Liburutegiak

- Edozein programa egiten dugunean:Adb:

```
#include <stdio.h>
#include <stdlib.h>
 int main(){
 printf(“kaixo”);
 exit(0);
}
```

Liburutegiak

- C lengoaian liburutegiak erabiltzen dira:

gcc programa.c -o programa

ldd programa

linux-vdso.so.1 => (0x00007fff01dfe000)

libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f80dd5af000)

/lib64/ld-linux-x86-64.so.2 (0x00007f80dd990000)

- Zein paketetan datorren ikusteko:

dpkg -S /lib/x86_64-linux-gnu/libc.so.6

libc6: /lib/x86_64-linux-gnu/libc.so.6

Liburutegiak

- Paketearen informazioa ikusteko:

  $ dpkg -l libc6

Deseado=Desconocido/Instalar/Eliminar/Purgar/Retener

| Estado=No/Instalado/Config-files/Desempaquetado/Medio-conf/Medio-inst/espera-disparo/pendiente-disparo

|/ Err?=(ninguno)/Requiere-reinst (Estado,Err: mayúsc.=malo)

||/ Nombre Versión Descripción

+++-==============-==============-
==

ii libc6 2.13-20ubuntu5 Embedded GNU C Library: Shared libraries

libc.Liburutegiak

- Sistemako programa askok erabiltzen dituzten libc paketeko funtzioak.
- Pakete honetan daude liburutegi partekatu garrantzitsuenak, hau da, *C*ko liburutegi estandarra eta liburutegi estandar matematikoa. Bi liburutegi hauek barik Linux sistemak ez du funtzionatuko.

Liburutegiak

- *C*ko liburutegi estandarrak erabiliko dituen programa bat garatu nahi badugu, libc6-dev instalatu beharko dugu.

$sudo aptitude install libc6-dev

- Libc6-dev paketearen edukia:GNU C Library:

  Development Libraries and Header Files

  $sudo dpkg -L libc6-dev

Liburutegiak

- Cko liburutegi estandarrak erabiliko dituen programa bat garatzeko, /usr/include-n instalatuko diren goiburu fitxategiez gain (Adb:/usr/include/stdio.h), exekutagarria sortzeko beharko dituen objetu artxiboak ere instalatu behar dira(Adb:/usr/lib/x86-64-linux-gnu/libc.so edo libc.a)

sudo dpkg -S /usr/include/stdio.h

libc6-dev: /usr/include/stdio.h

Liburutegiak

- C-n idatzitako pogramek liburutegiak eta .h (goiburu) fitxategiak erabiltzen dituzte.
- C liburutegiko goiburu artxiboetan:
  - Liburutegietan barneratuta dauden funtzioen erazagupenak ageri dira.
  - motak berriak eta makroak daude.
  - Aurre-prozesatzailea direktiba hauek prozesatzeaz arduratzen da: #include, #define, ...
  - Goiburuak /usr/include-n daude.

Liburutegia

- Liburutegiko artxiboak:
  - Estatikoak: Liburutegiko kodea exekutagarriaren barnean sartzen da.
  - Dinamikoak: Liburutegiko kodea exekuzioan zehar kargatzen da.

Liburutegiak

- Liburutegi dinamikoen abantailak estatikoekin parekatuz:
  - Exekutagarriek memoria gutxiago behar dute. Nahiz eta hainbat programetatik deitu, behin bakarrik kargatzen baitira.
  - Liburutegian aldaketak egiterakoan, ez da aplikazioa berriz konpilatu behar.

Liburutegiak antolatzea

- Sartu funtzioen kodeak .c luzapeneko iturri fitxategi batean.
- Sartu funtzioen erezagupenak eta erabiliko diren mota berriak (typedefs, structs…) izen berdineko .h goiburu fitxategi batean.
- Bihurtu .c → .o, eta sartu .o-ak liburutegi batean.
- Liburutegiak /lib eta /usr/lib –en daude. (http//www.pathname.com/fhs/pub/fhs-2.3.html)

/usr/lib : Libraries for programming and packages

  - Adb:/usr/lib/x86-64-linux-gnu/libc.so (garapen)

/lib : Essential shared libraries and kernel modules

  - Adb:$ldd programa.exe

    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (erabilpen)

Liburutegiak antolatzea

liburutegia1.h:

```
#ifndef _LIBURUTEGIA_1_H
#define _LIBURUTEGIA_1_H
int gehiketa (int a, int b);
int kenketa (int a, int b);
#endif
```

liburutegia1.c:

```
int gehiketa (int a, int b)
{
    return a+b;
}
int kenketa (int a, int b)
{
   return a-b;
}
```

Liburutegiak antolatzea

- Gatazka:

“fitxategia1.h”-an #include<liburutegia1.h> dago.

“fitxategia2.h”–an #include<liburutegia1.h> dago.

“fitxategia3.c”–n #include<fitxategia1.h> eta #include<fitxategia2.h> lerroak daude.

- Ekiditzeko:

LIBURUTERIA_1_H fitxategia lehen aldiz sartzerakoan, definitu gabe egongo da. Beraz #ifndef - #endif multzoan sartuko da eta funtzioen eredu eta motak zehaztuko dira, baita LIBURUTEGIA_1_H bera ere. Bigarrenez sartzerakoan, LIBURUTEGIA_1_H jadanik zehaztuta egongo da, eta ez da berriz sartuko #ifndef - #endif multzoan.

Liburutegi Estatikoa sortzeko

1. Liburutegia1-eko objetu modulua sortu:

gcc -o liburutegia1.o -c liburutegia1.c

- ar-k liburutegia sortzen du, baina edozein motako fitxategiak paketatzeko balio du (EZ objetu moduluak bakarrik). Gainera aukera desberdinak ditu:barnean zein fitxategi dauden ikusteko, fitxategi horiek ezabatu, ordezkatu, etab.

ar rv liborokor.a liburutegia1.o "r" aukerak ordezkatu edo barneratu egiten ditu objetu moduluak liburutegian. "v" aukerak mezuen bidez adierazten du "verbose".Adibidez:

ar rv liborokor.a liburutegia1.o

ar: creating liborokor.a

a - liburutegia1.o

Liburutegi Estatikoa

- **ranlib**-ek artxibo hasieran bertan dauden funtzioen aurkibide sortzen du. Bertan funtzioen deskribapen eta estekak daude, liburutegi osoa irakurri gabe ebatzi ahal izateko identifikagailuak daude.

ranlib liborokor.a

Liburutegi Estatikoa

- **ranlib**-ek artxibo hasieran bertan dauden funtzioen aurkibide sortzen du. Bertan funtzioen deskribapen eta estekak daude, liburutegi osoa irakurri gabe ebatzi ahal izateko identifikagailuak daude.

ranlib liborokor.a

Liburutegi Estatikoa

- Aurkibidea ikusteko:
  nm -s liborokor.a

Archive index:
gehiketa in liburutegia1.o
kenketa in liburutegia1.o
liburutegia1.o:
0000000000000000 T gehiketa
0000000000000014 T kenketa

Liburutegi Estatikoa

- Liburutegiko funtzio bat erabiltzen duen exekutagarria sortzeko:

  Hau da aplikazioa.c:

  #include <stdio.h> →/usr/include

  #include “liburutegia1.h”

  int main (void)
  {printf(“%d”,gehiketa(8,9));return 0;}

  gcc –L . –l. –o aplikazioa aplikazioa.c – lorokor -static

  ./aplikazioa

Liburutegi Estatikoa

gcc –o nireprograma nireprograma.c –I<path1>-I <path2>…-L <path1> -L <path2>… -lorokor static

Adb: gcc –L . –I. –o aplikazioa aplikazioa.c – lorokor

Adb: gcc –L /home/kepa/lana –I /home/kepa/lana –o aplikazioa aplikazioa.c /home/kepa/lana/liborokor.a static

-I <path> .h-ak non dauden jakiteko.(“/usr/include” ez den beste katalogo batean daudenean erabili.)

-L<path> .a edo .so non dauden jakiteko.(“/lib edo /usr/lib” ez den beste katalogo batean daudenean erabili)

-lorokor eginez, liburutegiaren izena adierazten dugu, lib aurrizkia eta .a luzapenik gabe (konpilatzean automatikoki jartzen baitira).

Liburutegi Dinamikoa sortzeko

1. Liburutegia1.c-eko objetu modulua sortu:

   gcc -c liburutegia1.c -o liburutegia1.o -fPIC

2. Estekatzaileak liburutegia sortzen du:

ld –o liborokor.so liburutegia1.o –shared

-o liborokor.so aukerak liburutegiari eman nahi diogun izena adierazteko da.

-shared aukerak liburutegi bat (eta EZ exekutagarri bat) egin behar duela adierazteko da

-fPIC options enable ``position independent code'' generation, a requirement for shared libraries

Liburutegi Dinamikoa

3. Behin liburutegia sortuta, gure programa liburutegiarekin lotzeko:

gcc –o aplikazioa aplikazioa.c –I<path> -I<path>…

-L<path> -L<path2>…-Bdynamic liburutegia1 liburutegia2

Adb: **gcc –o aplikazioa aplikazioa.c –I. –L . –Bdynamic liborokor.so**

-Bdynamic aukerarekin liborokor.a hartu beharrean liborokor.so hartzen du.

Liburutegi Dinamikoa

4. Behin exekutagarria konpilatuta dugunean, azken pauso bat falta da. Programak liburutegi dinamikoan dagoan funtzio bati deitzerakoan, gure liburutegi dinamikoak bilatzera nora joan behar den sistemari esango diogu, horretarako, LD_LIBRARY_PATH aldagaia moldatuko dugu.
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/kepa/bin

   Eta ondoren idatzi : export LD_LIBRARY_PATH

5. Orain exekutatu dezakegu:./aplikazioa