GDB调试

分析源代码

```
#include <stdio.h>
#include <stdlib.h>
void print_array(int *arr, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int *array = (int *)malloc(5 * sizeof(int));
    if (array == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }

    for (int i = 0; i < 5; i++) {
        array[i] = i + 1;
    }

    printf("Original array: ");
    print_array(array, 5);

    int *ptr = array + 5;
    *ptr = 10;

    printf("Modified array: ");
    print_array(array, 5);

    free(array);
    return 0;
}
```

源代码定义了一个打印函数用于输出数组，主函数定义了一个数组array包含五个整数，并且定义了一个整数指针指向了array数组的第五个整数，之后将其改为10。

结果输出

```
(gdb) run
Starting program: /home/uuu/task/week3/source
Original array: 1 2 3 4 5
Modified array: 1 2 3 4 5
[Inferior 1 (process 3269) exited normally]
```

发现输出的数组第五个整数并不是10，分析应该是ptr指针并没有正确指向对应的地址，在gdb中打上断点输出ptr的值进行分析：

```
(gdb) p *ptr
$2 = 0
(gdb) p array
$3 = (int *) 0x5555555592a0
(gdb) p ptr
$4 = (int *) 0x5555555592b4
```

发现ptr指向的并不是数组中的5根据地址可以得出，应该地址是多了一个整数（一个整数是4个字节），所以将`int *ptr = array + 5;`改为`int *ptr = array + 4;`即可

```
(gdb) run
Starting program: /home/uuu/task/week3/source_modify
Original array: 1 2 3 4 5
Modified array: 1 2 3 4 10
```

QEMU环境搭建

下载Linux内核代码

在终端内输入指令

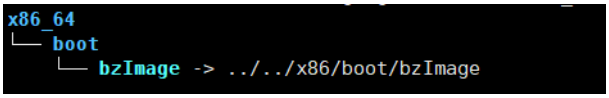
```
git clone -b staging-testing git://git.kernel.org/pub/scm/linux/kernel/git/gregkh/staging.git
```

下载linux-6.16.0内核源码，之后配置并编译内核

```
cd staging # cd linux-6.16.0
make x86_64_defconfig
make menuconfig
```

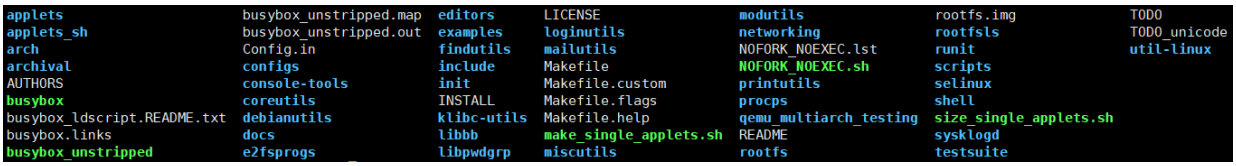
输入对应指令进入配置菜单，启用内核debug，关闭地址随机化，不然断点处无法停止。之后开始编译内核使用`make -j 12`进行编译。

编译完成后在arch/x86_64/boot/bzImage中会出现对应的内核文件bzImage如下图所示：



配置Busybox

下载Busybox并解压其文件夹内容如图所示：



将配置设置为静态编译后`make -j 12`进行编译

制作rootfs

接下来制作rootfs镜像文件，并把busybox安装到其中。 使用dd命令创建文件，并格式化为ext4文件系统。使用代码：

```
dd if=/dev/zero of=rootfs.img bs=1M count=512
mkfs.ext4 rootfs.img
```

创建用于挂载该镜像文件的目录 `rootfs`，使用 `mount` 命令将 `rootfs.img` 挂载到`rootfs`目录，编译 busybox 并写入`rootfs` 目录中。

```
mkdir rootfs
sudo mount -t ext4 -o loop rootfs.img ./rootfs
sudo make install CONFIG_PREFIX=./rootfs
```

对写入的busybox进行补充配置。

```
cd rootfs/
sudo mkdir proc dev etc home mnt
sudo cp -r ../examples/bootfloppy/etc/* etc/
```

最后，卸载rootfs.img

```
cd ..
sudo umount rootfs
```

至此，一个带有rootfs的磁盘镜像制作完成。

启动QEMU

使用如下命令启动无GUI的qemu：

```
qemu-system-arm -M vexpress-a9 -m 512M -kernel ~/linux/arch/arm/boot/zImage -dtb ~/linux/arch/arm/boot/dts/arm/vexpress-v2p-ca9.dtb -nog
```

启动完成后使用`uname -a`查看内核版本如图所示：

共享文件夹

根据上述创建的rootfs根文件目录可以直接在其中加入自己创建的程序以及驱动作为共享文件夹

首先创建临时挂载点

```
mkdir -p ./tmpfs  # 创建临时挂载目录
```

然后使用mount命令将镜像文件挂载到临时目录

```
sudo mount -o loop ./disk.img tmpfs/
```

使用cp命令将需要的程序复制到对应的文件夹中例如：

```
sudo cp ~/hello_arm ./tmpfs/home
```

最后卸载挂载镜像

```
sudo umount tmpfs
```

git上传流程

```
git add . #添加所有文件

git commit -m "提交说明（例如：首次提交）"

git push -u origin main #上传

git status #查看缓存区的内容
```