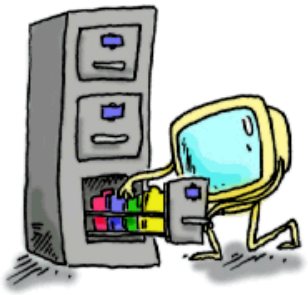
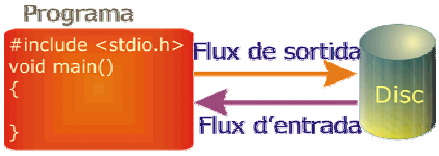
Tema 9: Fitxers en C

- 9.1 Introducció
- 9.2 Escriptura a fitxers
- 9.3 Lectura de fitxers
- 9.4 Modus i funcions de lectura/escriptura
- 9.5 Fitxers d'accés aleatori
- 9.6 Pas d'arguments per nom de programa



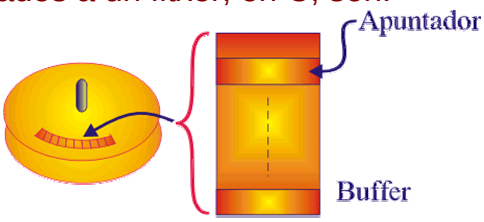
9.1 Fitxers

- Fins ara s'han vist estructures de dades que s'emmagatzemen en memòria principal (amb propietats de ràpidesa i mateix temps d'accés per a totes les dades). Quan hi ha moltes dades a tractar i que han de ser guardades es necessiten dispositius d'emmagatzement externs, no tant ràpids i més massius: discs, cintes, ...
- L'intercanvi de dades amb els dispositius externs es realitza mitjançant fluxs d'informació cap i des del dispositiu i s'organitza en fitxers. Un fitxer és una seqüència de bits emmagatzemats d'acord amb el protocol del sistema software.
- Quant aquests bits s'agrupen en paraules de 8 bits interpretats pel codi ASCII aleshores el fitxer és de tipus text. Els fitxers gràfics s'agrupen en paraules de 32 bits, representant píxels de color.
- Els noms dels fitxers s'han d'elegir de forma adequada al tipus de dada que contenen. Així, els fitxers de codi font en C tenen l'extensió .c o .cpp (VisualC), els executables .exe, quan es generen sortides de dades solen ser .dat, ...
- El flux de dades és un camí organitzat per connectar-se amb el dispositiu extern. Associat amb el camí hi ha associat un modus que determina el sentit del corrent: d'entrada (llegeix dades d'un arxiu), de sortida (envia dades cap al fitxer) o d'entrada/sortida. Tot el flux es realitza a través d'un buffer de memòria intermig que regula la transmissió de dades.



9.2 Escriptura a fitxers

- Els passos a seguir en l'escriptura de dades a un fitxer, en C, són:
  - Declarar el punter d'accés a arxiu.
  - Obrir el fitxer per a ser escrit.
  - Realitzar la transferència de dades
  - Tancar el fitxer.

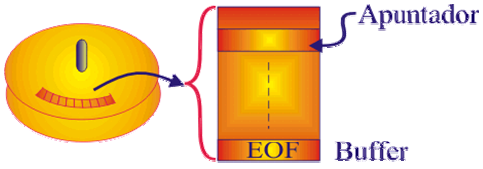


  - Exemple
    - Guardar en el fitxer primer.dat un conjunt de noms (un per línia) entrats per teclat.

```
#include <stdio.h>
#include <string.h>
void main()
{
   char nom[20];
   FILE *pt;                          ← Punter a variable FITXER
   pt=fopen("Nom.dat", "w");          ← Obertura del fitxer "Nom.dat" per a escriptura
   do
   {
        gets(nom);
        fprintf(pt, "%s%c\n", nom,'\n'); ← Posar el nom en el lloc del fitxer apuntat per pt
   }while (strlen(nom)>0);
   fclose(pt);                        ← Tancament del fitxer
}
```


9.3 Lectura de fitxers

- La lectura de fitxers segueix un procés similar al de lectura.
  - Declarar el punter d'accés a arxiu.
  - Obrir el fitxer per a ser llegit.
  - Realitzar la transferència de dades
  - Tancar el fitxer.
  - Ara, però, s'ha d'anar en compta en conèixer quan s'ha arribat al final del fitxer. Per això es compta amb la variable EOF que s'activa en arribar al final.



  - Exemple
    - Guardar en el fitxer primer.dat una sèrie de noms entrats des d'entrada estàndar.

```
#include <stdio.h>
   void main()
   {
      char car;
      FILE *pt;                       ← Punter a variable FITXER
      pt=fopen("Nom.dat","r");        ← Obertura del fitxer "Nom.dat" per a lectura
      while ( (car=getc(pt)) !=EOF)
            printf("%c", car);
      fclose(pt);                     ← Tancament del fitxer
   }
```

9.4 Modus i funcions de lectura/escriptura (I)

- Els modus de lectura/escriptura de fitxers en C són:
  - "a" → Obert per a afegir dades. Les dades es posen al final del fitxer . Si no existeix, es crea.
  - “w" → Obert per escriptura.
  - “r" → Obert per lectura. El fitxer ha d’existir.
  - “r+" →Obert per lectura/escriptura. Si no existeix, es crea de nou.
  - “w+" → Obert per lectura/escriptura. Ha d’existir l’arxiu.
  - "w+" → Obert per lectura/escriptura des del començament.
  - “_b” → Una b al final indica que es treballa amb un fitxer de tipus binari.
- Les funcions per treballar amb fitxers són:
  - Caràcter o string:
    - fputc(caràcter, punter_a_arxiu)→ Escriure caràcter
    - fputs(string, punter_a_arxiu)→ Escriure string
    - fgetc(punter_a_arxiu) → Llegir caràcter
    - fgets(string,tamany,punter_a_arxiu) → Llegir string
  - Amb dades de diferent tipus:
    - fprintf(FILE *fluxe, const char *format, arguments)
    - fscanf(FILE *fluxe, const char *format, arguments)

Els fitxers tipus text emmagatzemen els nombres com caràcters, enlloc de com a valors numèrics, motiu pel què no s’empra de forma eficient l’espi de disc.

En modus binari els nombres s’emmagatzemen com en memòria: només dos octets per nombre.

Per defecte els fitxers s’emmagatzemen en modus text. Per tant, en cas d’emprar el modus binari per escriure s’ha de vigilar que la lectura també es faci de forma binària!!

9.4 Modus i funcions de lectura/escriptura (II)

- Treballant amb fputs, fgets!

  Es llegeix una frase de teclat i s’emmagatzema per línies.

```
#include <stdio.h>
void main()
{
    char fr[30],nom_fitxer[]=“Frase.dat”;
    FILE *pt;
    pt=fopen(nom_fitxer, "w");          //Obrint el fitxer per escriptura
    do
    {
        gets(fr);
        fputs(fr,pt);                   //posar el string a pt
        fputc('\n',pt);                 //posar al fitxer un salt de ratlla per fer salt de línia
    }while (fr[0]=‘\0’);
    fclose(pt);

    pt=fopen (nom_fitxer, "r");         //Obrint el fitxer per lectura
    while (fgets(fr,30,pt)!=NULL)       //Llegir fins a 30 cars o salt de línia i posar-los a fr
        printf("%s",fr);
    fclose(pt);
}
```

El nom del fitxer no és res més que una constant que pot ser predefinida per una cadena de caràcters.

fgets() és una funció que pot valer NULL en cas de no llegir res.

9.4 Modus i funcions de lectura/escriptura (III)

- Mesclant tipus de dades: escribint i llegint un enter i una cadena per línies!

És un exemple senzill de com treballar amb dades mesclades, però s'ha d'anar en compta per què no facilita informació sobre possibles problemes que hi poden haver en el tractament de dades.

```
#include <stdio.h>
void main()
{
      int i, n;
      char s[20];
      FILE *pt;
      if ( (pt=fopen("Dades.dat", "w")) == NULL) exit (-1); //Obertura del fitxer per escriptura
      for (i=0;i<3;i++)
      {
            fflush(stdin);
            scanf("%i%s", &n, &s);
            fprintf(pt, "%i %s%c", n, s,'\n');      //Escribint amb fprintf un enter i un string
      }
      fclose(pt);
       if ( (pt=fopen("Dades.dat", "r")) == NULL)   exit (-1); //Obertura del fitxer per lectura
      while (fscanf(pt,"%i%s",  &n, s) != EOF)     //Llegint amb fscanf un enter i un string
            printf("%i %s\n",n, s);
      fclose(pt);
}
```

fopen() és una funció que val NULL quan el fitxer no s'obre bé

La funció exit() et treu del programa i segueix en la rutina de tractament d'errors, cas de tenir-la prevista.

9.5 Fitxers d'accés aleatori (I)

- L'accés a la informació emmagatzemada s'ha de realitzar d'acord amb la forma d'organització dels arxius.
- Fins ara s'han tractat fitxers d'accés seqüencial. La informació s'ha anat posant al fitxer i, tal com s'ha posat, s'ha de llegir. En obrir el fitxer l'apuntador ens adreça al començament del fitxer i, conforme es va llegint la informació, el punter apunta als caràcters que segueixen.
- Dels diferents exemples realitzats, només en el cas de lectura de la informació caràcter a caràcter fins que s'arriba al final del fitxer es té seguretat de què la informació es llegeix de forma correcta.
- L'altra forma usual d'organització de la informació és l'organització aleatòria de la informació. L'accés aleatori a l'arxiu permet llegir dades emmagatzemades dins l'arxiu sense necessitat d'haver de llegir les dades que hi ha abans.
- En C l'accés aleatori permet realitzar la lectura/escriptura d'informació de fitxers per blocs d'informació, entre aquests, directament per estructures.

9.5 Fitxers d'accés aleatori (II)

- Les funcions de lectura/escriptura de registres són:
  - fwrite(dir_dades, tamany, n, punter_a_arxiu) → Escriptura de registres
  - fread(dir_dades, tamany, n, punter_a_arxiu) → Lectura de registres

  on dir_dades és adreça on es troben els elements a tractar, tamany és el nombre d'octets que es llegeixen/escriuen i n el número d'elements a escriure/llegir

  Tant fread() com fwrite() retornen com a valors el nombre de registres llegits/escrits.
- El mètode de treball amb accés aleatori és el següent:
  - El punter a arxiu indica on tindrà lloc la propera operació dins l'arxiu.
  - En obrir-se un arxiu, el punter apunta a la posició 0.
  - Quan s'escriuen dades, queda apuntant al final de les dades escrites.
  - Si s'afegeixen dades, primer el punter accedeix a la última posició de l'arxiu.
  - Si es vol canviar la posició del punter a arxiu en C s'ha d'emprar la funció fseek().
- El format de la funció fseek() és el següent:

  fseek(punter_a_arxiu, desplaçament, mode)

  on mode és el tipus d'accés que es realitza i pot valer:

  mode = SEEK_SET (0) → des del començament de l'arxiu

  mode = SEEK_CUR (1) → des de la posició actual

  mode = SEEK_END (2) → des del final de l'arxiu
- La funció fseek() torna un valor 0 quan no es pot posicionar en el fitxer.

9.5 Fitxers d'accés aleatori (III)

- Exemple: Treballant amb fread(), fwrite() i fseek()

```
#include <stdio.h>
#include <stdlib.h>
struct personal
{
      char nom[30];
      long DNI;
};
void main()
{
      int i;
      personal pers, pr[4];
      FILE *pf;
                  //Entrada de registres en el fitxer
      pf=fopen("f_in.dat", "w");  //Obertura del fitxer
      for (i=0; i<4; i++)
      {
            printf("Entra nom i DNI: ");
            scanf("%s%i",&pers.nom, &pers.DNI);
            if (fwrite(&pers, sizeof(pers),1,pf) !=1) exit(-1);
      }
      fclose(pf);  //Tancament del fitxer
```

El tamany de l'estructura serà:

30*1 + 1*4 + 2= 36 bytes!

Escriptura d'1 registre. El valor retornar per fwrite serà 1.

9.5 Fitxers d'accés aleatori (IV)

```
pf=fopen("f_in.dat", "r");    //Obertura del fitxer

            //Lectura registre a registre
printf("\n--Llegint registre a registre--\n");
while (fread(&pers, sizeof(personal),1,pf)==1)
      printf("%s %i\n",pers.nom, pers.DNI);

            //Lectura 4 registres de cop
rewind(pf);    //Posicionament pt a l'inici del fitxer

printf("\n--Llegint 4 registres de cop--\n");
while ((fread(&pr, sizeof(personal),4,pf))==4)
      for (i=0;i<4;i++) printf("%s %i\n",pr[i].nom, pr[i].DNI);

            //Lectura del registre 2
printf("\n--Llegint el registre 2--\n");
fseek(pf,2*sizeof(personal), SEEK_SET);
if (fread(&pers, sizeof(personal),1,pf)==1)
printf("%s %i\n",pers.nom, pers.DNI);
fclose(pf);    //Tancament del fitxer

}
```

Lectura d'1 registre. El valor retornar per fread serà 1

Lectura de 4 registres. El valor retornar per fread serà 4. El resultat es posa en un array: personal pr[4] !!

Posicionament en el 3r registre

Lectura del 3r registre

9.6 Pas d'arguments per nom de programa

- C permet especificar una sèrie d'arguments quan es crida el programa.
- Aquests arguments es passen dins de la funció main() emprant els arguments de línia de comandes:
  - *int* argn → Compta el número d'arguments passats.
  - *char* *arg[ ] → On arg[ ] és un vector de punters a aquests arguments.
- Format: main(int identificador, char **identificador[ ])
- Exemple:

```
main(int argn, char *arg[])
{
        int num;
        printf("#arguments = %d\n", argn);
        for (num = 0; num < argn; num++)
        printf("Argument %d = %s \n", num, arg[num]);
}
```

Execució: Davant la crida del programa:

Programa Avui Plou

Dóna el resultat:

#Arguments = 2

Argument 0 = Avui

Argument 1 = Plou