

# 浙江大学

## 本科实验报告

课程名称: 计算机体系结构

姓 名: 王俊

学 院: 海洋学院

专 业: 海洋工程与技术

学 号: 3170100186

指导教师: 翁恺

2020 年 11 月 13 日

## Labs——发现跳转延时槽

课程名称: 计算机体系结构

实验类型: 综合

实验项目名称: 发现跳转延时槽

学生姓名: 王俊 专业: 海洋工程与技术 学号: 3170100186

同组学生姓名: None

指导老师: 翁恺

实验地点: 曹光彪西-301

实验日期: 2020 年 11 月 13 日

### 一、实验目的和要求

#### Experiment Purpose:

- Install the MIPS cross compiler
- write and adjust the C language program, and find the jump delay slot in the MIPS assembly code generated by the compilation.

#### Experiment Apparatus:

- Computer (Intel Core i5 or higher, 4GB RAM or higher) system
- Ubuntu 18.04
- Buildroot

#### Experimental Materials:

No

## 二、实验内容和原理

### 2.1.mips delay slot

The main purpose of introducing a delay slot is to improve the efficiency of the pipeline, which is divided into the following two types:

#### 1. Branch delay slot:

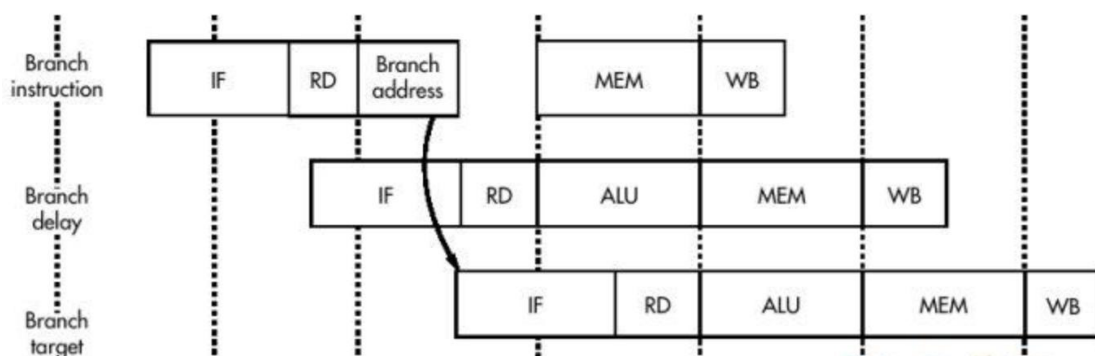
Branch delay slot (Branch delay slot) is simply an instruction located after the branch instruction, which is always executed regardless of whether the branch occurs or not, and the following code shows that the instruction located in the branch delay slot precedes the target. The instruction is submitted (commit means execution).

Jal 30 <---- 30 represents the target address of the jump jal will set the value of the ra register

nop / li a1, 4 <---- this place is a delay slot

..... <----- The value of the ra register becomes the address of this place

.....

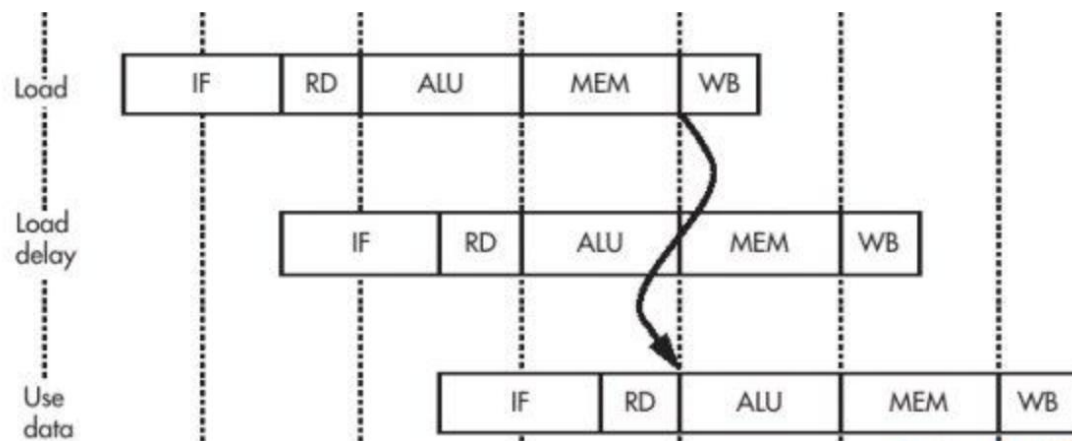


#### 2. Storage delay slot:

The data of the data load instruction can only be obtained from the cache or memory after the ALU stage of the next instruction is started, so the next instruction cannot use the data. If the next instruction compulsorily

reference the data, then the instruction will stop running (in the ALU Stage), wait for the data load instruction to complete before starting to run.

As shown below:



## 2.2.Delay slot exception return address is the address of the previous instruction:

The general CPU branch jump instruction flow is: branch jump instruction -> target jump address instruction.

But the branch jump instruction flow of MIPS is: branch jump instruction -> delay slot instruction -> instruction of the target jump address, and the delay slot instruction is inserted in the middle operation.

If the PC returns to the delay slot instruction address when the interrupt returns after the delay slot address is interrupted, the re-executed instruction flow is: delay slot instruction -> (delay slot instruction address + 4) address instruction, no jump Up!

This is not the interrupted instruction flow at all. In order to restore the

original instruction flow, the jump instruction before the delay slot needs to be reloaded into the pipeline (the instruction in the delay slot will be executed again, but it is not necessarily an exception) .

So the address returned after the delay slot interrupt is the address of the previous jump instruction.

### 三、实验过程

#### 3.1.MIPS 交叉编译环境的搭建 (Buildroot)

##### ● what is Buildroot ?

### About Buildroot

Buildroot is a tool that simplifies and automates the process of building a complete Linux system for an embedded system, using cross-compilation.

In order to achieve this, Buildroot is able to generate a cross-compilation toolchain, a root filesystem, a Linux kernel image and a bootloader for your target. Buildroot can be used for any combination of these options, independently (you can for example use an existing cross-compilation toolchain, and build only your root filesystem with Buildroot).

Buildroot is useful mainly for people working with embedded systems. Embedded systems often use processors that are not the regular x86 processors everyone is used to having in his PC. They can be PowerPC processors, MIPS processors, ARM processors, etc.

Buildroot supports numerous processors and their variants; it also comes with default configurations for several boards available off-the-shelf. Besides this, a number of third-party projects are based on, or develop their BSP<sup>1</sup> or SDK<sup>2</sup> on top of Buildroot.<sup>3</sup>

##### ● Buildroot 工程源码的下载，官方提供了两种方法。

1.通过 github 的方式获取:

The buildroot repository can be browsed online through cgit at <http://git.buildroot.net/buildroot>. To grab a copy of the repository use

```
git clone git://git.buildroot.net/buildroot
```

Or if you're behind a firewall blocking git:

```
git clone https://git.buildroot.net/buildroot
```

Please use the native git protocol if at all possible, as it's a lot more efficient than HTTP.

If you are not already familiar with using Git, we recommend you visit [the Git website](#).

Once you've checked out a copy of the source tree, you can update your source tree at any time so it is in sync with the latest and greatest by entering your buildroot directory and running the command:

```
git pull
```

## 2. 直接下载的方式

### Tarballs

You can also obtain daily snapshots of the latest Buildroot source tree if you want to follow development, but cannot or do not wish to use Git.

- You can download the [latest snapshot](#) or view recent [daily snapshots](#).
- You can also [browse the source tree online](#).

Older versions can be downloaded from the [release archive](#).

<https://blog.csdn.net/wzh0000mm>

### ● 这里我选择了第二种:

```
# 安装依赖库
$ sudo apt-get update
$ sudo apt-get install libncurses5-dev patch

# 下载源码
$ wget https://buildroot.org/downloads/snapshots/buildroot-snapshot.tar.bz2
$ tar -jxvf buildroot-snapshot.tar.bz2

# 进行编译的配置
$ cd buildroot
$ make clean
$ make menuconfig
```

安装依赖库:

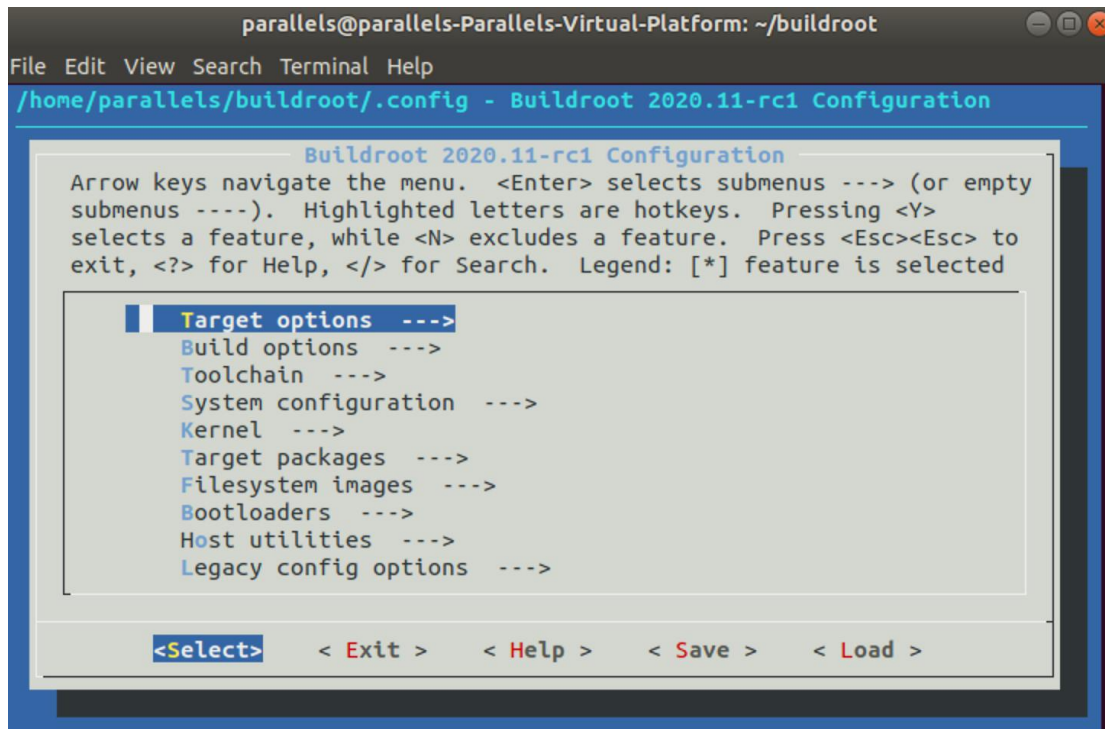
```
$ sudo apt-get update
```

```
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
```

```
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ sudo apt-get install
libncurses5-dev patch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libtinfo-dev
Suggested packages:
  ncurses-doc diffutils-doc
The following NEW packages will be installed:
  libncurses5-dev libtinfo-dev
The following packages will be upgraded:
  patch
1 upgraded, 2 newly installed, 0 to remove and 683 not upgraded.
```

然后下载解压文件，进行编译:

```
parallels@parallels-Parallels-Virtual-Platform:~$ cd buildroot
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ make clean
rm -rf /home/parallels/buildroot/output/target /home/parallels/buildroot/output/
images /home/parallels/buildroot/output/host \
/home/parallels/buildroot/output/build /home/parallels/buildroot/output/
staging \
/home/parallels/buildroot/output/legal-info /home/parallels/buildroot/ou
tput/graphs /home/parallels/buildroot/output/per-package
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ make menuconfig
```



- 将优化等级设置成 level 2

原来:

```
() directories that should be skipped when stripping  
gcc optimization level (optimize for size) --->
```

设置后:

```
() directories that should be skipped when stripping  
gcc optimization level (optimization level 2) --->
```

- 设置 mips 语言

```
Target Architecture (MIPS (little endian)) --->  
Target Binary Format (ELF) --->  
Target Architecture Variant (Generic MIPS32) --->  
[*] Use soft-float (NEW)
```

- 查看 kernel, 并设置:

```
parallels@parallels-Parallels-Virtual-Platform:~$ uname -r  
4.19.0-041900-generic
```

然后输入指令 make, 并等待:

经过一段时间的编译完成以后, 在 Buildroot 的根目录下会增加一个 output 文件, 其中包含已经编译好的文件。可以在 buildroot/output/host/usr/bin 目录下找到生成的交叉编译工具, mips 的编译器就是该目录下的 mipsel-linux-gcc。



```
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ cd output
parallels@parallels-Parallels-Virtual-Platform:~/buildroot/output$ cd host
parallels@parallels-Parallels-Virtual-Platform:~/buildroot/output/host$ cd bin
parallels@parallels-Parallels-Virtual-Platform:~/buildroot/output/host/bin$ ls
```

并找到目录下的 mipsel-linux-gcc

```
mipsel-linux-gcc
mipsel-linux-gcc-9.3.0
mipsel-linux-gcc-9.3.0.br_real
mipsel-linux-gcc-ar
mipsel-linux-gcc.br_real
mipsel-linux-gcc-nm
mipsel-linux-gcc-ranlib
mipsel-linux-gcov
mipsel-linux-gcov-dump
mipsel-linux-gcov-tool
mipsel-linux-gprof
```

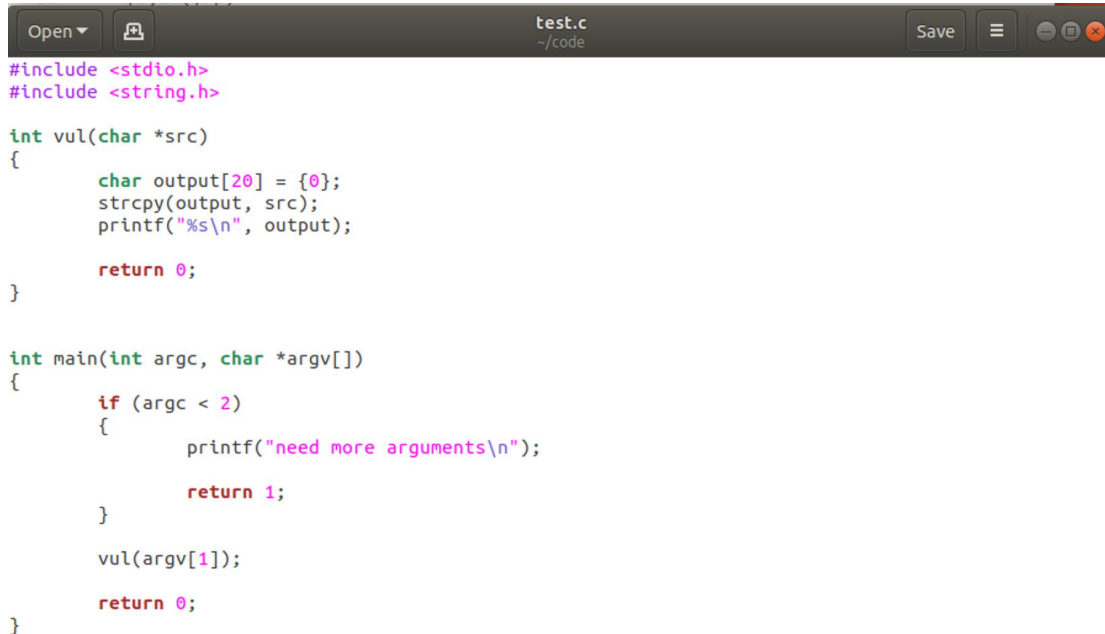
创建环境变量,然后我们就可以使用了

```
parallels@parallels-Parallels-Virtual-Platform:~$ export PATH=$PATH:~/buildroot/
output/host/usr/bin
```

## 四、实验结果分析

### ● Test.c:

```
parallels@parallels-Parallels-Virtual-Platform:~/code$ mipsel-linux-gnu-gcc test.
c -S test.s
```



```
#include <stdio.h>
#include <string.h>

int vul(char *src)
{
    char output[20] = {0};
    strcpy(output, src);
    printf("%s\n", output);

    return 0;
}

int main(int argc, char *argv[])
{
    if (argc < 2)
    {
        printf("need more arguments\n");

        return 1;
    }

    vul(argv[1]);

    return 0;
}
```

Use the command we can get:



```

test.c x test.s
move $fp,$sp
lui $28,%hi(__gnu_local_gp)
addiu $28,$28,%lo(__gnu_local_gp)
.cprestore 16
sw $4,28($fp)
lw $2,%got(__stack_chk_guard)($28)
lw $2,0($2)
sw $2,52($fp)
sw $0,32($fp)
sw $0,36($fp)
sw $0,40($fp)
sw $0,44($fp)
sw $0,48($fp)
addiu $2,$fp,32
lw $5,28($fp)
move $4,$2
lw $2,%call16(strcpy)($28)
move $25,$2
.reloc 1f,R_MIPS_JALR,strcpy
1: jalr $25
nop

lw $28,16($fp)
addiu $2,$fp,32
move $4,$2
lw $2,%call16(puts)($28)
move $25,$2
.reloc 1f,R_MIPS_JALR,puts
1: jalr $25
nop

lw $28,16($fp)
move $2,$0
lw $3,%got(__stack_chk_guard)($28)
lw $4,52($fp)

Plain Text Tab Width: 8 Ln 136, Col 21

$L3:
move $sp,$fp
lw $31,60($sp)
lw $fp,56($sp)
addiu $sp,$sp,64
jr $31
nop

```

We can see the **jalr**, **jal**, **jr** and the following is **nop**, which is the branch delay slot;

## ● Test2.c:

```

test2.c x test.s x test2.c x
#include <stdio.h>
#include <string.h>

int max(int a, int b, int c){
    int temp = 0;
    if(a >= b) temp = a;
    else temp = b;
    if(temp <= c) temp =c;
    return temp;
}

int main(int argc, char *argv[])
{
    int a,b,c,d;
    a = 1;
    b = 2;
    c = 3;

    int maxofabc = max(a,b,c);
    d = 4;
    int maxofbcd = max(b,c,d);
    return 0;
}

```

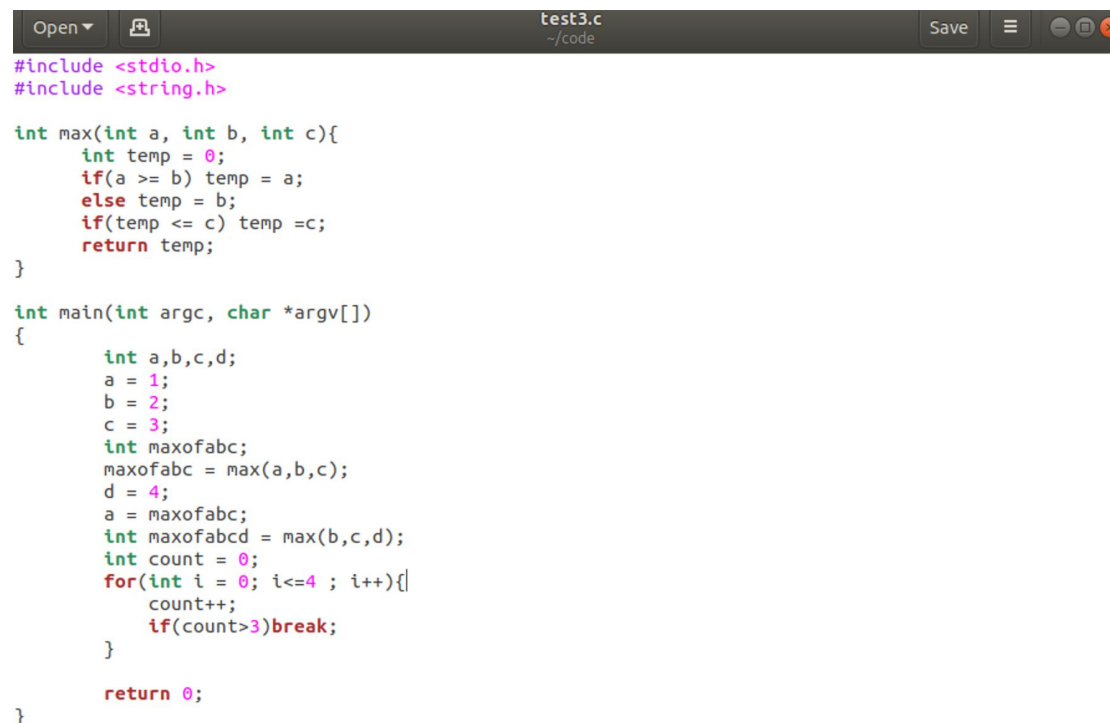
In the test2.s, we can see:

```
$L3:
    lw      $3,4($fp)
    lw      $2,24($fp)
    slt     $2,$2,$3
    bne     $2,$0,$L4
    nop

    lw      $2,24($fp)
    sw      $2,4($fp)
```

We can see the **bne** and the following is **nop**, which is the branch delay slot;

### ● Test3.c



```
test3.c
~/code
Save

#include <stdio.h>
#include <string.h>

int max(int a, int b, int c){
    int temp = 0;
    if(a >= b) temp = a;
    else temp = b;
    if(temp <= c) temp = c;
    return temp;
}

int main(int argc, char *argv[])
{
    int a,b,c,d;
    a = 1;
    b = 2;
    c = 3;
    int maxofabc;
    maxofabc = max(a,b,c);
    d = 4;
    a = maxofabc;
    int maxofabcd = max(b,c,d);
    int count = 0;
    for(int i = 0; i<=4; i++){
        count++;
        if(count>3)break;
    }

    return 0;
}
```

And in test3.s we can find:

```
    lw      $2,16($fp)
    sw      $2,4($fp)
    .option pic0
    b       $L3
    nop
```

We can see the **b** and the following is **nop**, which is the branch delay slot;

当进行了优化等级修改后

```

lw      $6,40($fp)
lw      $5,32($fp)
lw      $4,28($fp)
.option pic0
jal     max
move    $2,$0

```

the branch delay slot is filled with a move instruction;

So the statement following any branch jump statement is called a branch delay slot. In fact, when the program executes to the branch statement, when it just fills the address to jump to (in the code counter), before completing this instruction, the instruction following the branch statement is executed. This is because of the pipelining effect, several instructions are being executed at the same time, but at different stages. The branch delay slot is often used to complete some related work such as parameter initialization, rather than being wasted.

延时槽功能的关闭:

添加之前，有延迟槽

<pre> ASM temp.S 1 .org 0x0 2 #.set noreorder 3 .set noat 4 .global _start 5 _start: 6 bnel: 7   sll \$8,\$8,1 8   bne \$8,\$1,bnel 9   addi \$10,\$0,10 10  j end 11   addi \$10,\$0,10 12 end: </pre>	<pre> ASM temp.asm 1 2 temp.om:....file format elf32-tradbigmips 3 4 5 Disassembly of section .text: 6 7 00000000 &lt;_start&gt;: 8   0: 00084040 sll t0,t0,0x1 9   4: 1501fffe bne t0,at,0 &lt;_start&gt; 10  8: 00000000 nop 11  c: 08000006 j 18 &lt;end&gt; 12 10: 200a000a addi t2,zero,10 13 14: 200a000a addi t2,zero,10 14 </pre>
---	---

添加之后，延迟槽消失:

<pre> ASM temp.S 1 .org 0x0 2 .set noreorder 3 .set noat 4 .global _start 5 _start: 6 bnel: 7   sll \$8,\$8,1 8   bne \$8,\$1,bnel 9   addi \$10,\$0,10 10  j end 11   addi \$10,\$0,10 12 end: </pre>	<pre> ASM temp.asm 1 2 temp.om:....file format elf32-tradbigmips 3 4 5 Disassembly of section .text: 6 7 00000000 &lt;_start&gt;: 8   0: 00084040 sll t0,t0,0x1 9   4: 1501fffe bne t0,at,0 &lt;_start&gt; 10  8: 200a000a addi t2,zero,10 11  c: 08000005 j 14 &lt;end&gt; 12 10: 200a000a addi t2,zero,10 13 </pre>
--	---

所以，可以证明跳转指令后面的 nop 是延时槽;

## 五、实验总结与反思

### 1.在下载 libncurses5-dev patch 的时候出现了问题

```
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ sudo apt-get install libncurses5-dev patch
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
```

问题原因：主要是因为 apt 还在运行。

解决方案：

- 1: 查找所有 apt 相关的进程，并用命令杀死。

```
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ ps afx|grep apt
32752 pts/0    S+      0:00      \_ grep --color=auto apt
29942 ?        S       0:06      \_ /usr/lib/apt/methods/http
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ sudo kill -9 29942
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ sudo kill -9 32752
```

- 2: 删除锁定文件

锁定的文件会阻止 Linux 系统中某些文件或者数据的访问，这个概念也存在于 Windows 或者其他的操作系统中。

一旦你运行了 apt-get 或者 apt 命令，锁定文件将会创建于 /var/lib/apt/lists/、/var/lib/dpkg/、/var/cache/apt/archives/ 中。

这有助于运行中的 apt-get 或者 apt 进程能够避免被其它需要使用相同文件的用户或者系统进程所打断。当该进程执行完毕后，锁定文件将会删除。

所以：1: 移除对应目录下的锁文件：2: 强制重新配置软件包：3: 更新软件包源文件

```
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ sudo rm /var/lib/dpkg/lock
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ sudo dpkg --configure -a
parallels@parallels-Parallels-Virtual-Platform:~/buildroot$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
Building dependency tree
```

### 2.在编译的时候出现了问题

```
parallels@parallels-Parallels-Virtual-Platform:~$ mipsel-linux-gcc -o ./code/test ./code/test.c -static
./code/test.c:1:10: fatal error: stdio.h: No such file or directory
1 | #include <stdio.h>
  | ~~~~~
```

找不到头文件，于是直接使用第三种方法进行下载：

其实 MIPS 程序的交叉编译工具 gcc 不必非得自己编译得到，网站 <https://uclibc.org/>上已

经有编译后的 mips 程序的交叉编译工具提供给我们下载，在 ubuntu 系统上，可以使用脚本 config.sh 进行下载，脚本文件的内容如下：

### Config.sh:

```
1  #!/bin/bash
2  #this file aim to download xcompiler and install it
3  download_url="https://www.uclibc.org/downloads/binaries/0.9.30.1/cross-compiler-"
4
5
6  function download_bins {
7      echo "ready to download: $1.tar.bz2"
8      wget "${download_url}$1.tar.bz2" "--no-check-certificate"
9      echo "ready to tar: $1.tar.bz2"
10     tar -jxf $install_path/cross-compiler-$1.tar.bz2
11     sed -i 's\export PATH=$PATH:$install_path/cross-compiler-$1/bin\ /etc/profile
12 }
13
14 if [ $# == 0 ]
15 then
16     echo "Usage: $0 intall_path"
17     exit -1
18 elif [ $# == 1 ]
19 then
20     install_path=$1
21 fi
22
23 cd $install_path
24 download_bins mips
25 download_bins mipsel
26 download_bins armv4l
27 download_bins i686
28 download_bins x86_64
29
30 source /etc/profile
31
32 echo "Done"
```

```
parallels@parallels-Parallels-Virtual-Platform:~/bash$ vim config.sh
parallels@parallels-Parallels-Virtual-Platform:~/bash$ chmod +x config.sh
parallels@parallels-Parallels-Virtual-Platform:~/bash$ sudo ./config.sh
Usage: ./config.sh intall_path
parallels@parallels-Parallels-Virtual-Platform:~/bash$ sudo ./config.sh ./ready
to download: mips.tar.bz2
ready to download: mips.tar.bz2
--2020-11-15 13:04:18-- https://www.uclibc.org/downloads/binaries/0.9.30.1/cros
s-compiler-mips.tar.bz2
Resolving www.uclibc.org (www.uclibc.org)... 140.211.167.122
Connecting to www.uclibc.org (www.uclibc.org)|140.211.167.122|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20904891 (20M) [application/x-bzip2]
Saving to: 'cross-compiler-mips.tar.bz2'
```