Fonaments dels Sistemes Operatius

Departament d'Informàtica de Sistemas i Computadores (DISCA)
Universitat Politècnica de València

fSO

Pràctica 1
Programació en C (I)

Contingut

1 Objectius

Editar programes en llenguatge C i compilar-los en LINUX amb *gcc (GNU Compiler Collection*).

- Treballar amb programes en C continguts en un únic arxiu
- Aprendre a compilar i executar programes senzills
- Detectar errors bàsics de compilació
- Aprendre a comunicar-se amb el programa llegint i escrivint en el terminal
- Adquirir experiència amb " if …else if", "switch" i "while"

¡ADVERTENCIA¡: En aquesta pràctica s'assumeix que l'alumne ja sap programar en altre llenguatge.

2 Ferramentes necessàries

Per a crear programes en C és necessari un editor *(xemacs, kate, vi*, etc.) i un compilador de C.
Aquesta pràctica es realitza en sistema *Unix (Linux)* amb compilador *gcc.* Per a editar arxius es recomana utilitzar l'editor *kate*. En l'Annex1 podeu consultar aspectes del gcc i de les funcions printf() i scanf().

2.1 Etapes per a crear un programa en C

¡IMPORTANT!
Cal tindre en compte les següents etapes bàsiques per a crear programes en C durant totes les pràctiques de FSO:

- Utilitzar un editor per a escriure el codi font del programa i guardar-lo en arxius amb extensió *". c"*.
  ```
  $ kate myprogram.c &
  ```
- Si s'utilitza l'opció "-o" amb el compilador i no es detecten errors en el programa font, automàticament es genera un arxiu amb el codi executable. Per tant, al compilador li proporcionarem almenys dos noms d'arxiu: el del codi font i on ha d'emmagatzemar-se el codi executable.
  ```
  $ gcc myprogram.c  -o codeprogram
  ```
- Per a executar el programa heu d'utilitzar el nom de l'arxiu que conté codi executable
  ```
  $ ./codeprogram
  ```

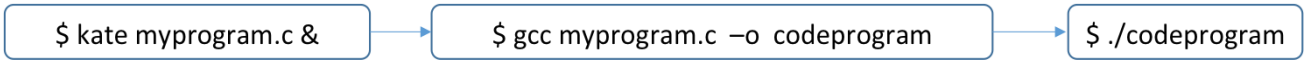

Figura 1: Etapes bàsiques per a crear programes en C

¡RECORDEU! Un programa en C necessita sempre una funció *main()*. En C totes les variables han de ser declarades abans de ser utilitzades i totes les instruccions s'escriuen en minúscules i acaben en ";".

2.2 Warnings i errors

És important diferenciar entre warnings i errors quan s'observa el resultat de compilar un programa. Un **warning** (avís) indica que el codi és ambigu i pot ser interpretat de manera diferent d'un compilador a un altre, però l'executable pot ser creat. Un **error** indica que no s'ha pogut compilar completament el programa i per tant l'executable no pot crear-se.
Els errors més típics són: **errors de compilació i errors d'enllaçat**. Els errors de compilació solen donar-se quan s'omet alguna cosa en el codi font com per exemple un ";". Els errors d'enllaçat són més subtils i solen donar-se quan el programa es troba repartit en diversos arxius (a estudiar més avant).

2.3 Ferramenta "*man*" per a trobar informació

"*man*" (manual) és una ferramenta de UNIX que te una entrada per a tots els comandaments, funcions i crides al sistema disponibles. Per a saber tot allò relacionat amb l'aplicació *man* heu de ficar:

```
$ man man
```

La documentació de les crides al sistema es troba en la secció 2 del manual, mentre que la de les funcions està en la secció 3. Per exemple, per a consultar la funció printf() o sqrt() heu de teclejar:

```
$ man 3 printf
```

```
$ man 3 sqrt
```

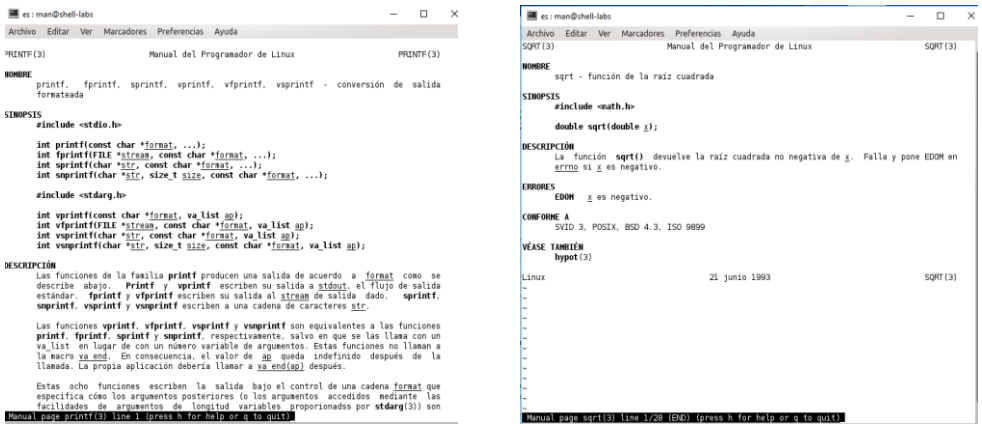

Figura 2: Pàgines de *man* per a les funcions *printf()* i *sqrt()*.

En qualsevol pàgina del manual la secció SYNOPSIS, inclou informació de l'ús de la funció. Per exemple, en el cas de la Figura 2:

#include <stdio.h>

significa que per a utilitzar la funció printf() s'ha de tindre aquest #include en l'arxiu del programa. També descriu els paràmetres necessaris per a invocar-la i allò que retorna rere la seua execució. La secció DESCRIPTION proporciona una menuda descripció d'allò que fa la funció.

Per a eixir d'una pàgina de man cal que polsar la tecla "q".

3 Programes en C

Aprendre C requereix practicar realitzant programes i modificant-los.

¡IMPORTANT! Recordeu mantenir sempre una còpia de la versió dels vostres programes quan funcionen i abans de fer grans canvis en ell.

3.1 Exercici 1.1 Creeu un programa en C: Editeu, compileu i executeu

Creeu l'arxiu "numbers.c" amb el contingut mostrat en la Figura 3, que correspon a l'estructura bàsica d'un programa en C. El programa "numbers" ha d'escriure informació (nombres) en pantalla, per això s'inclou la biblioteca <stdio.h> en el codi font i s'invoca la funció printf →Escriptura en pantalla.

Els passos a seguir són:

```
$ kate numbers.c &
```

Creeu l'arxiu i escriviu en ell les instruccions. Recordeu salvar cada cert temps per a no perdre informació.

Compileu el vostre codi codi font amb: (per a recuperar la consola polsar Enter)

```
$ gcc numbers.c  -o numbers
```

Comproveu si el compilador us mostra errors.

Es instructiu utilitzar l'opció -v de gcc per a obtenir un informe detallat de tots els passos de compilació:

```
$ gcc -v -o numbers numbers.c
```

Quan ja hageu solucionat tots els errors executeu-lo invocant l'arxiu que conté el codi executable

```
$ ./numbers
```

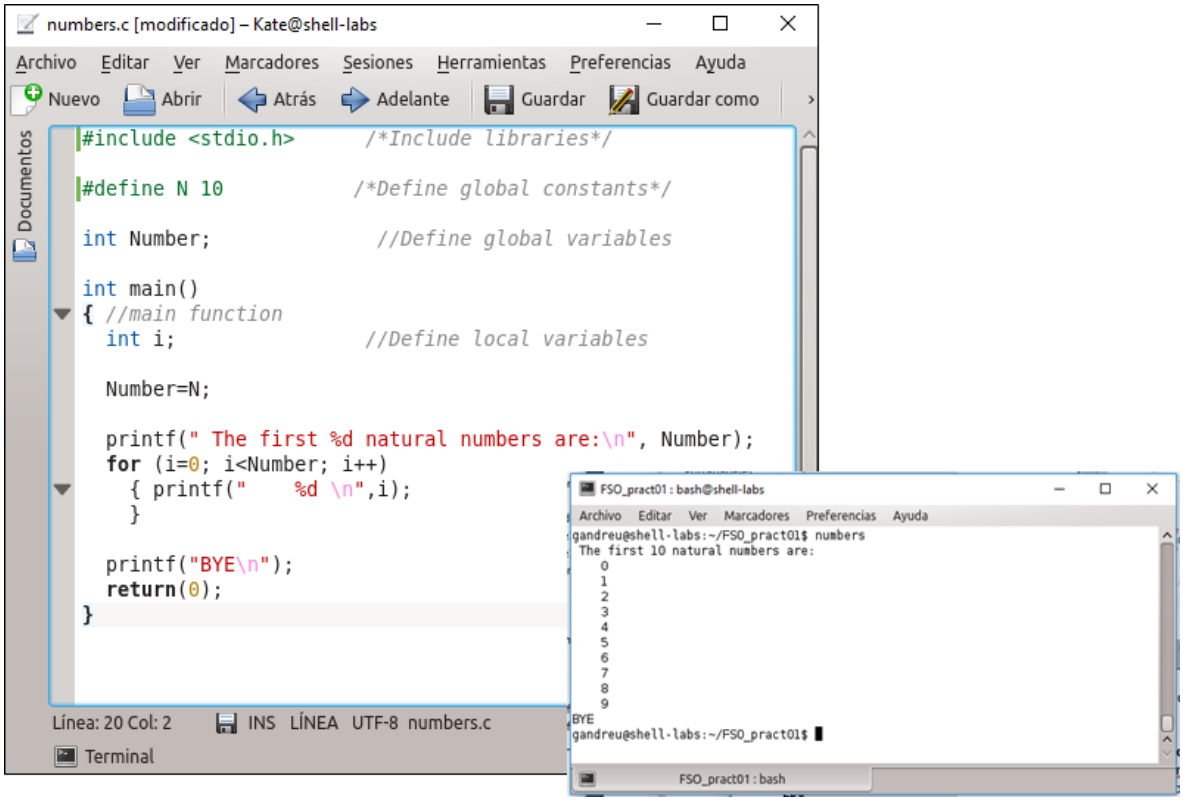

Figura 3: Contingut de l'arxiu "numbers.c" en l'editor kate i resultat de l'execució.

3.2 Exercici 1.2 Errors bàsics de compilació i *warnings*

Editeu l'arxiu "*numbers.c*" i realitzeu cadascuna de les accions que es proposen en la taula. rere cada acció heu de salvar l'arxiu, compilar i identificar el tipus d'error i el missatge que li mostra el compilador. Tingau en compte que abans de fer un canvi ha de desfer el canvi anterior.

Acció	Nº de línia de l'error	¿S'ha generat l'executable?	Error o warning
Comenteu la declaració de variable i → //int i;			
Substituir "%d" per "%f"			
Elimineu un ";"			
Substituïu "main" per altre nom "many"			

3.3 Exercici 1.3 Lectura de teclat: funció "scanf()"

Per a que *"numbers"* puga escriure un conjunt diferent de nombres s'ha de canviar el valor de la constant global N per altre i tornar a compilar. Tanmateix, el llenguatge C proporciona altres opcions sense necessitat de modificar el codi font i compilar cada vegada.

Copieu "*numbers.c*" en altre arxiu "*numbers2.c*". Editeu l'arxiu "*numbers2.c*" i modifiqueu-lo per a que pregunte a l'usuari quants nombres vol visualitzar i llisca el valor introduït per l'usuari. Utilitzeu la funció scanf() (Exemple: scanf("%d", &Number);. El funcionament que es requereix és el que s'aprecia en la figura 4.

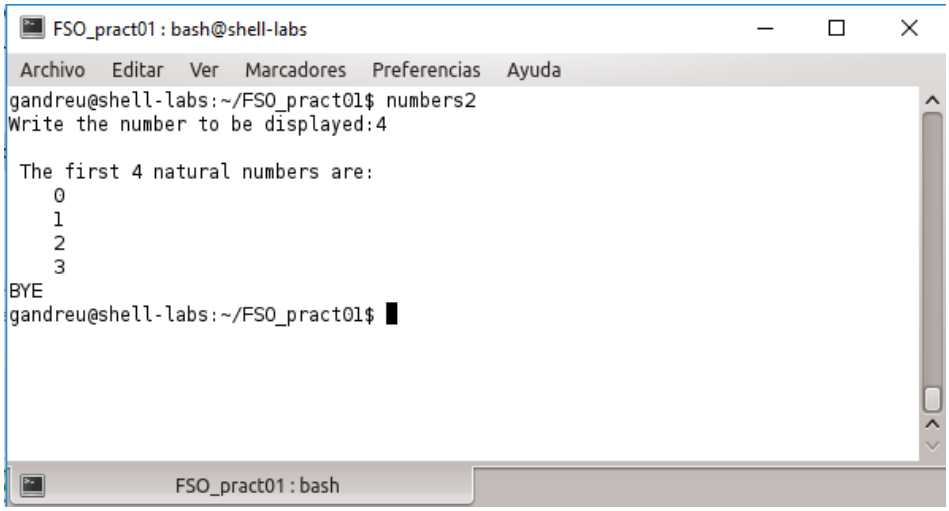

Figura 4: Execució de "*numbers2*", visualitza els nombres sol·licitats per l'usuari.

3.4 Exercici 1.4 Control de fluix "if …. else if"

Descarregueu de Poliformat el arxiu "*atm.c*" (*automatic teller machine*), conté el codi mostrat en figura 5, compileu-lo i executeu-lo. "*atm*" simula les operacions d'un caixer automàtic i està escrit d'una forma bàsica utilitzant "if … else if …". Comproveu el seu funcionament executant-lo tantes vegades com a opcions permeta.

```
#include <stdio.h>

#define InitBalance 1000
float Balance;

int main()
{ int operation, value;
  float income, withdraw;

  printf("\nWelcome to the FSO ATM\n");
  Balance=InitBalance;
  operation=0;
  printf("\nIndicate operation to do:\n");
  printf(" 1.Cash Income\n 2.Cash Withdrawal\n 3.Balance Enquiry\n");
  printf(" 4.Account Activity\n 5.Change PIN\n 6.Exit\n\n");
  printf(" Operation:");
  value=scanf("%d",&operation);

  if(operation==1){
    printf(" Cash Income\n");
    printf("\n Enter the amount to deposit:");
     scanf("%f",&income);
     Balance=Balance+income;
    printf(" Successful income\n");

  } else if(operation==2){
    printf(" Cash Withdrawal\n");
    printf("\n Enter the amount to withdraw:");
    scanf("%f",&income);

     if(Balance>income){
       Balance=Balance-income;
     }else{
       printf(" Operation does not allowed\n");
       printf("   Not enough cash\n");
     }

  }else if(operation==3){
   printf(" Balance Enquiry\n");

  }else if((operation==4)|| (operation==5)){
   printf(" This operation is not implemented");

 }else if(operation==6){
   printf(" EXIT\n");

  }else if(operation>6){
   printf(" ERROR: This opertaion does not applied\n");

  }

  printf("\n\n Current Balance: %.2f Euros", Balance);
  printf("\n\n Thanks \n\n");
   return(0);
}
```

```
gandreu@shell-labs:~/FSO_pract01$ atm

Welcome to the FSO ATM

Indicate operation to do:
 1.Cash Income
 2.Cash Withdrawal
 3.Balance Enquiry
 4.Account Activity
 5.Change PIN
 6.Exit

 Operation:2
 Cash Withdrawal

 Enter the amount to withdraw:270


 Current Balance: 730.00 Euros

 Thanks

gandreu@shell-labs:~/FSO_pract01$
```

Figura 5: Codi font i exemple d'execució del programa "*atm.c*".

3.5 Exercici 1.5 Control de fluix "switch"

Per a decisions múltiples on es comprove si una variable coincideix amb un valor dels d'un conjunt de valors es aconsellable utilitzar l'estructura:

```
switch(exp){
  case exp-const: ------------------------;
        break;
  case exp-const: ------------------------;
        break;
  default: ---------------------------------;
        break;
  }
```

Copieu l'arxiu "*atm.c"* en "*atm2.c"* utilitzant l'ordre del *Shell "$cp atm.c atm2.c"*. Modifiqueu "*atm2.c*" utilitzant la proposició *switch{}* reemplaçant els "*if else*" que comproven el valor de la variable "*oper*" per la sentència "*case*" corresponent. El funcionament ha de ser el mateix que amb "*if … else*". Una vegada aconseguit contesteu a les següents qüestions:

Pregunta	Resposta
¿Quina expressió heu utilitzat en *switch*(exp)?	
¿Per què és necessari el *break;*?	
¿Com heu resolt la comprovació de la línia 43? `}else if((operation==4)|| (operation==5)){`	
¿Per què es necessari el default:?	

3.6 Exercici 1.6 Bucle "while"

Copieu "*atm2.c*" en "*atm3.c"*. Modifiqueu atm3.c utilitzant la proposició "*while(exp){}"*, de manera que permeta més d'una operació en cada execució, mentre no es seleccione l'operació d'eixida (6). Una vegada resolt intenteu contestar a les següents qüestions:

Nota. Per a sagnar diverses línies al mateix temps seleccionar les línies a sagnar i polsar en Ferramentes -> Aliniar

Pregunta	Resposta
¿Quina expressió heu utilitzat en while(exp)?	

4 Propostes per a treballar pel vostre compte

El programa "*atm.c"* pot ser millorat, aquí s'indiquen algunes funcions per si l'alumne desitja practicar:

- Funció Saldo: Afegir funció que pregunte si ver o no el saldo (funcions de la biblioteca string.h.)
- Funció Idioma: Afegir funció que permeta seleccionar l'idioma del menú.

- Funció Seguretat: Afegir funció de seguretat que comprove la seua autentificació.
- Funció Moviments: En aquest cas ha de treballar amb arxius. Afegir una funció que emmagatzeme les operacions en un arxiu i siga capaç de recuperar-les per a mostrar-les al client

5 Annex 1: GCC ("GNU Compiler Collection")

GCC ("GNU Compiler Collection") engloba un conjunto de compiladores desenvolupats dins del projecte GNU, per tant, és software lliure. Actualment inclou compiladors per a C, C++, Objective C, Fortran, Ada. GCC pren un programa font i genera un programa executable binari per a la màquina on s'executa. GCC pot generar codi per a Intel x86, ARM, Alpha, Power PC, etc.. Es utilitzat com a compilador de desenvolupament en la majoria de plataformes. Així, Linux, Mac OS X, iOS (iPhone e iPad) estan íntegrament compilats amb GCC.

5.1 Sintaxi i opcions de GCC

La sintaxi d'ús del compilador gcc és la següent:

```
$ gcc [-options] [source_files] [object_files] -o output_file...
```

Les opcions van precedides d'un guió, com es habitual en UNIX, cada opció pot constar de diverses lletres i no poden agrupar-se diverses opcions rere un mateix guió. Algunes opcions requereixen després un nom d'arxiu o directori, altres no. Finalment, poden donar-se diversos noms d'arxiu a incloure en el procés de compilació.
GCC té infinitat d'opcions, a continuació, es detallen algunes:

Opció	Descripció
-c	realitza el preprocessament, compilació i acoblament, obtenint l'arxiu en codi objecte (.o).
-S	realitza el preprocessament i compilació, obtenint l'arxiu en assemblador.
-E	realitza només el preprocessament, enviat el resultat a l'eixida estàndard
-o arxiu	indica el nombre del arxiu de eixida (l'executable).
-Iruta	especifica la ruta del directori dels arxius a incloure en el programa font. No duu espai entre la I i la ruta, així: -I/usr/include. Per defecte no es necessari indicar les rutes de les "include" estàndard.
-Lruta	especifica la ruta del directori dels arxius de biblioteca amb codi objecte de les funcions referenciades en el programa font. No duu espai entre la L i la ruta, així: -L/usr/lib. Per defecte no és necessari indicar les rutes de les biblioteques estàndard.
-lNOMBRE	NOMBRE: biblioteca enllaçar junt amb el programa. Li diu al compilador quina biblioteca ha d'incloure amb el programa desenvolupat. Per a modificar el conjunt de biblioteques que utilitza l'enllaçador per defecte. Per exemple: -lm inclouria la biblioteca lib**m**.so (biblioteca matemàtica)
-v	Verbose on; mostra els comandaments executats en cada etapa de compilació i la versió dels mateixos.

5.2 Fases de compilació

El gcc, a l'igual que altres compiladores, es poden distingir 4 etapes en el procés de compilació (Figura 6):

- **Preprocessat**: Aquesta etapa interpreta les directives al preprocessador. Entre altres coses, les variables inicialitzades amb #define són substituïdes en el codi pel seu valor en tots els llocs on apareix el seu nom i s'inclou el font dels #include.
  Exemple;

```
$ gcc -E numbers.c > numbers.pp
 Examine numbers.pp amb
$ more numbers.pp
 Podreu comprovar que la variable N ha sigut substituïda pel seu valor
```

- **Compilació**: transforma el codi C en el llenguatge assemblador propi del processador de la nostra màquina. Si la màquina destí difereix de la del desenvolupament es denomina compilació creuada.

```
   $ gcc -S numbers.c
realitza les dues primeres etapes creant l'arxiu numbers.s;  examinant-lo amb
  $ more numbers.s
pot vore's el programa en llenguatge assemblador.
```

- **Assemblador**: transforma el programa escrit en llenguatge assemblador a codi objecte, un arxiu binari en llenguatge de màquina executable per el processador. No es freqüent realitzar només l'assemblat el normal és realitzar totes les etapes anteriors fins obtindre el codi objecte així:

```
 $ gcc -c numbers.c
on es crea l'arxiu numbers.o a partir de numbers.c. Pot verificar-se el tipus
d'arxiu usant el comandament
 $ file numbers.o
```

- **Enllaçat**: Les funcions de C/C++ incloses en el nostre codi, tal com a printf(), es troben ja compilades i assemblades en biblioteques existents en el sistema. Es precís incorporar d'alguna manera el codi binari d'aquestes funcions al nostre executable. En això consisteix l'etapa d'enllaçat, on es reuneixen un o més mòduls en codi objecte amb el codi existent en les biblioteques. L'enllaçador es denomina ld. El comandament per a enllaçar

```
 $ ld -o cercle cercle.o -lc
```

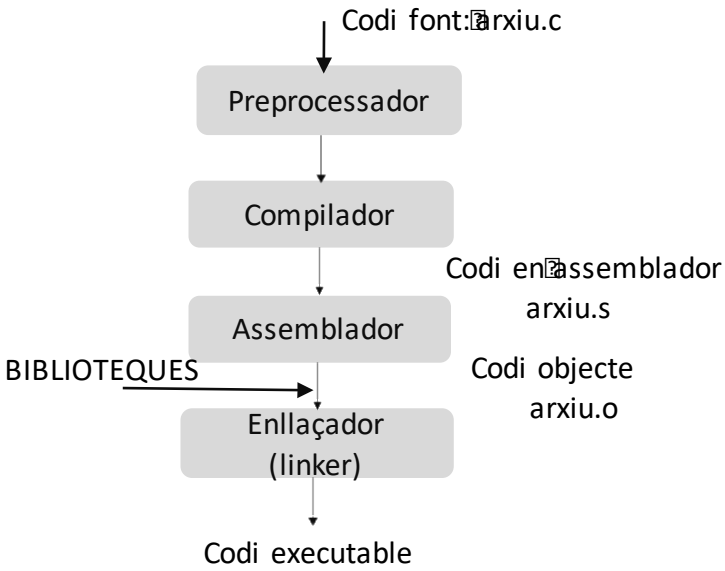

Figura 6: Fases de compilació.

5.3 Totes les fases en un sol pas

En programa amb un únic arxiu font tot el procés anterior pot fer-se en un sol pas:

```
  $ gcc -o numbers numbers.c
```

No es crea el arxiu numbers.o; el codi objecte intermig es crea i destrueix sense veure-ho l'operador, però el programa executable apareix allí i funciona.

Es instructiu usar l'opción -v de gcc per a obtindre un informe detallat de tots els passos de compilació:

```
$ gcc -v -o numbers numbers.c
```

Exemples d'ús:

```
$ gcc hola.c -o hola
 Compila el programa hola.c i genera un arxiu executable hola
$ gcc hola.c
  Compila el programa hola.c i genera un arxiu executable a.out.
$ gcc -o hola hola.c
  Compila el programa hola.c i genera un arxiu executable hola.
```

6 Annex 2: Biblioteca stdio.h funcions printf() i scanf()

En llenguatge C per a l'entrada/eixida s'ha d'utilitzar les funcions d'entrada i eixida estàndard proporcionades per la biblioteca estàndard de llenguatge C stdio.h, com a són printf i scanf, entre altres.

6.1 La funció printf()

La funció printf() envia una cadena de text amb format a l'eixida estàndard (la pantalla). La cadena de format conté dos tipus d'objectes: caràcters ordinaris i especificacions de conversió. Els caràcters ordinaris són copiats al fluix d'eixida, mentre que les especificacions de conversió provoquen la conversió i impressió dels valors dels arguments. Cada especificació de conversió ha d'anar precedida de el símbol %.

La sintaxi bàsica per a invocar aquesta funció és:

```
printf(<cadena_de_control> [, <llista_de_arguments> ] )
```

Exemple:

Codi font	Resultat en eixida estàndard

`#include <stdio.h>`
`int main(){`
`  int num1 = 10;`
`  printf("Number %d", num1);`
`  return 0;`
`}`
	`Number 10`

Figura 7: Exemple de programa amb printf()

La <cadena_de_control> és, en sí mateixa, una cadena de caràcters, que s'han d'escriure entre cometes dobles ("). Els arguments o paràmetres serveixen per a transferir dades entre funcions o programes. Quan un programa invoca la funció printf() amb una llista d'arguments, aquests valors (si no hi ha error) es mostren en pantalla junt amb la <cadena_de_control>. Per tant, la <cadena_de_control> ha d'indicar el format d'eixida de les dades que es van a mostrar en pantalla i pot constar de:

- Text ordinari (text normal).
- Especificadors de format.
- Seqüències d'escapament.

Tabla 1: Especificacions de format per a imprimir amb printf()

%caràcter	Tipus d'argument que requereix→ convertit a
%c	Int (sencer)→ char (caràcter): escriu un caràcter simple
%d, %i	Int (sencer)→ Int : escriu un sencer amb signe en base decimal

%e, %E	double → reals(double): escriu un nombre flotant amb exponent, en notació científica indicant l'exponent amb "e"
%f	double → reals: escriu un nombre en punt flotant sense exponent, format de punt flotant mmm.ddd
%g, %G	double → escriu l'opció més curta entre "%e" i "%f" o entre "%E" i "%F"
%o	Int → escriu un sencer en octal sense signe, sense zeros inicials
%s	Char * → imprimeix una cadena de caràcters fins trobar el caràcter '\0'.
%u	Int → escriu un sencer decimal sense signe
%x, %X	Int → escriu un sencer hexadecimal sense signe, sense el prefix 0x
%p	Void * → imprimeix un punter

Els especificadors de format estableixen el format d'eixida per pantalla dels arguments.
Les seqüències d'escapament ens permeten donar format a la informació mostrada per l'eixida estàndard. Una seqüència d'escapament sempre representa a un caràcter ASCII i s'escriu amb el caràcter barra invertida (\) . Aquests caràcters es poden classificar en:

- Gràfics: els quals es corresponen amb símbols utilitzats per a escriure, exemple: ",\, etc...
- No gràfics: representen accions, per exemple, moure el cursor al principi de la línia següent, etc..

Tabla 2: Seqüències d'escapament més utilitzades.

Seqüències d'escapament	Significat /Acció
\n	Nova línia (ASCII →010)
\t	Tabulació horitzontal (ASCII →009)
\v	Tabulació vertical
\f	Nova pàgina
\r	Retorn de carro
\\	Mostra el caràcter barra invertida (ASCII → 092)
\"	Mostra el caràcter cometa doble (ASCII →034)

6.2 La funció scanf()

La funció scanf() permet assignar valors a variables des de teclat, el seu format és similar a printf. La funció scanf() de la biblioteca estàndard stdio.h del llenguatge C assigna a una o més variables, un o més valors (dades) rebuts des de l'entrada estàndard (teclat). La sintaxi de la seu crida és:

scanf(<cadena_de_control> [, <llista_d_arguments>])

En la <cadena_de_control>, s'ha d'indicar el format d'entrada de les dades que es van a arreplegar per teclat. Per a això, s'ha d'utilitzar els especificadors de format.

Per exemple: scanf("%c %d %f %s", &ch, &i, &x, cad);

Codi font (exemple-A2)	Resultat en eixida estàndard

`#include <stdio.h>`
`int main(){`
`  int num;`
`  printf("\n Write an integer: ");`
`  scanf("%d", &num);`
`  return 0;`
`}`
	Write an integer: 4

Figura 8 : Exemple de programa amb scanf()

A l'executar el codi del exemple A2 en la memòria es reservarà espai per a la variable *num*. i si el usuari tecleja, per exemple, un 4, en pantalla se vorà allò mostrat en la columna dreta. Donat que la variable num és de tipus sencer, en la cadena de control s'escriu l'especificador d'un nombre sencer (%d). Per altra banda, la variable num està precedida del caràcter ampersand (&)

Observeu que en la funció scanf() els nom de les variables van **precedits de &** excepte quan correspon a una cadena de caràcters. Es tracta d'un nou operador, l'operador adreça o punter a variable. L'operador adreça (&) sempre actua sobre un operant (normalment una variable) per a obtenir l'adreça de memòria d'aquesta variable. Les variables es troben emmagatzemades en espais de memòria i la funció scanf() requereix les adreces de memòria de les variables, d'ací que vagen precedides de &.