$ gcc -c \*.c

To crate relocatable files of main.c and swap.c. We get corresponding two ‘.o’ files.

$ gcc \*.o -o myexe.out

To link and create the executable. Here it links two file and creates one exe file.

$ objdump -d -M intel main.o

$ objdump -d -M intel myexe.out | less

To check the content(assembly) in object file in intel format.

$ objdump -t main.o

$ readelf -s main.o

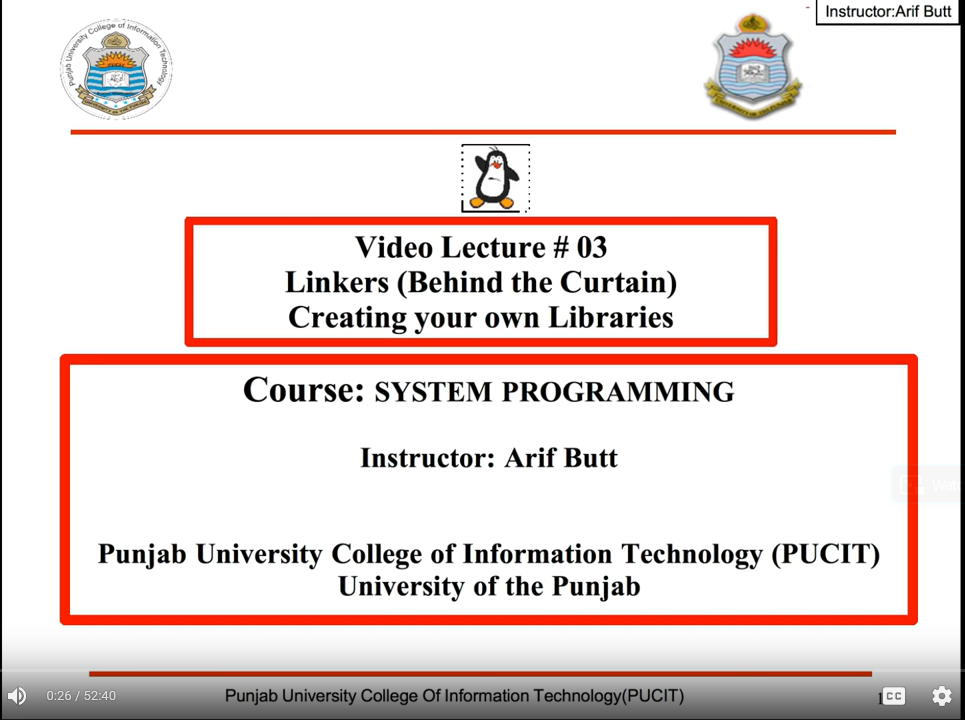
$ nm main.o

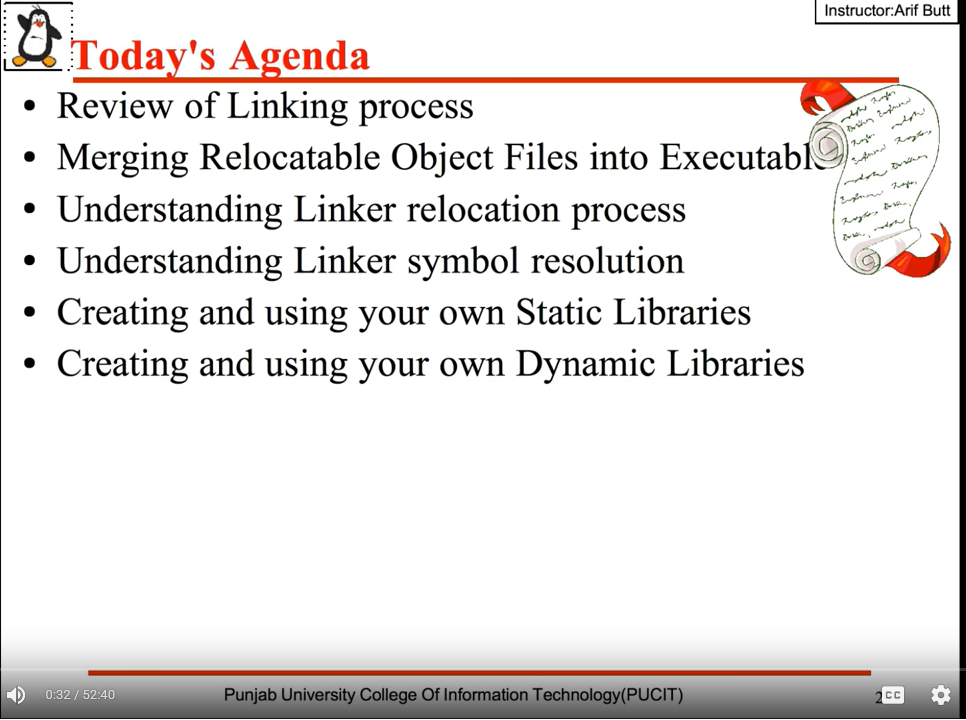
To see symbol table

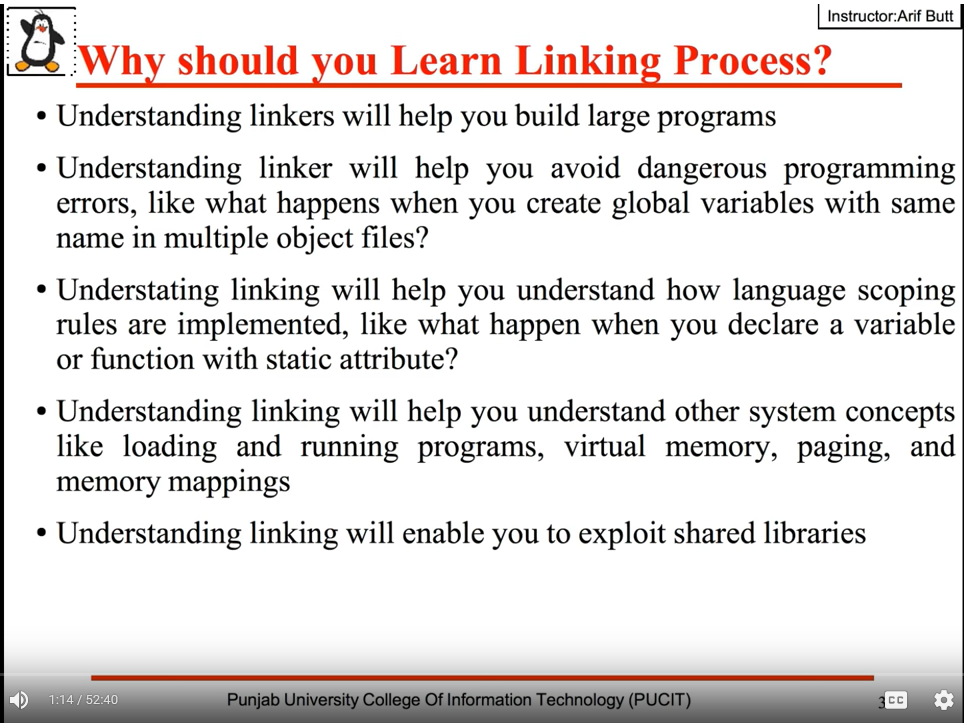
$ gcc --static main.o swap.o -o myexe2

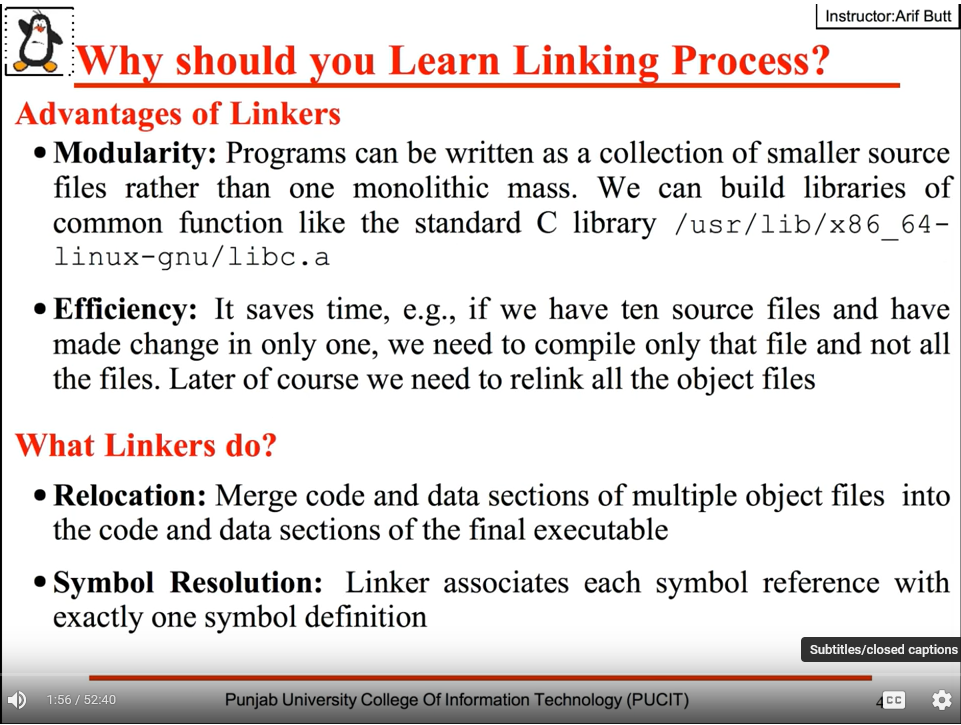
To generate statically linked executable.

$ nm myexe2 | less

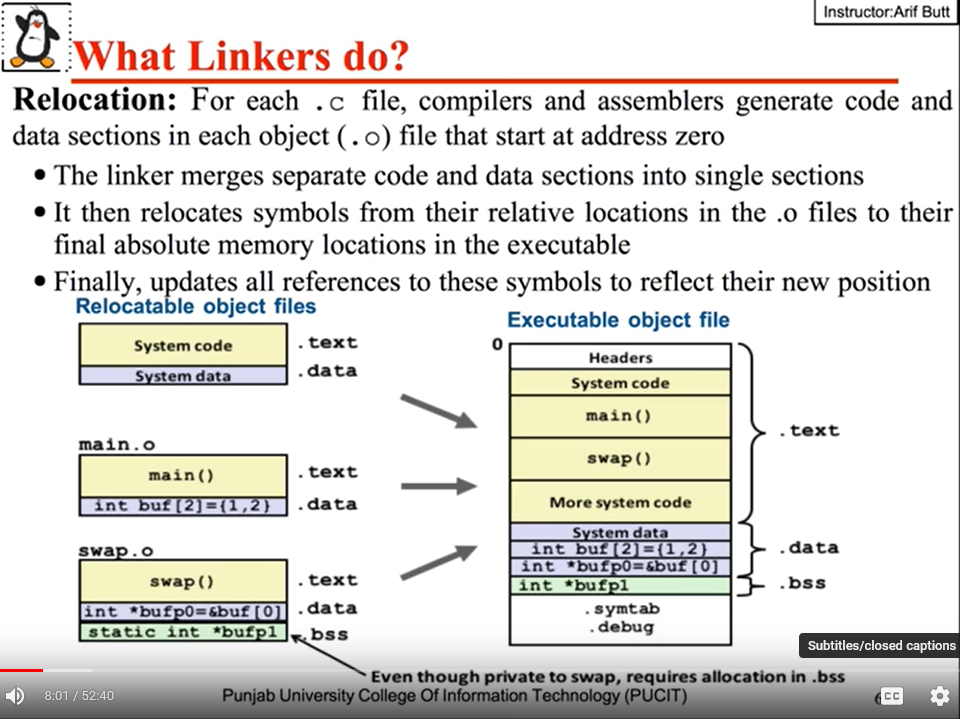


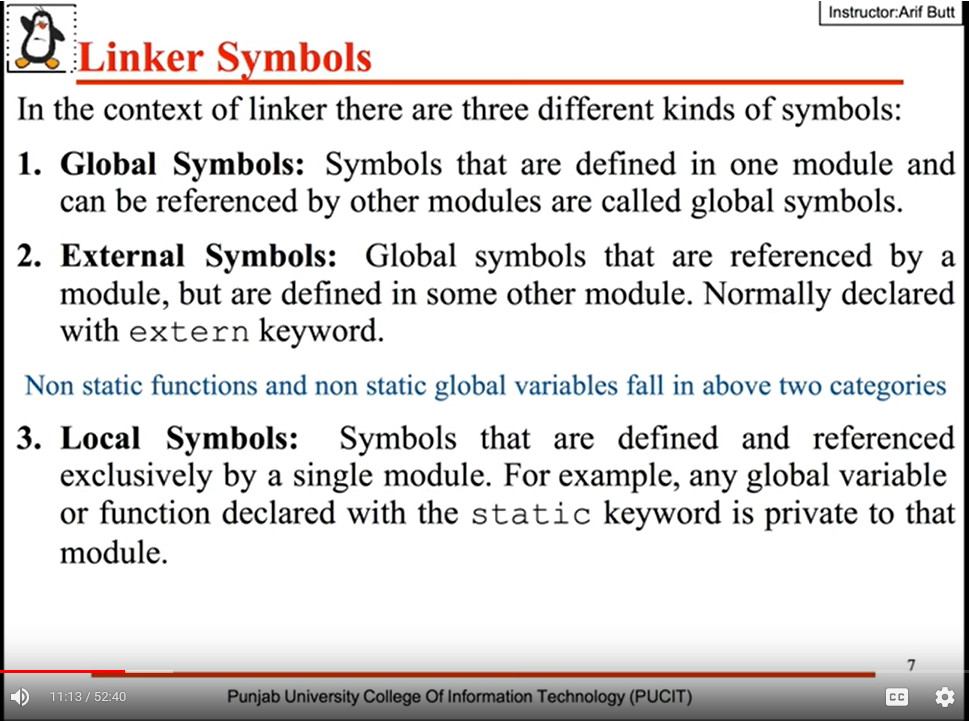


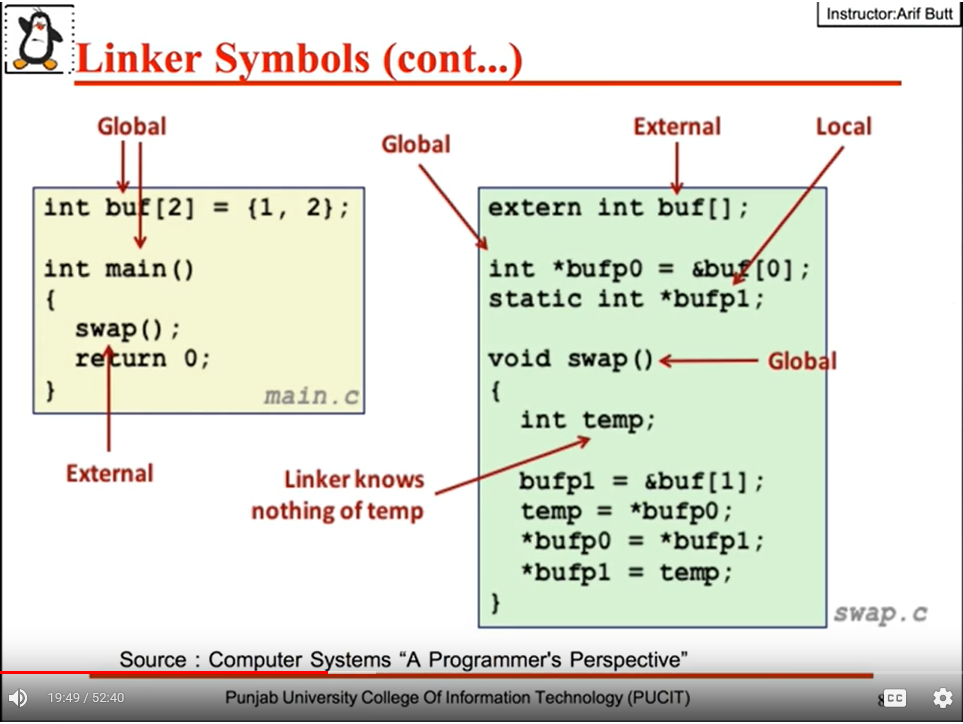


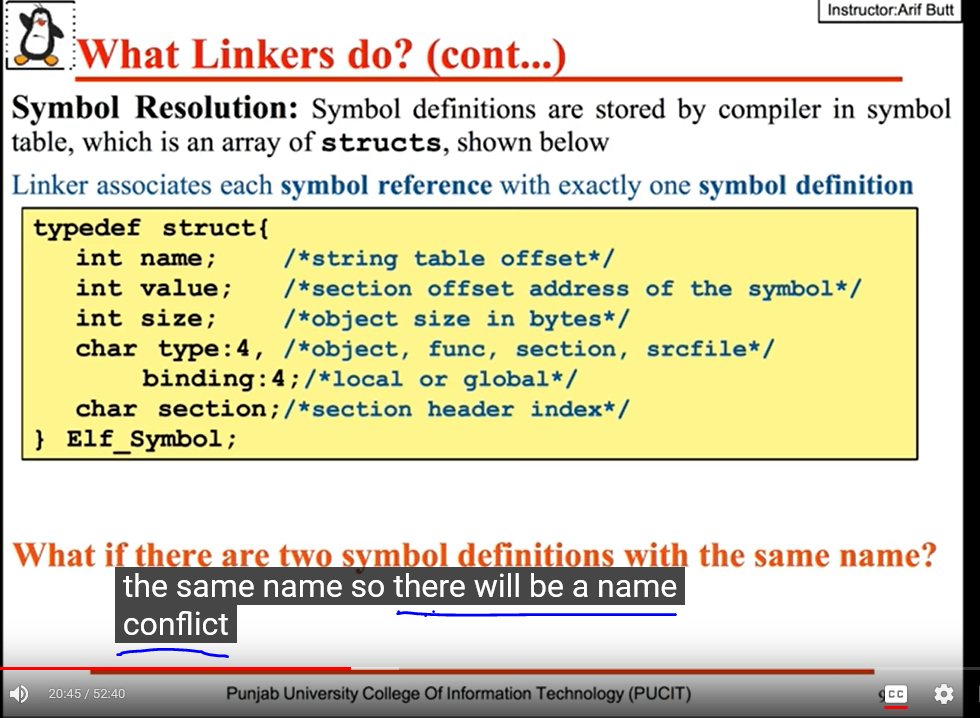


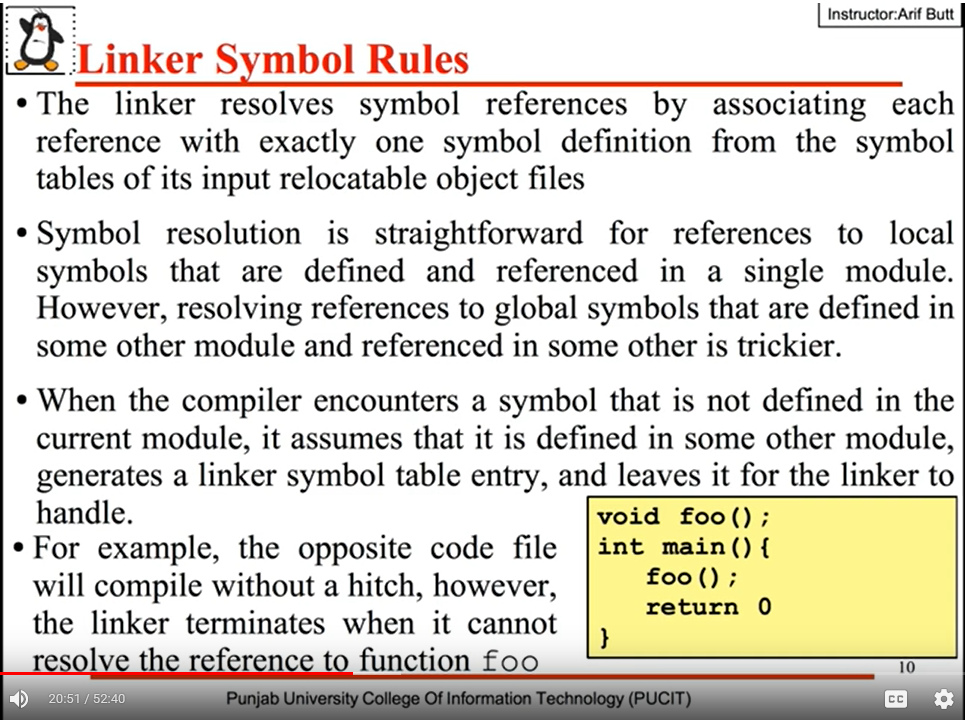


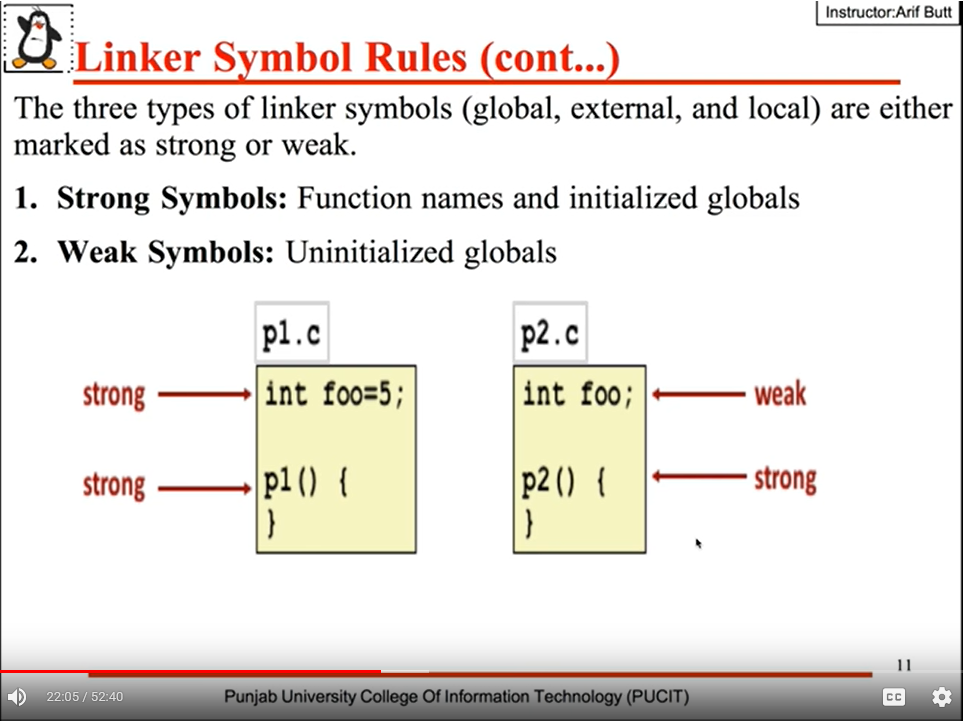


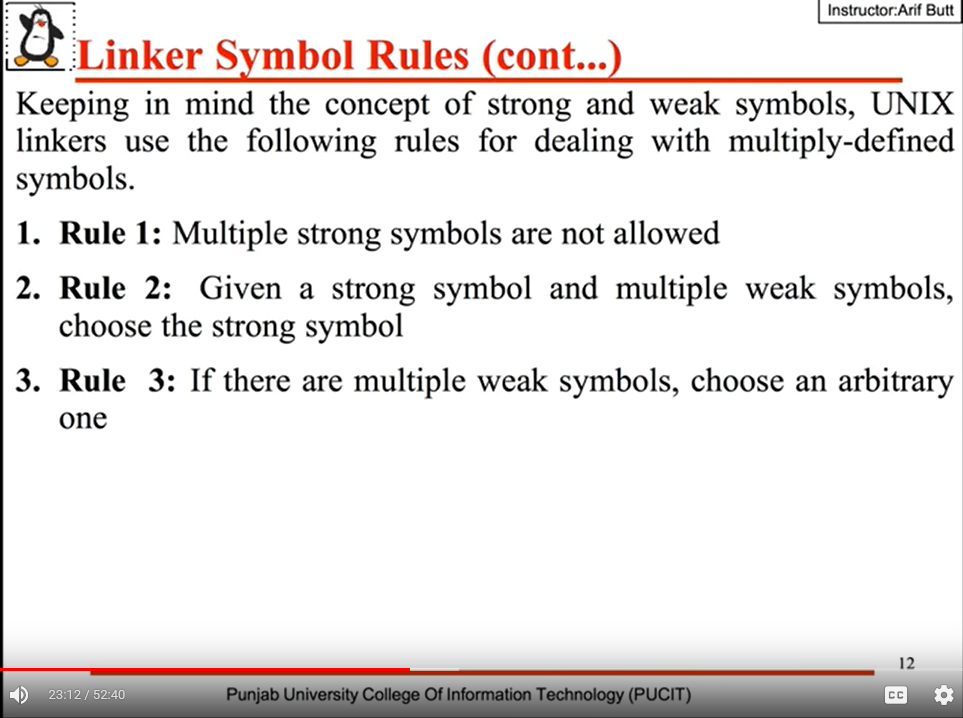
















$ man ar

Librarian or archiver to crate a library with the name of ‘libc.a’

$ ar -x /usr/lib/x86\_64-linux-gnu/libc.a

$ ar -x /usr/lib/gcc/x86\_64-redhat-linux/4.4.4/libgcc.a

To extract(-x) all ‘.o’ files from standard C library.

$ mkdir temp

$ cd temp; ar -x /usr/lib/x86\_64-linux-gnu/libc.a

To copy standard C library relocatable object into temp dir.

$ ls | wc

To check the number of relocatable object files in the standard c library.

$ ls | grep printf.o

**Creating Static Libraries:**

$ gcc -c \*.c

First we need to create object files.

$ ar -rs librohithmath.a myadd.o mysub.o mydiv.o mymul.o

$ ar -rs librohithmath.a \*.o

Next step is to archive all the object files and create a library.

-rs (r: will create library, s: will generate an index to the contents of this archive for speedy linking).

librohithmath.a library name(lib\_\_.a) it has to be start with ‘lib’ and end with ‘.a’ extension.

$ ar -t librohithmath.a

To see the contents of the library

**Writhing the driver program and use the library which we created:**

$ gcc -c prog1.c -I../

If header file is not present in present dir then use ‘-I<path>’.

$ nm prog1.o

To check the symbol table of prog1.o file

$ gcc prog1.o -o myexe

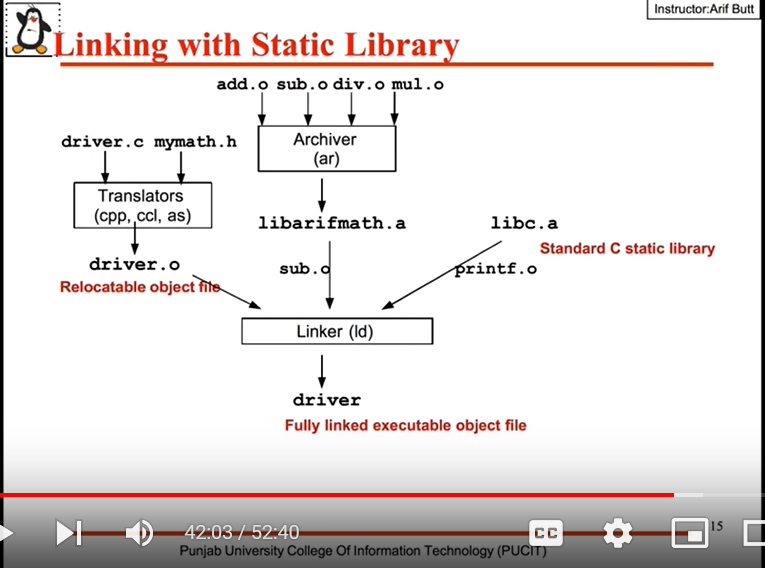
Not we will get error except ‘printf’ symbol, its because GCC implicitly link every program with the standard C library we don’t need to mention that. But for user defined libraries we need to explicitly the name of the library, option and linker name of the library.

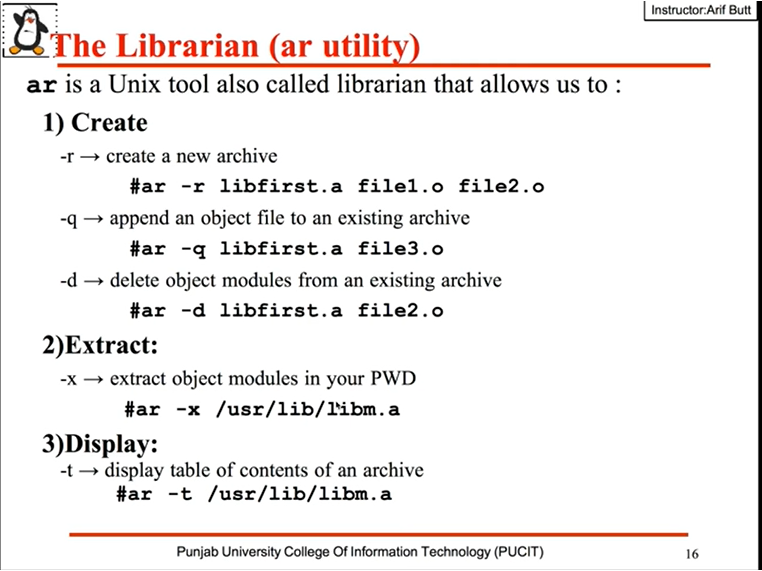
$ gcc prog1.o -o myexe -lrohithmath

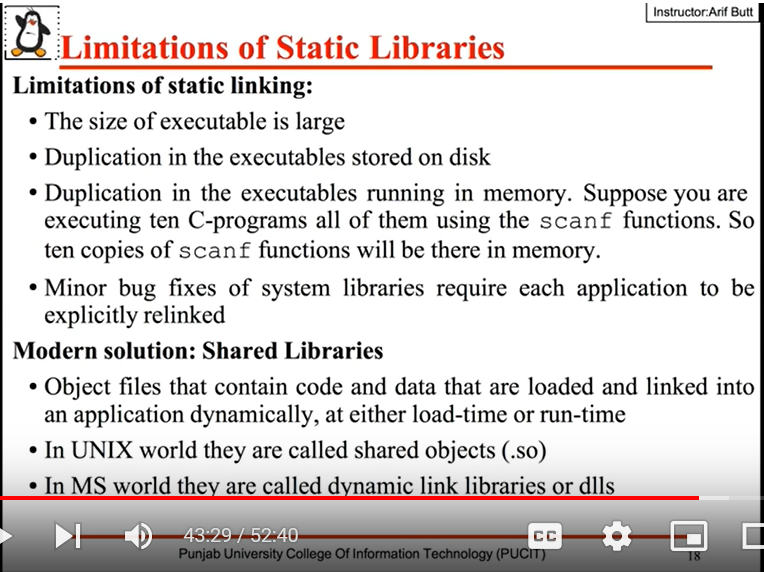
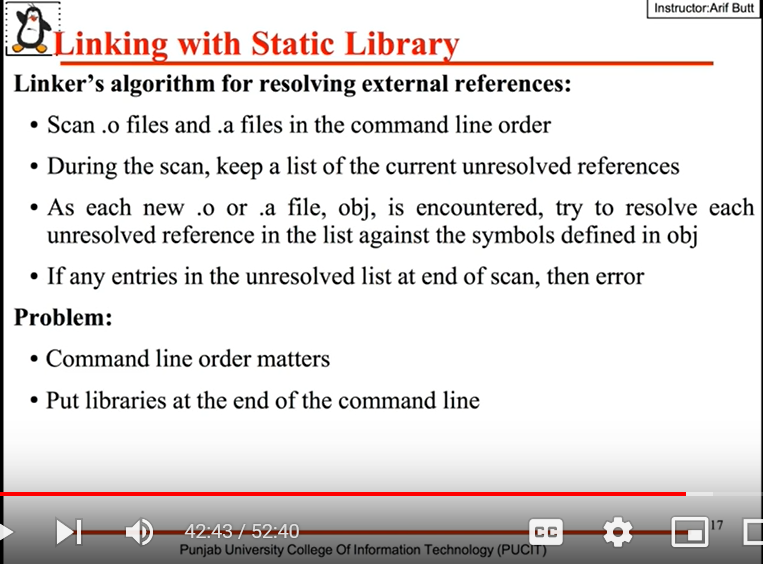
If the library file is in standard location then above command will execute fine.

$ gcc prog1.o -o myexe -lrohithmath -L../

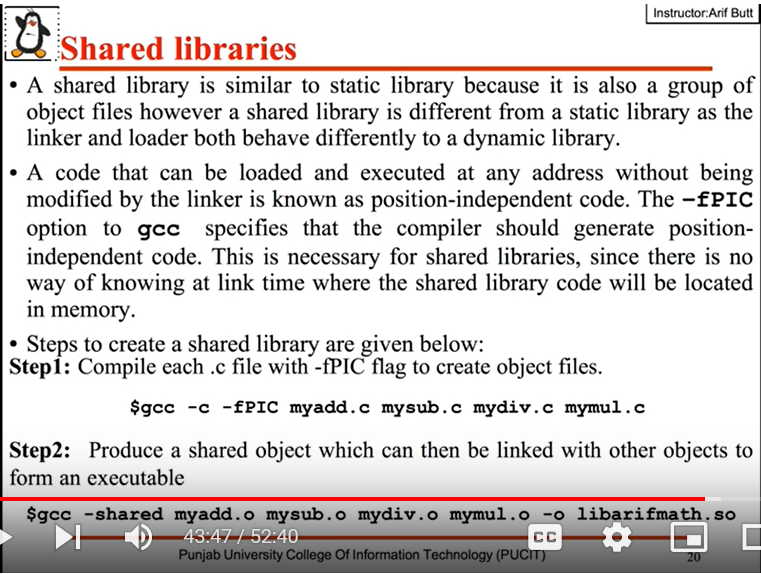
‘-L<path>’ will search for the library in given path.











$ gcc -c -fPIC \*.c

To create dynamic/shared libraries we need to generate object files using ‘-fPIC’.

$ gcc -shared \*.o -o librohithmath.so

To create shared library.

$ objdump -d -M intel librohithmath.so | less

To get the assembly code from the object/executable files. Use /<string> to search.

**Writing Driver program:**

$ gcc -c prot1.c -I../

$ gcc -c prot1.c -I ../

To create object file.

$ gcc prog1.o -o myexe -lrohithmath -L../

$ gcc prog1.o -o myexe -l rohithmath -L ../

To link with libraries

$ file myexe

To get the file info

$ ldd myexe

To list the dynamic dependencies of an executable.

$ nm myexe

If we run myexe we will get error stating user defined library unable to load. Reason is loader is searching for libraries at standard location. We can solve it in two ways we can copy library and past it in standard location or, update the environment variable LD library path.

$ export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/mnt/SysPrgmg/03/dynamiclib

$ ./myexe

Now it will run perfectly.

