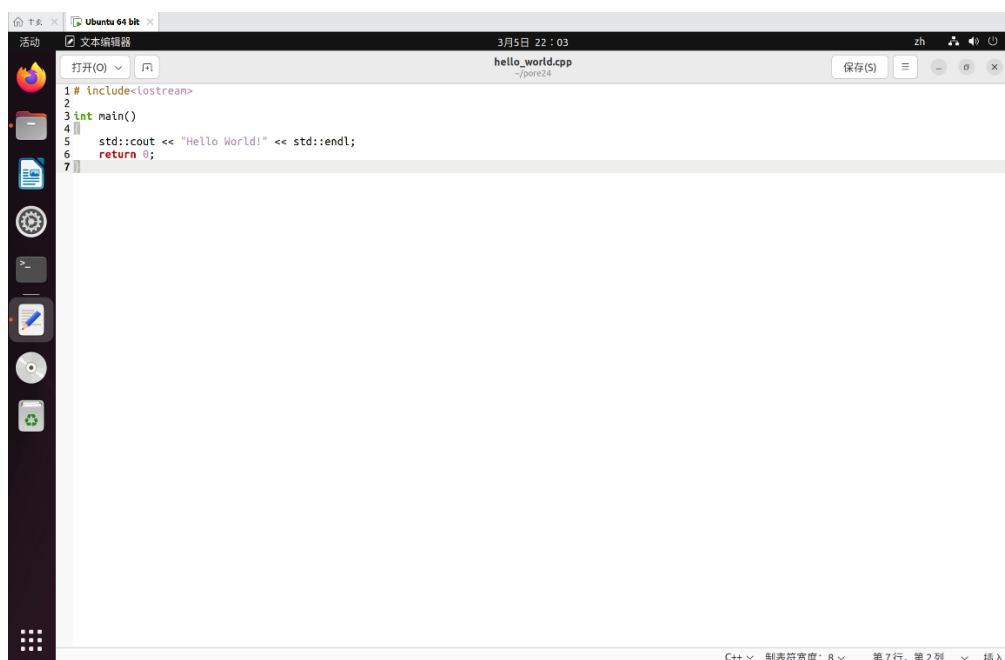


Lab2 访问 C++类中的私有成员

杨乙 21307130076 信息安全

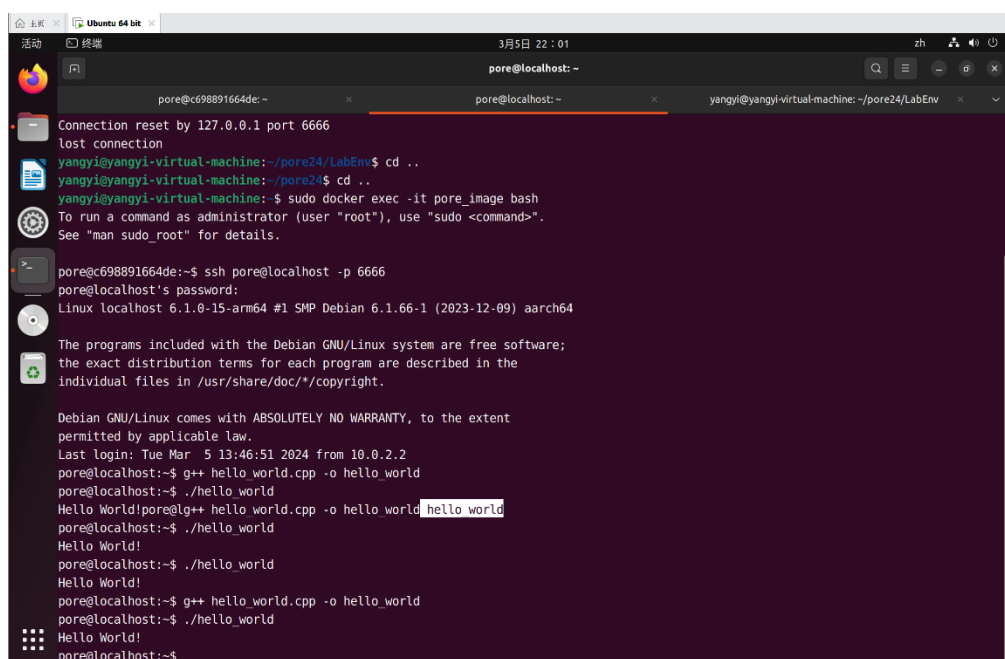
Task1

1. HelloWorld 程序截图：



```
1 #include<iostream>
2
3 int main()
4 {
5     std::cout << "Hello World!" << std::endl;
6     return 0;
7 }
```

2. qemu 虚拟机执行二进制文件输出截图：



```
pore@localhost: ~
pore@localhost: ~
yangyi@yangyi-virtual-machine: ~/pore24/LabEnv$ cd ..
yangyi@yangyi-virtual-machine: ~/pore24$ cd ..
yangyi@yangyi-virtual-machine: $ sudo docker exec -it pore_image bash
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

pore@c698891664de:~$ ssh pore@localhost -p 6666
pore@localhost's password:
Linux localhost 6.1.0-15-arm64 #1 SMP Debian 6.1.66-1 (2023-12-09) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar 5 13:46:51 2024 from 10.0.2.2
pore@localhost:~$ g++ hello_world.cpp -o hello_world
pore@localhost:~$ ./hello_world
Hello World!pore@g++ hello_world.cpp -o hello_world hello world
pore@localhost:~$ ./hello_world
Hello World!
pore@localhost:~$ ./hello_world
Hello World!
pore@localhost:~$ g++ hello_world.cpp -o hello_world
pore@localhost:~$ ./hello_world
Hello World!
pore@localhost:~$
```

Task2

1. classA 内存布局:

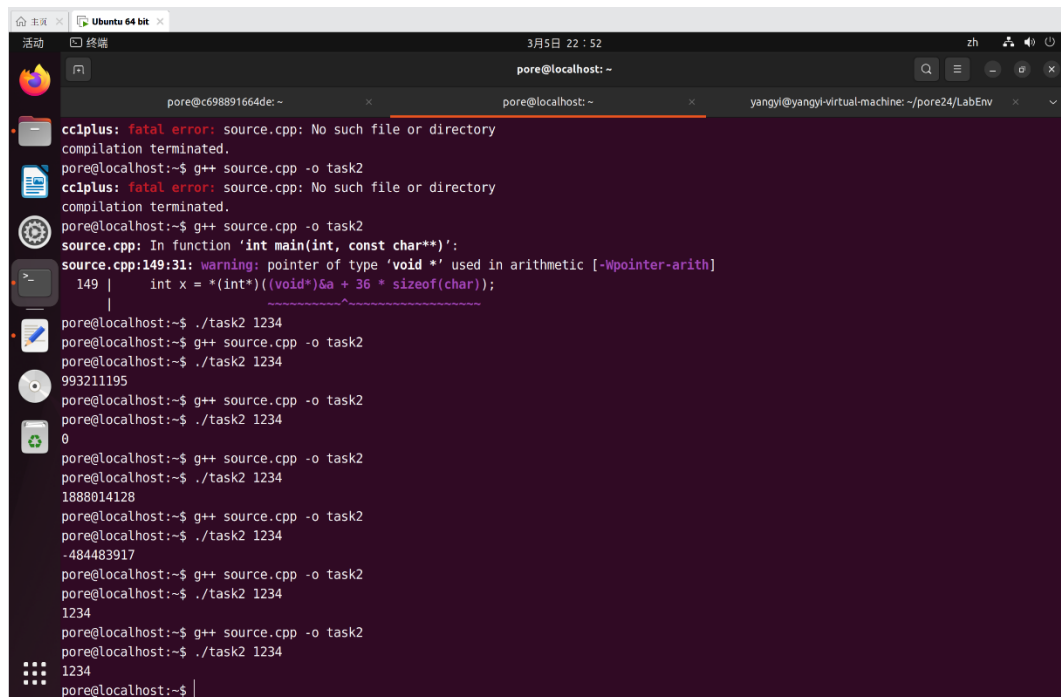
数字 1: 成员变量相对于对象内存起始位置的偏移量

数字 2: 成员变量占用字节数

文字: 计算过程

// 虚函数表指针	0	8	指针按 8B 对齐, 占 8B
char var_slCmNhpAkJ;	8	1	char 按 1B 对齐, 占 1B
char var_oJtyRgsipC;	9	1	char 按 1B 对齐, 占 1B
char var_GamptJtzVn;	10	1	char 按 1B 对齐, 占 1B
			空 1B
short var_FrCydvdgFx;	12	2	short 按 2B 对齐, 占 2B
short var_cysbDjRwMN;	14	2	short 按 2B 对齐, 占 2B
long long var_wZJoFTGZis;	16	8	long long 按 8B 对齐, 占 8B
char var_akbHmPSMIT;	24	1	char 按 1B 对齐, 占 1B
			空 1B
short var_vWvJpokbCx;	26	2	short 按 2B 对齐, 占 2B
int var-TaVbTFJZHq;	28	4	int 按 4B 对齐, 占 4B
char var_CpIPWIXozm;	32	1	char 按 1B 对齐, 占 1B
			空 3B
int x;	36	4	int 按 4B 对齐, 占 4B
char var_mNfPaLBcXn;	40	1	char 按 1B 对齐, 占 1B
			空 3B
int var_oXAcBMJVnE;	44	4	int 按 4B 对齐, 占 4B
long long var_TEDgYhSfzG;	48	8	long long 按 8B 对齐, 占 8B
short var_PZkcTDEqYF;	56	2	short 按 2B 对齐, 占 2B
			空 6B
long long var_TSEYnDWGgf;	64	8	long long 按 8B 对齐, 占 8B
long long var_AcxkVzLyIA;	72	8	long long 按 8B 对齐, 占 8B
long long var_uoRmmTyRUY;	80	8	long long 按 8B 对齐, 占 8B
short var_YvCzCJMFvZ;	88	2	short 按 2B 对齐, 占 2B
			空 2B
int var_JQgVFmqUxU;	92	4	int 按 4B 对齐, 占 4B
long long var_SaQJgQRtFa;	96	8	long long 按 8B 对齐, 占 8B

2. 输出截图



The screenshot shows a terminal window with the following content:

```
pore@c698891664de: ~  
pore@localhost: ~  
yangyl@yangyl-virtual-machine: ~/pore24/LabEnv  
cc1plus: fatal error: source.cpp: No such file or directory  
compilation terminated.  
pore@localhost:~$ g++ source.cpp -o task2  
cc1plus: fatal error: source.cpp: No such file or directory  
compilation terminated.  
pore@localhost:~$ g++ source.cpp -o task2  
source.cpp: In function 'int main(int, const char**):  
source.cpp:149:31: warning: pointer of type 'void *' used in arithmetic [-Wpointer-arith]  
149 |     int x = *(int*)((void*)&a + 36 * sizeof(char));  
      |  
pore@localhost:~$ ./task2 1234  
pore@localhost:~$ g++ source.cpp -o task2  
pore@localhost:~$ ./task2 1234  
993211195  
pore@localhost:~$ g++ source.cpp -o task2  
pore@localhost:~$ ./task2 1234  
0  
pore@localhost:~$ g++ source.cpp -o task2  
pore@localhost:~$ ./task2 1234  
1888014128  
pore@localhost:~$ g++ source.cpp -o task2  
pore@localhost:~$ ./task2 1234  
-484483917  
pore@localhost:~$ g++ source.cpp -o task2  
pore@localhost:~$ ./task2 1234  
1234  
pore@localhost:~$ g++ source.cpp -o task2  
pore@localhost:~$ ./task2 1234  
1234  
pore@localhost:~$
```

3. 代码思路:

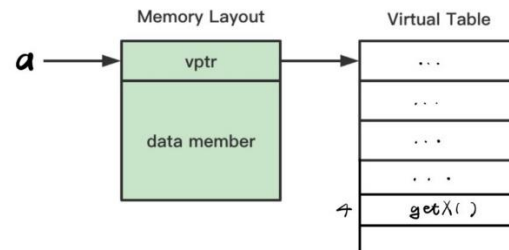
```
int x = *(int*)((unsigned long long)&a + 36 * sizeof(char));  
printf("%d\n", x);
```

因为变量 x 相对于对象 a 在内存中起始地址偏移量为 36B，因此先把 a 的起始地址强制类型转化为 unsigned long long，再加上 36B 的偏移量；再强制转化为 int 指针类型后，再取出指针指向的数，即为 x 的值

Task3

1. 调用 getX() 的原理:

- (1) 首先要确定 getX() 的地址。因为 getX() 是第五个虚函数，所以对象 a 的虚函数表如下:



要先获得 vptr 的值，再通过偏移量或下标访问 getX() 的地址:

```
// 通过偏移量访问
typedef unsigned long long u64; // 指针为 8B
u64 vptr = *(u64*)&a;           // vptr 的值
u64 getXptr = *((u64*)(vptr + 4 * sizeof(u64)));
// getX 函数指针的值
// vptr 加上 4*8B 的偏移量
// 先转化为 u64*类型
// 再取指针指向的值即为函数地址
// 转化为函数指针类型

func f = (func)getXptr;

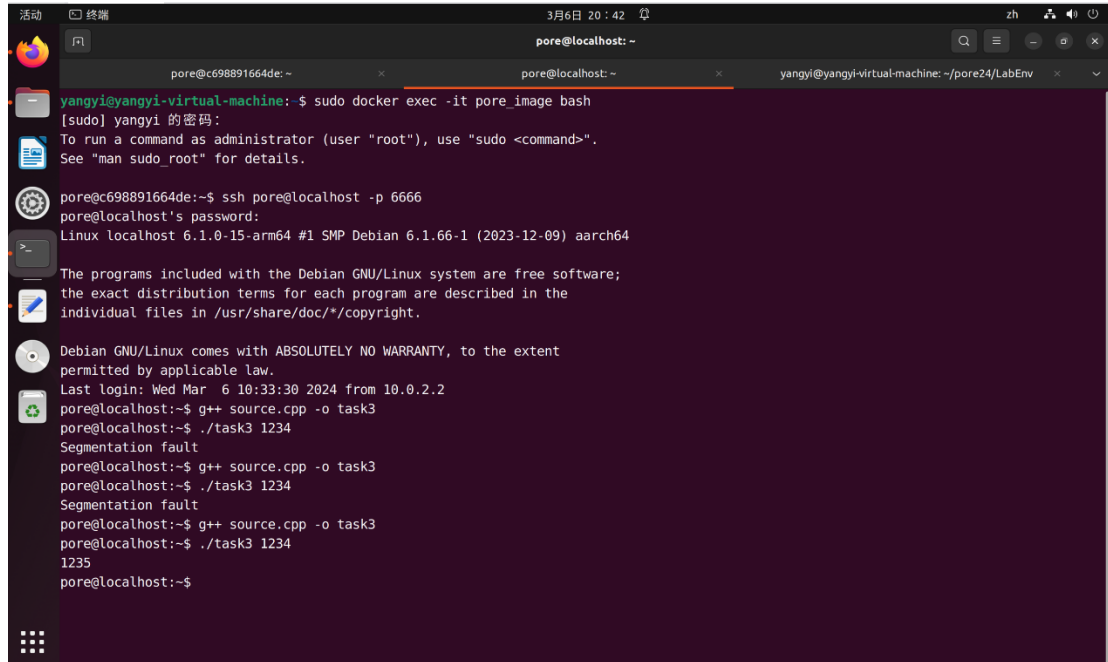
// 通过下标访问
typedef unsigned long long u64; // 指针为 8B
u64* vptr = (u64*)&a;           // vptr 的指针
u64* vtable = (u64*)*vptr;      // vptr 值，即 vtable 地址
func f = (func)(vtable[4]);     // 将 vtable 看作 u64 类型的数组
// 访问下标为 4 的元素
// 转化为函数指针类型
```

- (2) 其次要通过函数指针调用 getX()。要注意编译器在编译成员函数时会额外添加一个参数，传入当前对象的指针，用于访问成员变量。因此函数指针要定义形参 A*，调用函数指针时要传入参数&a，用于访问成员变量 x

```
typedef int(*func)(A*);          // 定义形参 A*
// .....
printf("%d\n", f(&a));           // 传入参数&a
```

（我在实验中一开始忽略了函数参数，导致函数将错误的值当作了当前对象的指针，访问了无意义的地址，返回结果是一个很大的负数）

2. 输出截图：



```
3月6日 20:42
pore@localhost: ~
yangyi@yangyi-virtual-machine: ~$ sudo docker exec -it pore_image bash
[sudo] yangyi 的密码:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

pore@c698891664de:~$ ssh pore@localhost -p 6666
pore@localhost's password:
Linux localhost 6.1.0-15-arm64 #1 SMP Debian 6.1.66-1 (2023-12-09) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar  6 10:33:30 2024 from 10.0.2.2
pore@localhost:~$ g++ source.cpp -o task3
pore@localhost:~$ ./task3 1234
Segmentation fault
pore@localhost:~$ g++ source.cpp -o task3
pore@localhost:~$ ./task3 1234
Segmentation fault
pore@localhost:~$ g++ source.cpp -o task3
pore@localhost:~$ ./task3 1234
1235
pore@localhost:~$
```