Week02 Assignment

> 班级：192115 学号：19373257 姓名：黄泽桓

1. 你安装好了 GCC、GDB、Make 工具了吗？记录安装命令

我的系统是：CentOS 7.6

安装命令：

```
yum -y install gcc
yum -y install gdb
```

2. 简述 GCC、GDB、Make 工具的作用

- GCC：原GNU C语言编译器，后扩展为支持更多编程语言，作为编译器，它能够预处理、编译、连接和汇编 C/C++ 语言，生成系统的可执行文件；
- GDB：调试 C/C++ 程序，可执行程序的启动、断点处停顿、查看过程的变量状态
- Make：控制可执行文件和其它一些从源代码来的非源码文件版本的软件，主要是读入 makefile 的文件，通过执行这个文件中的指令，来构建多个程序的依赖关系，从而编译和安装这个程序。

3. 尝试练习使用 GDB 命令

源码如下：

```
// gdbTest.c
#include<stdio.h>
int main()
{
    int a,b;
    scanf("%d%d",&a,&b);
    a++;
    b-=5;

    if(a == 0)
        a += 5;
    else
        b += 5;
    printf("%d",a*b);
    return 0;
}
```

- 使用 gcc 命令编译源代码，生成可调试的执行文件 `gcc -g gdbTest.c -o gdbTest`
- 依次写入 `gdb -q`, `file gdbTest` 和 `list`（展示源代码）

- 设置断点，利用 `break` 和 `disable`

```
(gdb) break 10
Breakpoint 1 at 0x4005d1: file gdbTest.c, line 10.
(gdb) break 15 if a==1
Breakpoint 2 at 0x4005ec: file gdbTest.c, line 15.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000004005d1 in main at gdbTest.c:10
2       breakpoint     keep y   0x00000000004005ec in main at gdbTest.c:15
        stop only if a==1
(gdb) disable 2
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000004005d1 in main at gdbTest.c:10
2       breakpoint     keep n   0x00000000004005ec in main at gdbTest.c:15
        stop only if a==1
```

- 使用 `run`, `print`, `next`, `continue` 命令进行调试

```
(gdb) run
Starting program: /home/zewan/Doc/Lab/02/gdbTest
5
6

Breakpoint 1, main () at gdbTest.c:10
10              if(a == 0)
Missing separate debuginfos, use: debuginfo-install glibc-2.17-307.el7.1.x86_64
(gdb) print a
$1 = 6
(gdb) next
13                      b += 5;
(gdb) continue
Continuing.
36[Inferior 1 (process 23432) exited normally]
```

- `q` 退出 gdb 调试

4. 请阐述静态链接库和动态链接库的异同点

- 静态链接库：和程序直接静态连接，占据较大内存
- 动态链接库：动态的共享库，在程序运行时与程序连接，占据内存小

5. 请阐述 Make 命令工具如何确定哪些文件需要重新生成，而哪些不需要生成

Make 工具根据 “Makefile” 特殊文件中的内容，来确定哪些文件需要生成，哪些不需要。

Makefile 文件中，规则、指令、目标这些信息提供了执行内容。当 make 命令第一次执行时，它扫描 Makefile，找到目标及其依赖后。如果这些依赖自身也是目标，则继续为这些依赖扫描 Makefile 建立其依赖关系，然后编译它们。一旦所有依赖都编译完成之后，则编译主目标。

6. 请简述 Make 中的伪目标的作用是什么

1. 当我们只需完成一系列操作，而不需要创建目标文件时，伪目标起作用，它能避免生成太多不必要的文件；
2. 提高执行 make 时的效率，省略不必要的扫描；
3. 当需要生成若干个可执行文件时，声明例如“all”的伪目标，可以节省多行命令，简洁高效。