To generate a static library (object code archive file):

● Compile: cc-Wall-c ctest1.c ctest2.c

● Create library "libctest.a" by archiving it:

ar-cvq libctest.a ctest1.o ctest2.o

● List files in library:

ar-t libctest.a

● Linking with the library:

o Library in current directory:

cc-o executable-name prog.c libctest.a

o

Library in not in current directory:

cc-o executable-name prog.c-L/path/to/library-directory-lctest

● size executable-name

To generate a shared object (Dynamically linked object library file):

● Compile: gcc-Wall-fPIC-c \*.c

Here PIC meaning generate position independent code

● Create shared library "libctest.so"

gcc-shared-o libctest.so \*.o

● export LD\_LIBRARY\_PATH=.:LD\_LIBRARY\_PATH

● Linking with the library:

gcc-L . prog.c-l ctest-o prog

● size prog

Compare the size of shared and static libraries using size command

**Fork**

#include<stdio.h>

#include<string.h>

#include<sys/types.h>

#include<stdlib.h>

#include<unistd.h>

#include<wait.h>

intmain(intargc,char\*argv[]){

printf("MainFunction:\n");

intretval=1;

pid\_tpid=fork();

if(pid<0){

printf("Errorinforkoperation\n");

}

if(pid==0){

printf("PIDforChildprocess:%d\nPIDofitsparentprocess:%d\n",getpid(),getppid());

execl("./binsearch",argv[1],NULL);

}

else{

printf("PIDofparentprocess:%d\n",getpid());

wait(&retval);

if(WIFEXITED(retval)==1)

{

printf("Childterminatednormally\n");

}

else{

printf("Childterminatedabnormally\n");

exit(0);

}

}

return0;

}