作业三

1.

设计程序的代码部分：

```c
#include<stdlib.h>
int main()
{
    int * p = NULL;
    free(p);
    return 0;
}
```

我们对其编译运行：

null

null.c

```
oslab@oslab-virtual-machine:~/桌面/zy3$ ./null
oslab@oslab-virtual-machine:~/桌面/zy3$
```

发现能正常编译运行，不会报错。

2.

```
oslab@oslab-virtual-machine:~/桌面/zy3$ gdb null
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 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 "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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"...
Reading symbols from null...
(gdb) run
Starting program: /home/oslab/桌面/zy3/null
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Inferior 1 (process 4792) exited normally]
```

可以看到显示了 Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".表示其加载了 libthread_db 这个线程调试支持库。

4.

程序代码部分:

```c
#include<stdlib.h>
int main()
{
    int * p = (int *)malloc(sizeof(int));
    return 0;
}
```

运行该程序:

```
oslab@oslab-virtual-machine:~/桌面/zy3$ gdb malloc
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 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 "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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"...
Reading symbols from malloc...
(gdb) run
Starting program: /home/oslab/桌面/zy3/malloc
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Inferior 1 (process 5440) exited normally]
```

发现并没有报错出现，但是程序本身具有内存泄漏的问题，所以我们使用 gdb 调试来查看该问题。在 mian 处设置断点，然后往下执行的时候查看其指向内存地址：

```
Breakpoint 1, main () at /home/oslab/桌面/zy3/malloc.c:4
4               int * p = (int *)malloc(sizeof(int));
(gdb) n
5               return 0;
(gdb) print *p
$4 = 0
(gdb) print p
$5 = (int *) 0x5555555592a0
(gdb) n
6       }
(gdb) print p
$6 = (int *) 0x5555555592a0
(gdb) n
```

我们发现 gdb 调试并不能显示出内存泄漏的问题，我们需要使用 Valgrind。

```
==8078== Memcheck, a memory error detector
==8078== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==8078== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==8078== Command: ./malloc
==8078==
==8078==
==8078== HEAP SUMMARY:
==8078==     in use at exit: 4 bytes in 1 blocks
==8078==   total heap usage: 1 allocs, 0 frees, 4 bytes allocated
==8078==
==8078== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==8078==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck
d64-linux.so)
==8078==    by 0x10915E: main (malloc.c:4)
==8078==
==8078== LEAK SUMMARY:
==8078==    definitely lost: 4 bytes in 1 blocks
==8078==    indirectly lost: 0 bytes in 0 blocks
==8078==      possibly lost: 0 bytes in 0 blocks
==8078==    still reachable: 0 bytes in 0 blocks
==8078==         suppressed: 0 bytes in 0 blocks
==8078==
==8078== For lists of detected and suppressed errors, rerun with: -s
==8078== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

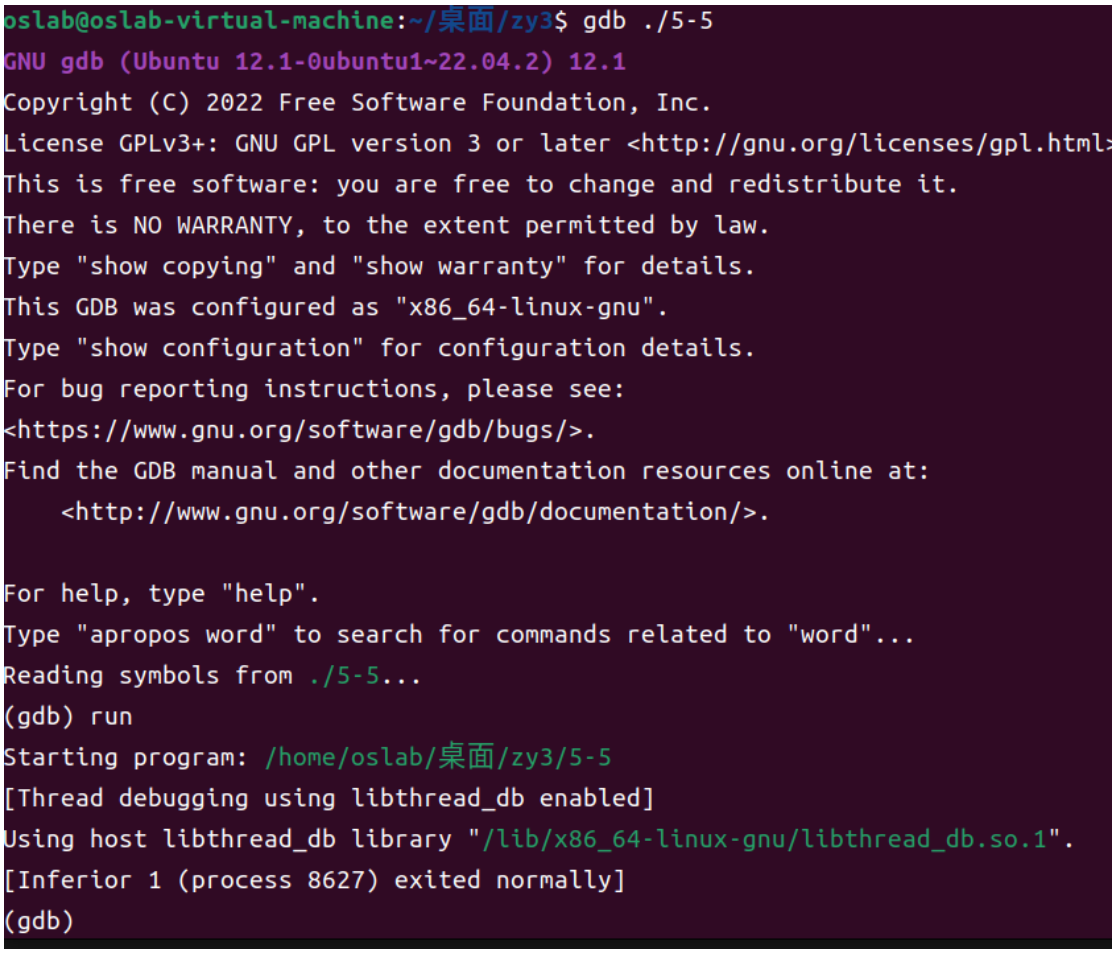
从结果中我们可以看到，程序在退出时仍有 4 个字节的内存未释放（很可能就是 malloc(sizeof(int)) 分配的那部分）。“definitely lost” 部分就说明存在明确的内存泄漏。

5.

程序代码部分：

```c
#include<stdlib.h>
int main()
{
    int * data = (int *)malloc(100*sizeof(int));
    data[100] = 0;
    free(data);
    return 0;
}
```

运行结果：




可以正常运行没有报错。再使用 valgrind 工具运行:

```
oslab@oslab-virtual-machine:~/桌面/zy3$ valgrind --leak-check=full ./5-5
==8835== Memcheck, a memory error detector
==8835== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==8835== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==8835== Command: ./5-5
==8835==
==8835== Invalid write of size 4
==8835==    at 0x10918D: main (5-5.c:5)
==8835==  Address 0x4a981d0 is 0 bytes after a block of size 400 alloc'd
==8835==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memche
d64-linux.so)
==8835==    by 0x10917E: main (5-5.c:4)
==8835==
==8835==
==8835== HEAP SUMMARY:
==8835==     in use at exit: 0 bytes in 0 blocks
==8835==   total heap usage: 1 allocs, 1 frees, 400 bytes allocated
==8835==
==8835== All heap blocks were freed -- no leaks are possible
==8835==
==8835== For lists of detected and suppressed errors, rerun with: -s
==8835== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

我们可以看到，其中有：

```
==8835== Invalid write of size 4
==8835==    at 0x10918D: main (5-5.c:5)
```

说明文件的第五行也就是内存分配有问题。

```
==8835==  Address 0x4a981d0 is 0 bytes after a block of size 400 alloc'd
==8835==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-am
d64-linux.so)
==8835==    by 0x10917E: main (5-5.c:4)
```

说明了存在越界写入的问题。

6.

程序代码部分：

```c
#include<stdlib.h>
#include<stdio.h>
int main()
{
    int * p = (int *)malloc(100*sizeof(int));
    p[1] = 10;
    p[2] = 20;
    free(p);
    printf("%d %d\n", p[1],p[2]);
    return 0;
}
```

运行结果:

```
oslab@oslab-virtual-machine:~/桌面/zy3$ ./5-6
5 -531510379
oslab@oslab-virtual-machine:~/桌面/zy3$ ./5-6
5 -434138107
oslab@oslab-virtual-machine:~/桌面/zy3$ ./5-6
6 -1674011839
oslab@oslab-virtual-machine:~/桌面/zy3$ ./5-6
5 -670909204
oslab@oslab-virtual-machine:~/桌面/zy3$
```

看到每次运行之后打印出的值都不一样。