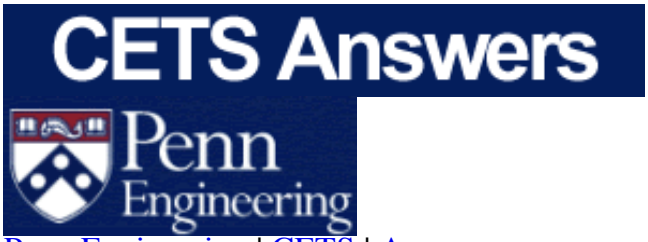




# How do I use gcc, g++, and gdb?

The C compiler on eniac is gcc. Its C++ counterpart is g++.

To compile a C or C++ program:

```
% gcc file.c
```

or

```
% g++ file.c
```

This compiles `file.c` into an executable binary named `a.out`.

Here are a few options to gcc and g++:

**-o *outputfile***
: To specify the name of the output file. The executable will be named `a.out` unless you use this option.

**-g**
: To compile with debugging flags, for use with gdb.

**-L *dir***
: To specify directories for the linker to search for the library files.

**-l *library***
: This specifies a library to link with.

**-I *dir***
: This specifies a directories for the compile to search for when looking for include files.

---

The debugger is gdb. Here is a typical example of a gcc/gdb session:

```
% cat hello.c
#include<stdio.h>

main() {
    int count;

    for (count=0;count<10;count++)
       printf("Hello from CETS!\n");
}
% gcc -g hello.c
% gdb ./a.out
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.13 (sparc-sun-solaris2.3),
Copyright 1994 Free Software Foundation, Inc...
(gdb) b main
Breakpoint 1 at 0x10784: file hello.c, line 6.
(gdb) r
Starting program: /home1/b/bozo/./a.out
```

```
Breakpoint 1, main () at hello.c:6
6           for (count=0;count<10;count++)
(gdb) s
7               printf("Hello from CETS!\n");
(gdb) p count
$1 = 0
(gdb) disp count
1: count = 0
(gdb) set count=8
(gdb) s
Hello from CETS!
6           for (count=0;count<10;count++)
1: count = 8
(gdb)
7               printf("Hello from CETS!\n");
1: count = 9
(gdb) c
Continuing.
Hello from CETS!

Program exited with code 01.
(gdb) q
%
```

Here are a few gdb commands:

**help**
: Will give you help on most gdb functions. If you wish for help on a specific command, type `help` *`command`*.

**b** ***function-name***
: To set a breakpoint at a function.

**r** ***args***
: To run the program. It will run until it reaches a breakpoint.

**s**
: To single-step through lines of code.

**c**
: To continue until the next breakpoint.

**p** ***variable***
: To print a variable's value.

**q**
: To quit gdb.