作业一，使用GDB调试程序

一，设置断点在main函数入口，使用next单步跳跃执行。

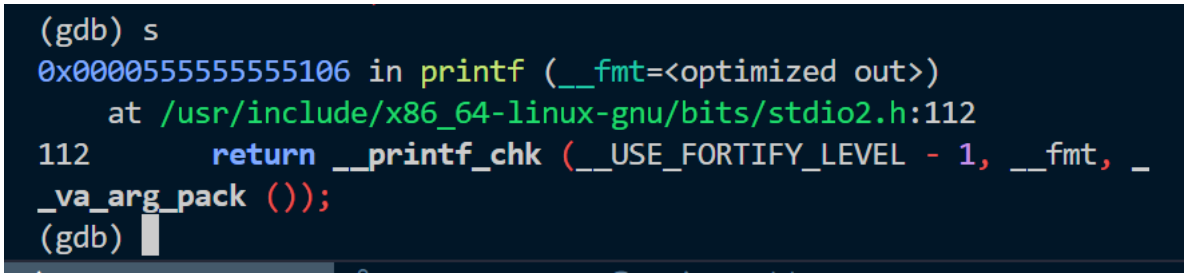
```
rdtsc.c
27  int main(int argc, char **argv)
28  {
29      int tmp=0;
30      uint64_t cycles1, cycles2;
31      struct timespec ts1, ts2;
32
33      printf("這個程式是量測一個指令執行的時間, 但CPU可同時執行數
34      printf("因此這些量測方法比較適合量測大範圍的程式碼\n\n");
35

問题  输出  调试控制台  终端  端口

(gdb) b main
Breakpoint 1 at 0x1100: file rdtsc.c, line 28.
(gdb) r
Starting program: /home/tan/Desktop/linux-system-programming/ch0
2/rdtsc
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread
_db.so.1".

Breakpoint 1, main (argc=1, argv=0x7fffffffdce8) at rdtsc.c:28
28      {
(gdb) n
33          printf("這個程式是量測一個指令執行的時間，但CPU可同
時執行數十個指令\n");
(gdb)
```

二，step命令，遇到函数会进入



```
(gdb) s
0x0000555555555106 in printf (__fmt=<optimized out>)
    at /usr/include/x86_64-linux-gnu/bits/stdio2.h:112
112       return __printf_chk (__USE_FORTIFY_LEVEL - 1, __fmt, _
_va_arg_pack ());
(gdb)
```

三，打印变量的值，列出局部变量

```
(gdb) n
40          clock_gettime(CLOCK_MONOTONIC, &ts1);
(gdb) info local
tmp = 1
cycles1 = 4362236243124
cycles2 = 4363673102177
ts1 = {tv_sec = 0, tv_nsec = 0}
ts2 = {tv_sec = 0, tv_nsec = 0}
__PRETTY_FUNCTION__ = "main"
(gdb)
```

四，打印调用栈，以及调用参数，有的参数会被编译器优化掉。

```
(gdb) bt
#0  __GI___clock_gettime (clock_id=1, tp=0x7fffffffdb90)
    at ../sysdeps/unix/sysv/linux/clock_gettime.c:30
#1  0x0000555555555166 in main (argc=<optimized out>,
    argv=<optimized out>) at rdtsc.c:40
(gdb) info local
tmp = 1
cycles1 = 4362236243124
cycles2 = 4363673102177
ts1 = {tv_sec = 0, tv_nsec = 0}
ts2 = {tv_sec = 0, tv_nsec = 0}
__PRETTY_FUNCTION__ = "main"
(gdb)
```

上下调整当前栈帧

```
(gdb) down
#0  __GI___clock_gettime (clock_id=1, tp=0x7fffffffdb90)
    at ../sysdeps/unix/sysv/linux/clock_gettime.c:30
30
(gdb) info locals
r = <optimized out>
vdso_time64 = <optimized out>
resultvar = <optimized out>
__arg2 = <optimized out>
__arg1 = <optimized out>
_a2 = <optimized out>
_a1 = <optimized out>
(gdb)
```

五，watch监听变量的修改

```
(gdb) watch cycles1
Watchpoint 3: cycles1
(gdb) info watchpoints
Num     Type           Disp Enb Address    What
3       watchpoint     keep y              cycles1
(gdb)
```

六，修改程序，故意访问错误的内存地址

添加如下代码

```
int *memerr = (int *)0;
printf("dereference of null pointer %d", *memerr);
```

```
tan@pop-os:~/Desktop/linux-system-programming/ch02$ make
gcc -g -O3 rdtsc.c -o rdtsc
tan@pop-os:~/Desktop/linux-system-programming/ch02$ ./rdtsc
Segmentation fault (core dumped)
```

发生段错误，程序收到11号信号，默认将该程序终止，当然这个信号处理函数是可以由用户自定义，JVM就自定义了这个信号的处理函数。

```
sighandler_t signal(int signum, sighandler_t handler);
```