C PROGRAMAZIO–LENGOAIA (XII)

Sarrera/Irteera

Iñaki Alegria & Montse Maritxalar

C-z egin daitekeen sarrera/irteera, liburutegi estandarra erabiliz egiten da. Erizpidearen arabera ondoko sarrera/irteera-motak bereiz daitezke:

1) Fitxategiari begira: estandarra/esplizitua

2) Formatuari begira: formaturik gabekoa/formatuduna

3) Atzipen-motari begira: sekuentziala/zuzenekoa

Orain arte ikusitako **scanf** eta **print** funtzioek fitxategi estandarrekin lan egiten dute, formatudunak dira eta atzipen sekuentziala egiten dute.

Sarrera/irteera fitxategi estandarren gainean

C-k maneiatzen dituen fitxategi estandarrak hiru dira: **stdin** sarrerarako, **stdout** irteerarako eta **stderr** errore-mezuetarako. Fitxategi hauek sistema eragileak prestatuak (irekiak) ematen dizkio programa nagusiari, honek zuzenean erabil ditzan. Fitxategi estandarrak teklatua eta pantaila dira besterik ezean, baina sistema eragilearen bidez berrelbidera daitezke. UNIX-ean eta DOS-ean **<** eta **>** karaktereak horretarako erabiltzen dira. C-ren liburutegian aipatutako **scanf** eta **print** funtzioen gain badaude sarrera/irteera fitxategi estandarrekin lan egiteko beste funtzio batzuk eta garrantzitsuenak honako hauek dira: **getchar, putchar, gets** eta **puts.** Lauak formaturik gabekoak dira eta lehenengo biek karaktere bakar bat irakurri edo idazten duten bitartean, besteek lerro oso bat maneiatzen dute. Haien erazagupena **stdio.h** fitxategian dago, gainerako sarrera/irteerako funtzio guztiena bezala, eta hauexek dira:

```
extern int getchar ( );
extern int putchar (int);
extern char * gets (char*);
extern int puts (char*);
```

1. eta 2. programetan **getc** eta **gets** funtzioak nola erabili ikus daiteke. Bi adibideetan, lerro bat irakurri

1. programa. getchar funtzioaren erabilpena.

```
# include <stdio.h>
int irak_lerro_luz (lerroa)
char lerroa []; /*irakurtzeko bufferra*/
{
  int c, i = Ø
  do  {  c = getchar ( );
         lerroa [i ]= c;
         i++;
      }
  while (c! = '\n' && c! = EOF);
  lerroa [i] = '\Ø'; /*string bukaera */
  return (i);
}
```

```
# include <stdio.h>
int irak_lerro_luz (lerroa)
char lerroa [];
{
  int i ;
  gets (lerroa);
  for (i = Ø; lerroa [i]! = '\Ø"; i++); ; /*ez du gorputzik
        edukiko */
  return (i+1);
}
```

2. programa. gets funtzioaren erabilpena.

eta lerroak duen luzera kalkulatzen duen funtzioa definitzen da.

Sarrera/Irteera fitxategiak erabiliz

C-rako fitxategi-kontzeptua orokorra da; bertan dispositiboak (teklatua, pantaila, inprimagailua, ...) zein memoria lagungarriko (disko, disket, zinta, ...) fitxategiak sartzen bait dira.

Beste lengoaietan bezala fitxategi bat erabiltzeko fitxategi hori ireki egin behar da, dagokion kontrol-informazioa memorian karga dadin eta sistema eragilearen sarrera/irteerako oinarrizko funtzioek kontsulta dezaten. Kontrol-informazio hori FILE izeneko egitura batean metatzen da (stdio.h fitxategian dago definituta) fitxategia irekitzen denean, eta erakusle batez erabiliko dugu, zeren eta fitxategi baten gaineko funtzioak deitzean bere FILE egituraren erreferentzia argumentu gisa aipatu egin behar bait da.

modua	onartutako eragiketak	uneko posizioa
"r"	irakurketak	hasiera
"w"	idazketak (fitxategi berriak sortuz)	hasiera
"a"	idazketak (gehikuntzak)	fitxategiaren bukaera
"r+"	irakurketak + idazketak (eguneratzea)	hasiera
"w+"	irakurketak + idazketak (fitxategi berria sortuz)	hasiera
"a+"	irakurketak + idazketak (gehikuntzak)	fitxategiaren bukaera

2. taula. fopen funtzioa erabiltzeko moduak.

Beraz, fitxategiak erabiltzeko, C programetan **FILE *** motako aldagai bat definituko da, irekitzean (**fopen** funtzioaz) emaitza jaso dezan eta gainerako sarrera/irteerako funtzioetan parametro gisa erabili ahal izateko.

1. taulan FILE egitura maneiatzen duten sarrera/irteerako funtzio batzuen zerrenda azaltzen da, bertan parametro eta emaitzaren datu-mota zein azalpen motz bat agertzen direlarik.

1. taulan azaldutako funtzio guztiak, **fseek**-a izan ezik, atzipen sekuentzialean erabiltzen dira, karaktereak (fgetc, fputc), lerroak (fgets, fputs), erregistroak (fread, fwrite) edo egoera bereziak (feof, ferror) maneiatzeko beti irekitako fitxategi baten gainean. **fp** jarri behar den tokian **stdin** edo **stdout** zehazten bada, funtzioa sarrera edo irteera estandarraren gainean burutzen da.

Irakurketak eta idazketak uneko posizioan egiten dira posizio hori **fopen** funtzioaz hasieratu eta atzipen bakoitzean eguneratu egiten delarik. **fopen** funtzioaren parametroa nolakoa denaren arabera, uneko posizioa aukeratzeaz gain buru daitezkeen eragiketak definitzen dira, 2. taulan azaltzen den legez.

3. programan S/I-ko funtzioak erabiltzen dira, UNIXean erabiltzen den **cat** izeneko programa idazteko. Programa honen bidez argumentu gisa pasatzen diren fitxategiak idazten dira, bata bestearen atzean irteera estandarrean. Argumentuak **argc** eta **argv** aldagaien bidez maneiatzen dira, 10. kapituluan ikusi genuen bezala.

1. taula. Sarrera/irteerako funtzioak.

Funtzioaren prototipoa	Azalpena
FILE * **fopen** (char * izena, int modua)	Fitxategia prestatu eta FILE egitura lortu
int **fclose** (FILE * fp)	Fitxategia itxi
int **fgetc** (FILE* fp)	Karaktere bat irakurri
int **fputc** (int c, FILE * fp)	Karaktere bat idatzi
int **fgets** (char * buf, int luz, FILE * fp)	Lerro bat irakurri (luzera mugatua)
int **fputs** (char * buf, FILE * fp)	Lerro bat idatzi
int **fread** (void * buf, int osa_luz, int osa_kop, FILE * fp)	Irakurri luzera finkoko zenbait osagai
int **fwrite** (void * buf, int osa_luz, int osa_kop, FILE *fp)	Idatzi luzera finkoko zenbait osagai
int **fscanf** (FILE*fp, char * formatua,...)	Scanf fitxategi batetik (formatuduna)
int **fprintf** (FILE * fp, char * formatua,...)	printf fitxategi batera (formatuduna)
int **ferror** (FILE * fp)	Errore-kodea lortu
int **feof** (FILE * fp)	Fitxategi bukaerako egoeraz galdetzeko)
int **fseek** (FILE * fp, long pos, int modua)	Posizio jakin batean kokatu
long **ftell** (FILE * fp)	Posizioa lortu

```
# include <stdio.h>
main (argc, argv)
int argc;
char * argv [ ];
{  FILE * fp
   int i;
   if  (argc < 2) erantsi-fitxat (stdin);
      else
      for  ( i = 1; i < argc; i++)
          {
            fp = fopen (argv [i], "r");
            if (fp = = NULL) {  fprintf (satderr, "errorea irekitzean "\n")'
                                  exit (1);
                               }
              else erantsi-fitxat (fp);
           close (fp);
   exit (o);
}
void erantsi-fitxat (file)
FILE * file;
{  char c;
   do  {  c = fgetc (file);
          fputc (c, stdout);
       }
   while (c! = EOF);
}
```

3. programa. Fitxategien kateaketa burutzen duen cat programa.

4. programa. Zuzeneko atzipenaren erabilera.

```
        # include "zuzen.h"
int     zuzen_atzi (fitx, pos, buffer, luz, eragik)
        FILE * fitx
        long pos;
        char * buffer;
        int luz;
        char eragik;
        {
            int stat;
            stat = fseek (fitx, pos, SEEK_SET);
            if (stat! = Ø) return (stat);
            if (eragik = = IRAKUR)
                  stat = fread (buffer, luz, 1, fitx);
              else stat = fwrite (buffer, luz, 1, fitx);
            return (stat);
        }
```

Zuzeneko atzipena

Adibideetan ikusi dugunez, liburutegiko oinarrizko S/Iko funtzioek (fgetc, fputc, freda, fwrite, ...) atzipen sekuentziala bideratzen dute, zeren fitxategia irekitzean hartzen duen uneko posizioa erabili eta eguneratzen bait dute.

Zuzeneko atzipena burutzeko aipatutako funtzioak **fseek**-arekin konbinatu behar dira; azken honek posizio jakin batean kokatzeko ahalmena eskaintzen bait du. **fseek** funtzioaren parametroak fitxategiaren deskribatzailea (**FILE *** motakoa), desplazamendua eta modua dira. Hiru modu berezi daude, bakoitzak desplazamendua aplikatzeko puntu desberdina aukeratzen duelarik. SEEK_SET fitxategiaren hasiera aukeratzen du, SEEK_CUR uneko posizioa eta SEEK_END fitxategiaren bukaera.

```
# include <string.h>
# include <stdio.h>
# define EZ Ø
# define BAI 1'
int. bilatu (fitx, string)
     FILE * fitx;
     char * string;
     {   long uneko_pos;
         uneko_pos = ftell (fitx);
         if( aurkitu (fitx, string) = = EZ)
            {   fseek (fitx, uneko_pos, SEEK_SET); /* zegoen posizioa*/
                return (EZ);
            }
         else return (BAI);
     }
int aurkitu (fitx, st)
     FILE * fitx;
     char * st;
     {
         char * buff [MAX];
         long pos;
         while (fgets (buff, MAX-1, fitx)! = NULL)
            if (strstr (buff, st)! = NULL) /* bilatu st buff-ean */
                  return (BAI);
         return (EZ)
     }
```

5. programa. fseek eta ftell funtzioen erabilpena.

4. programan zuzen_atzi funtzioa definitzen da, zeinak parametro baten arabera posizio jakin batean dagoen luzera finkoko eremu bat irakurtzen edo idazten duen.

ftell funtzioak ere lagun dezake; zuzeneko atzipenean uneko posizioa itzultzen bait digu. 5. programan editoreek duten bilaketa-laguntza nola programatzen den ikustearekin batera, ftell funtzioaren erabilpenaren adibidea dugu.