

OS大作业（救救共享）

克隆该仓库：

```
1 | git clone https://gitcode.com/M0rtzz/zzu-cs-os-design.git  
2 | cd zuu-cs-os-design
```

第一题

①准备

```
1 | mkdir score  
2 | touch score/1.txt score/2.txt score/3.txt
```

内容如下：

1.txt

```
1 | 李娜 郑州大学 2021级 1班 85  
2 | 张伟 郑州大学 2021级 1班 72  
3 | 刘波 郑州大学 2021级 1班 93  
4 | 王芳 郑州大学 2021级 1班 68  
5 | 陈丽 郑州大学 2021级 1班 78  
6 | 杨洋 郑州大学 2021级 1班 91  
7 | 赵静 郑州大学 2021级 1班 87  
8 | 孙强 郑州大学 2021级 1班 58  
9 | 周浩 郑州大学 2021级 1班 82  
10 | 吴婷 郑州大学 2021级 1班 79
```

2.txt

```
1 | 萧燕 郑州大学 2021级 2班 88  
2 | 方莹 郑州大学 2021级 2班 76  
3 | 高翔 郑州大学 2021级 2班 95  
4 | 何敏 郑州大学 2021级 2班 70  
5 | 江燕 郑州大学 2021级 2班 81  
6 | 孔涛 郑州大学 2021级 2班 90  
7 | 李莉 郑州大学 2021级 2班 73  
8 | 马超 郑州大学 2021级 2班 55  
9 | 宁娜 郑州大学 2021级 2班 84
```

3.txt

```
1 | 彭辉 郑州大学 2021级 3班 92  
2 | 齐东 郑州大学 2021级 3班 74  
3 | 任梅 郑州大学 2021级 3班 89  
4 | 沈磊 郑州大学 2021级 3班 52  
5 | 陶杰 郑州大学 2021级 3班 80  
6 | 田芳 郑州大学 2021级 3班 86  
7 | 王维 郑州大学 2021级 3班 75  
8 | 吴雯 郑州大学 2021级 3班 69
```

②代码

```

1 #!/bin/zsh
2
3 # -----
4 # @file: score.sh
5 # @brief: 合并分数文件，并输出年级排名前十、各分数区间人数、平均分
6 # @author: M0rtzz
7 # @date: 2024-04-30
8 # -----
9
10 rm -f ./score/sorted_scores.txt
11
12 # 按分数逆序排序
13 {
14     for file in ./score/*.txt; do
15         cat "${file}" | awk '{print $1, $4, $5}'
16         # 文件之间添加换行符
17         printf "\n"
18     done
19 } | sort -k3 -rn >./score/sorted_scores.txt
20
21 # 输出年级排名前十
22 echo "年级排名前十："
23 head -n 10 ./score/sorted_scores.txt
24
25 # 统计各分数区间人数
26 echo "60以下人数："
27 awk '$3 < 60 {count++} END {print count}' ./score/sorted_scores.txt
28
29 echo "60-70人数："
30 awk '$3 >= 60 && $3 < 70 {count++} END {print count}'
31     ./score/sorted_scores.txt
32
33 echo "70-80人数："
34 awk '$3 >= 70 && $3 < 80 {count++} END {print count}'
35     ./score/sorted_scores.txt
36
37 echo "80-90人数："
38 awk '$3 >= 80 && $3 < 90 {count++} END {print count}'
39     ./score/sorted_scores.txt
40
41 # 平均分
42 echo -n "平均分："
43 awk '{sum+=$3; count++} END {print sum/count}' ./score/sorted_scores.txt
44
45 # 删除后三行的换行符
46 sed -i '$d' ./score/sorted_scores.txt
47 sed -i '$d' ./score/sorted_scores.txt
48 sed -i '$d' ./score/sorted_scores.txt

```

③运行

```
1 | sudo chmod +x score.sh
2 | ./score.sh
```

```
1 | cat ./score/sorted_scores.txt
```

```
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design 100x31
> cat ./score/sorted_scores.txt
高翔 2班 95
刘波 1班 93
彭辉 3班 92
杨洋 1班 91
孔涛 2班 90
任梅 3班 89
萧燕 2班 88
赵静 1班 87
田芳 3班 86
李娜 1班 85
宁娜 2班 84
周浩 1班 82
江燕 2班 81
陶杰 3班 80
吴婷 1班 79
陈丽 1班 78
方莹 2班 76
王维 3班 75
齐东 3班 74
李莉 2班 73
张伟 1班 72
何敏 2班 70
吴雯 3班 69
王芳 1班 68
孙强 1班 58
马超 2班 55
沈磊 3班 52
```

第二题

①安装LLVM工具集（使用清华源）及依赖库

Reference: <https://mirrors.tuna.tsinghua.edu.cn/help/llvm-apt/>

```
1 | wget -O - https://apt.llvm.org/llvm-snapshot.gpg.key | sudo apt-key add -
```

```
1 | echo -e "deb [arch=amd64] https://mirrors.tuna.tsinghua.edu.cn/llvm-apt/focal/
  llvm-toolchain-focal main\n# deb-src [arch=amd64]
  https://mirrors.tuna.tsinghua.edu.cn/llvm-apt/focal/ llvm-toolchain-focal
  main" | sudo tee /etc/apt/sources.list.d/llvm-apt.list > /dev/null && sudo cp
  /etc/apt/sources.list.d/llvm-apt.list /etc/apt/sources.list.d/llvm-
  apt.list.save
```

```
1 | sudo apt update -y && sudo apt upgrade -y && sudo apt install clang clangd
  llvm clang-format liblldb-19-dev
```

```
1 | clang --version
```

```
m0rtzz@m0rtzz:~ m0rtzz@m0rtzz:~ 100x24 base 17:04:33
❯ clang --version
Ubuntu clang version 19.0.0 (++20240421052952+48324f0f7b26-1~exp1~20240421173100.1000)
Target: x86_64-pc-linux-gnu
Thread model: posix
InstalledDir: /usr/lib/llvm-19/bin
❯
m0rtzz@m0rtzz:~ 17:04:35
```

②构建工具

```
1 CC = clang
2 CFLAGS = -std=gnu11 -Wall -g -O2 -pthread # 避免使用-std=c11, 否则将无法使用一些必要的函数和类型
3 SRC_FILES := $(wildcard *.c)
4 OUT_DIR := out
5 OUT_FILES := $(patsubst %.c,$(OUT_DIR)/%.out,$(SRC_FILES))
6
7 all: $(OUT_FILES)
8
9 $(OUT_DIR)/%.out: %.c | $(OUT_DIR)
10      $(CC) $< -o $@ $(CFLAGS)
11
12 $(OUT_DIR):
13      mkdir -p $(OUT_DIR)
14
15 .PHONY: clean
16
17 clean:
18      rm -rf $(OUT_DIR)
```

③代码

```
1 /**
2 * @file pv.c
3 * @brief 生产者、计算者、消费者问题
4 * @author M0rtzz E-mail : m0rtzz@outlook.com
5 * @version 1.0
6 * @date 2024-04-30
7 *
8 */
```

```

9
10 #include <time.h>
11 #include <stdio.h>
12 #include <ctype.h>
13 #include <unistd.h>
14 #include <stdlib.h>
15 #include <stdbool.h>
16 #include <pthread.h>
17 #include <semaphore.h>
18
19 #define BUFFER_SIZE 5
20 #define THREAD_NUM 3
21
22 /**
23 * @brief 两缓冲区
24 */
25 char buffer_1[BUFFER_SIZE];
26 char buffer_2[BUFFER_SIZE];
27
28 /**
29 * @brief empty_i 表示 buffer_i 中空闲位置信号量, full_i 表示 buffer_i 中填充数据
30 信号量
31 */
32 sem_t empty_1, full_1, empty_2, full_2;
33
34 /**
35 * @brief 生产者
36 * @param arg
37 * @return void*
38 */
39 void *producerFunc(void *arg)
40 {
41     while (true)
42     {
43         // 初始化随机数种子
44         srand((unsigned int)time(NULL));
45         // 生成随机小写字母
46         char item = 'a' + rand() % 26;
47
48         // P
49         sem_wait(&empty_1);
50         // 放入 buffer_1
51         for (int i = 0; i < BUFFER_SIZE; i++)
52         {
53             if (buffer_1[i] == '\0')
54             {
55                 buffer_1[i] = item;
56                 break;
57             }
58
59             // V
60             sem_post(&full_1);
61
62             sleep(1);
63         }
}

```

```

64         return NULL;
65     }
66
67 /**
68 * @brief 计算者
69 * @param arg
70 * @return void*
71 */
72 void *calculatorFunc(void *arg)
73 {
74     while (true)
75     {
76         // P
77         sem_wait(&full_1);
78         // 从 buffer_1 中取出字母，转换为大写字母后放入 buffer_2
79         for (int i = 0; i < BUFFER_SIZE; i++)
80         {
81             if (buffer_1[i] != '\0')
82             {
83                 buffer_2[i] = toupper(buffer_1[i]);
84                 buffer_1[i] = '\0';
85                 break;
86             }
87         }
88
89         // V
90         sem_post(&empty_1);
91         sem_post(&full_2);
92
93         sleep(1);
94     }
95
96
97     return NULL;
98 }
99
100 /**
101 * @brief 消费者
102 * @param arg
103 * @return void*
104 */
105 void *consumerFunc(void *arg)
106 {
107     while (true)
108     {
109         // P
110         sem_wait(&full_2);
111         // 从 buffer_2 中取出字符并打印到屏幕上
112         for (int i = 0; i < BUFFER_SIZE; i++)
113         {
114             if (buffer_2[i] != '\0')
115             {
116                 printf("%c\n", buffer_2[i]);
117                 buffer_2[i] = '\0';
118                 break;
119             }

```

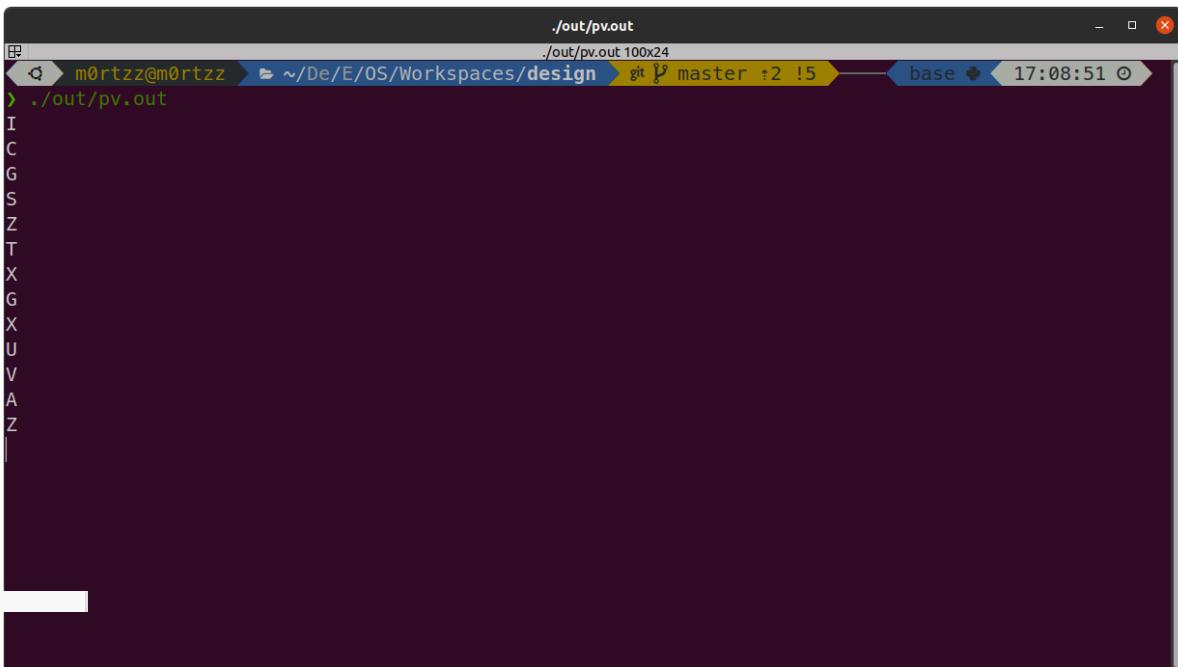
```

120     }
121
122     // v
123     sem_post(&empty_2);
124
125     sleep(1);
126 }
127
128     return NULL;
129 }
130
131 signed main()
{
132     pthread_t threads[THREAD_NUM];
133
134     // 初始化信号量
135     sem_init(&empty_1, 0, BUFFER_SIZE);
136     sem_init(&full_1, 0, 0);
137     sem_init(&empty_2, 0, BUFFER_SIZE);
138     sem_init(&full_2, 0, 0);
139
140     // 创建线程
141     pthread_create(&threads[0], NULL, producerFunc, NULL);
142     pthread_create(&threads[1], NULL, calculatorFunc, NULL);
143     pthread_create(&threads[2], NULL, consumerFunc, NULL);
144
145     // 等待线程结束
146     for (int i = 0; i < THREAD_NUM; ++i)
147         pthread_join(threads[i], NULL);
148
149     // 销毁信号量
150     sem_destroy(&empty_1);
151     sem_destroy(&full_1);
152     sem_destroy(&empty_2);
153     sem_destroy(&full_2);
154
155     return EXIT_SUCCESS;
156 }
157 }
```

④运行

```

1 | make all
2 | ./out/pv.out
```



A screenshot of a terminal window titled ".out/pv.out". The terminal shows the command `./out/pv.out` being run. The background of the terminal is dark, and the text is white. The terminal window has a header bar with the title and some status information.

第三题

①代码（构建工具同上）

```
1 /**
2 * @file socket_server.c
3 * @brief 服务器
4 * @author M0rtzz E-mail : m0rtzz@outlook.com
5 * @version 1.0
6 * @date 2024-05-01
7 *
8 */
9
10 #include <stdio.h>
11 #include <string.h>
12 #include <stdbool.h>
13 #include <stdlib.h>
14 #include <signal.h>
15 #include <unistd.h>
16 #include <pthread.h>
17 #include <semaphore.h>
18 #include <arpa/inet.h>
19 #include <sys/types.h>
20 #include <sys/socket.h>
21 #include <netinet/in.h>
22 #define MAX_LINK_NUM 5 // 最大连接数
23
24 // 用已连接客户端套接字
25 int client_sockfd[MAX_LINK_NUM];
26 // 服务器套接字
27 int server_sockfd;
28 // 当前连接数
29 int cur_link = 0;
30 // 同步服务器连接数的信号量
31 sem_t mutex;
```

```

32 // 答案
33 int secret_num = 0;
34
35 /**
36 * @brief 服务器与客户端的收发通信函数, n为连接数组序号
37 * @param n
38 */
39 void rcv_snd(int n)
40 {
41     int retval;
42     char recv_buf[1024];
43     char send_buf[1024];
44     pthread_t tid;
45     tid = pthread_self();
46
47     printf("服务器线程id=%lu使用套接字%d与客户机对话开始...\n", tid,
48            client_sockfd[n]);
49
50     // 发送随机数给客户端
51     // sprintf(send_buf, "答案是%d", secret_number);
52     // write(client_sockfd[n], send_buf, strlen(send_buf));
53
54     do
55     {
56         memset(recv_buf, 0, 1024);
57         int rcv_num = read(client_sockfd[n], recv_buf, sizeof(recv_buf));
58         if (rcv_num > 0)
59         {
60             int guess = atoi(recv_buf); // 将客户端的输入转换为整数
61
62             if (guess < secret_num)
63             {
64                 strcpy(send_buf, "小了");
65             }
66             else if (guess > secret_num)
67             {
68                 strcpy(send_buf, "大了");
69             }
70             else
71             {
72                 strcpy(send_buf, "猜对了");
73                 write(client_sockfd[n], send_buf, strlen(send_buf));
74                 break;
75             }
76             write(client_sockfd[n], send_buf, strlen(send_buf));
77         }
78     } while (true);
79
80     printf("服务器线程id=%lu与客户机对话结束\n", tid);
81     close(client_sockfd[n]);
82     // 关闭服务器监听套接字
83     close(server_sockfd);
84     client_sockfd[n] = -1;
85     cur_link--;
86     printf("当前连接数为: %d\n", cur_link);

```

```

87     sem_post(&mutex);
88     pthread_exit(&retval);
89 }
90
91 signed main()
92 {
93     // 初始化随机数生成器
94     srand(time(NULL));
95
96     // 生成1到100的随机数
97     secret_num = rand() % 100 + 1;
98
99     printf("答案是: %d\n", secret_num);
100
101    socklen_t client_len = 0;
102    // 服务器端协议地址
103    struct sockaddr_in server_addr;
104    // 客户端协议地址
105    struct sockaddr_in client_addr;
106    // 连接套接字数组循环变量
107    int i = 0;
108    server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
109    // 指定网络套接字
110    server_addr.sin_family = AF_INET;
111    // 接受所有IP地址的连接
112    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
113    // 绑定到9736端口
114    server_addr.sin_port = htons(9736);
115    bind(server_sockfd, (struct sockaddr *)&server_addr,
116          sizeof(server_addr)); // 协议套接字命名为server_sockfd
117    printf("1、服务器开始listen...\n");
118    // 创建连接数最大为MAX_LINK_NUM的套接字队列，监听命名套接字，listen不会阻塞，它向
119    // 内核报告套接字和最大连接数
120    listen(server_sockfd, MAX_LINK_NUM);
121    // 忽略子进程停止或退出信号
122    signal(SIGCHLD, SIG_IGN);
123
124    for (i = 0; i < MAX_LINK_NUM; i++)
125        client_sockfd[i] = -1; // 初始化连接队列
126
127    while (true)
128    {
129        // 搜索空闲连接
130        for (i = 0; i < MAX_LINK_NUM; i++)
131            if (client_sockfd[i] == -1)
132                break;
133
134        // 如果达到最大连接数，则客户等待
135        if (i == MAX_LINK_NUM)
136        {
137            printf("已经达到最大连接数%d，请等待其它客户释放连接...\n",
138                  MAX_LINK_NUM);
139            // 阻塞等待空闲连接
140            sem_wait(&mutex);

```

```

140         // 被唤醒后继续监测是否有空闲连接
141         continue;
142     }
143
144     client_len = sizeof(client_addr);
145     printf("2、服务器开始accept...i=%d\n", i);
146     client_sockfd[i] = accept(server_sockfd, (struct sockaddr
147 *)&client_addr, &client_len);
148     // 当前连接数增1
149     cur_link++;
150     // 可用连接数信号量mutex减1
151     sem_wait(&mutex);
152     printf("当前连接数为: %d(<=%d)\n", cur_link, MAX_LINK_NUM);
153     // 输出客户端地址信息
154     printf("连接来自:连接套接字号=%d,IP地址=%s,端口号=%d\n",
155     client_sockfd[i], inet_ntoa(client_addr.sin_addr),
156     ntohs(client_addr.sin_port));
157     pthread_create(malloc(sizeof(pthread_t)), NULL, (void *)(&recv_snd),
158     (void *)i);
159 }
160
161 return EXIT_SUCCESS;
162 }
```

```

1 /**
2 * @file socket_client.c
3 * @brief 客户机
4 * @author M0rtzz E-mail : m0rtzz@outlook.com
5 * @version 1.0
6 * @date 2024-05-01
7 *
8 */
9
10 #include <stdio.h>
11 #include <string.h>
12 #include <stdbool.h>
13 #include <stdlib.h>
14 #include <signal.h>
15 #include <unistd.h>
16 #include <pthread.h>
17 #include <semaphore.h>
18 #include <arpa/inet.h>
19 #include <sys/types.h>
20 #include <sys/socket.h>
21 #include <netinet/in.h>
22
23 signed main()
24 {
25     // 客户端套接字描述符
26     int sockfd;
27     // 地址结构体的长度
28     int len = 0;
29     // 套接字协议地址
30     struct sockaddr_in address;
31     // 发送消息缓冲区
```

```

32     char snd_buf[1024];
33     // 接收消息缓冲区
34     char rcv_buf[1024];
35     // connect函数调用的结果
36     int result;
37     // 接收消息长度
38     int rcv_num;
39     // 客户进程标识符
40     pid_t cpid;
41     sockfd = socket(AF_INET, SOCK_STREAM, 0);
42
43     if (sockfd < 0)
44     {
45         perror("客户端创建套接字失败! \n");
46         return EXIT_FAILURE;
47     }
48
49     // 使用网络套接字
50     address.sin_family = AF_INET;
51     address.sin_addr.s_addr = inet_addr("127.0.0.1"); // 服务器地址
52     // 服务器所监听的端口
53     address.sin_port = htons(9736);
54
55     if (inet_aton("127.0.0.1", &address.sin_addr) < 0)
56     {
57         printf("inet_aton error.\n");
58         return -EXIT_FAILURE;
59     }
60
61     len = sizeof(address);
62     // 获取客户进程标识符
63     cpid = getpid();
64     printf("1、客户机%d开始connect服务器... \n", cpid);
65     result = connect(sockfd, (struct sockaddr *)&address, len);
66
67     if (result == -1)
68     {
69         perror("客户机connect服务器失败! \n");
70         exit(EXIT_FAILURE);
71     }
72
73     printf("-----客户机%d与服务器线程对话开始... \n", cpid);
74
75     // 客户机与服务器循环发送接收消息
76     do
77     {
78
79         printf("2.客户机%d--->服务器:sockfd=%d,请输入客户机要发送给服务器的消息: ", cpid, sockfd);
80         // 发送缓冲区清零
81         memset(snd_buf, 0, 1024);
82         scanf("%s", snd_buf); // 键盘输入欲发送给服务器的消息字符串
83         // 将消息发送到套接字
84         write(sockfd, snd_buf, sizeof(snd_buf));
85
86         if (strcmp(snd_buf, "quit", 2) == 0)

```

```

87         break; // 若发送"quit", 则结束循环, 通信结束
88
89     // 接收缓冲区清零
90     memset(rcv_buf, 0, 1024);
91     printf("客户机%d, sockfd=%d 等待服务器回应...\n", cpid, sockfd);
92     rcv_num = read(sockfd, rcv_buf, sizeof(rcv_buf));
93     printf("客户机%d, sockfd=%d 从服务器接收的消息长度=%lu\n", cpid, sockfd,
94            strlen(rcv_buf));
95     // 输出客户机从服务器接收的消息
96     printf("3.客户机%d<---服务器:sockfd=%d,客户机从服务器接收到的消息是:
97           %s\n", cpid, sockfd, rcv_buf);
98
99     sleep(1);
100
101 } while (strncmp(rcv_buf, "猜对了", 2) != 0); // 如果收到"!q", 则结束循环,
102   通信结束
103     printf("-----客户机%d, sockfd=%d 与服务器线程对话结束-----\n",
104 cpid, sockfd);
105     // 关闭客户机套接字
106     close(sockfd);
107
108     return EXIT_SUCCESS;
109 }
```

②运行

```

1 make all
2 ./out/socket_server.out
3 ./out/socket_client.out # 不多于5个
```

The screenshot shows a terminal window with three tabs: '服务器' (Server), '客户机24268' (Client 24268), and '客户机24258' (Client 24258). The '服务器' tab contains the command `./out/socket_server.out`. The '客户机24268' tab shows the output of the client program, which includes a sequence of numbers (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25) followed by the message '答案是: 25'. The '客户机24258' tab shows the output of another client instance.

```
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design
服务器          客户机24268          客户机24258
m0rtzz@m0rtzz ~ /De/E/OS/Workspaces/design git master ?12 base 15:14:26
> ./out/socket_client.out
1. 客户机24268开始connect服务器...
-----客户机24268与服务器线程对话开始...
2. 客户机24268-->服务器:sockfd=3,请输入客户机要发送给服务器的消息: 12
客户机24268,sockfd=3 等待服务器回应...
客户机24268,sockfd=3 从服务器接收的消息长度=6
3. 客户机24268<--服务器:sockfd=3,客户机从服务器接收到的消息是: 小了
2. 客户机24268-->服务器:sockfd=3,请输入客户机要发送给服务器的消息: 25
客户机24268,sockfd=3 等待服务器回应...
客户机24268,sockfd=3 从服务器接收的消息长度=9
3. 客户机24268<--服务器:sockfd=3,客户机从服务器接收到的消息是: 猜对了
-----客户机24268,sockfd=3 与服务器线程对话结束-----
m0rtzz@m0rtzz ~ /De/E/0/W/design git master ?12 37s base 15:15:09
>
```

```
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design
服务器          客户机24268          客户机24258
m0rtzz@m0rtzz ~ /De/E/OS/Workspaces/design git master ?12 base 15:14:27
> ./out/socket_client.out
1. 客户机24258开始connect服务器...
-----客户机24258与服务器线程对话开始...
2. 客户机24258-->服务器:sockfd=3,请输入客户机要发送给服务器的消息: 55
客户机24258,sockfd=3 等待服务器回应...
客户机24258,sockfd=3 从服务器接收的消息长度=6
3. 客户机24258<--服务器:sockfd=3,客户机从服务器接收到的消息是: 大了
2. 客户机24258-->服务器:sockfd=3,请输入客户机要发送给服务器的消息: 24
客户机24258,sockfd=3 等待服务器回应...
客户机24258,sockfd=3 从服务器接收的消息长度=6
3. 客户机24258<--服务器:sockfd=3,客户机从服务器接收到的消息是: 小了
2. 客户机24258-->服务器:sockfd=3,请输入客户机要发送给服务器的消息: 25
客户机24258,sockfd=3 等待服务器回应...
客户机24258,sockfd=3 从服务器接收的消息长度=9
3. 客户机24258<--服务器:sockfd=3,客户机从服务器接收到的消息是: 猜对了
-----客户机24258,sockfd=3 与服务器线程对话结束-----
m0rtzz@m0rtzz ~ /De/E/0/W/design git master ?12 36s base 15:15:06
>
```

第四题

①源码编译安装gdb版Bochs (x86模拟器)

安装版本为Bochs-2.2.6，原因是：

https://github.com/oldlinux-web/oldlinux-files/blob/master/bochs/README_FIRST

```
7 When running the Linux kernel 0.1x in Bochs environment with version number
8 newer or equal to 2.2.6, you may encounter an problem at the bootstrap period
9 with the error message "HD controller not ready". There are two general
10 methods you can use to fix this problem.
```

因不想使用官方推荐的 `aptitude` 工具（此工具一般用于解决依赖问题，它会 `autoremove` 系统中的软件包），但此系统使用 `apt` 安装时没有遇见依赖问题，所以我还是使用了 `apt`：

```

1 # @file: linux/src/code/setup.sh
2 # @line: 80
3 sudo apt update && sudo apt install libgtk2.0-dev

```

```

79 if [ "$1" == "-d"]; then
80     # sudo apt-get install aptitude && sudo apt install libgtk2.0-dev
81     sudo apt update && sudo apt install libgtk2.0-dev
82     ./configure --enable-gdb-stub --enable-disasm

```

```

1 cd linux/src/code/
2 ./setup.sh

```

① Note

以下为最后运行Bochs时出现BUG后经过code spelunking后解决问题的过程【本仓库上传的源码鄙人已解决BUG:】：

1) [linux/src/code//bochs-2.2.6/gdbstub.cc](#):

```

1 // @brief: add
2 // @line: 492-494
3 else if (last_stop_reason == GDBSTUB_STOP_NO_REASON)
4 {
5     write_signal(&buf[1], SIGSEGV);
6 }

```

```

487 if (last_stop_reason == GDBSTUB_EXECUTION_BREAKPOINT || 
488     last_stop_reason == GDBSTUB_TRACE)
489 {
490     write_signal(&buf[1], SIGTRAP);
491 }
492 else if (last_stop_reason == GDBSTUB_STOP_NO_REASON)
493 {
494     write_signal(&buf[1], SIGSEGV);
495 }
496 else
497 {
498     write_signal(&buf[1], 0);
499 }
500 put_reply(buf);
501 break;
502 }

```

2) [linux/src/code/bochs-2.2.6/cpu/cpu.cc](#):

```

1 // @brief: comment
2 // @line: 143-147
3 // #if BX_GDBSTUB
4 //     if (bx_dbg.gdbstub_enabled) {
5 //         return;
6 //     }
7 // #endif

```

```

142 //·BUG:·fix·bug
143 //·#if·BX_GDBSTUB
144 //·····if·(bx_dbg.gdbstub_enabled)·{
145 //·····return;
146 //·····}
147 //·#endif
148 ···}

```

②改写内核源码过程 ([linux/linux-0.12/](#))

本仓库中的 [linux/linux-0.12](#) 鄙人已修改过源码，未修改过的源码tarball包为[linux/linux-0.12-pure-unmodified.tar.gz](#)。

1) [include/unistd.h](#)

```

1 // @brief: add
2 // @line: 149, 150
3 #define __NR_m0rtzz 87
4 #define __NR_ashore 88

```

```

148 #define __NR_uselib→86
149 #define __NR_m0rtzz·87
150 #define __NR_ashore·88
151
152 #define _syscall0( type, name ) \

```

2) [include/linux/sys.h](#)

```

1 // @brief: add
2 // @line: 92
3 extern int sys_m0rtzz();
4 extern int sys_ashore();
5
6 fn_ptr sys_call_table[] = { sys_setup, sys_exit, sys_fork, sys_read,
7 sys_write, sys_open, sys_close, sys_waitpid, sys_creat, sys_link,
8 sys_unlink, sys_execve, sys_chdir, sys_time, sys_mknod, sys_chmod,
9 sys_chown, sys_break, sys_stat, sys_lseek, sys_getpid, sys_mount,
10 sys_umount, sys_setuid, sys_getuid, sys_stime, sys_ptrace, sys_alarm,
11 sys_fstat, sys_pause, sys_utime, sys_stty, sys_gtty, sys_access,
12 sys_nice, sys_ftime, sys_sync, sys_kill, sys_rename, sys_mkdir,
13 sys_rmdir, sys_dup, sys_pipe, sys_times, sys_prof, sys_brk, sys_setgid,
14 sys_getgid, sys_signal, sys_geteuid, sys_getegid, sys_acct, sys_phys,
15 sys_lock, sys_ioctl, sys_fcntl, sys_mpx, sys_setpgid, sys_ulimit,
16 sys_uname, sys_umask, sys_chroot, sys_ustat, sys_dup2, sys_getppid,
17 sys_getpgrp, sys_setsid, sys_sigaction, sys_sgetmask, sys_ssetmask,
18 sys_setreuid,sys_setregid, sys_sigsuspend, sys_sigpending, sys_sethostname,
19 sys_setrlimit, sys_getrlimit, sys_getrusage, sys_gettimeofday,

```

```
20 |     sys_settimeofday, sys_getgroups, sys_setgroups, sys_select, sys_symlink,
21 |     sys_lstat, sys_readlink, sys_uselib, sys_m0rtzz, sys_ashore};
```

```
91 | extern int sys_uselib();
92 | extern int sys_m0rtzz();
93 | extern int sys_ashore();
94 |
95 | fn_ptr sys_call_table[] = { sys_setup, sys_exit, sys_fork, sys_read,
96 |     sys_write, sys_open, sys_close, sys_waitpid, sys_creat, sys_link,
97 |     sys_unlink, sys_execve, sys_chdir, sys_time, sys_mknod, sys_chmod,
98 |     sys_chown, sys_break, sys_stat, sys_lseek, sys_getpid, sys_mount,
99 |     sys_umount, sys_setuid, sys_getuid, sys_stime, sys_ptrace, sys_alarm,
100 |    sys_fstat, sys_pause, sys_utime, sys_stty, sys_gtty, sys_access,
101 |    sys_nice, sys_ftime, sys_sync, sys_kill, sys_rename, sys_mkdir,
102 |    sys_rmdir, sys_dup, sys_pipe, sys_times, sys_prof, sys_brk, sys_setgid,
103 |    sys_getgid, sys_signal, sys_geteuid, sys_getegid, sys_acct, sys_phys,
104 |    sys_lock, sys_ioctl, sys_fcntl, sys_mpx, sys_setpgid, sys_ulimit,
105 |    sys_uname, sys_umask, sys_chroot, sys_ustat, sys_dup2, sys_getppid,
106 |    sys_getpgrp, sys_setsid, sys_sigaction, sys_sgetmask, sys_ssetmask,
107 |    sys_setreuid, sys_setregid, sys_sigsuspend, sys_sigpending, sys_sethostname,
108 |    sys_setrlimit, sys_getrlimit, sys_getrusage, sys_gettimeofday,
109 |    sys_settimeofday, sys_getgroups, sys_setgroups, sys_select, sys_symlink,
110 |    sys_lstat, sys_readlink, sys_uselib, sys_m0rtzz, sys_ashore};
```

4) kernel/ststem_calls.s

```
1 | # @brief: change
2 | # @line: 63
3 | nr_system_calls = 84
```

```
62
63 | nr_system_calls = 84
64
```

5) kernel/m0rtzz.c

```
1 | /**
2 | * @file m0rtzz.c
3 | * @brief 实现自定义的系统调用函数
4 | * @author M0rtzz
5 | * @version 1.0
6 | * @date 2024-04-30
7 | */
8 |
9 | #include <errno.h>
10 | #include <string.h>
11 | #include <asm/segment.h>
12 |
13 | char msg[30]; // 全局变量，用于存储用户传递的消息
14 |
15 | /**
16 | * @brief 实现 `sys_m0rtzz` 系统调用函数，将用户提供的字符串拷贝到内核空间
17 | * @param str 用户提供的字符串指针
18 | * @return 返回拷贝的字符个数，如果超过30个字符，则返回负值错误码
19 | */
20 | int sys_m0rtzz(const char *str)
```

```

21 {
22     int i;
23     char tmp[40]; // 临时缓冲区, 用于存储从用户空间读取的字符串
24
25     // 从用户空间逐个字符读取, 直到遇到结束符或者缓冲区满为止
26     for (i = 0; i < 40; i++)
27     {
28         // 从用户空间读取一个字符并存储到临时缓冲区中
29         tmp[i] = get_fs_byte(str + i);
30
31         if (tmp[i] == '\0') // 如果读取到字符串结束符, 则退出循环
32             break;
33     }
34
35     // 统计读取的字符个数
36     i = 0;
37     while (i < 40 && tmp[i] != '\0')
38         i++;
39
40     int len = i;
41
42     // 如果读取的字符个数超过30个, 则返回错误码
43     if (len > 30)
44         return -(EINVAL);
45
46     // 将读取的字符串拷贝到全局变量msg中
47     strcpy(msg, tmp);
48
49     // 返回拷贝的字符个数
50     return i;
51 }
52
53 /**
54 * @brief 实现 `sys_ashore` 系统调用函数, 将内核空间中的消息拷贝到用户提供的缓冲区中
55 * @param str 用户提供的缓冲区指针, 用于存储消息
56 * @param size 缓冲区的大小
57 * @return 返回拷贝的字符个数, 如果缓冲区大小不足, 则返回负值错误码
58 */
59 int sys_ashore(char *str, unsigned int size)
60 {
61     int len = 0;
62
63     // 统计全局变量msg中的字符个数
64     for (; msg[len] != '\0'; len++)
65         ;
66
67     // 如果全局变量msg中的字符个数超过了缓冲区的大小, 则返回错误码
68     if (len > size)
69         return -(EINVAL);
70
71     // 将全局变量msg中的消息拷贝到用户提供的缓冲区中
72     int i;
73     for (i = 0; i < size; i++)
74     {
75         put_fs_byte(msg[i], str + i); // 将字符逐个写入用户空间
76     }

```

```

77     if (msg[i] == '\0') // 如果遇到字符串结束符，则退出循环
78         break;
79     }
80
81     // 返回拷贝的字符个数
82     return i;
83 }
```

6) kernel/Makefile

```

1 # @brief: add
2 # @line: 29
3 OBJS = sched.o sys_call.o traps.o asm.o fork.o \
4       panic.o printk.o vsprintf.o sys.o exit.o \
5       signal.o mktime.o m0rtzz.o
```

```

26
27 OBJS...= sched.o sys_call.o traps.o asm.o fork.o \
28 → panic.o printk.o vsprintf.o sys.o exit.o \
29 → signal.o mktime.o m0rtzz.o
30
```

```

1 # @brief: add
2 # @line: 48
3 m0rtzz.s m0rtzz.o: m0rtzz.c ../include/asm/segment.h ../include/string.h
  ..../include/errno.h
```

```

47 ##### Dependencies:
48 | m0rtzz.s.m0rtzz.o: m0rtzz.c ../include/linux/kernel.h ../include/unistd.h
49 |   exit.s.exit.o:: exit.c ../include/errno.h ../include/signal.h \
```

③编译linux-0.12

```

1 cd linux/oslab/
2 code run.sh # 修改脚本
```

```

1#!/bin/sh
2
3 lock_file="./hdc.img.lock"
4
5 if [ -f "${lock_file}" ]; then
6     rm "${lock_file}"
7 fi
8
9 export OSLAB_PATH=$(dirname `which $0`)
10
11 if [ ! -e "hdc.img" ]; then
12     tar -xvJf hdc.tar.xz
13 fi
14
15 if [ "$1" ] && [ "$1" = "-m" ]
16 then
```

```

17 (cd ..linux-0.12; make clean; make; cp Image ..oslab/Image)
18 elif [ "$1" ] && [ "$1" = "-g" ]
19 then
20 ${OSLAB_PATH}/bochs/bochs-gdb -q -f ${OSLAB_PATH}/bochs/bochsrc-gdb.bxrc & \
21 gdb -x ${OSLAB_PATH}/bochs/.gdbrc ..linux-0.12/tools/system
22 else
23 bochs -q -f ${OSLAB_PATH}/bochs/bochsrc.bxrc
24 fi

```

```
1 | ./run.sh -m # 编译内核源码
```

④编写代码

```

1 | cd linux/oslab/
2 | touch mount.sh umount.sh

```

```

1 |#!/usr/bin/sudo /bin/zsh
2 |
3 |# -----
4 |# @file: mount.sh
5 |# @brief: 挂载文件系统，此文件系统是linux-0.11的文件系统映像，但不影响使用，在此不过多赘述
6 |# @author: M0rtzz
7 |# @date: 2024-05-01
8 |# -----
9 |
10|mount_folder="hdc"
11|
12|if [ ! -d "${mount_folder}" ]; then
13|    mkdir "${mount_folder}"
14|fi
15|
16|export OSLAB_PATH=$(cd $(dirname "${BASH_SOURCE[0]}") >/dev/null && pwd)
17|mount -t minix -o loop,offset=1024 ${OSLAB_PATH}/hdc.img ${OSLAB_PATH}/hdc

```

```

1 |#!/usr/bin/sudo /bin/zsh
2 |
3 |# -----
4 |# @file: umount.sh
5 |# @brief: 卸载文件系统
6 |# @author: M0rtzz
7 |# @date: 2024-05-01
8 |# -----
9 |
10|umount ./hdc

```

现在可以在本地直接访问Linux文件系统而不需要在模拟器终端中访问：

```

1 | sudo chmod +x mount.sh
2 | ./mount.sh

```

```
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab 100x24
> ./mount.sh
> m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab git: master !8 ?6 base 19:29:18 ⓘ
> ls hdc/
bin dev etc image Image m0rtzz shoelace tmp user var
> m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab git: master !8 ?7 base 19:29:21 ⓘ
> m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab git: master !8 ?7 base 19:29:36 ⓘ
>
```

之后需要修改此文件系统下的 `/usr/include/unistd.h` 并在此系统编写我们的上层C语言代码来调用我们之前编写进内核的系统调用函数：

```
1 | code hdc/
```

```
1 // @file: hdc/usr/include/unistd.h
2 // @brief: add
3 // @line: 132, 133
4 #define __NR_m0rtzz 87
5 #define __NR_ashore 88
```

```
131 #define __NR_setregid 71
132 #define __NR_m0rtzz 87
133 #define __NR_ashore 88
```

```
1 /**
2 * @file hdc/usr/root/m0rtzz_1.c
3 * @brief 使用系统调用 `__NR_m0rtzz`
4 * @author M0rtzz E-mail : m0rtzz@outlook.com
5 * @version 1.0
6 * @date 2024-05-01
7 *
8 */
9
10 /**
11 * @brief 启用系统调用
12 */
13 #define __LIBRARY__
14 #include <stdio.h>
15 #include <errno.h>
16 #include <stdlib.h>
17 #include <unistd.h>
18
```

```

19 _syscall1(int, m0rtzz, const char *, str);
20
21 int main(int argc, char **argv)
22 {
23     m0rtzz(argv[1]);
24
25     return EXIT_SUCCESS;
26 }
```

```

1 /**
2 * @file hdc/usr/root/m0rtzz_2.c
3 * @brief 使用系统调用 `__NR_ashore`
4 * @author M0rtzz E-mail : m0rtzz@outlook.com
5 * @version 1.0
6 * @date 2024-05-01
7 *
8 */
9
10 /**
11 * @brief 启用系统调用
12 */
13 #define __LIBRARY__
14 #include <stdio.h>
15 #include <errno.h>
16 #include <stdlib.h>
17 #include <unistd.h>
18
19 _syscall2(int, ashore, char *, str, unsigned, size);
20
21 int main()
22 {
23     char s[30];
24     ashore(s, 30);
25     printf("The string is: %s\n", s);
26
27     return EXIT_SUCCESS;
28 }
```

```

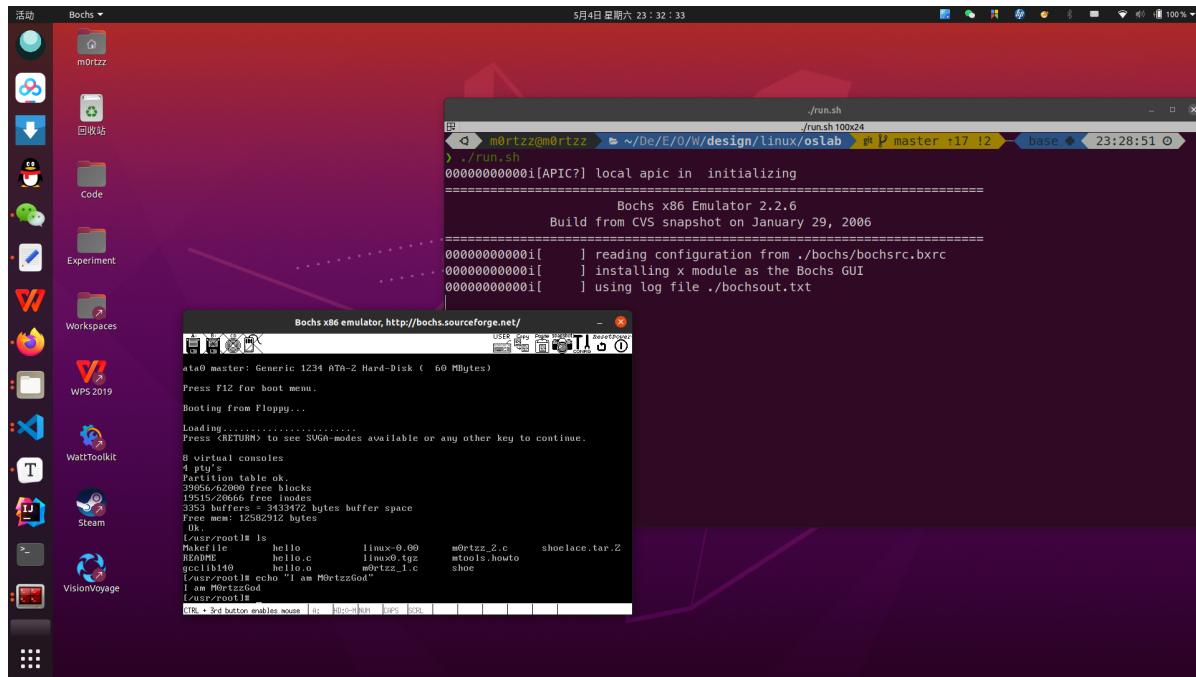
1 # @file: hdc/usr/root/Makefile
2
3 CC = gcc
4
5 all:m0rtzz_1.out m0rtzz_2.out
6
7 %.out: %.c
8     $(CC) $< -o $@
9
10 .PHONY: clean
11
12 clean:
13     rm -f *.out
```

⑤进入linux-0.12编译并运行代码

1) 以普通模式进入linux-0.12

```
1 | cd linux/oslab/
2 | ./run.sh # 以普通模式进入linux-0.12
```

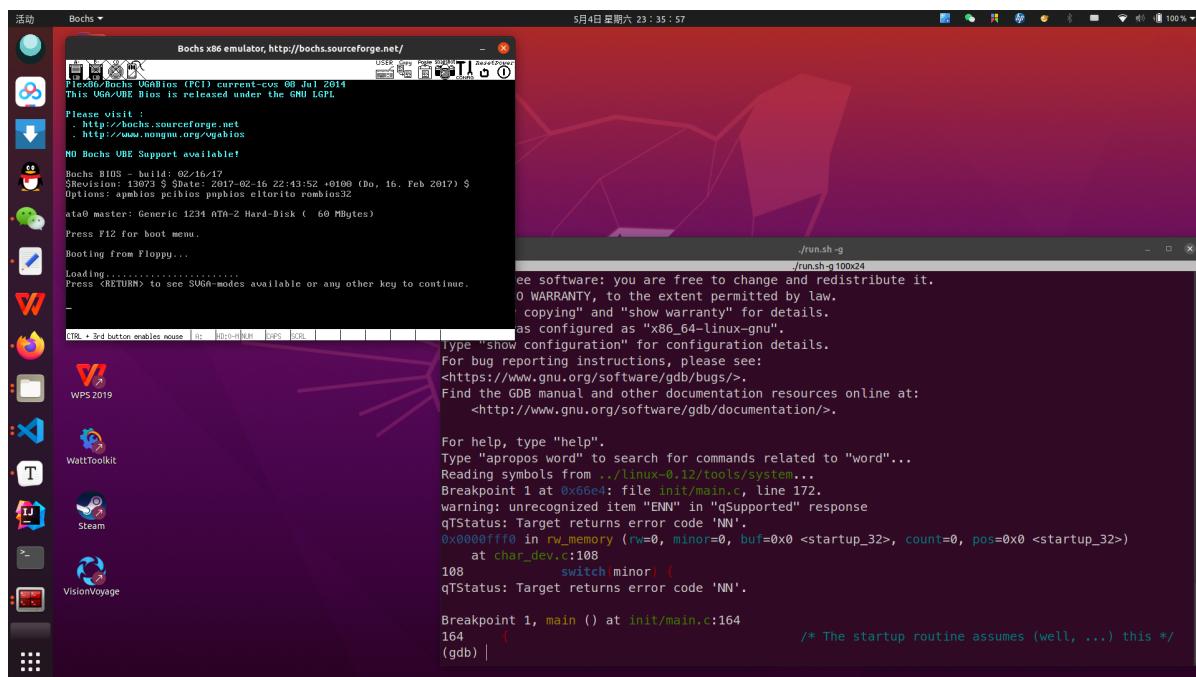
进入系统之后输入基础命令发现正常使用，无BUG：



2) 以gdb模式进入linux-0.12

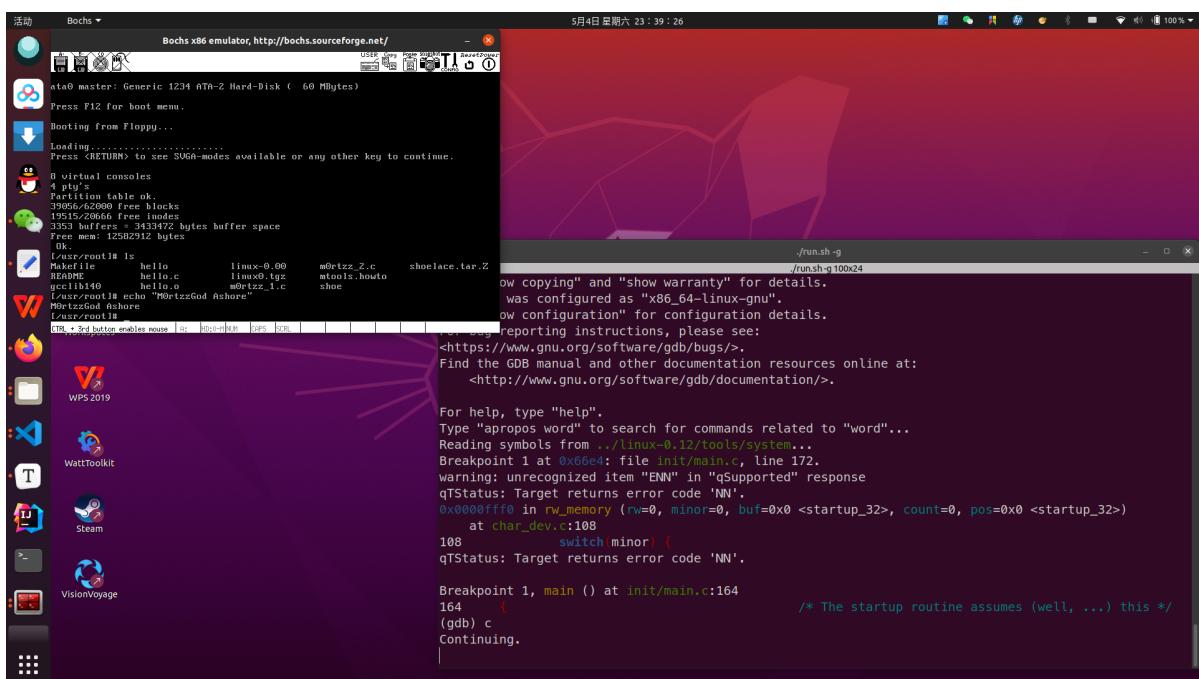
```
1 | cd linux/oslab/
2 | ./run.sh -g # 以gdb模式进入linux-0.12
```

一开始本地终端进入gdb模式但模拟器终端没有进入文件系统：

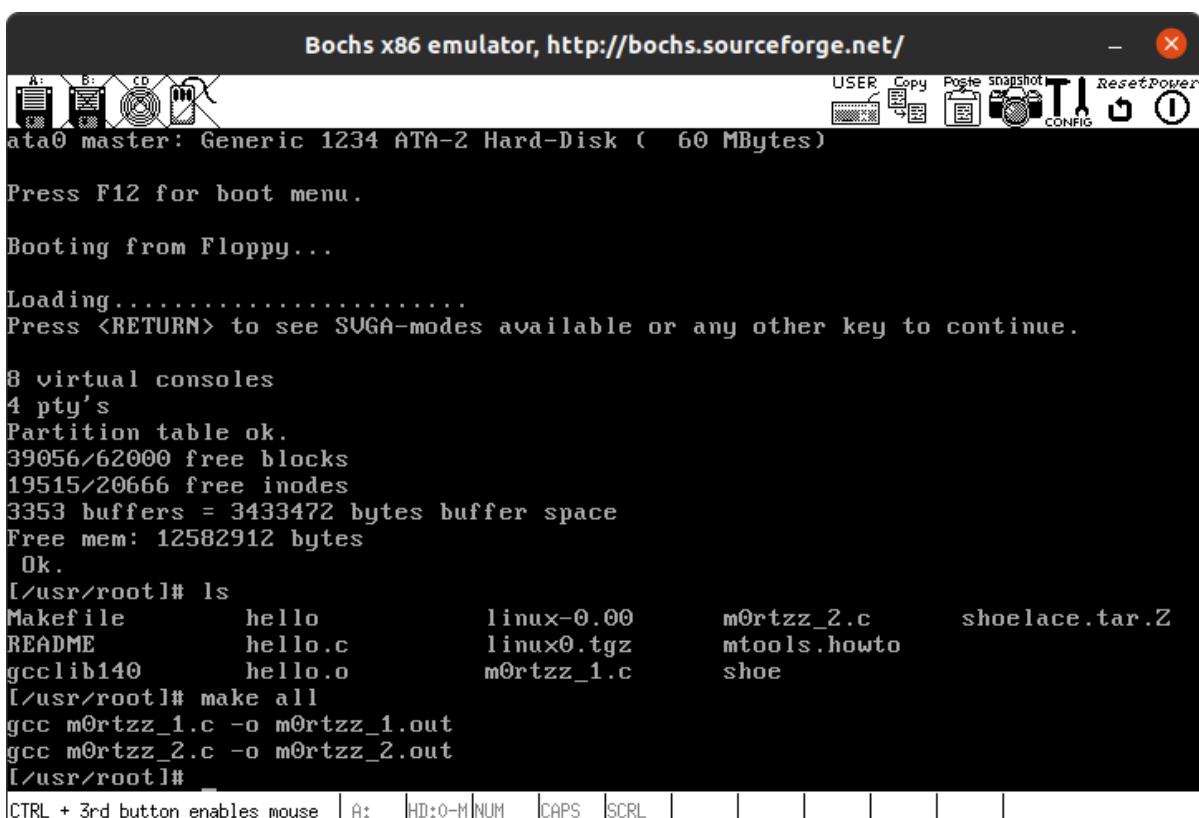


我们需要在本地终端按 **c** 键 (continue) 并回车进入文件系统后就可以正常使用命令了：

README



```
1 | ls
2 | make all
```



```
1 | ./m0rtzz_1.out str
2 | ./m0rtzz_2.out
```

```

Bochs x86 emulator, http://bochs.sourceforge.net/
[ /usr/root ]# ./mOrtzz_1.out I_am_M0rtzzGod
[ /usr/root ]# ./mOrtzz_2.out
The string is: I_am_M0rtzzGod
[ /usr/root ]# -

```

IPS: 64.868M A: NUM CAPS SCRL HD:0-M

⑥退出linux-0.12

1) 普通模式退出linux-0.12

直接点击模拟器终端右上角的 即可退出：

```

Bochs x86 emulator, http://bochs.sourceforge.net/
[ /usr/root ]# ls
Makefile      hello      linux-0.00      m0rtzz_2.c      shoelace.tar.Z
README       hello.c     linux0.tgz      mtools.howto
gcclib140    hello.o     m0rtzz_1.c      shoe
[ /usr/root ]# echo "I am M0rtzzGod"
I am M0rtzzGod
[ /usr/root ]#

```

CTRL + 3rd button enables mouse A: HD:0-M NUM CAPS SCRL

```
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab 100x24
> ./run.sh
00000000000i[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.2.6
Build from CVS snapshot on January 29, 2006
=====
00000000000i[      ] reading configuration from ./bochs/bochsrc.bxrc
00000000000i[      ] installing x module as the Bochs GUI
00000000000i[      ] using log file ./bochsout.txt
X connection to :1 broken (explicit kill or server shutdown).
> m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab 100x24
>
```

2) gdb模式退出linux-0.12

点击模拟器终端右上角的 后在本地终端输入 键并回车即可退出：

```
Bochs x86 emulator, http://bochs.sourceforge.net/
ata0 master: Generic 1234 ATA-2 Hard-Disk ( 60 MBytes)
Press F12 for boot menu.

Booting from Floppy...
Loading.....
Press <RETURN> to see SVGA-modes available or any other key to continue.

8 virtual consoles
4 pty's
Partition table ok.
39056/62000 free blocks
19515/20666 free inodes
3353 buffers = 3433472 bytes buffer space
Free mem: 12582912 bytes
Ok.
[/usr/root]# ls
Makefile      hello      linux-0.00      m0rtzz_2.c      shoelace.tar.Z
README       hello.c     linux0.tgz     mtools.howto
gcclib140    hello.o     m0rtzz_1.c      shoe
[/usr/root]# echo "M0rtzzGod Ashore"
M0rtzzGod Ashore
[/usr/root]#
CTRL + 3rd button enables mouse | A: | HD:0-M | NUM | CAPS | SCRL |
```

```
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab 100x24
<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 ..linux-0.12/tools/system...
Breakpoint 1 at 0x66e4: file init/main.c, line 172.
warning: unrecognized item "ENN" in "qSupported" response
qtStatus: Target returns error code 'NN'.
0x0000ffff0 in rw_memory (rw=0, minor=0, buf=0x0 <startup_32>, count=0, pos=0x0 <startup_32>)
  at char_dev.c:108
108          switch(minor) {
qtStatus: Target returns error code 'NN'.

Breakpoint 1, main () at init/main.c:164
164  {
(gdb) c
Continuing.
X connection to :1 broken (explicit kill or server shutdown).
Remote connection closed
(gdb) q
m0rtzz@m0rtzz ~/De/E/0/W/design/l/oslab git master +17 !2
```

⑦卸载文件系统

```
1 | sudo chmod +x umount.sh
2 | ./umount.sh
```

```
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab
m0rtzz@m0rtzz:~/Desktop/Experiment/OS/Workspaces/design/linux/oslab 100x24
[sudo] m0rtzz 的密码:
m0rtzz@m0rtzz ~/De/E/0/W/design/l/oslab git master +2 !2 ?1
ls hdc/
bin dev etc image Image init shoelace lmp usr var
m0rtzz@m0rtzz ~/De/E/0/W/design/l/oslab git master +2 !2 ?1
./umount.sh
m0rtzz@m0rtzz ~/De/E/0/W/design/l/oslab git master +2 !2
ls hdc/
m0rtzz@m0rtzz ~/De/E/0/W/design/linux/oslab git master +2 !2
```