1 SYSTEMS PROGRAMMING LABORATORY I - Getting Started

- **Examples:**
  - Compile and run the code.
  - Analyze the code and output.

1. If you don't know the name of the *man(1)* page you want, you can perform a keyword search. The following shows how you could search for information about changing owners of a file:

   ```
   $ man -k owner
   ```

2. Another way this can be done on most systems is to use the apropos(1) command:

   ```
   $ apropos owner
   ```

3. About gcc; try following commands

   ```
   $ gcc --version
   $ type cc
   cc is /usr/bin/cc
   $ ls -li /usr/bin/cc
   $ type gcc
   gcc is hashed (/usr/bin/gcc)
   $ ls -li /usr/bin/gcc
   ```

   Since both /usr/bin/cc and /usr/bin/gcc link to the same i-node 7951 (some number) in the example, you know that these two files are linked to the same executable file.

4. Compiling with GCC; following programs are given, compile and link. main.c reciprocal.cpp reciprocal.hpp
   Steps are;

   ```
   $ gcc -c main.c
   $ gcc -E main.c -o main.pp
   ```

   Examine **main.pp**.

   ```
   $ gcc -x cpp-output -c main.pp -o main.o
   $ g++ -c reciprocal.cpp
   $ g++ -c -D NDEBUG reciprocal.cpp
   $ g++ -c -D NDEBUG=3 reciprocal.cpp
   $ g++ -c reciprocal.cpp
   ```

Check the size of **reciprocal.o**.

```
$ g++ -c -O2 reciprocal.cpp
```

Compare the new size with the previous.

```
$ g++ -o reciprocal main.o reciprocal.o
$ ./reciprocal 7
The reciprocal of 7 is 0.142857
```

Link with a library say **libncurses**.

```
$ g++ -o reciprocal main.o reciprocal.o -lncurses
```

If it is not in your path, locate it by **locate** command; type

```
$locate  libncurses
```

Then say it is the path; /usr/local/lib/ncurses. Give the location of this library by

```
% g++ -o reciprocal main.o reciprocal.o -L/usr/local/lib/ncurses -lncurses
```

Check the sizes and compare with the cases you compiled without library flags.

```
% g++ -o reciprocal reciprocal.o main.o -lncurses -static
```

Check the sizes and compare with the cases you compiled without **static**.

5. Error Checking and Warnings Consider pedant.c. Steps are;

```
%gcc pedant.c -o pedant
```

this code compiles without complaint.

```
%gcc -ansi pedant.c -o pedant
```

Again, no complaint.

```
%gcc -pedantic pedant.c -o pedant
pedant.c: In function ‘main?:
pedant.c:9: warning: ANSI C does not support ‘long long?
```

The code compiles, despite the emitted warning. (These messages may be different.)

```
%gcc -pedantic-errors pedant.c -o pedant
pedant.c: In function ‘main?:
pedant.c:9: ANSI C does not support ‘long long?
```

With **-pedantic- errors**, however, it does not compile. GCC stops after emitting the error diagnostic.

- **Exercises:**

  1. Write

     `$man gcc`

     and read once to see how many possible flags we have.

  2. Compile, link and execute the following codes with the appropriate flags; $-Wall$, $-ansi$, $-pedantic$, $-pedantic-errors$, $-DSHOW\_PID$, ... asgn1.c asgn2.c asgn3.c uargs.c uvars.c Analyse and compare the outputs for different flags.

  3. Modify the files makefile and Makefile so that able to compile the files given above in one makefile with appropriate flags. We do not a specific purpose, just increase your ability to write a makefile.