Errutinen espezifikazioa C programazio-lengoaian

Kontzeptuak

Iturburu-kodea, konpilazioa, konpilazio banatua, objektu-kodea, estekatzea, fitxategi exekutagarria, karga, exekuzioa, erreferentzien ebazpena.

Enuntziatua

Emandako adibideak erabiliz praktikatu gcc konpiladorearekin.

Urratsak

Proposatutako ariketak egin, ***gcc*** konpiladorearekin praktikatuz.

Dokumentazioa

- UNIXeko emandako apunteak.
- C lengoaiako ikastaroko apunteak.
- gcc konpiladorearen aukeren apunteak.
- C programen iturburu kodea.
- UNIXeko laguntza (*man*).

gcc konpiladorea

Urratsak konpilazioan (banan-banan edota pausu bakarrean)

- Aurreprozesaketa: aurreprozesadorearen aginduak (#define…) interpretatzen dira, iturburu fitxategietan behar diren ordezkapenak eginez. Adibidez:

```
more adibide.c
```

```
#define TAMAINA 50
main(int argc, char *argv[]) {
   int taula[TAMAINA];
   int i;

   for (i = 0; i < TAMAINA; i++) {
      taula[i] = i*i;
      printf("%d balioaren karratua %d da\n", i, taula[i]);
   }
}
```

```
gcc –E adibide.c > adibide.pp
more adibide.pp
```

```
main(int argc, char *argv[]) {
   int taula[50];
   int i;

   for (i = 0; i < 50; i++) {
      taula[i] = i*i;
      printf("%d balioaren karratua %d da\n", i, taula[i]);
   }
}
```

- Konpilazioa: C iturburu kodea makinaren prozesadorearen mihiztadura lengoaiara itzultzen da.

- Mihiztadura: mihiztadura kodea objektu kodera itzultzen da (kode bitarra, prozesadoreagatik exekutagarria dena).

- Estekatzea: programa osatzen duten objektu modulu desberdinak (iturburu fitxategiak konpilatuak, erabilitako liburutegiak…) estekatu egiten dira, fitxategi exekutagarri bakarra sortuz.

gcc konpiladorearen sintaxia

gcc opzioak C_fitx_izena obj_modulu_izena

Opzioak:

-c objektu fitxategia soilik lortzeko (estekatu gabe)

-g araztatzailearentzako (*debugger*) informazioa sortzeko

-llib_izena estekatzaileari (*linker*) esaten dio lib_izena liburutegian objektu moduluak bilatzeko esaten dio

-Lkatalogoa liburutegiak dauden katalogoa adierazteko

-o exek_izena lortzen den exekutagarriari izena emateko

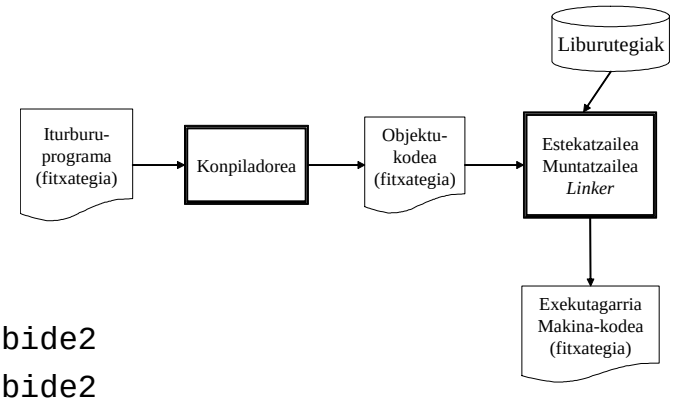
-E aurreprozesatzea soilik egiteko

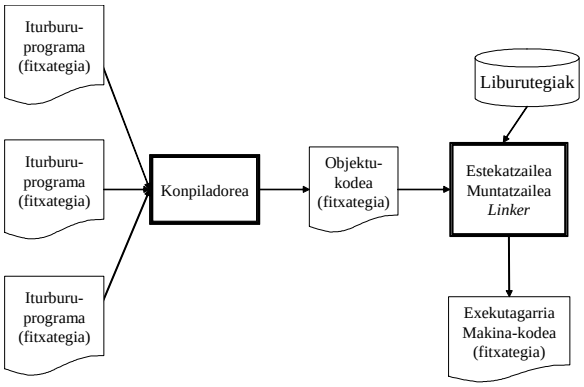
-S mihiztadura kodea soilik lortzeko

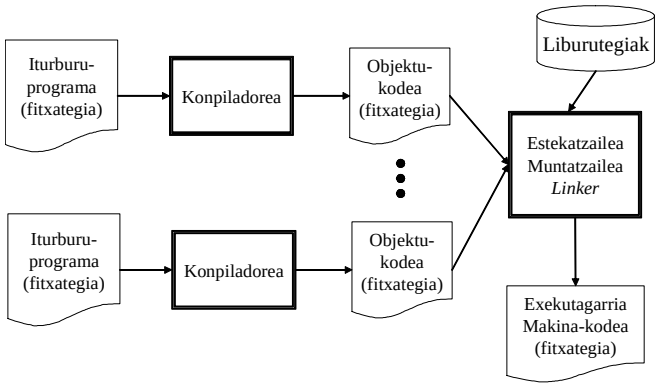
-D sinboloak definitzeko (baldintzako konpilazioa, makroak)

Adibideak

```
gcc adibide.c (emaitza: a.out)
gcc –c adibide.c
gcc adibide.o –o adibide
gcc adibide.c –o adibide
gcc –c adibide2.c
gcc –c errutinak.c
gcc adibide2.o errutinak.o –o adibide2
gcc adibide2.c errutinak.c –o adibide2
```







ld tresna (*link-editor*)

Moduluak estekatzen ditu

```
ld   opzioak   obj_modulu_izenak
```

Opzioak:

-lliburutegia	objektu-moduluak aurkitzeko liburutegia
-Lkatalogoa	liburutegiari dagokion katalogoa
-o exek_izena	lortzen den exekutagarriari izena emateko
-M	moduluen mapa lortzen da

ar tresna

Objektu-moduluen liburutegi bat sortu eta eguneratzen du.

```
ar   opzioak   liburutegia   obj_modulu_izenak
```

Opzioak:

d	osagairen bat ezabatzeko
m	osagairen bat bukaeran jartzeko
p	osagairen bat inprimatzeko
r	osagairen bat ordezkatzeko
t	edukien taula listatzeko
x	osagairen bat ateratzeko
c	liburutegia sortzeko
u	aldatuak baino ez ordezkatzeko
v	aldaketei buruzko informazioa lortzeko

Liburutegiak

(1) Izena zera izan behar da: `lib`*`izen`*`.a`
Adibidez: `libmod.a`

(2) Sorkuntzarako: `ar rcv lib`*`izen`*`.a`
Adibidez: `ar rcv libmod.a`

(3) Objektu modulu-bat gehitzeko: `ar r lib`*`izen`*`.a `*`modulu`*`.o`
Adibidez: `ar r libmod.a cambiar.o`

(4) Konpilazioa liburutegia erabiliz:

`gcc -L`*`kat`* `-o `*`exekutagarria iturburua`* `-l`*`izen`*
Adibidez: `gcc -L. -o proba proba.c -l`**`mod`**

libmod.a liburutegian bilatzen dira objektu-moduluak

(5) Gauza bera estekatzailetik:

`ld -L`*`kat`* `-l`*`izen`* `-o `*`exkeutagarria objektua`*`.o`
Adibidez: `ld -L. -l`**`mod`** `-o proba proba.o`

gdb tresna (arazketa)

Sintaxia: **`gdb`** `exekutagarria`

Aginduak:

help	aginduen laburpena
run	exekutatu programa
next	exekutatu agindu bat
step	funtzioaren barruan sartu
list	kodea ikusi
break	eten-puntu bat definitu
print	aldagai baten balioa azaldu orain
display	aldagai baten balioa azaldu eten-aldi guztietan
where	exekuzio-puntua eta funtzioen pila ikusi

Enuntziatua

1. ariketa. Taulen kudeaketarako C lengoaian egindako funtzio multzo bat (*taulak.c*) ematen da. Funtzioak erabiliz zenbakiak sailkatzeko programa nagusi bat programatu, konpilatu eta probatu.

2. ariketa. Ilaren kudeaketarako C lengoaian egindako funtzio multzo bat (*ilarak.c*) ematen da. Objektu-modulua sortuko da multzo horretatik. Beste bi fitxategi ematen dira ere (*ilarak.h* eta *asignatu.h*) funtzioen erazagupena ez errepikatzeko eta memoriaren kudeaketa egiteko. *proba.c* programa —eginda ematen dena— zenbaki osokoak irakurri, ilararatu eta idazten ditu. Programa honetan oinarriturik sailkatzeko programa bat eraiki beharko da.

Urratsak

1. sailkatu.c programa idatzi, taulak.h sartuz (#include agindua).
2. sailkatu.c fitxategia konpilatu, sailkatu programa sortuz. Probatu.
3. ilarak.c fitxategia konpilatu, ilarak.o sortuz.
4. proba programa sortu eta probatu.
5. Aldatu proba zenbakiak sailkatu ditzan.

```
/* taulak.h */

extern void irakur_taula(int taula[], int kop);
extern void inprimatu_taula(int taula[], int kop)
extern void sailkatu(int taula[], int kop);
```

```
/* taulak.c */

#include <stdio.h>

void irakur_taula(int taula[], int kop)
{
   int i;

   printf("Sakatu %d zenbaki:\n", kop);
   for (i = 0; i < kop; i++)
      scanf("%d", &taula[i]);
}

void inprimatu_taula(int taula[], int kop)
{
   int i;

   printf("Taularen edukina:");
   for (i = 0; i < kop; i++)
```

```
      printf(" %d", taula[i]);
   printf("\n");
}

int txikiena(int taula[], int kop)
{
   int i, ind_txik = 0;

   for (i = 1; i < kop; i++)
      if (taula[i] < taula[ind_txik]) ind_txik = i;
   return(ind_txik);
}

void sailkatu(int taula[], int kop)
{
   int i, ind;
   int tnp;

   for (i = 0; i < kop-1; i++) {
      ind = txikiena(&taula[i], kop-i);
      tnp = taula[i];
      taula[i] = taula[i+ind];
      taula[i+ind] = tnp;
   }
}
```

```
/* ilarak.h */

struct item {
   struct item *ondoko;
   int lehen;
   char *infor;
};

struct ilara {
   struct item *lehena;
   struct item *azkena;
};

#define NIL (struct item*)0
#define BOOL short

void sortu(struct ilara *p);
BOOL huts(struct ilara *p);
struct item *lehena(struct ilara *p);
struct item *atera(struct ilara *p, int leh);
void lotu(struct ilara *p, struct item *pelem);
void txertatu(struct ilara *p, struct item *pelem);
```

```
/* asignatu.h */

#define MAXELEM 100

struct item taula_elem[MAXELEM];

struct item *hartu_elementu()
```

```
{
   static int i = 0;

   if (i < MAXELEM) return(&taula_elem[i++]);
   else return(NIL);
}
```

```
/* ilarak.c */

#include "ilarak.h"

void sortu(struct ilara *p)
{
   p->lehena = NIL;
   p->azkena = NIL;
}

BOOL huts(struct ilara *p)
{
   return(p->lehena == NIL);
}

struct item *lehena(struct ilara *p)
{
   struct item *paux;

   paux = p->lehena;
   if (paux != NIL) {
      p->lehena = paux->ondoko;
      if (p->azkena == paux)
         p->azkena = NIL;
   }
   return(paux);
}

void lotu(struct ilara *p, struct item *pelem)
{
   struct item *paux;

   paux = p->azkena;
   if (paux == NIL)
      p->lehena = pelem;
   else
      paux->ondoko = pelem;
   pelem->ondoko = NIL;
   p->azkena = pelem;
}

struct item *atera(struct ilara *p, int leh)
{
   struct item *paux, *paur;

   paur = NIL;

   for (paux = p->lehena; paux != NIL; paux = paux->ondoko)
   {
      if (paux->lehen > leh) paur = paux;
      if (paux->lehen == leh)
```

```
      {
         if (paur != NIL) paur->ondoko = paux->ondoko;
         else p->lehena = paux->ondoko;
         if (paux->ondoko == NIL) p->azkena = paur;
         return(paux);
      }
      if(paux->lehen < leh) return(NIL);
   }
   return(NIL);
}

void txertatu(struct ilara *p, struct item *pelem)
{
   struct item *paux, *paur;

   paur = NIL;

   for (paux = p->lehena; paux != NIL; paux = paux->ondoko)
   {
      if (pelem->lehen <= paux->lehen) paur = paux;
      else break;
   }
   pelem->ondoko = paux;
   if (paur != NIL) paur->ondoko = pelem;
   else p->lehena = pelem;
   if (paux == NIL) p->azkena = pelem;
}
```

```
/* proba.c */

#include <stdio.h>
#include "ilarak.h"
#include "asignatu.h"

struct ilara ilara_bat;

main()
{
   int datu;
   struct item *elem;

   sortu(&ilara_bat);
   while (scanf("%d", &datu) != EOF)
   {
      if ((elem = hartu_elementu()) == NIL) break;
      elem->lehen = datu;
      lotu(&ilara_bat, elem);
   }
   printf("\n");
   while (!huts(&ilara_bat))
   {
      elem = lehena(&ilara_bat);
      printf("%d\n", elem->lehen);
   }
}
```

IZENA

ordenatu – Zenbaki osoak txikitik handira ordenatzen duen Unix filtroa.

ERABILPENA

./ordenatu

DESKRIBAPENA

ordenatu Unix filtroak zenbaki sekuentzia bat sarrera estandarretik irakurtzen du (lerroko zenbaki bat), eta irteera estandarrean irakurritako sekuentzia txikitik handira ordenaturik idazten du.

Sarrerako sekuentzia fitxategi-bukaera kodearen bidez amaitzen da (^D teklatuan).

Filtro honek ez du argumenturik.

BATERAGARRITASUNA

ordenatu filtroak edozein Unix sisteman funtzionatu beharko luke.

ERRORE EZAGUNAK

ordenatu filtroak gehienez 100 zenbaki onartzen ditu. Zenbaki gehiago sartu ezkero, sartutako lehen 100 zenbakiak ordenatuko ditu soilik.

Sartutako zenbakiak int motakoak izan behar dira. Bestela, emaitza edozein izan daiteke.

EGILEA

Anonimoa

ERABILPEN ESKUBIDEAK

Copyleft lizentzia <http://www.gnu.org/licenses/gpl.html>. Bermerik gabe.

IKUSI ERE

sort (1)