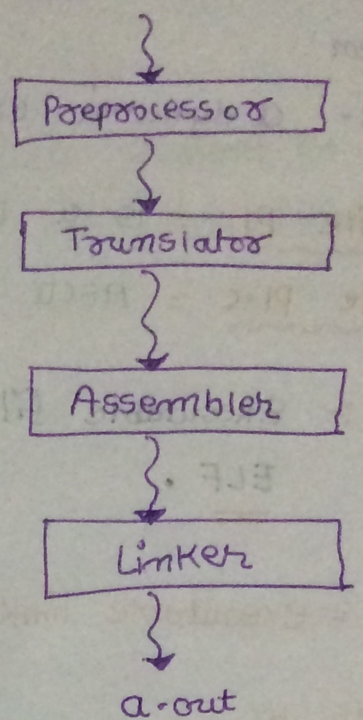
Question :- Default the library is connected to static or dynamic.

Answer :- dynamic

```
cc P1.c - o P2
cc - static P1.c - o P1
```



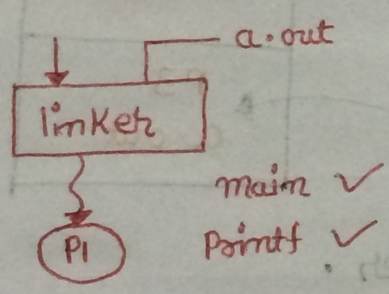
Full Form for ldd

List dynamic dependencies.

```
ldd P2
linux-gate.so.1 ⇒ (0x00d3e000)
libc.so.6 ⇒ libi386-linux-gnu
            (libc.so.6 (0x0085e000))
```

Static

1) cc - static P1.c - o P1



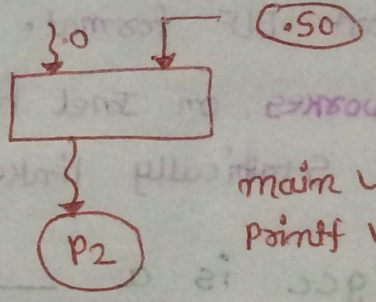
2) Size P1 is ~~small~~ large

3) text data bss dec hex
   546673 2024 7112 5558 87621

4) It does not require dependency.

dynamic

1) cc P1.c - o P2



2) Size P2 is small.

3) text 900 data 256 bss 8 dec 1164 hex 48c file P2

4) It require's dependency.

ldd Command [lddp2] :- (Imp)

It display's the dependancy for that executable file.

(*) .so means shared object

→ if we create a P3.c then only P2 library is not load once again only P3 library is loaded into RAM.

(*) P1 is static :-

⇒ If we write ldd command ldd P1

then

Note :- a dynamic executable.

(*) file P1 = is a ELF [executable link File].

file P1.c = ASCII text file.

Format :- Executable files has formate type which is known as ELF.

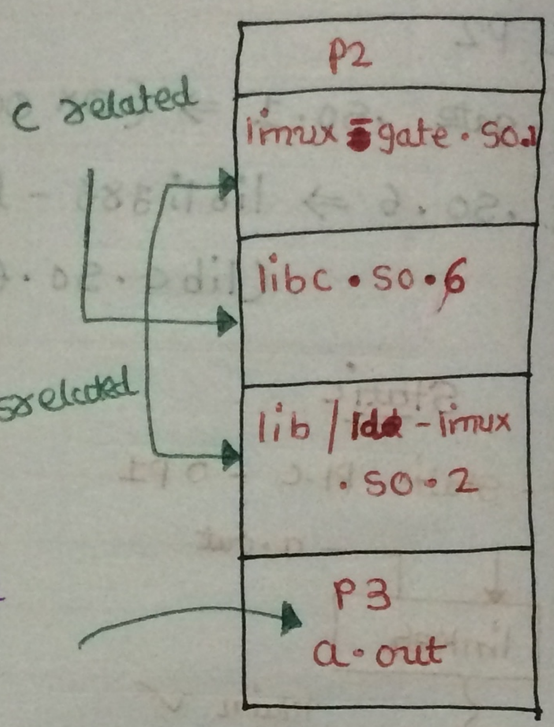
ELF = executable link file

(*) file P1 :-

⇒ Previous Version of Linux COFF format.

⇒ Now all new Version of Linux Support ELF format.

⇒ It workes on Intel hardware and it is Statically linked.



* armgcc is a _____ Compiler.

(*) file P2 :-

⇒ It is dynamically linked.

⇒ Prefer the dynamic if OS. is there.

(*) hm P2 command :-(dynamic)

⇒ It gives a information Such as

1) memory Address :-

2) memory section :-

3) Symbol name :- (function name)

memory Address

08048304 T main

T ← text file

main ← function name

U Printf

* nm P1 command :- (static)

08048ddo T Printf

(text file)

* In executable file not only our program is present in this present dependancy an lookuptable and all other information.

Vi main.c

Prg.c

```
#include <stdio.h>
main()
{
    int i=10, j=20, K;
    Printf ("Hello World ---\n");
    K = Sum (i,j);
    Printf ("K = %d\n", K);
    K = mul (i,j);
    Printf ("K = %d\n", K);
```

* Vi print.c

```
#include <stdio.h>
Void print (char *P)
{
    Printf ("%s", P);
}
```

* Vi Sum.c

```
int Sum (int i, int j)
{
    return (i+j);
}
```

* Vi mul.c

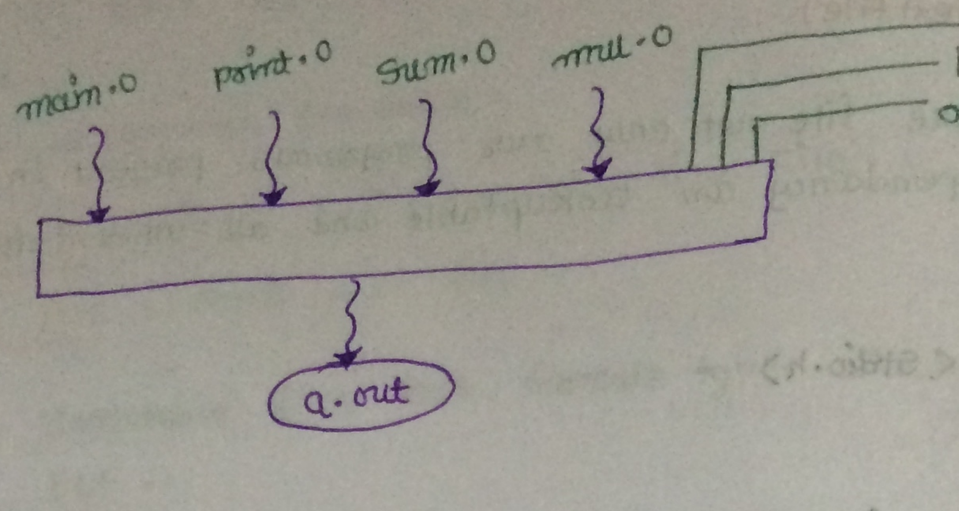
```
int mul (int i, int j)
{
    return (i*j);
}
```

fpic = position independence code

* To compile multiple files

* cc main.c print.c Sum.c

* cc -main.c print.c sum.c mul.c



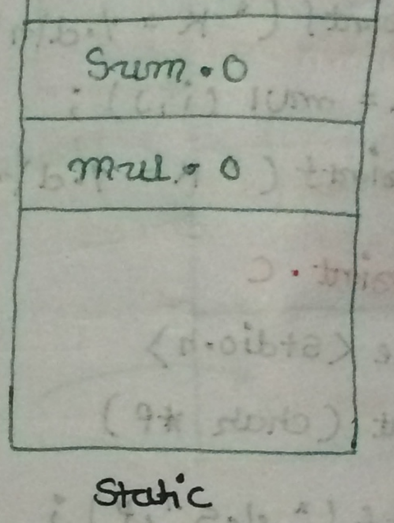
main ✓
Print ✓
Sum ✓
mul ✓
Printf ✗

☞ How to create our own static library?

1) cc -c print.c

⇒ cc -c -fpic print.c -o print.o
⇒ cc -c -fpic sum.c -o sum.o
⇒ cc -c -fpic mul.c -o mul.o

libabc.a



The command to create a static library is.

or ar rcs libabc.a print.o sum.o mul.o

ar rcs libabc.a print.o sum.o mul.o

This argv consits of series of letters.

r c s

Replacing Creation indexing

To Show How many object files in library.

* ar -t libabc.a

Print.o
Sum.o
mul.o

it display the file present in the library.

⊛ To delet a object files from library:

$ ar -d libabc.a print.o

⊛ To add the deleted object files from library.

$ ar -r libabc.a print.o (add new file in library)

⊛ To show the table of content of the library.

$ ar -tv libabc.a

Ans ==
```
rw-rw-r --- 2018/2018  856 Oct 16 19:37 2015 sum.o
rw-rw-r --- 2018/2018  852 Oct 16 19:37 2015 mul.o
rw-rw-r --- 2018/2018 1240 Oct 16 19:36 2015 print.o
```

Fat 32
Fat 16
} system

⊛ File P1 :-

⇒ ELF 32bit LSB executable.
⇒ Intel 80386
⇒ Version 1 (GNU/Linux), Statically linked
⇒ GNU/Linux 2.6.24
⇒ Build ID [sha 1] = 0xff40b8c0ldc3 ---
⇒ not stripped.

./dynamic is not compile using static like as

cc -static P1.c libdynamic.so -o P2
Then error generate.

⊛ Ldd P2 :-

* linux-gate.so.1 ⇒ (0xb77bd000) : OS related
* libc.so.6 ⇒ /lib/i386-linux-gnu/libc.so.6 : C related
* lib/ld-linux.so.2 : OS related

⊛ How to Creat our own dynamic library.

cc -shared -o libmma.so sum.o mul.o print.o

libmma.so
sum
mul

Que:- why static library is not in coloured?
why dynamic is in coloured?

Que :- How to link with our own created static and dynamic library ?

→ (For static) cc main.c libabc.a

Que :- is these executable file is statically linked for dynamic linked ?

Ans Dynamically

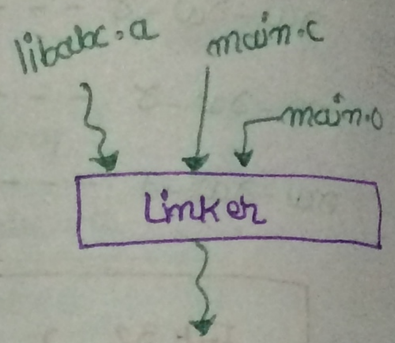
→ because printf is not there printf is in predefined library so.

For dynamic

dot slash

↓

cc main.c libmmno.so



main ✓
sum ✓
mul ✓
Print ✓
Printf ✗
↑
dynamic

ldd a.out

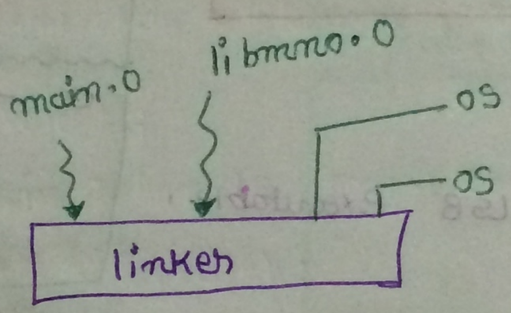
./a.out ↵

Hello world

K = 30

K = 200

main ✓
Print ✗
sum ✗
mul ✗
Printf ✗



1) main advantage of dynamic library is it saves the memory in RAM.

2) Fixing the bug is very easy. (Imp)

→ without disturping exciting file without uninstallation of our application if we want to fix the bug. if u want to add new feature go fo dynamic library creation.

→ Cross compiler support static library.

Que :- What is the meaning of static linking and dynamic linking ?

→ Linking or binding is a concept related to calling function and called function.

One :- Static linking :-

→ if calling function knows where exactly call function present at compile time then the linking between them is called as Static linking.

One :- dynamic linking :-

1) If calling function knows where exactly call function present at load time then linking betn them is called as dynamic load time linking.

2) if calling function knows where exactly call function present at runtime then linking betn them is called as dynamic run time linking.

→ if we write cc -static p1.c
at compile time blinding
betn main and printf.

→ if we write p1.c ↵
at load time blinding
betn main and printf

(*)
```
#include <stdio.h>
main()
{
    int i=10, j=20, k, op;
    printf("Enter number ----\n");
    Scanf("%d %d", &i, &j);
    Printf("Enter the op 1) Sum\n 2) mul\n");
    Scanf("%d", &op);
    if (op==1)
    {
        K = Sum(i,j);
        printf("K = %d\n", K);
    }
    else if (op==2)
    {
        K = mul(i,j);
        printf("K = %d\n", K);
    }
```

```
else
    printf (" Unknown number Enter\n ");
    while (1);
}
```

cc main.c error / undefined reference

cc main.c ./libmmo.so ↵

Static ↘

cc -static main.c libbc.a

./a.out

10

20

☞ To acheive the dynamic runtime linking without Function pointer it is not possible.

```
# include <dlfcn.h>
```

1) Void * dlopen (const char * filename , int flag);

2) char * dlerror (Void);

3) Void * dlsym (Void * handle , const char * symbol)

4) int dlclose (Void * handle);

1) dlopen :- It is useful for opening the dynamic library.

2) dl~~error~~ sym :- It is useful for finding the symbol address (function address) in library.

3) dlerror :- It is useful for displaying a proper error message.

4) dlclose :- It is useful for closing the library if required.

Flag in 1'St function :-

1) RTLD_Now :- is called early blinding.

2) RTLD_LASY :- it is called lasy (lazy) binding.

(*) Commana

cc -sh

⇒ Dynamic

How to link libraries?

to link wit

to link wi

⇒ compiler inside the library.

(*) advan

⇒ You same t that t

(*) fixin

Pl
Sh li cd

⇒ B

when

⇒