GCC Tool

gcc - GNU project C and C++ compiler
compiler is a process of transulation.
build is a process of create packages.

Step1: Source file (first.c)
Step2: Compile
a. Preprocessor (first.c) to (first.i) gcc -E -v first.c -o first.i
b. Compile (first.i) to (first.s) gcc -S -v first.i -o first.s
c. Assembler (first.s) to (first.o) gcc -c -v first.s -o first.o
Step3: Build
a. Linker tool. gcc first.o -o first
b. Runtime code (Library Code)

objdump -D first.o | more (Relocatable binary code). Platform independent

objdump -D first | more (Executable binary code). Platform dependent.

Differnce between Linker and Assembler:

1. Relocatable binaries carry binary equivalent of the routines found in source.
2. Executale binaries carry instructions found in relocatable with additional **runtime code**.

Runtime code:
Runtime code is responsible to initalize the address space with additional resources required & use by functionality code.

********** ELF File Format**********
 # ~/ERTOS/gcc$ file first.c
first.c: ASCII text
 # ~/ERTOS/gcc$ file first.i
first.i: ASCII C program text
 # ~/ERTOS/gcc$ file first.s
first.s: ASCII assembler program text
 # ~/ERTOS/gcc$ file first.o
*first.o: **ELF** 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not stripped*
 # ~/ERTOS/gcc$ file first
*first: **ELF** 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.15, not stripped*
****

Executale and Linkable Format (ELF) is a standard binary file format.

readelf -a first | more (readelf is a details of ELF file)

creation of exec file using Dynamic linker
gcc first.c -o first (by default Dynamic)
creation of exec file using Static linker
gcc -static first.c -o firststatic

 # ~/ERTOS/gcc$ file firststatic

firststatic: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.6.15, not stripped

 # ~/ERTOS/gcc$ file first

first: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.15, not stripped

Differnce between Static Linker and Dynamic Linker:

1. Size is Different.
2. Static Linkers use static libraries which are appended the executable image at build time.
3. Dynamic Linkers use dynamic libraries which carrys symbolic reference in exectable and physically loaded at runtime.

Procedure for Creation of static libraries:

Step1: Implementaion of Source code.

one.c

two.c

Step2: Compile source code up to object file.

gcc -c one.c two.c

Step3: Use UNIX archive tools create library image.

ar -rcs libourown.a one.o two.o

Procedure for Creation of Dynamic libraries (Shared Libraries):

Step1: Implementaion of Source code.

one.c

two.c

Step2: Compile source to create position independent relocatale.

gcc -c -fpic one.c

gcc -c -fpic two.c

Step3:

gcc -shared -o libourown.so one.o two.o

~/ERTOS/gcc/lib$ file libourown.a
libourown.a: current ar archive
~/ERTOS/gcc/lib$ file libourown.so
libourown.so: ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked, not stripped
~/ERTOS/gcc/lib$ ar -t libourown.a
one.o
two.o
~/ERTOS/gcc/lib$ ar -t libourown.so
ar: libourown.so: File format not recognized

~/ERTOS/gcc/lib$ ar -t /usr/lib/libc.a

gcc app.c -I./ -o app ./libourown.a

gcc app.c -I./ -o appdyn ./libourown.so

~/ERTOS/gcc/lib$ file app
app: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.15, not stripped

~/ERTOS/gcc/lib$ file appdyn
appdyn: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.15, not stripped

~/ERTOS/gcc/lib$ gcc -static app.c -I./ -o app ./libourown.a
~/ERTOS/gcc/lib$ file app
app: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.6.15, not stripped.