Week02 Assignment

> 班级：192112
> 学号：19373073
> 姓名：何潇龙

1. 你安装好了 GCC、GDB、Make 工具了吗？记录安装命令

我的系统是：Ubuntu 16.04.10

安装命令：

```
sudo apt-get install build-essential
```

2. 简述 GCC、GDB、Make 工具的作用

GCC 是 Linux 平台下最常用的编译程序，把代码编译成二进制文件。

GDB是LINUX用来调试C和 C++程序的调试器。可以运行程序，设置所有的能影响程序运行的参数和环境。 控制程序在指定的条件下停止运行。当程序停止时，可以检查程序的状态。修改程序的错误，并重新运行程序。 动态监视程序中变量的值。

Make 是一个自动化的 程序自动维护工具。它根据 Makefile 所描述的"依赖关系" 自动决定项目的那些部分需要重新编译。

3. 尝试练习使用 GDB 命令

- 任意编写一个 C 程序，通过截图等方式，说明你在调试过程中的一些尝试

C程序：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void bubbleSort(int k[],int n){
    int i,j,flag=1;
    int temp;
    for(i=n-1;i>0 && flag==1;i--){
        flag=0;
        for(j=0;j<i;j++)
            if(k[j]>k[j+1]){
                temp=k[j];
                k[j]=k[j+1];
                k[j+1]=temp;
                flag=1;
            }
    }
}
int main()
{
    int n,m,i,tmp=0;
    scanf("%d",&n);
```

```
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    bubbleSort(a,n);
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    return 0;
}
```

1. 首先打开gdb工具

```
charlot@charlot-virtual-machine:~/Desktop/hello$ gdb
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
```

2. 用`file [文件名]`引入调试文件

```
(gdb) file bubbleSort
Reading symbols from bubbleSort...done.
```

3. 用`list（l）`命令列举出原代码

```
(gdb) l
6               int i,j,flag=1;
7               int temp;
8               for(i=n-1;i>0&&flag==1;i--){
9               flag=0;
10              for(j=0;j<i;j++)
11                      if(k[j]>k[j+1]){
12                              temp=k[j];
13                              k[j]=k[j+1];
14                              k[j+1]=temp;
15                              flag=1;
(gdb) l
16                      }
17              }
18      }
19      int main()
20      {
21              int a[1000];
22              int n,m,i;
23              scanf("%d",&n);
24              for(i=0;i<n;i++)
25                      scanf("%d",&a[i]);
(gdb) l
26              bubbleSort(a,n);
27              for(i=0;i<n;i++)
28                      printf("%d",a[i]);
29              return 0;
30      }
(gdb)
```

4. 用`b 26`在第26行增加断点，`run`执行程序，用`s`进入函数执行，`n`执行下一条语句，`p`打印变量当前值

```
(gdb) b 26
Breakpoint 1 at 0x8048609: file bubbleSort.c, line 26.
(gdb) run
Starting program: /home/charlot/Desktop/hello/bubbleSort
10
5 3 6 4 7 2 1 9 8 10

Breakpoint 1, main () at bubbleSort.c:26
26              bubbleSort(a,n);
(gdb) s
bubbleSort (k=0xbfffdf6c, n=10) at bubbleSort.c:6
6               int i,j,flag=1;
(gdb) n
8               for(i=n-1;i>0&&flag==1;i--){
(gdb) n
9               flag=0;
(gdb) n
10              for(j=0;j<i;j++)
(gdb) n
11                      if(k[j]>k[j+1]){
(gdb) n
12                              temp=k[j];
(gdb) n
13                              k[j]=k[j+1];
(gdb) n
14                              k[j+1]=temp;
(gdb) n
15                              flag=1;
(gdb) n
10              for(j=0;j<i;j++)
(gdb) n
11                      if(k[j]>k[j+1]){
(gdb) n
10              for(j=0;j<i;j++)
(gdb) n
11                      if(k[j]>k[j+1]){
(gdb) n
12                              temp=k[j];
(gdb) n
13                              k[j]=k[j+1];
(gdb) n
14                              k[j+1]=temp;
(gdb) n
15                              flag=1;
(gdb) n
10              for(j=0;j<i;j++)
(gdb) n
11                      if(k[j]>k[j+1]){
(gdb) n
10              for(j=0;j<i;j++)
(gdb) p j
$1 = 3
(gdb) p i
$2 = 9
(gdb) p k[1]
$3 = 5
(gdb)
```

5. 用`continue`执行断点后的命令，用`q`退出调试。

```
(gdb) continue
Continuing.
12345678910[Inferior 1 (process 2244) exited normally]
(gdb) q
charlot@charlot-virtual-machine:~/Desktop/hello$
```

4. 请阐述静态链接库和动态链接库的异同点

同

共享代码，代码封装，都会产生lib文件

异

静态链接库：lib包含函数代码本身，当要使用时，连接器会找出程序所需的函数，然后将它们拷贝到执行文件，由于这种拷贝是完整的，所以一旦连接成功，静态程序库也就不再需要了。

动态链接库：lib包含了函数所在的dll文件和文件中函数位置的信息，某个程序在运行中要调用某个动态链接库函数的时候，操作系统首先会查看所有正在运行的程序，看在内存里是否已有此库函数的拷贝了。如果有，则让其共享那一个拷贝；只有没有才链接载入。

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

根据被修改的文件以及依赖关系确定。如果某个源程序文件被修改，那么依赖这个源程序文件的所有目标文件，都需要重新编译：如果仅修改了某几个源文件，则只重新编译这几个源文件；如果某个头文件被修改了，则重新编译所有包含该头文件的源文件。

6. 请简述 Make 中的伪目标的作用是什么

“伪目标”并不是一个文件，只是一个标签。使用伪目标可以避免只执行命令的目标和工作目录下的实际文件出现名字冲突并提高执行make时的效率