id_fi2c,"%i",id_fi2);
    sprintf(numProcesosx,"%i",numProcesos);
    //sprintf(a4, "%i", id_bustias[numProcesos]);
    sprintf(missatgeE, "%d;%d;%d;%f", f1[numProcesos].f, f1[numProcesos].c,f1[numProcesos].d,f1[numProcesos].r);
    sendM(id_bustias[numProcesos],missatgeE,100);
    //*nfActius++;      /* inicialitza variable compartida */
    *nfActius=*nfActius+1;

    execlp("./fantasma4", "fantasma4", id_taulerx, n_fil1x, n_colx, id_fil1c, id_fi2c, retardx, a5, numProcesosx, id_mis1, id_mis2, id_mis3, id_mi

    exit(0);
  }
  else if (tpid[numProcesos] > 0) numProcesos++;


  xocs=0;
}
```

Fitxer cocos4 (pare)

2. Al main del procés pare, hem fet un bucle per establir un identificador (fent ini_mis) per a cada fantasma, tenint 9 identificadors de les 9 bústies, un per a cada fantasma.

```
for (int i = 0; i < 9 ; i++)
{
  id_bustias[i] = ini_mis();      /* crear una bustia I
  //fprintf(stderr, "\nPrimer: %i", id_mis[i]);
}

sprintf (id_mis1, "%i", id_bustias[0]);
sprintf (id_mis2, "%i", id_bustias[1]);
sprintf (id_mis3, "%i", id_bustias[2]);
sprintf (id_mis4, "%i", id_bustias[3]);
sprintf (id_mis5, "%i", id_bustias[4]);
sprintf (id_mis6, "%i", id_bustias[5]);
sprintf (id_mis7, "%i", id_bustias[6]);
sprintf (id_mis8, "%i", id_bustias[7]);
sprintf (id_mis9, "%i", id_bustias[8]);
    /********** bucle principal del joc **********/
```

3. Passem cada identificador per la funció execlp. A més de passar aquests identificadors, també passem el fantasma actual que es crearà ("numProcesosx"), y el número de fantasmes que hi ha actius ja creats i printats a la partida del joc ("nfActiusx").

```
tpid[numProcesos]=fork();   /* crea un nou proces (fill)*/
if(tpid[numProcesos]==(pid_t)0)
{


  //id_bustias[numProcesos] = ini_mis();    /* crear bustia IPC */
  //sprintf(a4,"%i",id_bustias[numProcesos]);  /* convertir identif. bustia en string */
  sprintf(numProcesosx,"%i",numProcesos);

  sprintf(missatgeE, "%d;%d;%d;%f", f1[numProcesos].f, f1[numProcesos].c,f1[numProcesos].d,f1[numProcesos].r);
  sendM(id_bustias[numProcesos],missatgeE,100);

  execlp("./fantasma4", "fantasma4", id_taulerx, n_fillx, n_colx, id_fi1c, id_fi2c, retardx, a5, numProcesosx, id_mis1, id_mis2, id_mis3, id_mis4, id_mis5, id_mis6, id_mis7, id_mis8, id_mis9, nfActiusx,(char *)0);
  exit(0);
}
else if (tpid[numProcesos] > 0) numProcesos++;
```

4. Ara, en el procés fill, s'obté la informació que contenia el execlp.
5. Una vegada obtinguts els identificadors de les bústies de cada fantasma, hem fet un receive del fantasma de la bústia del fantasma actual, per tal d'obtenir el missatge que ens ha enviat el procés pare amb la informació corresponent del fantasma creat actual. Com que la informació d'aquest missatge està en forma de String i cada paràmetre separat per ";", fem un strtok i un atof per cada ; que s'obtingui i anem assignant cada valor de cada paràmetre amb la variable del fantasma corresponent.

```
//NUM_PROCES//
id_fantasma = atoi(ll_args[8]);

//BUSTIA//
for (int i = 0; i < 9; i++)
{
    id_bustia[i] = atoi(ll_args[i + 9]);    // Carreguem els ids de les busties en un array de caracters
}

id_nfActius= atoi(ll_args[18]);
nfActius=map_mem(id_nfActius);



receiveM(id_bustia[id_fantasma],a4);

token = strtok(a4, ";");

// Guardar cada salida en las variables del objeto f1
f1.f = atoi(token);

token = strtok(NULL, ";");
f1.c = atof(token);

token = strtok(NULL, ";");
f1.d = atof(token);

token = strtok(NULL, ";");
f1.r = atof(token);
```

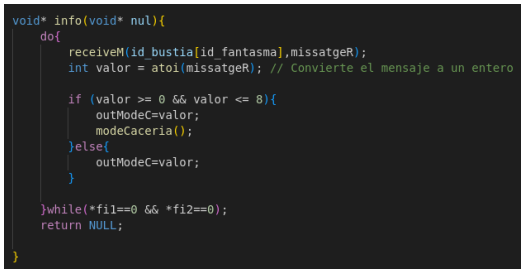
6. Una vegada tenim els valors que necessita el fantasma per començar amb la seva execució i ja ens funcionava el pas de missatges correctament, vam intentar fer la part del mode cacera i normal:

a. Primer vam fer el funcionament de la funció prismàtics per tal de que un fantasma tingui visió per inspeccionar les posicions del laberint que té al seu davant, és a dir, des de la seva posició i totes les posicions consecutives en la direcció actual de moviment, fins que trobin un obstacle: un caràcter de paret del laberint, un altre fantasma o el menjacocos. Quan l'obstacle sigui el menjacocos, enviarà un missatge amb el identificador del fantasma que ha vist al menjacocos; aquest missatge serà enviat per tots els fantasmes que estiguin actius al joc ja creats en partida (variable compartida "nfActiusx").

   En cas de que el fantasma no ha vist al menjacocos al haver fet la inspecció de les posicions consecutives el tauler, enviarà un missatge de -1 als fantasmes actius per a que no entrin o surtin del mode cacera.

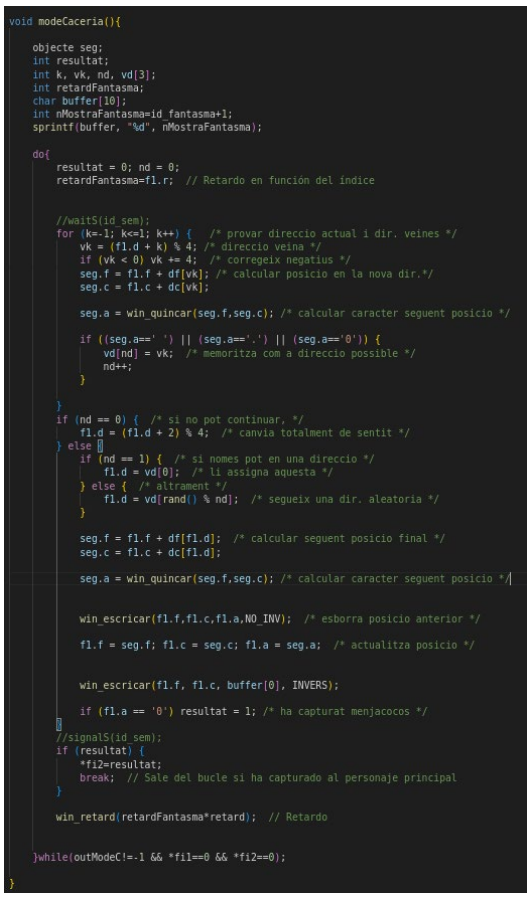
```
//prismatics
aux.f=seg.f;
aux.c=seg.c;
aux.a=seg.a;

acabaBucle=false;
//caza=false;
while (!acabaBucle) {
    aux.f = df[f1.d]+aux.f;
    aux.c = dc[f1.d]+aux.c;
    aux.a = win_quincar(aux.f, aux.c);
    if ((aux.a == '+') || (aux.a >= '1' && aux.a <= '9') || (aux.a == '0')) {
        acabaBucle=true;
        if(aux.a == '0'){
            caza = true;
            sprintf(missatgeE, "%d", id_fantasma);
            for(int i =0; i<*nfActius;i++){
                if(i!=id_fantasma){
                    sendM(id_bustia[i],missatgeE,25);
                }

            }

        }else{
            outModeC=-1;
            caza=false;
            sprintf(missatgeE, "%d", outModeC);
            for(int i =0; i<*nfActius;i++){
                if(i!=id_fantasma){
                    sendM(id_bustia[i],missatgeE,25);
                }

            }
        }

    }

}
```

b. Per tal de rebre aquests missatges emets pel fantasma que ha vist al menjacocos, hem fet un thread a la funció anomenada "info" per tal de llegir periòdicament la seva bústia. Quan la llegeix, comprova si el missatge es un número del 0 al 8 (que correspondrà al número/identificador del fantasma que ha emès el missatge), que significarà que aquest fantasma ha vist al menjacocos i per tant entrarà en mode cacera, o es un -1 que significa que no ha vist al menjacocos i, en aquest cas, indica que s'ha de sortir del mode cacera (si es que es trobaven en aquest mode) o directament que no s'ha d'entrar en aquest mode.



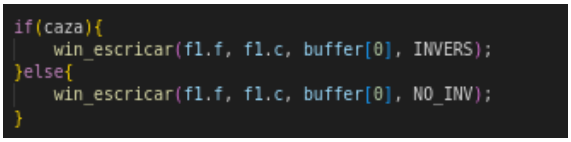
```
void* info(void* nul){
    do{
        receiveM(id_bustia[id_fantasma],missatgeR);
        int valor = atoi(missatgeR); // Convierte el mensaje a un entero

        if (valor >= 0 && valor <= 8){
            outModeC=valor;
            modeCaceria();
        }else{
            outModeC=valor;
        }

    }while(*fi1==0 && *fi2==0);
    return NULL;

}
```

c. Si el fantasma entra en mode cacera, executarà el mateix codi que a la fase anterior (fantasma3), però en aquest cas, el fantasma es printarà en mode invers, simbolitzant que ha entrat en mode cacera. Aquest mode s'executarà indefinidament fins que el fantasma llegeix un receive amb un -1 o per finalització del joc. Com que el fantasma que es trobi en aquest mode no serà el que haurà vist al fantasma, no farà la inspecció de posicions (prismàtics) perquè només ho farà el fantasma que hagi vist al menjacocos.



```
void modeCaceria(){

    objecte seg;
    int resultat;
    int k, vk, nd, vd[3];
    int retardFantasma;
    char buffer[10];
    int nMostraFantasma=id_fantasma+1;
    sprintf(buffer, "%d", nMostraFantasma);

    do{
        resultat = 0; nd = 0;
        retardFantasma=f1.r;  // Retardo en función del índice


        //waitS(id_sem);
        for (k=-1; k<=1; k++) {   /* provar direccio actual i dir. veines */
            vk = (f1.d + k) % 4; /* direccio veina */
            if (vk < 0) vk += 4;  /* corregeix negatius */
            seg.f = f1.f + df[vk]; /* calcular posicio en la nova dir.*/
            seg.c = f1.c + dc[vk];

            seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

            if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0')) {
                vd[nd] = vk;  /* memoritza com a direccio possible */
                nd++;
            }

        }
        if (nd == 0) {  /* si no pot continuar, */
            f1.d = (f1.d + 2) % 4;  /* canvia totalment de sentit */
        } else {
            if (nd == 1) {  /* si nomes pot en una direccio */
                f1.d = vd[0];  /* li assigna aquesta */
            } else {  /* altrament */
                f1.d = vd[rand() % nd];  /* segueix una dir. aleatoria */
            }

            seg.f = f1.f + df[f1.d];  /* calcular seguent posicio final */
            seg.c = f1.c + dc[f1.d];

            seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */


            win_escricar(f1.f,f1.c,f1.a,NO_INV);  /* esborra posicio anterior */

            f1.f = seg.f; f1.c = seg.c; f1.a = seg.a;  /* actualitza posicio */


            win_escricar(f1.f, f1.c, buffer[0], INVERS);

            if (f1.a == '0') resultat = 1; /* ha capturat menjacocos */
        }
        //signalS(id_sem);
        if (resultat) {
            *fi2=resultat;
            break;  // Sale del bucle si ha capturado al personaje principal
        }

        win_retard(retardFantasma*retard);  // Retardo


    }while(outModeC!=-1 && *fi1==0 && *fi2==0);
}
```

d. Per últim, el fantasma que ha vist al menjacocos, seguirà executant el codi principal (mode normal) però tindrà en compte si s'ha de printar en mode invers o en mode no invers (dependrà del si ha vist o no al menjacocos). Aquest fet es controlarà amb una variable booleana.



```
if(caza){
    win_escricar(f1.f, f1.c, buffer[0], INVERS);
}else{
    win_escricar(f1.f, f1.c, buffer[0], NO_INV);
}
```

- Fins aquí es on hem pogut arribar en aquesta fase. No hem pogut implementar el concepte de travessar les parets ni el fet de que quan un fantasma es trobi en mode cacera, segueixi al menjacocos.
- També cal dir, que aquesta fase, alhora de fer la part del pas de missatges per tal d'avisar als demes fantasmes que ha vist al menjacocos, a vegades no es printen en mode invers o no invers a temps. Sospitem que és pel fet de la sincronització i que a vegades, al deixar de estar en mode cacera o entra, executa el codi principal i no arriba a executar el codi del mode cacera (o a l'inrevés), ja que quan els fantasmes entren en mode cacera, es mouen el doble de posicions (no es mouen en una unitat).

3. Implementació

3.1 Cocos1

```
/*****************************************************************************/
/*                                                                           */
/*                                                                  cocos1.c */
/*                                                                           */
/*                                                                           */
/*    Programa inicial d'exemple per a les practiques 2.1 i 2.2 de FSO.      */
/*    Es tracta del joc del menjacocos: es dibuixa un laberint amb una       */
/*    serie de punts (cocos), els quals han de ser "menjats" pel menja-      */
/*    cocos. Aquest menjacocos es representara amb el caracter '0', i el     */
/*    moura l'usuari amb les tecles 'w' (adalt), 's' (abaix), 'd' (dreta)    */
/*    i 'a' (esquerra). Simultaniament hi haura un conjunt de fantasmes,     */
/*    representats per numeros de l'1 al 9, que intentaran capturar al       */
/*    menjacocos. En la primera versio del programa, nomes hi ha un fan-     */
/*    tasma.                                                                 */
/*                                                                           */
/*    Evidentment, es tracta de menjar tots els punts abans que algun fan-   */
/*    tasma atrapi al menjacocos.                                            */
/*                                                                           */
/*  Arguments del programa:                                                  */
/*                                                                           */
/*    per controlar la posicio de tots els elements del joc, cal indicar     */
/*    el nom d'un fitxer de text que contindra la seguent informacio:        */
/*                                                  n_fil1 n_col fit_tauler creq */
/*                                                  mc_f mc_c mc_d mc_r      */
/*                                                  f1_f f1_c f1_d f1_r      */
```

```
/*                                                                      */
/*    on 'n_fil1', 'n_col' son les dimensions del taulell de joc, mes una   */
/*    fila pels missatges de text a l'ultima linia. "fit_tauler" es el nom  */
/*    d'un fitxer de text que contindra el dibuix del laberint, amb num. de */
/*    files igual a 'n_fil1'-1 i num. de columnes igual a 'n_col'. Dins     */
/*    d'aquest fitxer, hi hauran caracter ASCCII que es representaran en    */
/*    pantalla tal qual, excepte el caracters iguals a 'creq', que es visua-*/
/*    litzaran invertits per representar la paret.                          */
/*    Els parametres 'mc_f', 'mc_c' indiquen la posicio inicial de fila i   */
/*    columna del menjacocos, aixi com la direccio inicial de moviment      */
/*    (0 -> amunt, 1-> esquerra, 2-> avall, 3-> dreta). Els parametres      */
/*    'f1_f', 'f1_c' i 'f1_d' corresponen a la mateixa informacio per al    */
/*    fantasma 1. El programa verifica que la primera posicio del menja-    */
/*    cocos o del fantasma no coincideixi amb un bloc de paret del laberint.*/
/*                                                  'mc_r' 'f1_r' son dos reals que multipliquen
el retard del moviment.  */
/*    A mes, es podra afegir un segon argument opcional per indicar el      */
/*    retard de moviment del menjacocos i dels fantasmes (en ms);           */
/*    el valor per defecte d'aquest parametre es 100 (1 decima de segon).   */
/*                                                                      */
/*  Compilar i executar:
*/
/*    El programa invoca les funcions definides a 'winsuport.h', les        */
/*    quals proporcionen una interficie senzilla per crear una finestra     */
/*    de text on es poden escriure caracters en posicions especifiques de   */
/*    la pantalla (basada en CURSES); per tant, el programa necessita ser   */
/*    compilat amb la llibreria 'curses':                                   */
/*                                                                      */
/*                                                  $ gcc -Wall cocos0.c winsuport.o -o cocos0 -
lcurses                                                                   */
```

```
/*                                                    $ ./cocos0 fit_param [retard]
                                                                                */

/*
                                                                                */

/*  Codis de retorn:
*/

/*    El programa retorna algun dels seguents codis al SO:                      */

/*                                                    0  ==>  funcionament normal
                                                                                */

/*                                                    1  ==>  numero d'arguments incorrecte
                                                                                */

/*                                                    2  ==>  fitxer de configuracio no accessible
                                                                                */

/*                                                    3  ==>  dimensions del taulell incorrectes
                                                                                */

/*                                                    4 ==>  parametres del menjacocos incorrectes
                                                                                */

/*                                                    5     ==>     parametres     d'algun     fantasma
incorrectes                                                                     */

/*                                                    6  ==>  no s'ha pogut crear el camp de joc
                                                                                */

/*                                                    7  ==>  no s'ha pogut inicialitzar el joc
                                                                                */

/*****************************************************************************/



#include <stdio.h>                                  /* incloure definicions de funcions estandard
*/

#include <stdlib.h>                                 /* per exit() */

#include <unistd.h>                                 /* per getpid() */

#include "winsuport.h"                              /* incloure definicions de funcions propies */

#include <pthread.h>

#include <stdbool.h>

#include <string.h>
```

```
#define MAX_THREADS                    10
#define MAXFANTASMES 9
#define MIN_FIL 7                      /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80


                                       /*    definir    estructures
d'informacio */
typedef struct {                       /* per un objecte (menjacocos o fantasma)
*/
                                       int f;                  /*    posicio
actual: fila */
                                       int c;                  /*    posicio
actual: columna */
                                       int d;                  /*    direccio
actual: [0..3] */
   float r;          /* per indicar un retard relati */
                                       char a;                 /*    caracter
anterior en pos. actual */
} objecte;



/* variables globals */
int n_fil1, n_col;                     /* dimensions del camp de joc */
char tauler[70];                       /* nom del fitxer amb el laberint de joc */
char c_req;                                    /* caracter de pared del laberint
*/


objecte mc;                            /* informacio del menjacocos */
objecte f1[9];                                 /* informacio del fantasma 1 */


int df[] = {-1, 0, 1, 0};              /* moviments de les 4 direccions possibles */
int dc[] = {0, -1, 0, 1};              /* dalt, esquerra, baix, dreta */
```

```
int cocos;                                  /* numero restant de cocos per menjar */
int retard;                                 /* valor del retard de moviment, en mil.lisegons */
int fi1=0, fi2=0;
int nFantasmes=-1; //número de fantasma per guardar-ho en el vector a partir de la lectura del fitxer de joc
int numFantasmes=0; //número de fantasmes existents
int idFantasmes=1; //identificador thread fantasma
pthread_t tid[MAX_THREADS];

char infoTemps[64];
char infoCocos[12];
int minuts=0,segons=0,hora=0;


/* funcio per realitzar la carrega dels parametres de joc emmagatzemats */
/* dins d'un fitxer de text, el nom del qual es passa per referencia a  */
/* 'nom_fit'; si es detecta algun problema, la funcio avorta l'execucio */
/* enviant un missatge per la sortida d'error i retornant el codi per-  */
/* tinent al SO (segons comentaris al principi del programa).           */
void carrega_parametres(const char *nom_fit)
{
  FILE *fit;
  int ret = 0; //variable per a indicar errors
  fit = fopen(nom_fit,"rt");                  /* intenta obrir fitxer */
  if (fit == NULL)
  {                                           fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom_fit);
                                              exit(2);
  }

  if (!feof(fit)) fscanf(fit,"%d %d %s %c\n",&n_fil1,&n_col,tauler,&c_req);
```

```
 else {
                                                    fprintf(stderr,"Falten parametres al fitxer
\'%s\'\n",nom_fit);
                                                    fclose(fit);
                                                    exit(2);
                                                    }
 if ((n_fil1 < MIN_FIL) || (n_fil1 > MAX_FIL) ||
                                                    (n_col < MIN_COL) || (n_col > MAX_COL))
 {
                                                    fprintf(stderr,"Error: dimensions del camp de
joc incorrectes:\n");
                                                    fprintf(stderr,"\t%d =< n_fil1 (%d) =<
%d\n",MIN_FIL,n_fil1,MAX_FIL);
                                                    fprintf(stderr,"\t%d =< n_col (%d) =<
%d\n",MIN_COL,n_col,MAX_COL);
                                                    fclose(fit);
                                                    exit(3);
 }

 if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);
 else {
                                                    fprintf(stderr,"Falten parametres al fitxer
\'%s\'\n",nom_fit);
                                                    fclose(fit);
                                                    exit(2);
                                                    }
 if ((mc.f < 1) || (mc.f > n_fil1-3) ||
                                                    (mc.c < 1) || (mc.c > n_col-2) ||
                                                    (mc.d < 0) || (mc.d > 3))
 {
                                                    fprintf(stderr,"Error: parametres menjacocos
incorrectes:\n");
                                                    fprintf(stderr,"\t1 =< mc.f (%d) =< n_fil1-3
(%d)\n",mc.f,(n_fil1-3));
```

```
                                                    fprintf(stderr,"\t1 =< mc.c (%d) =< n_col-2
(%d)\n",mc.c,(n_col-2));
                                                    fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);
                                                    fclose(fit);
                                                    exit(4);
 }

 int iteracio=0;
 while((ret==0) && (!feof(fit)))
                                                    {
  iteracio=1;
  nFantasmes++;

  fscanf(fit,"%d %d %d %f\n",&f1[nFantasmes].f,&f1[nFantasmes].c,&f1[nFantasmes].d,&f1[nFantasmes].r);


  if ((f1[nFantasmes].f < 1) || (f1[nFantasmes].f > n_fil1-3) ||
   (f1[nFantasmes].c < 1) || (f1[nFantasmes].c > n_col-2) ||
   (f1[nFantasmes].d < 0) || (f1[nFantasmes].d > 3))
  {
   ret = 1; //error
   fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");
   fprintf(stderr,"\t1 =< f1.f (%d) =< n_fil1-3 (%d)\n",f1[nFantasmes].f,(n_fil1-3));
   fprintf(stderr,"\t1 =< f1.c (%d) =< n_col-2 (%d)\n",f1[nFantasmes].c,(n_col-2));
   fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",f1[nFantasmes].d);
   fclose(fit);
   exit(5);
  }
  if(ret==0)
  {
   numFantasmes++;
  }
```

```
  }
  if (iteracio == 0)
  {
     fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
     fclose(fit);
     exit(2);
  }
  else if(ret == 0)
  {
    fclose(fit);
    printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",
                                                          TEC_AMUNT,    TEC_AVALL,    TEC_DRETA,
TEC_ESQUER);
    printf("prem una tecla per continuar:\n");
    getchar();
  }




}




/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
void inicialitza_joc(void)
{
  int r,i,j;
  char strin[12];
  int nBucleFantasma=0;
  int nMostraFantasma=nBucleFantasma;
```

```
 char buffer[10];
 r = win_carregatauler(tauler,n_fil1-1,n_col,c_req);
 if (r == 0)
 {
  mc.a = win_quincar(mc.f,mc.c);
  if (mc.a == c_req) r = -6;                                   /* error: menjacocos sobre pared */
  else
  {
   while((nBucleFantasma<numFantasmes))  //si hi ha algún error, s'abortarà amb l'inicialització del joc
   {
    f1[nBucleFantasma].a = win_quincar(f1[nBucleFantasma].f,f1[nBucleFantasma].c);
    if (f1[nBucleFantasma].a == c_req) r = -7;                 /* error: fantasma sobre pared */
    else
    {
     cocos = 0;                                                  /* compta el numero total de cocos
*/
     for (i=0; i<n_fil1-1; i++)
      for (j=0; j<n_col; j++)
       if (win_quincar(i,j)=='.') cocos++;

     win_escricar(mc.f,mc.c,'0',NO_INV);
     nMostraFantasma=nBucleFantasma+1;
     sprintf(buffer, "%d", nMostraFantasma);
     win_escricar(f1[nBucleFantasma].f,f1[nBucleFantasma].c,buffer[0], NO_INV);
    }
    nBucleFantasma++;
   }
   if (mc.a == '.') cocos--;                                   /* menja primer coco */

   sprintf(strin,"Cocos: %d", cocos); win_escristr(strin);
  }
 }
```

```
  if (r != 0)
  {                                                    win_fi();
                                                       fprintf(stderr,"Error: no s'ha pogut inicialitzar
el joc:\n");
                                                       switch (r)
                                                       { case -1: fprintf(stderr,"  nom de fitxer
erroni\n"); break;
                                                         case -2: fprintf(stderr," numero de columnes
d'alguna fila no coincideix amb l'amplada del tauler de joc\n"); break;
                                                         case -3: fprintf(stderr," numero de columnes
del laberint incorrecte\n"); break;
                                                         case -4: fprintf(stderr," numero de files del
laberint incorrecte\n"); break;
                                                         case -5: fprintf(stderr," finestra de camp de
joc no oberta\n"); break;
                                                         case -6: fprintf(stderr,"  posicio inicial del
menjacocos damunt la pared del laberint\n"); break;
                                                         case -7: fprintf(stderr,"  posicio inicial del
fantasma damunt la pared del laberint\n"); break;
                                                       }
                                                       exit(7);
  }
}




void * mou_fantasma(void * index)
{
  int ind = (intptr_t) index;
  objecte seg;
  int resultat;
  int k, vk, nd, vd[3];
  int retardFantasma;
  int nMostraFantasma=ind;
```

```
char buffer[10];

do {  // Bucle propio de la función
 resultat = 0; nd = 0;
 retardFantasma=f1[ind].r;  // Retardo en función del índice
 for (k=-1; k<=1; k++) {   /* provar direccio actual i dir. veines */
  vk = (f1[ind].d + k) % 4; /* direccio veina */
  if (vk < 0) vk += 4;  /* corregeix negatius */
  seg.f = f1[ind].f + df[vk]; /* calcular pSosicio en la nova dir.*/
  seg.c = f1[ind].c + dc[vk];
  seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */
  if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0')) {
   vd[nd] = vk;  /* memoritza com a direccio possible */
   nd++;
  }
 }
 if (nd == 0) {  /* si no pot continuar, */
  f1[ind].d = (f1[ind].d + 2) % 4;  /* canvia totalment de sentit */
 } else {
  if (nd == 1) {  /* si nomes pot en una direccio */
   f1[ind].d = vd[0];  /* li assigna aquesta */
  } else {  /* altrament */
   f1[ind].d = vd[rand() % nd];  /* segueix una dir. aleatoria */
  }

  seg.f = f1[ind].f + df[f1[ind].d];  /* calcular seguent posicio final */
  seg.c = f1[ind].c + dc[f1[ind].d];
  seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */
  win_retard(retardFantasma*retard);  // Retardo
  win_escricar(f1[ind].f,f1[ind].c,f1[ind].a,NO_INV);  /* esborra posicio anterior */
  f1[ind].f = seg.f; f1[ind].c = seg.c; f1[ind].a = seg.a;  /* actualitza posicio */
```

```
      nMostraFantasma=ind+1;
      sprintf(buffer, "%d", nMostraFantasma);
      win_escricar(f1[ind].f,f1[ind].c,buffer[0],NO_INV);  /* redibuixa fantasma */
      if (f1[ind].a == '0') resultat = 1; /* ha capturat menjacocos */
    }
    if (resultat) {
      fi2=resultat;

      break;  // Sale del bucle si ha capturado al personaje principal
    }

  } while (!fi1 && !fi2);

  return (NULL);
}




/* funcio per moure el menjacocos una posicio, en funcio de la direccio de   */
/* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots  */
/* els cocos, i 0 altrament */
void * mou_menjacocos(void * null)
{
  char strin[12];
  objecte seg;
  int tecla, resultat;
  int retardMenjacocos;

  do{
    retardMenjacocos=mc.r;
    resultat=0;
```

```
   tecla = win_gettec();
   if (tecla != 0)
   switch (tecla)                                   /* modificar direccio menjacocos segons
tecla */
   {
    case TEC_AMUNT:                                 mc.d = 0; break;
    case TEC_ESQUER:  mc.d = 1; break;
    case TEC_AVALL:                                 mc.d = 2; break;
    case TEC_DRETA:                                 mc.d = 3; break;
    case TEC_RETURN:  resultat = -1; break;
   }
   seg.f = mc.f + df[mc.d];                        /* calcular seguent posicio */
   seg.c = mc.c + dc[mc.d];
   seg.a = win_quincar(seg.f,seg.c);               /* calcular caracter seguent posicio */
   if ((seg.a == ' ') || (seg.a == '.'))
   {
    win_retard(retardMenjacocos*retard);  // retard
    win_escricar(mc.f,mc.c,' ',NO_INV);             /* esborra posicio anterior */
    mc.f = seg.f; mc.c = seg.c;                             /* actualitza posicio */
    win_escricar(mc.f,mc.c,'0',NO_INV);             /* redibuixa menjacocos */
    if (seg.a == '.')
    {
     cocos--;
     sprintf(strin,"Cocos: %d", cocos); win_escristr(strin);
     if (cocos == 0) resultat = 1; //guanya l'usuari (1)
    }
   }
   fi1=resultat;
  }while(fi1==0 && fi2==0);
  return (NULL);
}
```

```
void * cronometre(void * index)
{
                                                    bool acaba=false;
                                                    do{
                                                     if(segons==60){
                                                                segons=0;
                                                                minuts++;
                                                                if(minuts==60){
                                                                          minuts=0;
                                                                          hora++;
                                                                }
   }


                                                     //win_escristr(infoTemps);
                                                     win_retard(1000);
   segons++;
                                                     if(((fi1)==1)||((fi2)==1)){
                                                       acaba=true;
                                                       //fprintf(stderr, "fi1:%i  \t fi2:%i",fi1,fi2);
                                                      }
                                                    }while(acaba==false);
                                                    return (NULL);
}


void* info(void* nul)
{
                                                    char info[50];
  int auxCocos=0;
```

```
  int auxHores=0;
  int auxMinuts=0;
  int auxSegons=0;
                                                        while(!fi1 && !fi2)
                                                        {
     if((auxCocos!=cocos) || (auxHores!=hora) || (auxMinuts!=minuts) || (auxSegons!=segons)){
        info[0]='\0';
        sprintf(infoTemps,"Temps %i:%i:%i",hora,minuts,segons);
        sprintf(infoCocos,"Cocos: %d", cocos);

        strcat(info, infoTemps);
        strcat(info, " - ");
        strcat(info, infoCocos);

        win_escristr(info);
        auxCocos=cocos;
        auxHores=hora;
        auxMinuts=minuts;
        auxSegons=segons;
     }

  }
  return (NULL);
}




/* programa principal                                                    */
int main(int n_args, const char *ll_args[])
{
  int rc;                                                  /* variables locals */
```

```
  //int p; //retard fantasma
  //srand(getpid());                                  /* inicialitza numeros aleatoris */

  if ((n_args != 2) && (n_args !=3))
  {                                                 fprintf(stderr,"Comanda: cocos0 fit_param
[retard]\n");
                                                    exit(1);
  }
  carrega_parametres(ll_args[1]);

  if (n_args == 3) retard = atoi(ll_args[2]);
  else retard = 100;

  rc = win_ini(&n_fil1,&n_col,'+',INVERS);          /* intenta crear taulell */
  if (rc == 0)                                      /* si aconsegueix accedir a l'entorn CURSES
*/
  {
    segons=0;
                                                    minuts=0;
                                                    hora=0;
    inicialitza_joc();
    fi1 = 0; fi2 = 0;

                                                    /********** bucle principal del joc
**********/

    pthread_create(&tid[1], NULL, mou_menjacocos, (void *)(intptr_t)0);

    for (int i = 0; i < numFantasmes; i++) {
      pthread_create(&tid[i], NULL, mou_fantasma, (void *)(intptr_t)i);
    }
```

```
  pthread_create(&tid[10], NULL, cronometre, (void *)(intptr_t)0);

  pthread_create(&tid[10], NULL, info, (void *)(intptr_t)0);
  //p++; if ((p%2)==0)                                        /* ralentitza fantasma a 2*retard */




  do
  {
    win_retard(retard);
  }while(!fi1 && !fi2);



    for(int j = 0; j < MAX_THREADS; j++)                    /*finalizacion de threads*/
    {
      pthread_join(tid[j], NULL);
    }
    win_fi();
    if (fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");
    else { if (fi1) printf("Ha guanyat l'usuari!\n");
      else printf("Ha guanyat l'ordinador!\n"); }

}
else
{
  fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");
                                                              switch (rc)
  { case -1: fprintf(stderr,"camp de joc ja creat!\n");
      break;
```

```
    case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");
      break;
    case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");
      break;
    case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");
      break;
   }
                                                              exit(6);
 }
 return(0);
}
```

3. Implementació

3.2 Cocos2

```
/*****************************************************************************/
/*                                                                           */
/*                              cocos1.c                                     */
/*                                                                           */
/*    Programa inicial d'exemple per a les practiques 2.1 i 2.2 de FSO.      */
/*    Es tracta del joc del menjacocos: es dibuixa un laberint amb una       */
/*    serie de punts (cocos), els quals han de ser "menjats" pel menja-      */
/*    cocos. Aquest menjacocos es representara amb el caracter '0', i el     */
/*    moura l'usuari amb les tecles 'w' (adalt), 's' (abaix), 'd' (dreta)    */
/*    i 'a' (esquerra). Simultaniament hi haura un conjunt de fantasmes,     */
/*    representats per numeros de l'1 al 9, que intentaran capturar al       */
/*    menjacocos. En la primera versio del programa, nomes hi ha un fan-     */
/*    tasma.                                                                 */
/*    Evidentment, es tracta de menjar tots els punts abans que algun fan-   */
/*    tasma atrapi al menjacocos.                                            */
/*                                                                           */
/*  Arguments del programa:                                                  */
/*    per controlar la posicio de tots els elements del joc, cal indicar     */
/*    el nom d'un fitxer de text que contindra la seguent informacio:        */
/*                  n_fil1 n_col fit_tauler creq                             */
/*                  mc_f mc_c mc_d mc_r                                      */
/*                  f1_f f1_c f1_d f1_r                                      */
/*                                                                           */
/*    on 'n_fil1', 'n_col' son les dimensions del taulell de joc, mes una    */
/*    fila pels missatges de text a l'ultima linia. "fit_tauler" es el nom   */
/*    d'un fitxer de text que contindra el dibuix del laberint, amb num. de  */
/*    files igual a 'n_fil1'-1 i num. de columnes igual a 'n_col'. Dins       */
/*    d'aquest fitxer, hi hauran caracter ASCCII que es representaran en     */
/*    pantalla tal qual, excepte el caracters iguals a 'creq', que es visua-*/
/*    litzaran invertits per representar la paret.                           */
```

```
/*    Els parametres 'mc_f', 'mc_c' indiquen la posicio inicial de fila i   */
/*    columna del menjacocos, aixi com la direccio inicial de moviment      */
/*    (0 -> amunt, 1-> esquerra, 2-> avall, 3-> dreta). Els parametres       */
/*    'f1_f', 'f1_c' i 'f1_d' corresponen a la mateixa informacio per al    */
/*    fantasma 1. El programa verifica que la primera posicio del menja-    */
/*    cocos o del fantasma no coincideixi amb un bloc de paret del laberint.*/
/*          'mc_r' 'f1_r' son dos reals que multipliquen el retard del moviment.  */
/*    A mes, es podra afegir un segon argument opcional per indicar el      */
/*    retard de moviment del menjacocos i dels fantasmes (en ms);           */
/*    el valor per defecte d'aquest parametre es 100 (1 decima de segon).   */
/*                                                                          */
/*  Compilar i executar:                                                    */
/*    El programa invoca les funcions definides a 'winsuport.h', les        */
/*    quals proporcionen una interficie senzilla per crear una finestra     */
/*    de text on es poden escriure caracters en posicions especifiques de   */
/*    la pantalla (basada en CURSES); per tant, el programa necessita ser   */
/*    compilat amb la llibreria 'curses':                                   */
/*                                                                          */
/*          $ gcc -Wall cocos0.c winsuport.o -o cocos0 -lcurses             */
/*          $ ./cocos0 fit_param [retard]                                   */
/*                                                                          */
/*  Codis de retorn:                                                        */
/*    El programa retorna algun dels seguents codis al SO:                  */
/*        0  ==>  funcionament normal                                       */
/*        1  ==>  numero d'arguments incorrecte                             */
/*        2  ==>  fitxer de configuracio no accessible                      */
/*        3  ==>  dimensions del taulell incorrectes                        */
/*        4  ==>  parametres del menjacocos incorrectes                     */
/*        5  ==>  parametres d'algun fantasma incorrectes                   */
/*        6  ==>  no s'ha pogut crear el camp de joc                        */
/*        7  ==>  no s'ha pogut inicialitzar el joc                         */
/*************************************************************************/
```

```
#include <stdio.h>              /* incloure definicions de funcions estandard */
#include <stdlib.h>             /* per exit() */
#include <unistd.h>             /* per getpid() */
#include "winsuport.h"          /* incloure definicions de funcions propies */
#include <pthread.h>
#include <stdbool.h>
#include <string.h>

#define MAX_THREADS 10
#define MAXFANTASMES 9
#define MIN_FIL 7               /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80


                                /* definir estructures d'informacio */
typedef struct {            /* per un objecte (menjacocos o fantasma) */
        int f;                          /* posicio actual: fila */
        int c;                          /* posicio actual: columna */
        int d;                          /* direccio actual: [0..3] */
 float r;       /* per indicar un retard relati */
        char a;                         /* caracter anterior en pos. actual */
} objecte;


/* variables globals */
int n_fil1, n_col;              /* dimensions del camp de joc */
char tauler[70];            /* nom del fitxer amb el laberint de joc */
```

```
char c_req;                         /* caracter de pared del laberint */

objecte mc;                 /* informacio del menjacocos */
objecte f1[9];                      /* informacio del fantasma 1 */

int df[] = {-1, 0, 1, 0};     /* moviments de les 4 direccions possibles */
int dc[] = {0, -1, 0, 1};     /* dalt, esquerra, baix, dreta */

int cocos;                          /* numero restant de cocos per menjar */
int retard;                   /* valor del retard de moviment, en mil.lisegons */
int fi1=0, fi2=0;
int nFantasmes=-1; //número de fantasma per guardar-ho en el vector a partir de la lectura del fitxer de joc
int numFantasmes=0; //número de fantasmes existents
int idFantasmes=1; //identificador thread fantasma
pthread_t tid[MAX_THREADS];
pthread_mutex_t fantasmamutex= PTHREAD_MUTEX_INITIALIZER;   /* crea un sem. Global*/
char infoTemps[64];
char infoCocos[12];
int minuts=0,segons=0,hora=0;

/* funcio per realitzar la carrega dels parametres de joc emmagatzemats */
/* dins d'un fitxer de text, el nom del qual es passa per referencia a  */
/* 'nom_fit'; si es detecta algun problema, la funcio avorta l'execucio */
/* enviant un missatge per la sortida d'error i retornant el codi per-        */
/* tinent al SO (segons comentaris al principi del programa).                  */
void carrega_parametres(const char *nom_fit)
{
  FILE *fit;
  int ret = 0; //variable per a indicar errors
  fit = fopen(nom_fit,"rt");                    /* intenta obrir fitxer */
  if (fit == NULL)
  {         fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom_fit);
```

```
        exit(2);
}

if (!feof(fit)) fscanf(fit,"%d %d %s %c\n",&n_fil1,&n_col,tauler,&c_req);
else {
        fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
        fclose(fit);
        exit(2);
        }
if ((n_fil1 < MIN_FIL) || (n_fil1 > MAX_FIL) ||
        (n_col < MIN_COL) || (n_col > MAX_COL))
{
        fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");
        fprintf(stderr,"\t%d =< n_fil1 (%d) =< %d\n",MIN_FIL,n_fil1,MAX_FIL);
        fprintf(stderr,"\t%d =< n_col (%d) =< %d\n",MIN_COL,n_col,MAX_COL);
        fclose(fit);
        exit(3);
}

if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);
else {
        fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
        fclose(fit);
        exit(2);
        }
if ((mc.f < 1) || (mc.f > n_fil1-3) ||
        (mc.c < 1) || (mc.c > n_col-2) ||
        (mc.d < 0) || (mc.d > 3))
{
        fprintf(stderr,"Error: parametres menjacocos incorrectes:\n");
        fprintf(stderr,"\t1 =< mc.f (%d) =< n_fil1-3 (%d)\n",mc.f,(n_fil1-3));
        fprintf(stderr,"\t1 =< mc.c (%d) =< n_col-2 (%d)\n",mc.c,(n_col-2));
```

```
        fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);
        fclose(fit);
        exit(4);
}

int iteracio=0;
while((ret==0) && (!feof(fit)))
        {
 iteracio=1;
 nFantasmes++;

 fscanf(fit,"%d %d %d %f\n",&f1[nFantasmes].f,&f1[nFantasmes].c,&f1[nFantasmes].d,&f1[nFantasmes].r);


 if ((f1[nFantasmes].f < 1) || (f1[nFantasmes].f > n_fil1-3) ||
  (f1[nFantasmes].c < 1) || (f1[nFantasmes].c > n_col-2) ||
  (f1[nFantasmes].d < 0) || (f1[nFantasmes].d > 3))
 {
  ret = 1; //error
  fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");
  fprintf(stderr,"\t1 =< f1.f (%d) =< n_fil1-3 (%d)\n",f1[nFantasmes].f,(n_fil1-3));
  fprintf(stderr,"\t1 =< f1.c (%d) =< n_col-2 (%d)\n",f1[nFantasmes].c,(n_col-2));
  fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",f1[nFantasmes].d);
  fclose(fit);
  exit(5);
 }
 if(ret==0)
 {
  numFantasmes++;
 }
```

```
  }
  if (iteracio == 0)
  {
     fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
     fclose(fit);
     exit(2);
  }
  else if(ret == 0)
  {
    fclose(fit);
    printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",
                          TEC_AMUNT, TEC_AVALL, TEC_DRETA, TEC_ESQUER);
    printf("prem una tecla per continuar:\n");
    getchar();
  }




}




/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
void inicialitza_joc(void)
{
  int r,i,j;
  //char strin[12];
  int nBucleFantasma=0;
  int nMostraFantasma=nBucleFantasma;
  char buffer[10];
```

```
r = win_carregatauler(tauler,n_fil1-1,n_col,c_req);
if (r == 0)
{
  mc.a = win_quincar(mc.f,mc.c);
  if (mc.a == c_req) r = -6;                    /* error: menjacocos sobre pared */
  else
  {
    while((nBucleFantasma<numFantasmes))  //si hi ha algún error, s'abortarà amb l'inicialització del joc
    {
      f1[nBucleFantasma].a = win_quincar(f1[nBucleFantasma].f,f1[nBucleFantasma].c);
      if (f1[nBucleFantasma].a == c_req) r = -7;     /* error: fantasma sobre pared */
      else
      {
        cocos = 0;                          /* compta el numero total de cocos */
        for (i=0; i<n_fil1-1; i++)
          for (j=0; j<n_col; j++)
            if (win_quincar(i,j)=='.') cocos++;

        win_escricar(mc.f,mc.c,'0',NO_INV);
        nMostraFantasma=nBucleFantasma+1;
        sprintf(buffer, "%d", nMostraFantasma);
        win_escricar(f1[nBucleFantasma].f,f1[nBucleFantasma].c,buffer[0], NO_INV);
      }
      nBucleFantasma++;
    }
    if (mc.a == '.') cocos--;           /* menja primer coco */

    sprintf(infoCocos,"Cocos: %d", cocos); //win_escristr(strin);
  }
}

if (r != 0)
```

```
 {        win_fi();
          fprintf(stderr,"Error: no s'ha pogut inicialitzar el joc:\n");
          switch (r)
          { case -1: fprintf(stderr,"  nom de fitxer erroni\n"); break;
            case -2: fprintf(stderr,"  numero de columnes d'alguna fila no coincideix amb l'amplada del tauler de
joc\n"); break;
            case -3: fprintf(stderr,"  numero de columnes del laberint incorrecte\n"); break;
            case -4: fprintf(stderr,"  numero de files del laberint incorrecte\n"); break;
            case -5: fprintf(stderr,"  finestra de camp de joc no oberta\n"); break;
            case -6: fprintf(stderr,"  posicio inicial del menjacocos damunt la pared del laberint\n"); break;
            case -7: fprintf(stderr,"  posicio inicial del fantasma damunt la pared del laberint\n"); break;
          }
          exit(7);
 }
}



void * mou_fantasma(void * index)
{
  int ind = (intptr_t) index;
  objecte seg;
  int resultat;
  int k, vk, nd, vd[3];
  int retardFantasma;
  int nMostraFantasma=ind;
  char buffer[10];

  do {  // Bucle propio de la función
    resultat = 0; nd = 0;
    retardFantasma=f1[ind].r;  // Retardo en función del índice
    pthread_mutex_lock(&fantasmamutex);
```

```
for (k=-1; k<=1; k++) {   /* provar direccio actual i dir. veines */
  vk = (f1[ind].d + k) % 4; /* direccio veina */
  if (vk < 0) vk += 4;  /* corregeix negatius */
  seg.f = f1[ind].f + df[vk]; /* calcular posicio en la nova dir.*/
  seg.c = f1[ind].c + dc[vk];

  seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

  if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0')) {
    vd[nd] = vk;  /* memoritza com a direccio possible */
    nd++;
  }
}
if (nd == 0) {  /* si no pot continuar, */
  f1[ind].d = (f1[ind].d + 2) % 4;  /* canvia totalment de sentit */
} else {
  if (nd == 1) {  /* si nomes pot en una direccio */
    f1[ind].d = vd[0];  /* li assigna aquesta */
  } else {  /* altrament */
    f1[ind].d = vd[rand() % nd];  /* segueix una dir. aleatoria */
  }

  seg.f = f1[ind].f + df[f1[ind].d];  /* calcular seguent posicio final */
  seg.c = f1[ind].c + dc[f1[ind].d];

  seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

  win_escricar(f1[ind].f,f1[ind].c,f1[ind].a,NO_INV);  /* esborra posicio anterior */


  f1[ind].f = seg.f; f1[ind].c = seg.c; f1[ind].a = seg.a;  /* actualitza posicio */
```

```
      nMostraFantasma=ind+1;
      sprintf(buffer, "%d", nMostraFantasma);

      win_escricar(f1[ind].f,f1[ind].c,buffer[0],NO_INV);  /* redibuixa fantasma */


      if (f1[ind].a == '0') resultat = 1; /* ha capturat menjacocos */
    }

    pthread_mutex_unlock(&fantasmamutex);

    if (resultat) {
      fi2=resultat;

      break;  // Sale del bucle si ha capturado al personaje principal
    }
    win_retard(retardFantasma*retard);  // Retardo
  } while (!fi1 && !fi2);

  return (NULL);
}




/* funcio per moure el menjacocos una posicio, en funcio de la direccio de   */
/* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots  */
/* els cocos, i 0 altrament */
void * mou_menjacocos(void * null)
{

  objecte seg;
```

```
int tecla, resultat;
int retardMenjacocos;

do{
 retardMenjacocos=mc.r;
 resultat=0;
 tecla = win_gettec();
 if (tecla != 0)
 switch (tecla)              /* modificar direccio menjacocos segons tecla */
 {
  case TEC_AMUNT:      mc.d = 0; break;
  case TEC_ESQUER:  mc.d = 1; break;
  case TEC_AVALL:        mc.d = 2; break;
  case TEC_DRETA:       mc.d = 3; break;
  case TEC_RETURN:  resultat = -1; break;
 }
 seg.f = mc.f + df[mc.d];           /* calcular seguent posicio */
 seg.c = mc.c + dc[mc.d];
 seg.a = win_quincar(seg.f,seg.c);         /* calcular caracter seguent posicio */

 if ((seg.a == ' ') || (seg.a == '.') || (seg.a >= '1' && seg.a <= '9'))
 {
  pthread_mutex_lock(&fantasmamutex);
  win_escricar(mc.f,mc.c,' ',NO_INV);              /* esborra posicio anterior */
  mc.f = seg.f; mc.c = seg.c;                      /* actualitza posicio */
  win_escricar(mc.f,mc.c,'0',NO_INV);              /* redibuixa menjacocos */
  pthread_mutex_unlock(&fantasmamutex);
  if(seg.a >= '1' && seg.a <= '9'){
   fi2 = 1;
  }
  if (seg.a == '.')
  {
```

```
      cocos--;
        //win_escristr(strin);
      if (cocos == 0) resultat = 1; //guanya l'usuari (1)
     }
   }

   fi1=resultat;
    win_retard(retardMenjacocos*retard);  // retard

 }while(fi1==0 && fi2==0);
 return (NULL);
}



void * cronometre(void * index)
{

        bool acaba=false;
        do{
                if(segons==60){
                        segons=0;
                        minuts++;
                        if(minuts==60){
                                minuts=0;
                                hora++;
                        }
   }


         //win_escristr(infoTemps);
         win_retard(1000);
```

```
   segons++;
            if(((fi1)==1)||((fi2)==1)){
                        acaba=true;
                        //fprintf(stderr, "fi1:%i  \t fi2:%i",fi1,fi2);
            }
           }while(acaba==false);
           return (NULL);
}



void* info(void* nul)
{
           char info[50];
  int auxCocos=0;
  int auxHores=0;
  int auxMinuts=0;
  int auxSegons=0;
           while(!fi1 && !fi2)
           {
     if((auxCocos!=cocos) || (auxHores!=hora) || (auxMinuts!=minuts) || (auxSegons!=segons)){
        info[0]='\0';
        sprintf(infoTemps,"Temps %i:%i:%i",hora,minuts,segons);
        sprintf(infoCocos,"Cocos: %d", cocos);
        strcat(info, infoTemps);
        strcat(info, " - ");
        strcat(info, infoCocos);

        win_escristr(info);
        auxCocos=cocos;
        auxHores=hora;
        auxMinuts=minuts;
```

```
        auxSegons=segons;
    }


                        //win_retard(retard);
 }
 return (NULL);
}




/* programa principal                                   */
int main(int n_args, const char *ll_args[])
{
 int rc;            /* variables locals */

 //int p; //retard fantasma
 //srand(getpid());                  /* inicialitza numeros aleatoris */

 if ((n_args != 2) && (n_args !=3))
 {        fprintf(stderr,"Comanda: cocos0 fit_param [retard]\n");
          exit(1);
 }
 carrega_parametres(ll_args[1]);

 if (n_args == 3) retard = atoi(ll_args[2]);
 else retard = 100;

 rc = win_ini(&n_fil1,&n_col,'+',INVERS);          /* intenta crear taulell */
 if (rc == 0)                  /* si aconsegueix accedir a l'entorn CURSES */
 {
```

```
inicialitza_joc();
fi1 = 0; fi2 = 0;

                        /********** bucle principal del joc **********/

pthread_create(&tid[0], NULL, mou_menjacocos, (void *)(intptr_t)0);

for (int i = 0; i < numFantasmes; i++) {
  pthread_create(&tid[i+1], NULL, mou_fantasma, (void *)(intptr_t)i);
}

pthread_create(&tid[10], NULL, cronometre, (void *)(intptr_t)0);

pthread_create(&tid[10], NULL, info, (void *)(intptr_t)0);
//p++; if ((p%2)==0)              /* ralentitza fantasma a 2*retard */




do
{
  win_retard(100);
}while(!fi1 && !fi2);



  for(int j = 0; j < MAX_THREADS; j++)                    /*finalizacion de threads*/
  {
    pthread_join(tid[j], NULL);
  }
```

```
    pthread_mutex_destroy(&fantasmamutex);
    win_fi();
    if (fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");
    else { if (fi1) printf("Ha guanyat l'usuari!\n");
      else printf("Ha guanyat l'ordinador!\n"); }

  }
  else
  {
   fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");
            switch (rc)
   { case -1: fprintf(stderr,"camp de joc ja creat!\n");
      break;
    case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");
      break;
    case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");
      break;
    case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");
      break;
   }
            exit(6);
  }
  return(0);
}
```

3. Implementació

3.3 Cocos3

Fitxer cocos3 (pare)

```
#include <stdint.h>                /* intptr_t for 64bits machines */
#include <pthread.h>
#include <stdio.h>                 /* incloure definicions de funcions estandard */
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "winsuport2.h"
#include "memoria.h"    /* incloure definicions de funcions propies */

#define MAX_THREADS 10
#define MAXFANTASMES 9
#define MIN_FIL 7                  /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80


                                   /* definir estructures d'informacio */
typedef struct {            /* per un objecte (menjacocos o fantasma) */
        int f;                             /* posicio actual: fila */
        int c;                             /* posicio actual: columna */
        int d;                             /* direccio actual: [0..3] */
 float r;          /* per indicar un retard relati */
        char a;                            /* caracter anterior en pos. actual */
```

```
} objecte;


/* variables globals */
int n_fil1, n_col;                  /* dimensions del camp de joc */
char tauler[70];            /* nom del fitxer amb el laberint de joc */
char c_req;                         /* caracter de pared del laberint */

objecte mc;                 /* informacio del menjacocos */
objecte f1[9];                      /* informacio del fantasma 1 */

int df[] = {-1, 0, 1, 0};      /* moviments de les 4 direccions possibles */
int dc[] = {0, -1, 0, 1};      /* dalt, esquerra, baix, dreta */

int cocos;                          /* numero restant de cocos per menjar */
int retard;                   /* valor del retard de moviment, en mil.lisegons */
int id_fi1, *fi1, id_fi2, *fi2;
int nFantasmes=-1; //número de fantasma per guardar-ho en el vector a partir de la lectura del fitxer de joc
int numFantasmes=0; //número de fantasmes existents
int idFantasmes=1; //identificador thread fantasma
pthread_t tid[MAX_THREADS];

//IDENTIFICADORS DE MEMORIA COMPARTIDA
void *p_tauler;
int id_tauler;                          /*Variable que guarda el tamany del tauler*/

int ncocos = 0,id_ncocos;
pid_t tpid[MAXFANTASMES];               /*Creacio de la taula de processos*/

char infoTemps[64];
char infoCocos[12];
int minuts=0,segons=0,hora=0;
```

```
/* funcio per realitzar la carrega dels parametres de joc emmagatzemats */
/* dins d'un fitxer de text, el nom del qual es passa per referencia a  */
/* 'nom_fit'; si es detecta algun problema, la funcio avorta l'execucio */
/* enviant un missatge per la sortida d'error i retornant el codi per-  */
/* tinent al SO (segons comentaris al principi del programa).           */
void carrega_parametres(const char *nom_fit)
{

  FILE *fit;
  int ret = 0; //variable per a indicar errors
  fit = fopen(nom_fit,"rt");                    /* intenta obrir fitxer */
  if (fit == NULL)
  {       fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom_fit);
          exit(2);
  }

  if (!feof(fit)) fscanf(fit,"%d %d %s %c\n",&n_fil1,&n_col,tauler,&c_req);
  else {
          fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
          fclose(fit);
          exit(2);
          }
  if ((n_fil1 < MIN_FIL) || (n_fil1 > MAX_FIL) ||
          (n_col < MIN_COL) || (n_col > MAX_COL))
  {
          fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");
          fprintf(stderr,"\t%d =< n_fil1 (%d) =< %d\n",MIN_FIL,n_fil1,MAX_FIL);
          fprintf(stderr,"\t%d =< n_col (%d) =< %d\n",MIN_COL,n_col,MAX_COL);
          fclose(fit);
```

```
        exit(3);
}

if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);
else {
        fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
        fclose(fit);
        exit(2);
        }
if ((mc.f < 1) || (mc.f > n_fil1-3) ||
        (mc.c < 1) || (mc.c > n_col-2) ||
        (mc.d < 0) || (mc.d > 3))
{
        fprintf(stderr,"Error: parametres menjacocos incorrectes:\n");
        fprintf(stderr,"\t1 =< mc.f (%d) =< n_fil1-3 (%d)\n",mc.f,(n_fil1-3));
        fprintf(stderr,"\t1 =< mc.c (%d) =< n_col-2 (%d)\n",mc.c,(n_col-2));
        fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);
        fclose(fit);
        exit(4);
}

int iteracio=0;
while((ret==0) && (!feof(fit)))
        {
 iteracio=1;
 nFantasmes++;

 fscanf(fit,"%d %d %d %f\n",&f1[nFantasmes].f,&f1[nFantasmes].c,&f1[nFantasmes].d,&f1[nFantasmes].r);


 if ((f1[nFantasmes].f < 1) || (f1[nFantasmes].f > n_fil1-3) ||
   (f1[nFantasmes].c < 1) || (f1[nFantasmes].c > n_col-2) ||
```

```
    (f1[nFantasmes].d < 0) || (f1[nFantasmes].d > 3))
  {
    ret = 1; //error
    fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");
    fprintf(stderr,"\t1 =< f1.f (%d) =< n_fil1-3 (%d)\n",f1[nFantasmes].f,(n_fil1-3));
    fprintf(stderr,"\t1 =< f1.c (%d) =< n_col-2 (%d)\n",f1[nFantasmes].c,(n_col-2));
    fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",f1[nFantasmes].d);
    fclose(fit);
    exit(5);
  }
  if(ret==0)
  {
    numFantasmes++;
  }



}

if (iteracio == 0)
{
    fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
    fclose(fit);
    exit(2);
}
else if(ret == 0)
{
  fclose(fit);
  printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",
                        TEC_AMUNT, TEC_AVALL, TEC_DRETA, TEC_ESQUER);
  printf("prem una tecla per continuar:\n");
  getchar();
}
```

```
}



/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
void inicialitza_joc(void)
{
 int r,i,j;
 char strin[12];
 int nBucleFantasma=0;
 int nMostraFantasma=nBucleFantasma;
 char buffer[10];
 r = win_carregatauler(tauler,n_fil1-1,n_col,c_req);
 win_update();
 if (r == 0)
 {
  mc.a = win_quincar(mc.f,mc.c);
  if (mc.a == c_req) r = -6;                    /* error: menjacocos sobre pared */
  else
  {
   while((nBucleFantasma<numFantasmes))  //si hi ha algún error, s'abortarà amb l'inicialització del joc
   {
    f1[nBucleFantasma].a = win_quincar(f1[nBucleFantasma].f,f1[nBucleFantasma].c);
    if (f1[nBucleFantasma].a == c_req) r = -7;     /* error: fantasma sobre pared */
    else
    {
     cocos = 0;                          /* compta el numero total de cocos */
```

```
        for (i=0; i<n_fil1-1; i++)
          for (j=0; j<n_col; j++)
            if (win_quincar(i,j)=='.') cocos++;

        win_escricar(mc.f,mc.c,'0',NO_INV);
        win_update();
        nMostraFantasma=nBucleFantasma+1;
        sprintf(buffer, "%d", nMostraFantasma);
        win_escricar(f1[nBucleFantasma].f,f1[nBucleFantasma].c,buffer[0], NO_INV);
        win_update();
       }
       nBucleFantasma++;
     }
     // win_escricar(f1[0].f,f1[0].c,'1', NO_INV);
     // win_escricar(f1[1].f,f1[1].c,'2', NO_INV);
     // win_escricar(f1[2].f,f1[2].c,'3', NO_INV);
     // win_escricar(f1[3].f,f1[3].c,'4', NO_INV);
     if (mc.a == '.') cocos--;           /* menja primer coco */

     sprintf(strin,"Cocos: %d", cocos);
     win_escristr(strin);
   }
  }

  if (r != 0)
  {         win_fi();
            fprintf(stderr,"Error: no s'ha pogut inicialitzar el joc:\n");
            switch (r)
            { case -1: fprintf(stderr,"  nom de fitxer erroni\n"); break;
              case -2: fprintf(stderr,"  numero de columnes d'alguna fila no coincideix amb l'amplada del tauler de
joc\n"); break;
              case -3: fprintf(stderr,"  numero de columnes del laberint incorrecte\n"); break;
              case -4: fprintf(stderr,"  numero de files del laberint incorrecte\n"); break;
```

```
            case -5: fprintf(stderr,"  finestra de camp de joc no oberta\n"); break;
            case -6: fprintf(stderr,"  posicio inicial del menjacocos damunt la pared del laberint\n"); break;
            case -7: fprintf(stderr,"  posicio inicial del fantasma damunt la pared del laberint\n"); break;
           }
           exit(7);
  }
}


/* funcio per moure el menjacocos una posicio, en funcio de la direccio de   */
/* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots  */
/* els cocos, i 0 altrament */
void * mou_menjacocos(void * null)
{
  //char strin[12];
  objecte seg;
  int tecla, resultat;
  int retardMenjacocos;

  do{
    retardMenjacocos=mc.r;
    resultat=0;
    tecla = win_gettec();
    if (tecla != 0)
    switch (tecla)              /* modificar direccio menjacocos segons tecla */
    {
      case TEC_AMUNT:      mc.d = 0; break;
      case TEC_ESQUER:  mc.d = 1; break;
      case TEC_AVALL:        mc.d = 2; break;
      case TEC_DRETA:       mc.d = 3; break;
      case TEC_RETURN:  resultat = -1; break;
    }
```

```
  seg.f = mc.f + df[mc.d];          /* calcular seguent posicio */
  seg.c = mc.c + dc[mc.d];
  seg.a = win_quincar(seg.f,seg.c);        /* calcular caracter seguent posicio */

  if ((seg.a == ' ') || (seg.a == '.') || (seg.a >= '1' && seg.a <= '9'))
  {
    win_escricar(mc.f,mc.c,' ',NO_INV);            /* esborra posicio anterior */
    mc.f = seg.f; mc.c = seg.c;                    /* actualitza posicio */
    win_escricar(mc.f,mc.c,'0',NO_INV);            /* redibuixa menjacocos */

    if(seg.a >= '1' && seg.a <= '9'){
      *fi2 = 1;
    }

    if (seg.a == '.')
    {
      cocos--;
      //sprintf(strin,"Cocos: %d", cocos);
      //win_escristr(strin);

      if (cocos == 0) resultat = 1; //guanya l'usuari (1)
    }
  }

  *fi1=resultat;
   win_retard(retardMenjacocos*retard);  // retard

 }while(*fi1==0 && *fi2==0);
 return (NULL);
}
```

```
void * cronometre(void * index)
{

        bool acaba=false;
        do{
                if(segons==60){
                        segons=0;
                        minuts++;
                        if(minuts==60){
                                minuts=0;
                                hora++;
                        }
  }


         //win_escristr(infoTemps);
         win_retard(1000);
  segons++;
         if(((*fi1)==1)||((*fi2)==1)){
                 acaba=true;
                 fprintf(stderr, "fi1:%i  \t fi2:%i",*fi1,*fi2);
         }
        }while(acaba==false);
        return (NULL);
}



void* info(void* nul)
{
        char info[50];
```

```
 int auxCocos=0;
 int auxHores=0;
 int auxMinuts=0;
 int auxSegons=0;
          while(!*fi1 && !*fi2)
          {
    if((auxCocos!=cocos) || (auxHores!=hora) || (auxMinuts!=minuts) || (auxSegons!=segons)){
       info[0]='\0';
       sprintf(infoTemps,"Temps %i:%i:%i",hora,minuts,segons);
       sprintf(infoCocos,"Cocos: %d", cocos);
       strcat(info, infoTemps);
       strcat(info, " - ");
       strcat(info, infoCocos);

       win_escristr(info);
       auxCocos=cocos;
       auxHores=hora;
       auxMinuts=minuts;
       auxSegons=segons;
    }


                          //win_retard(retard);
 }
 return NULL;
}
```

```
/* programa principal                          */
```

```
int main(int n_args, const char *ll_args[])
{
 int rc;              /* variables locals */
 int numProcesos = 0;
 //int p; //retard fantasma
 //srand(getpid());                    /* inicialitza numeros aleatoris */

 if ((n_args != 2) && (n_args !=3))
 {        fprintf(stderr,"Comanda: cocos0 fit_param [retard]\n");
          exit(1);
 }
 carrega_parametres(ll_args[1]);

 if (n_args == 3) retard = atoi(ll_args[2]);
 else retard = 100;

 rc = win_ini(&n_fil1,&n_col,'+',INVERS);          /* intenta crear taulell */
 if (rc != 0)                    /* si aconsegueix accedir a l'entorn CURSES */
 {
  //TAULER
  id_tauler = ini_mem(rc);   //crear zona de memòria compartida amb els bytes que ocupa el mapa del joc
          p_tauler = map_mem(id_tauler); //estableix un identificador
  //set_filcol(p_tauler,n_fil1, n_col);
  win_set(p_tauler,n_fil1,n_col);

  inicialitza_joc();

          char id_taulerx[10];
           char n_fil1x[10];
          char n_colx[10];
  sprintf(id_taulerx, "%d", id_tauler);
                    sprintf(n_fil1x, "%d", n_fil1);
```

```
                sprintf(n_colx, "%d", n_col);


//FI1 i FI2//
char id_fi1c[10];
char id_fi2c[10];
id_fi1 = ini_mem(sizeof(int));  /* crear zona mem. compartida */
        fi1 = map_mem(id_fi1);          /* obtenir adres. de mem. compartida */
        *fi1 = 0;                       /* inicialitza variable compartida */
        sprintf(id_fi1c,"%i",id_fi1);

        id_fi2 = ini_mem(sizeof(int));  /* crear zona mem. compartida */
        fi2 = map_mem(id_fi2);          /* obtenir adres. de mem. compartida */
        *fi2 = 0;
        sprintf(id_fi2c,"%i",id_fi2);


//RETARD
char retardx[10];
sprintf(retardx, "%d", retard);

                        /********** bucle principal del joc **********/

pthread_create(&tid[0], NULL, mou_menjacocos, (void *)(intptr_t)0);

pthread_create(&tid[10], NULL, cronometre, (void *)(intptr_t)0);

pthread_create(&tid[10], NULL, info, (void *)(intptr_t)0);


//FANTASMES//
char f1_fx[10];
```

```
          char f1_cx[10];
         char f1_dx[10];
  char f1_rx[10];
  char id_fantasmax[10];


  for(int i=0; i<numFantasmes; i++)
                  {
                        tpid[numProcesos]=fork();   /* crea un nou proces (fill)*/
                        if(tpid[numProcesos]==(pid_t)0)
                        {
    sprintf(id_fantasmax, "%d", i);
                          sprintf(f1_fx, "%d", f1[i].f);
                          sprintf(f1_cx, "%d", f1[i].c);
                          sprintf(f1_dx, "%d", f1[i].d);
                          sprintf(f1_rx, "%f", f1[i].r);

                              execlp("./fantasma3", "fantasma3", id_taulerx, n_fil1x, n_colx, id_fantasmax,
f1_fx, f1_cx, f1_dx, f1_rx, id_fi1c, id_fi2c, retardx,(char *)0);
                              fprintf(stderr, "error: no puc executar el process fill");
     exit(0);
                        }
                        else if (tpid[numProcesos] > 0) numProcesos++;    /* branca del pare */
                  }


   do
  {
    win_retard(retard);
    win_update();

  }while(!*fi1 && !*fi2);
```

```
 for(int j = 0; j < MAX_THREADS; j++)                    /*finalizacion de threads*/
        {
                pthread_join(tid[j], NULL);
        }


        for (int i = 0; i < numProcesos; i++){
                waitpid(tpid[i], NULL, 0);      /* espera finalitzacio d'un fill */
        }



 win_fi();

 elim_mem(id_tauler);     /* elimina zones de memoria compartida */
          elim_mem(id_fi1);
          elim_mem(id_fi2);

 if (*fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");
 else { if (*fi1) printf("Ha guanyat l'usuari!\n");
   else printf("Ha guanyat l'ordinador!\n"); }

}
else
{
 fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");
          switch (rc)
 { case -1: fprintf(stderr,"camp de joc ja creat!\n");
    break;
```

```
    case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");
      break;
    case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");
      break;
    case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");
      break;
   }
            exit(6);
 }
 return(0);
}
```

Fitxer fantasma3 (fill)

```
#include <stdint.h>                /* intptr_t for 64bits machines */
#include <pthread.h>
#include <stdio.h>                 /* incloure definicions de funcions estandard */
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "winsuport2.h"
#include "memoria.h"    /* incloure definicions de funcions propies */

#define MAX_THREADS 10
#define MAXFANTASMES 9
#define MIN_FIL 7                  /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80



                                   /* definir estructures d'informacio */
typedef struct {            /* per un objecte (menjacocos o fantasma) */
        int f;                             /* posicio actual: fila */
        int c;                             /* posicio actual: columna */
        int d;                             /* direccio actual: [0..3] */
 float r;         /* per indicar un retard relati */
        char a;                            /* caracter anterior en pos. actual */
} objecte;
```

```
/* variables globals */
int n_fil1, n_col;                    /* dimensions del camp de joc */
char tauler[70];            /* nom del fitxer amb el laberint de joc */
char c_req;                           /* caracter de pared del laberint */

objecte mc;                 /* informacio del menjacocos */
objecte f1;                           /* informacio del fantasma 1 */

int df[] = {-1, 0, 1, 0};     /* moviments de les 4 direccions possibles */
int dc[] = {0, -1, 0, 1};     /* dalt, esquerra, baix, dreta */

int cocos;                            /* numero restant de cocos per menjar */
int retard;                    /* valor del retard de moviment, en mil.lisegons */
int id_fi1, *fi1, id_fi2, *fi2;

pthread_t tid[MAX_THREADS];

//IDENTIFICADORS DE MEMORIA COMPARTIDA
void* p_tauler;
int id_tauler;                              /*Variable que guarda el tamany del tauler*/
char id_fi1c[10],id_fi2c[10];
pid_t tpid[MAXFANTASMES];                /*Creacio de la taula de processos*/
int id_fantasma;

int main(int n_args, char *ll_args[])
{
   //TAULER//
          id_tauler = atoi(ll_args[1]);     /* obtenir dimensions del camp de joc */
   n_fil1 = atoi(ll_args[2]);
          n_col = atoi(ll_args[3]);
```

```
        //FANTASMA//
id_fantasma=atoi(ll_args[4]);
        f1.f=atoi(ll_args[5]);
        f1.c=atoi(ll_args[6]);
        f1.d=atoi(ll_args[7]);
        f1.r=atoi(ll_args[8]);

//FI1 y FI2//
id_fi1=atoi(ll_args[9]);
id_fi2=atoi(ll_args[10]);

//Retard//
retard=atoi(ll_args[11]);




p_tauler = map_mem(id_tauler);  /* obtenir adres. de mem. compartida */
        if (p_tauler == (int*) -1)
        {
        fprintf(stderr, "proces (%d): error en identificador de memoria \n",(int)getpid());
        exit(0);
        }
        win_set(p_tauler,n_fil1,n_col);  /* crea acces a finestra oberta pel proces pare */



        fi1 = map_mem(id_fi1);          /* obtenir adres. de mem. compartida */
        fi2 = map_mem(id_fi2);          /* obtenir adres. de mem. compartida */
```

```
objecte seg;
int resultat;
int k, vk, nd, vd[3];
int retardFantasma;
int nMostraFantasma=id_fantasma;
char buffer[10];

f1.a='.';

nMostraFantasma=id_fantasma+1;
sprintf(buffer, "%d", nMostraFantasma);


do {  // Bucle propio de la función
   win_set(p_tauler,n_fil1,n_col);
   resultat = 0; nd = 0;
   retardFantasma=f1.r;  // Retardo en función del índice

   for (k=-1; k<=1; k++) {   /* provar direccio actual i dir. veines */
      vk = (f1.d + k) % 4; /* direccio veina */
      if (vk < 0) vk += 4;  /* corregeix negatius */
      seg.f = f1.f + df[vk]; /* calcular posicio en la nova dir.*/
      seg.c = f1.c + dc[vk];

      seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

      if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0')) {
         vd[nd] = vk;  /* memoritza com a direccio possible */
         nd++;
```

```
    }
}
if (nd == 0) {  /* si no pot continuar, */
    f1.d = (f1.d + 2) % 4;  /* canvia totalment de sentit */
} else {
    if (nd == 1) {  /* si nomes pot en una direccio */
        f1.d = vd[0];  /* li assigna aquesta */
    } else {  /* altrament */
        f1.d = vd[rand() % nd];  /* segueix una dir. aleatoria */
    }

    seg.f = f1.f + df[f1.d];  /* calcular seguent posicio final */
    seg.c = f1.c + dc[f1.d];

    seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

    win_escricar(f1.f,f1.c,f1.a,NO_INV);  /* esborra posicio anterior */


    f1.f = seg.f; f1.c = seg.c; f1.a = seg.a;  /* actualitza posicio */


    win_escricar(f1.f,f1.c,buffer[0],NO_INV);  /* redibuixa fantasma */


    if (f1.a == '0') resultat = 1; /* ha capturat menjacocos */
}



if (resultat) {
    *fi2=resultat;
```

```
            break;  // Sale del bucle si ha capturado al personaje principal
        }
        win_retard(retardFantasma*retard);  // Retardo
    } while (*fi1==0 && *fi2==0);

}
```

3. Implementació

3.4 Cocos4

Fitxer cocos4 (pare)

```
#include <stdint.h>                /* intptr_t for 64bits machines */
#include <pthread.h>
#include <stdio.h>                 /* incloure definicions de funcions estandard */
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "winsuport2.h"
#include "memoria.h"    /* incloure definicions de funcions propies */
#include "semafor.h"
#include "missatge.h"
#include <signal.h>



#define MAX_THREADS 3
#define MAXFANTASMES 9
#define MIN_FIL 7                  /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80


                                   /* definir estructures d'informacio */
typedef struct {            /* per un objecte (menjacocos o fantasma) */
```

```
        int f;                          /* posicio actual: fila */
        int c;                          /* posicio actual: columna */
        int d;                          /* direccio actual: [0..3] */
 float r;       /* per indicar un retard relati */
        char a;                         /* caracter anterior en pos. actual */
} objecte;


/* variables globals */
int n_fil1, n_col;              /* dimensions del camp de joc */
char tauler[70];            /* nom del fitxer amb el laberint de joc */
char c_req;                     /* caracter de pared del laberint */

objecte mc;             /* informacio del menjacocos */
objecte f1[9];                  /* informacio del fantasma 1 */

int df[] = {-1, 0, 1, 0};   /* moviments de les 4 direccions possibles */
int dc[] = {0, -1, 0, 1};   /* dalt, esquerra, baix, dreta */

int cocos;                      /* numero restant de cocos per menjar */
int retard;                 /* valor del retard de moviment, en mil.lisegons */
int id_fi1, *fi1, id_fi2, *fi2;
int nFantasmes=-1; //número de fantasma per guardar-ho en el vector a partir de la lectura del fitxer de joc
int numFantasmes=0; //número de fantasmes existents
int idFantasmes=1; //identificador thread fantasma
pthread_t tid[MAX_THREADS];
int numProcesos = 0;
char missatgeE[100];

//TAULER//
char id_taulerx[10];
char n_fil1x[10];
```

```
char n_colx[10];


//FI1 i FI2//
char id_fi1c[10];
char id_fi2c[10];

//RETARD
char retardx[10];


//IDENTIFICADORS DE MEMORIA COMPARTIDA
void *p_tauler;
int id_tauler;                          /*Variable que guarda el tamany del tauler*/

int ncocos = 0,id_ncocos;
pid_t tpid[MAXFANTASMES];               /*Creacio de la taula de processos*/

char infoTemps[64];
char infoCocos[12];
int minuts=0,segons=0,hora=0;


//NÚMERO DE XOCS (FASE 4)
int xocs=0;
char auxXocs[2];

//SEMÀFORS
int id_sem;
int id_bustias[9];
char a4[20], a5[20];
```

```
char id_mis1[25], id_mis2[25], id_mis3[25], id_mis4[25], id_mis5[25], id_mis6[25], id_mis7[25], id_mis8[25],
id_mis9[25];
char numProcesosx[25];
int *nfActius;
int id_nfActius;
char nfActiusx[25];

/* funcio per realitzar la carrega dels parametres de joc emmagatzemats */
/* dins d'un fitxer de text, el nom del qual es passa per referencia a  */
/* 'nom_fit'; si es detecta algun problema, la funcio avorta l'execucio */
/* enviant un missatge per la sortida d'error i retornant el codi per-  */
/* tinent al SO (segons comentaris al principi del programa).           */
void carrega_parametres(const char *nom_fit)
{

 FILE *fit;
 int ret = 0; //variable per a indicar errors
 fit = fopen(nom_fit,"rt");                    /* intenta obrir fitxer */
 if (fit == NULL)
 {        fprintf(stderr,"No s'ha pogut obrir el fitxer \'%s\'\n",nom_fit);
          exit(2);
 }

 if (!feof(fit)) fscanf(fit,"%d %d %s %c\n",&n_fil1,&n_col,tauler,&c_req);
 else {
          fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
          fclose(fit);
          exit(2);
          }
 if ((n_fil1 < MIN_FIL) || (n_fil1 > MAX_FIL) ||
          (n_col < MIN_COL) || (n_col > MAX_COL))
 {
          fprintf(stderr,"Error: dimensions del camp de joc incorrectes:\n");
```

```
        fprintf(stderr,"\t%d =< n_fil1 (%d) =< %d\n",MIN_FIL,n_fil1,MAX_FIL);
        fprintf(stderr,"\t%d =< n_col (%d) =< %d\n",MIN_COL,n_col,MAX_COL);
        fclose(fit);
        exit(3);
}

if (!feof(fit)) fscanf(fit,"%d %d %d %f\n",&mc.f,&mc.c,&mc.d,&mc.r);
else {
        fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
        fclose(fit);
        exit(2);
        }
if ((mc.f < 1) || (mc.f > n_fil1-3) ||
        (mc.c < 1) || (mc.c > n_col-2) ||
        (mc.d < 0) || (mc.d > 3))
{
        fprintf(stderr,"Error: parametres menjacocos incorrectes:\n");
        fprintf(stderr,"\t1 =< mc.f (%d) =< n_fil1-3 (%d)\n",mc.f,(n_fil1-3));
        fprintf(stderr,"\t1 =< mc.c (%d) =< n_col-2 (%d)\n",mc.c,(n_col-2));
        fprintf(stderr,"\t0 =< mc.d (%d) =< 3\n",mc.d);
        fclose(fit);
        exit(4);
}

int iteracio=0;
while((ret==0) && (!feof(fit)))
        {
  iteracio=1;
  nFantasmes++;

  fscanf(fit,"%d %d %d %f\n",&f1[nFantasmes].f,&f1[nFantasmes].c,&f1[nFantasmes].d,&f1[nFantasmes].r);
```

```
  if ((f1[nFantasmes].f < 1) || (f1[nFantasmes].f > n_fil1-3) ||
    (f1[nFantasmes].c < 1) || (f1[nFantasmes].c > n_col-2) ||
    (f1[nFantasmes].d < 0) || (f1[nFantasmes].d > 3))
  {
    ret = 1; //error
    fprintf(stderr,"Error: parametres fantasma 1 incorrectes:\n");
    fprintf(stderr,"\t1 =< f1.f (%d) =< n_fil1-3 (%d)\n",f1[nFantasmes].f,(n_fil1-3));
    fprintf(stderr,"\t1 =< f1.c (%d) =< n_col-2 (%d)\n",f1[nFantasmes].c,(n_col-2));
    fprintf(stderr,"\t0 =< f1.d (%d) =< 3\n",f1[nFantasmes].d);
    fclose(fit);
    exit(5);
  }
  if(ret==0)
  {
    numFantasmes++;
  }


}



if (iteracio == 0)
{
    fprintf(stderr,"Falten parametres al fitxer \'%s\'\n",nom_fit);
    fclose(fit);
    exit(2);
}
else if(ret == 0)
{
  fclose(fit);
```

```
   printf("Joc del MenjaCocos\n\tTecles: \'%c\', \'%c\', \'%c\', \'%c\', RETURN-> sortir\n",
                        TEC_AMUNT, TEC_AVALL, TEC_DRETA, TEC_ESQUER);
   printf("prem una tecla per continuar:\n");
   getchar();
 }




}



/* funcio per inicialitar les variables i visualitzar l'estat inicial del joc */
void inicialitza_joc(void)
{
  int r,i,j;
  char strin[12];
  int nBucleFantasma=0;
  //int nMostraFantasma=nBucleFantasma;
  //char buffer[10];
  r = win_carregatauler(tauler,n_fil1-1,n_col,c_req);
  win_update();
  if (r == 0)
  {
    mc.a = win_quincar(mc.f,mc.c);
    if (mc.a == c_req) r = -6;                  /* error: menjacocos sobre pared */
    else
    {
      while((nBucleFantasma<numFantasmes))  //si hi ha algún error, s'abortarà amb l'inicialització del joc
      {
```

```
      f1[nBucleFantasma].a = win_quincar(f1[nBucleFantasma].f,f1[nBucleFantasma].c);
      if (f1[nBucleFantasma].a == c_req) r = -7;     /* error: fantasma sobre pared */
      else
      {
        cocos = 0;                      /* compta el numero total de cocos */
        for (i=0; i<n_fil1-1; i++)
          for (j=0; j<n_col; j++)
            if (win_quincar(i,j)=='.') cocos++;

        win_escricar(mc.f,mc.c,'0',NO_INV);
        win_update();
      }
      nBucleFantasma++;
    }

    if (mc.a == '.') cocos--;           /* menja primer coco */

    sprintf(strin,"Cocos: %d", cocos);
    win_escristr(strin);
   }
  }

  if (r != 0)
  {         win_fi();
            fprintf(stderr,"Error: no s'ha pogut inicialitzar el joc:\n");
            switch (r)
            { case -1: fprintf(stderr,"  nom de fitxer erroni\n"); break;
              case -2: fprintf(stderr,"  numero de columnes d'alguna fila no coincideix amb l'amplada del tauler de
joc\n"); break;
              case -3: fprintf(stderr,"  numero de columnes del laberint incorrecte\n"); break;
              case -4: fprintf(stderr,"  numero de files del laberint incorrecte\n"); break;
              case -5: fprintf(stderr,"  finestra de camp de joc no oberta\n"); break;
              case -6: fprintf(stderr,"  posicio inicial del menjacocos damunt la pared del laberint\n"); break;
```

```
            case -7: fprintf(stderr,"  posicio inicial del fantasma damunt la pared del laberint\n"); break;
          }
          exit(7);
 }
}


/* funcio per moure el menjacocos una posicio, en funcio de la direccio de   */
/* moviment actual; retorna -1 si s'ha premut RETURN, 1 si s'ha menjat tots  */
/* els cocos, i 0 altrament */
void * mou_menjacocos(void * null)
{
 //char strin[12];
 objecte seg;
 int tecla, resultat;
 int retardMenjacocos;


 do{
  retardMenjacocos=mc.r;
  resultat=0;
  tecla = win_gettec();
  if (tecla != 0)
  switch (tecla)              /* modificar direccio menjacocos segons tecla */
  {
    case TEC_AMUNT:      mc.d = 0; break;
    case TEC_ESQUER:  mc.d = 1; break;
    case TEC_AVALL:        mc.d = 2; break;
    case TEC_DRETA:       mc.d = 3; break;
    case TEC_RETURN:  resultat = -1; break;
  }
  seg.f = mc.f + df[mc.d];           /* calcular seguent posicio */
```

```
seg.c = mc.c + dc[mc.d];
seg.a = win_quincar(seg.f,seg.c);        /* calcular caracter seguent posicio */
if ((seg.a == ' ') || (seg.a == '.') || (seg.a >= '1' && seg.a <= '9'))
{

 win_escricar(mc.f,mc.c,' ',NO_INV);            /* esborra posicio anterior */
 mc.f = seg.f; mc.c = seg.c;                    /* actualitza posicio */
 win_escricar(mc.f,mc.c,'0',NO_INV);            /* redibuixa menjacocos */
 if (seg.a == '.')
 {
  cocos--;
  //sprintf(strin,"Cocos: %d", cocos);
  //win_escristr(strin);

  if (cocos == 0) resultat = 1; //guanya l'usuari (1)
 }
 if(seg.a >= '1' && seg.a <= '9'){
  *fi2 = 1;
 }
}
else if (xocs<=2)
        {
                        xocs++;
        }
if ((xocs==2) && (numProcesos<numFantasmes))
{
 tpid[numProcesos]=fork();
                        if(tpid[numProcesos]==(pid_t)0)
                        {
  sprintf(id_fi1c,"%i",id_fi1);
  sprintf(id_fi2c,"%i",id_fi2);
  sprintf(numProcesosx,"%i",numProcesos);
```

```
	//sprintf(a4, "%i", id_bustias[numProcesos]);
	sprintf(missatgeE, "%d;%d;%d;%f", f1[numProcesos].f, f1[numProcesos].c,f1[numProcesos].d,f1[numProcesos].r);
	sendM(id_bustias[numProcesos],missatgeE,100);
	//*nfActius++;  /* inicialitza variable compartida */
	*nfActius=*nfActius+1;

	execlp("./fantasma4", "fantasma4", id_taulerx, n_fil1x, n_colx, id_fi1c, id_fi2c, retardx, a5, numProcesosx, id_mis1, id_mis2, id_mis3, id_mis4, id_mis5, id_mis6, id_mis7, id_mis8, id_mis9, nfActiusx, (char *)0);

	exit(0);
				}
				else if (tpid[numProcesos] > 0) numProcesos++;


			xocs=0;
   }

   *fi1=resultat;
    win_retard(retardMenjacocos*retard);  // retard

 }while(*fi1==0 && *fi2==0);
 return (NULL);
}


void * cronometre(void * index)
{

	bool acaba=false;
	do{
		if(segons==60){
```

```
                        segons=0;
                        minuts++;
                        if(minuts==60){
                                minuts=0;
                                hora++;
                        }
   }


            //win_escristr(infoTemps);
            win_retard(1000);
   segons++;
            if(((*fi1)==1)||((*fi2)==1)){
                    acaba=true;
                    //fprintf(stderr, "fi1:%i  \t fi2:%i",*fi1,*fi2);
            }
           }while(acaba==false);
  return (NULL);

}



void* info(void* nul)
{
            char info[50];
  int auxCocos=0;
  int auxHores=0;
  int auxMinuts=0;
  int auxSegons=0;
            while(!*fi1 && !*fi2)
            {
```

```
    if((auxCocos!=cocos) || (auxHores!=hora) || (auxMinuts!=minuts) || (auxSegons!=segons)){
       info[0]='\0';
       sprintf(infoTemps,"Temps %i:%i:%i",hora,minuts,segons);
       sprintf(infoCocos,"Cocos: %d", cocos);
       strcat(info, infoTemps);
       strcat(info, " - ");
       strcat(info, infoCocos);

       win_escristr(info);
       auxCocos=cocos;
       auxHores=hora;
       auxMinuts=minuts;
       auxSegons=segons;
    }


                              //win_retard(retard);
 }
 return NULL;
}




/* programa principal                                    */
int main(int n_args, const char *ll_args[])
{
 int rc;              /* variables locals */
 id_sem = ini_sem(1);    /* crear semafor IPC inicialment obert */
```

```
sprintf(a5,"%i",id_sem);    /* convertir identificador sem. en string */


if ((n_args != 2) && (n_args !=3))
{       fprintf(stderr,"Comanda: cocos0 fit_param [retard]\n");
        exit(1);
}
carrega_parametres(ll_args[1]);

if (n_args == 3) retard = atoi(ll_args[2]);
else retard = 100;

rc = win_ini(&n_fil1,&n_col,'+',INVERS);          /* intenta crear taulell */
if (rc != 0)                    /* si aconsegueix accedir a l'entorn CURSES */
{
 //TAULER
 id_tauler = ini_mem(rc);   //crear zona de memòria compartida amb els bytes que ocupa el mapa del joc
        p_tauler = map_mem(id_tauler); //estableix un identificador
 //set_filcol(p_tauler,n_fil1, n_col);
 win_set(p_tauler,n_fil1,n_col);

 inicialitza_joc();


 sprintf(id_taulerx, "%d", id_tauler);
                sprintf(n_fil1x, "%d", n_fil1);
                sprintf(n_colx, "%d", n_col);



 id_fi1 = ini_mem(sizeof(int));  /* crear zona mem. compartida */
        fi1 = map_mem(id_fi1);          /* obtenir adres. de mem. compartida */
```

```
        *fi1 = 0;                       /* inicialitza variable compartida */
        sprintf(id_fi1c,"%i",id_fi1);

        id_fi2 = ini_mem(sizeof(int));  /* crear zona mem. compartida */
        fi2 = map_mem(id_fi2);          /* obtenir adres. de mem. compartida */
        *fi2 = 0;
        sprintf(id_fi2c,"%i",id_fi2);


sprintf(retardx, "%d", retard);


id_nfActius = ini_mem(sizeof(int));     /* crear zona mem. compartida */
        nfActius = map_mem(id_nfActius);            /* obtenir adres. de mem. compartida */
        *nfActius = 1;                          /* inicialitza variable compartida */
sprintf(nfActiusx, "%d", id_nfActius);

for (int i = 0; i < 9 ; i++)
        {
                id_bustias[i] = ini_mis();          /* crear una bustia IPC per cada fantasma*/
                //fprintf(stderr, "\nPrimer: %i", id_mis[i]);
        }

sprintf (id_mis1, "%i", id_bustias[0]);
sprintf (id_mis2, "%i", id_bustias[1]);
sprintf (id_mis3, "%i", id_bustias[2]);
sprintf (id_mis4, "%i", id_bustias[3]);
sprintf (id_mis5, "%i", id_bustias[4]);
sprintf (id_mis6, "%i", id_bustias[5]);
sprintf (id_mis7, "%i", id_bustias[6]);
sprintf (id_mis8, "%i", id_bustias[7]);
sprintf (id_mis9, "%i", id_bustias[8]);
```

```
                    /********** bucle principal del joc **********/

  pthread_create(&tid[0], NULL, mou_menjacocos, (void *)(intptr_t)0);

  pthread_create(&tid[1], NULL, cronometre, (void *)(intptr_t)0);

  pthread_create(&tid[2], NULL, info, (void *)(intptr_t)0);


  //pthread_create(&tid[1], NULL, cronometre, (void *)(intptr_t)0);

  //FANTASMES//



                    tpid[numProcesos]=fork();   /* crea un nou proces (fill)*/
                    if(tpid[numProcesos]==(pid_t)0)
                    {


    //id_bustias[numProcesos] = ini_mis();    /* crear bustia IPC */
    //sprintf(a4,"%i",id_bustias[numProcesos]);  /* convertir identif. bustia en string */
    sprintf(numProcesosx,"%i",numProcesos);


    sprintf(missatgeE,                "%d;%d;%d;%f",                f1[numProcesos].f,
f1[numProcesos].c,f1[numProcesos].d,f1[numProcesos].r);
    sendM(id_bustias[numProcesos],missatgeE,100);


                        execlp("./fantasma4", "fantasma4", id_taulerx, n_fil1x, n_colx, id_fi1c,
id_fi2c, retardx, a5, numProcesosx, id_mis1, id_mis2, id_mis3, id_mis4, id_mis5, id_mis6, id_mis7, id_mis8,
id_mis9, nfActiusx,(char *)0);
    exit(0);
                    }
```

```
                        else if (tpid[numProcesos] > 0) numProcesos++;




do
{
  win_retard(retard);
  win_update();

}while(!*fi1 && !*fi2);




for(int j = 0; j < 1; j++)                          /*finalizacion de threads*/
        {
                pthread_join(tid[j], NULL);
        }


        for (int i = 0; i <= numProcesos; i++){
                waitpid(tpid[i], NULL, 0);      /* espera finalitzacio d'un fill */
        }




win_fi();

elim_mem(id_tauler);    /* elimina zones de memoria compartida */
          elim_mem(id_fi1);
          elim_mem(id_fi2);
elim_sem(id_sem);
```

```
  elim_mem(id_nfActius);

  if (*fi1 == -1) printf("S'ha aturat el joc amb tecla RETURN!\n");
  else { if (*fi1) printf("Ha guanyat l'usuari!\n");
    else printf("Ha guanyat l'ordinador!\n"); }



}
else
{
  fprintf(stderr,"Error: no s'ha pogut crear el taulell:\n");
            switch (rc)
  { case -1: fprintf(stderr,"camp de joc ja creat!\n");
      break;
    case -2: fprintf(stderr,"no s'ha pogut inicialitzar l'entorn de curses!\n");
      break;
    case -3: fprintf(stderr,"les mides del camp demanades son massa grans!\n");
      break;
    case -4: fprintf(stderr,"no s'ha pogut crear la finestra!\n");
      break;
  }
            exit(6);
}
return(0);
```

Fitxer fantasma4 (fill)

```
#include <stdint.h>                /* intptr_t for 64bits machines */
#include <pthread.h>
#include <stdio.h>                 /* incloure definicions de funcions estandard */
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "winsuport2.h"
#include "memoria.h"    /* incloure definicions de funcions propies */
#include "semafor.h"
#include "missatge.h"
#include <signal.h>

#define MAX_THREADS 10
#define MAXFANTASMES 9
#define MIN_FIL 7                  /* definir limits de variables globals */
#define MAX_FIL 25
#define MIN_COL 10
#define MAX_COL 80


                                   /* definir estructures d'informacio */
typedef struct {            /* per un objecte (menjacocos o fantasma) */
        int f;                              /* posicio actual: fila */
        int c;                              /* posicio actual: columna */
        int d;                              /* direccio actual: [0..3] */
 float r;         /* per indicar un retard relati */
```

```
        char a;                         /* caracter anterior en pos. actual */
} objecte;


/* variables globals */
int n_fil1, n_col;              /* dimensions del camp de joc */
char tauler[70];        /* nom del fitxer amb el laberint de joc */
char c_req;                     /* caracter de pared del laberint */

objecte mc;             /* informacio del menjacocos */
objecte f1;                     /* informacio del fantasma 1 */

int df[] = {-1, 0, 1, 0};   /* moviments de les 4 direccions possibles */
int dc[] = {0, -1, 0, 1};   /* dalt, esquerra, baix, dreta */

int cocos;                      /* numero restant de cocos per menjar */
int retard;                 /* valor del retard de moviment, en mil.lisegons */
int id_fi1, *fi1, id_fi2, *fi2;
int id_nfActius;
int *nfActius;
pthread_t tid[MAX_THREADS];

//IDENTIFICADORS DE MEMORIA COMPARTIDA
void* p_tauler;
int id_tauler;                      /*Variable que guarda el tamany del tauler*/
char id_fi1c[10],id_fi2c[10];
pid_t tpid[MAXFANTASMES];           /*Creacio de la taula de processos*/
int id_fantasma;
int id_bustia[MAXFANTASMES];
char a4[20];
char *token;
int id_sem;
```

```
bool caza;
char missatgeE[100];
char missatgeR[100];
bool cazaTodos=false;
int outModeC=0;
pthread_t tid[MAX_THREADS];



void modeCaceria(){

   objecte seg;
   int resultat;
   int k, vk, nd, vd[3];
   int retardFantasma;
   char buffer[10];
   int nMostraFantasma=id_fantasma+1;
   sprintf(buffer, "%d", nMostraFantasma);

   do{
      resultat = 0; nd = 0;
      retardFantasma=f1.r;  // Retardo en función del índice



      //waitS(id_sem);
      for (k=-1; k<=1; k++) {   /* provar direccio actual i dir. veines */
         vk = (f1.d + k) % 4; /* direccio veina */
         if (vk < 0) vk += 4;  /* corregeix negatius */
         seg.f = f1.f + df[vk]; /* calcular posicio en la nova dir.*/
         seg.c = f1.c + dc[vk];

         seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */
```

```
    if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0')) {
        vd[nd] = vk;  /* memoritza com a direccio possible */
        nd++;
    }

}
if (nd == 0) {  /* si no pot continuar, */
    f1.d = (f1.d + 2) % 4;  /* canvia totalment de sentit */
} else {
    if (nd == 1) {  /* si nomes pot en una direccio */
        f1.d = vd[0];  /* li assigna aquesta */
    } else {  /* altrament */
        f1.d = vd[rand() % nd];  /* segueix una dir. aleatoria */
    }

    seg.f = f1.f + df[f1.d];  /* calcular seguent posicio final */
    seg.c = f1.c + dc[f1.d];

    seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */


    win_escricar(f1.f,f1.c,f1.a,NO_INV);  /* esborra posicio anterior */

    f1.f = seg.f; f1.c = seg.c; f1.a = seg.a;  /* actualitza posicio */


    win_escricar(f1.f, f1.c, buffer[0], INVERS);

    if (f1.a == '0') resultat = 1; /* ha capturat menjacocos */
}
//signalS(id_sem);
```

```
      if (resultat) {
         *fi2=resultat;
         break;  // Sale del bucle si ha capturado al personaje principal
      }

      win_retard(retardFantasma*retard);  // Retardo


   }while(outModeC!=-1 && *fi1==0 && *fi2==0);

}




void* info(void* nul){
   do{
      receiveM(id_bustia[id_fantasma],missatgeR);
      int valor = atoi(missatgeR); // Convierte el mensaje a un entero

      if (valor >= 0 && valor <= 8){
         outModeC=valor;
         modeCaceria();
      }else{
         outModeC=valor;
      }

   }while(*fi1==0 && *fi2==0);
   return NULL;

}
```

```
int main(int n_args, char *ll_args[])
{
   //TAULER//
         id_tauler = atoi(ll_args[1]);     /* obtenir dimensions del camp de joc */
   n_fil1 = atoi(ll_args[2]);
         n_col = atoi(ll_args[3]);


   //FI1 y FI2//
   id_fi1=atoi(ll_args[4]);
   id_fi2=atoi(ll_args[5]);

   //Retard//
   retard=atoi(ll_args[6]);

   //SEMAFOR//
   id_sem = atoi(ll_args[7]);

   //NUM_PROCES//
   id_fantasma = atoi(ll_args[8]);

   //BUSTIA//
   for (int i = 0; i < 9; i++)
         {
               id_bustia[i] = atoi(ll_args[i + 9]); // Carreguem els ids de les busties en un array de caracters
         }
```

```
id_nfActius= atoi(ll_args[18]);
nfActius=map_mem(id_nfActius);



receiveM(id_bustia[id_fantasma],a4);

token = strtok(a4, ";");

// Guardar cada salida en las variables del objeto f1
f1.f = atoi(token);

token = strtok(NULL, ";");
f1.c = atof(token);

token = strtok(NULL, ";");
f1.d = atof(token);

token = strtok(NULL, ";");
f1.r = atof(token);



p_tauler = map_mem(id_tauler);  /* obtenir adres. de mem. compartida */
        if (p_tauler == (int*) -1)
        {
        fprintf(stderr, "proces (%d): error en identificador de memoria \n",(int)getpid());
        exit(0);
        }
        win_set(p_tauler,n_fil1,n_col);   /* crea acces a finestra oberta pel proces pare */
```

```
        fi1 = map_mem(id_fi1);          /* obtenir adres. de mem. compartida */
        fi2 = map_mem(id_fi2);          /* obtenir adres. de mem. compartida */

pthread_create(&tid[0], NULL, info, (void *)(intptr_t)0);

objecte aux;
objecte seg;
int resultat;
int k, vk, nd, vd[3];
int retardFantasma;
int nMostraFantasma;
char buffer[10];
nMostraFantasma=id_fantasma;


f1.a='.';

nMostraFantasma=id_fantasma+1;
sprintf(buffer, "%d", nMostraFantasma);
caza=false;
bool acabaBucle=false;

do {  // Bucle propio de la función
   win_set(p_tauler,n_fil1,n_col);
   resultat = 0; nd = 0;
   retardFantasma=f1.r;  // Retardo en función del índice


   //waitS(id_sem);
   for (k=-1; k<=1; k++) {   /* provar direccio actual i dir. veines */
      vk = (f1.d + k) % 4; /* direccio veina */
```

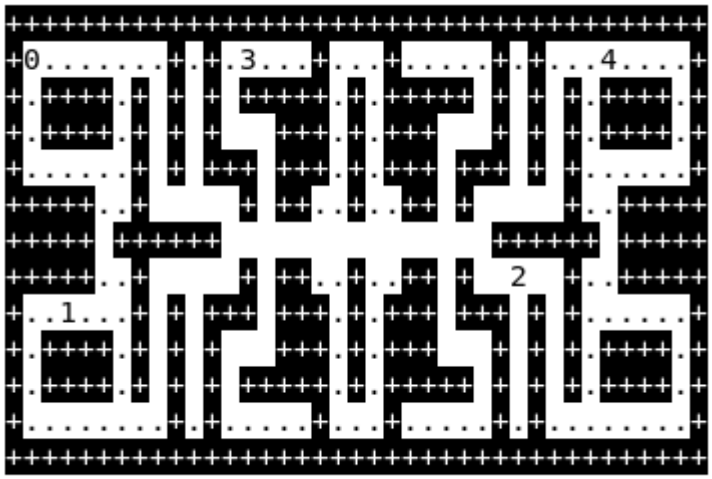
```
    if (vk < 0) vk += 4;  /* corregeix negatius */
    seg.f = f1.f + df[vk]; /* calcular posicio en la nova dir.*/
    seg.c = f1.c + dc[vk];

    seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

    if ((seg.a==' ') || (seg.a=='.') || (seg.a=='0')) {
       vd[nd] = vk;  /* memoritza com a direccio possible */
       nd++;
    }

}
if (nd == 0) {  /* si no pot continuar, */
    f1.d = (f1.d + 2) % 4;  /* canvia totalment de sentit */
} else {
    if (nd == 1) {  /* si nomes pot en una direccio */
       f1.d = vd[0];  /* li assigna aquesta */
    } else {  /* altrament */
       f1.d = vd[rand() % nd];  /* segueix una dir. aleatoria */
    }

    seg.f = f1.f + df[f1.d];  /* calcular seguent posicio final */
    seg.c = f1.c + dc[f1.d];

    seg.a = win_quincar(seg.f,seg.c); /* calcular caracter seguent posicio */

    //prismatics
    aux.f=seg.f;
    aux.c=seg.c;
    aux.a=seg.a;

    acabaBucle=false;
```

```
//caza=false;
while (!acabaBucle) {
   aux.f = df[f1.d]+aux.f;
   aux.c = dc[f1.d]+aux.c;
   aux.a = win_quincar(aux.f, aux.c);
   if ((aux.a == '+') || (aux.a >= '1' && aux.a <= '9') || (aux.a == '0')) {
      acabaBucle=true;
      if(aux.a == '0'){
         caza = true;
         sprintf(missatgeE, "%d", id_fantasma);
         for(int i =0; i<*nfActius;i++){
            if(i!=id_fantasma){
               sendM(id_bustia[i],missatgeE,25);
            }

         }

      }else{
         outModeC=-1;
         caza=false;
         sprintf(missatgeE, "%d", outModeC);
         for(int i =0; i<*nfActius;i++){
            if(i!=id_fantasma){
               sendM(id_bustia[i],missatgeE,25);
            }

         }
      }

   }

}
```

```
        win_escricar(f1.f,f1.c,f1.a,NO_INV);  /* esborra posicio anterior */

        f1.f = seg.f; f1.c = seg.c; f1.a = seg.a;  /* actualitza posicio */

        if(caza){
           win_escricar(f1.f, f1.c, buffer[0], INVERS);
        }else{
           win_escricar(f1.f, f1.c, buffer[0], NO_INV);
        }

        if (f1.a == '0') resultat = 1; /* ha capturat menjacocos */
      }
      //signalS(id_sem);
      if (resultat) {
         *fi2=resultat;
         break;  // Sale del bucle si ha capturado al personaje principal
      }

      win_retard(retardFantasma*retard);  // Retardo
   } while (*fi1==0 && *fi2==0);

}
```

4. Joc de proves

4.1 Cocos1



Podem observar el correcte funcionament de diferents factors, com per exemple el comptador de temps i de cocos, el correcte funcionament del printeig del mapa

La comanda que hem executat és:

```
milax@casa:~/Escriptori/Practica21/Practica21$ ./cocos1 joc22.txt 200
Joc del MenjaCocos
        Tecles: 'w', 's', 'd', 'a', RETURN-> sortir
prem una tecla per continuar:
```

En aquesta informació dins del joc22.txt on ens mostra la informació dels 4 fantasmes que s'han de mostrar.

```
1 1 0 1.0
6 13 3 2.0
6 24 1 1.5
6 14 3 2.0
6 25 1 1.5
```

(Cal indicar que en imatges es difícil de mostrar el correcte funcionament de figures que han de estar en moviment)

També podem comprovar que si en la comanda li passem un nombre petit aquest farà que tingui poc retard i en cas contrari quan més gran sigui el retard més lent anirà.
s

```
milax@casa:~/Escriptori/Practica21/Practica21$ ./cocos1 joc22.txt 50
milax@casa:~/Escriptori/Practica21/Practica21$ ./cocos1 joc22.txt 400
```

Pararem la partida en cas de que el menjacocos agafi tots els cocos o quan un fantasma el mati.

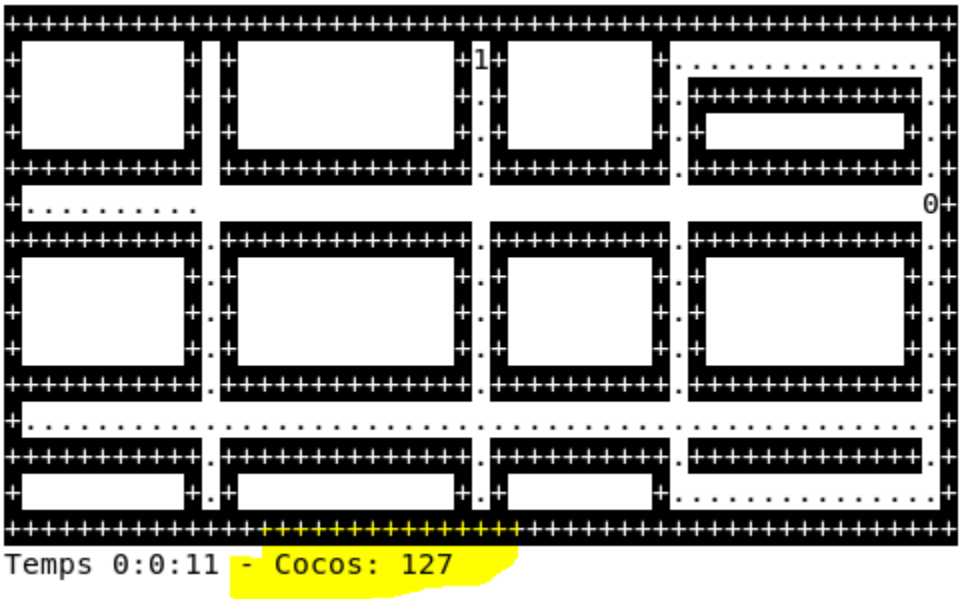
4. Joc de proves

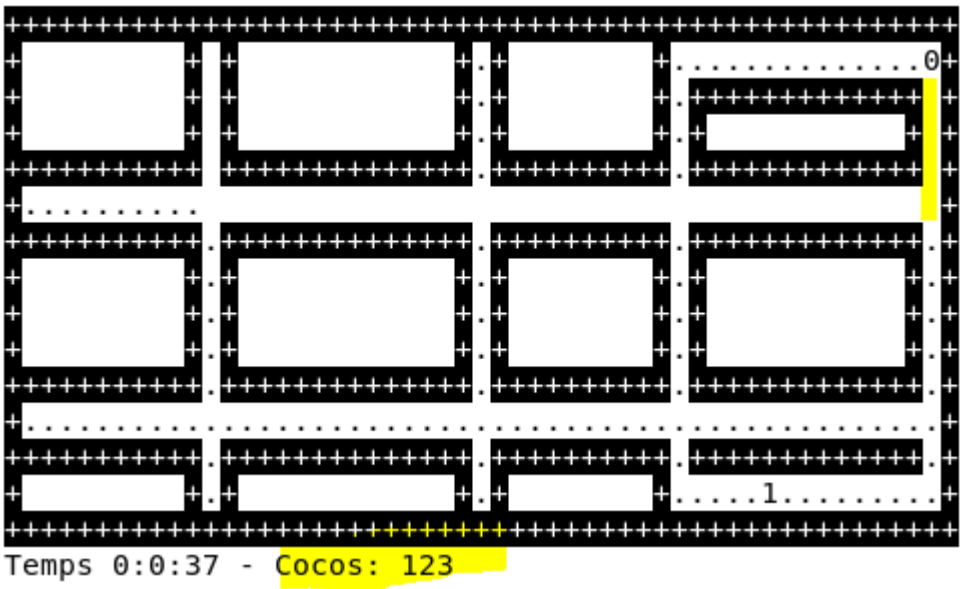
4.2 Cocos2

```
milax@casa:~/Escriptori/Practica21/Practica21$ ./cocos2 joc22.txt 200
Joc del MenjaCocos
        Tecles: 'w', 's', 'd', 'a', RETURN-> sortir
prem una tecla per continuar:
```

Amb el cocos 2 podem observar un joc mes fluït degut a la sincronització o com s'arreglen alguns mini bugs que podien aparèixer per falta de sincronització al cocos1.

En aquest cas provarem un altre mapa. També funciona les funcions esmentades anteriorment.



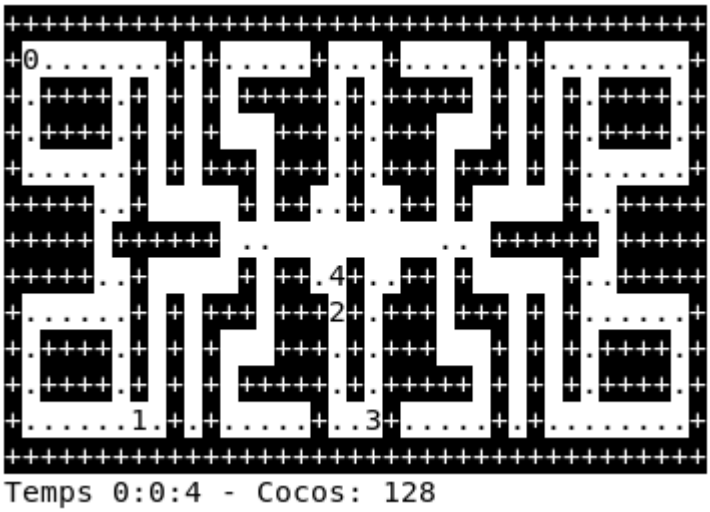



```
1 11 2 1.0
13 11 0 1.0
```

mapa joc31.txt

4. Joc de proves

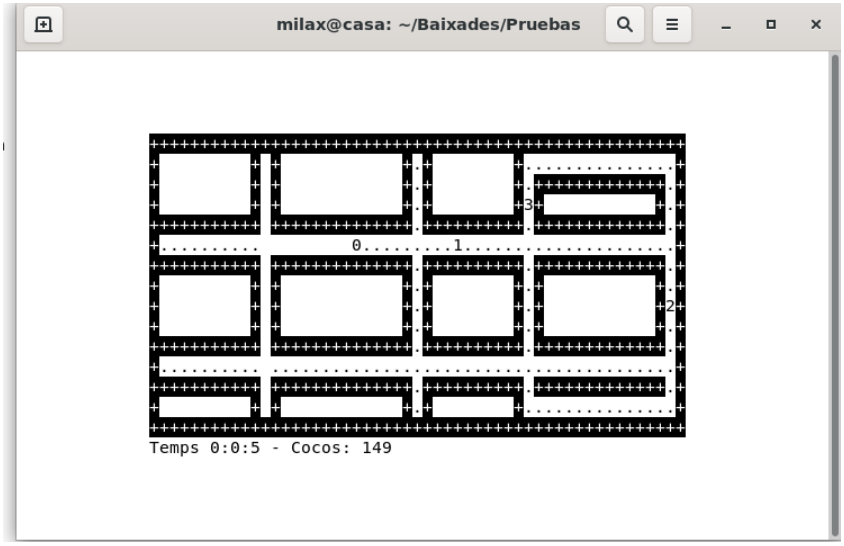
4.3 Cocos3



En cocos 3 comprovem que funcioni tot igual però aquesta vegada amb processos.

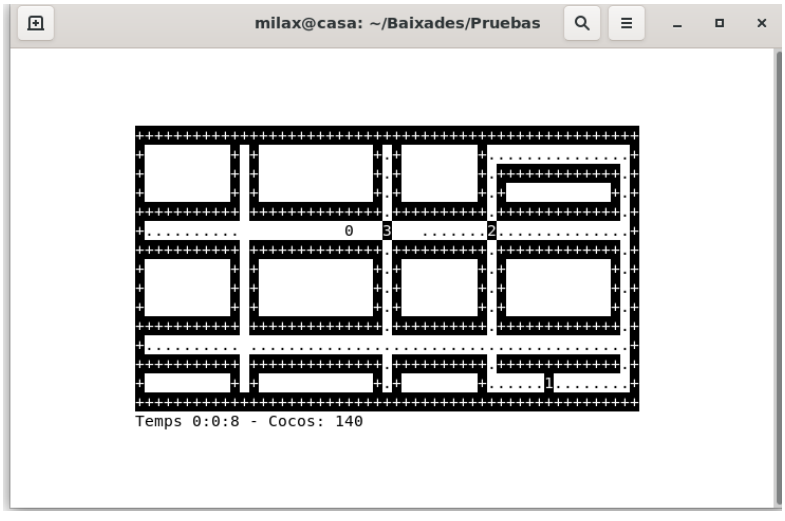
4. Joc de proves

4.4 Cocos4

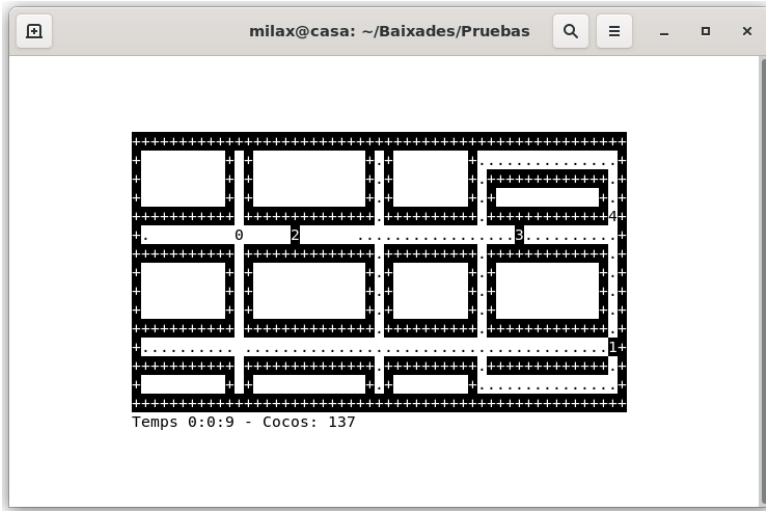


En aquest cas, podem veure que com el fantasma 1 va a la direcció a la que va el menjacocos, no la vist i per tant no s'activa el mode cacera i, per tant, no es mostren els fantasmes en invers.

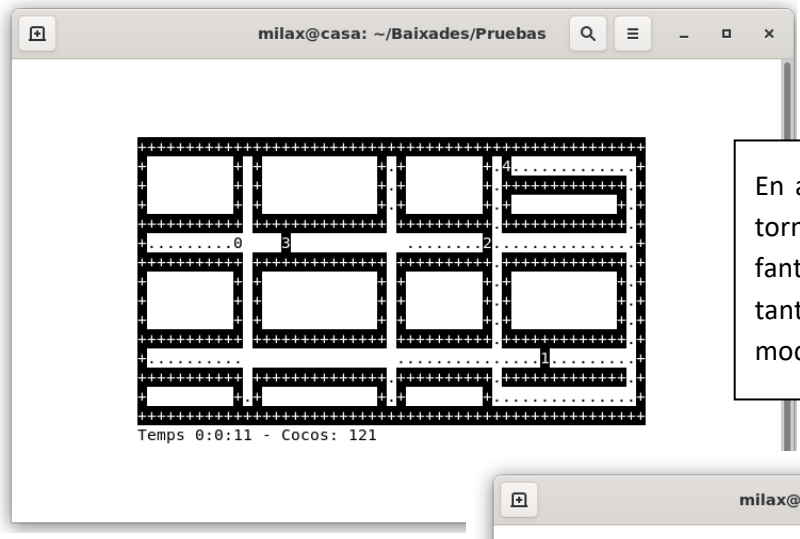
També, com que el menjacocos no s'ha xocat més de 2 cops quan s'ha creat el fantasma 3, no apareix el fantasma4.



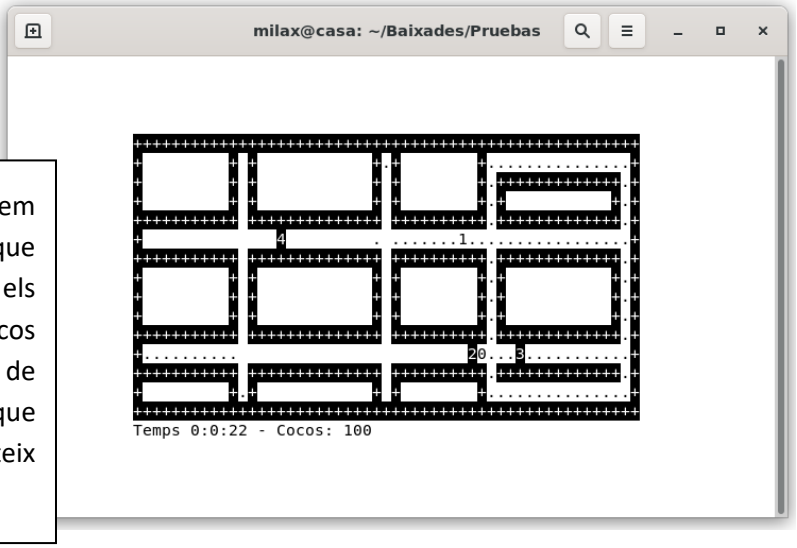
En aquest cas, com que el fantasma 3 ha vist al menjacocos, ha activat el mode cacera i per tant, ha provocat que tots el fantasmes es veuen de manera inversa. Com a l'exemple de dalt, com que el menjacocos no s'ha xocat més de 2 cops quan s'ha creat el fantasma 3, no apareix el fantasma4.



Per últim, en aquest exemple podem veure un dels errors que hem comentat anteriorment: el fantasma 2 ha vist al menjacocos i ha activat el mode cacera, però veiem que tots els fantasmes es veuen en invers menys el 4.



En aquest exemples podem veure que torna a funcionar correctament, on el fantasma 3 ha vist al menjacocos i, per tant, tots els fantasmes es veuen en mode invers.



Mentre que en aquest exemple, podem veure que un altre cop passa el que passava al segon exemple: els fantasmes 2 i 3 han vist al menjacocos però com podem veure, el 4 es veu de manera inversa mentre que el 1, que també s'hauria de veure del mateix mode, es veu en mode no invers.

5. Autoavaluació i conclusions

Aquesta pràctica ens ha ajudat a aprendre conceptes bàsics relacionats amb la programació amb múltiples fils d'execució (multithreading) i de múltiples processos per tal de fer múltiples tasques independents alhora. També ens ha ajudat a saber que són, per a que serveixen i com s'implementen els semàfors i les seccions crítiques.
A més, hem expandit el coneixement al llenguatge C, per tal d'implementar les funcions que se'ns demanaven perquè tot funcionés correctament.

Per últim, no hem sabut com implementar del tot i correctament la última tasca del mode cacera i mode normal ja que no teníem temps i no ens sortíem amb on podien estar els punts claus per tal de trobar el funcionament correcte dels modes.