

添加内核补丁

补丁 (patch) 是什么？

补丁 (patch) 是什么？

- 版本管理
- 代码之间的不同之处
 - 目录的区别
 - 文件的区别
 - 文件内的区别: "+"与"-"
- 作用:
 - New features
 - Bug fix
- 来源:
 - 自己编写代码
 - 别人编写代码

实验课打补丁流程：

- 1、选取补丁
- 2、制作补丁
- 3、打补丁
- 4、安装并验证补丁

1、选取补丁

1、选取补丁——补丁出处：

理论上讲，各大开源网站(如github，google的android源码目录)的每次[commit](#)记录都可以作为补丁添加到我们的内核中，但是由于开源网站上分支众多、每个分支的代码更改次数也相当多，工作量巨大，所以我们需要针对性的寻找我们感兴趣并且比较重要的补丁。

网站<http://kernelnewbies.org/LinuxVersions>上针对每个linux内核版本的重大新特性做了总结，并且大部分都提供了新特性的[详细介绍](#)和[验证方法](#)，从这里选取补丁可以大大减少我们的工作量。除此之外，git上的一些分支的代码也可以merge到已发布的内核中，需要我们筛选找到我们感兴趣的代码。



Kernel Hacking

- [Frontpage](#)
- [Kernel Hacking](#)
- [Kernel Documentation](#)
- [Kernel Glossary](#)
- [FAQ](#)
- [Found a bug?](#)
- [Kernel Changelog](#)
- [Upstream Merge Guide](#)

Projects

- [KernelJanitors](#)
- [KernelMentors](#)
- [KernelProjects](#)

Community

- [Why a community?](#)
- [Regional Kernelnewbies](#)
- [Personal Pages](#)
- [Upcoming Events](#)

References

- [Mailing Lists](#)

KernelNewbies : LinuxVersions

Last updated at 2020-01-27 18:13:24

This is a list of links to every changelog.

5.x

- [Linux_5.5](#) Released Sun, 26 Jan 2020 (63 days)
- [Linux_5.4](#) Released Sun, 24 Nov 2019 (70 days)
- [Linux_5.3](#) Released Sun, 15 September 2019 (70 days)
- [Linux_5.2](#) Released Sun, 7 July 2019 (63 days)
- [Linux_5.1](#) Released Sun, 5 May 2019 (63 days)
- [Linux_5.0](#) Released Sun, 3 March 2019 (70 days)

4.x

- [Linux_4.20](#) Released 23 December, 2018 (62 days)
- [Linux_4.19](#) Released 22 October, 2018 (71 days)
- [Linux_4.18](#) Released 12 August, 2018 (70 days)
- [Linux_4.17](#) Released 3 Jun, 2018 (63 days)
- [Linux_4.16](#) Released 1 Apr, 2018 (73 days)
- [Linux_4.15](#) Released 18 January, 2018 (77 days)
- [Linux_4.14](#) Released 12 November, 2017 (70 days)
- [Linux_4.13](#) Released 3 September, 2017 (63 days)
- [Linux_4.12](#) Released 2 July, 2017 (63 days)
- [Linux_4.11](#) Released 30 April, 2017 (70 days)
- [Linux_4.10](#) Released 19 February, 2017 (70 days)



Kernel Hacking

[Frontpage](#)

[Kernel Hacking](#)

[Kernel Documentation](#)

[Kernel Glossary](#)

[FAQ](#)

[Found a bug?](#)

[Kernel Changelog](#)

[Upstream Merge Guide](#)

Projects

[KernelJanitors](#)

[KernelMentors](#)

[KernelProjects](#)

Community

[Why a community?](#)

[Regional Kernelnewbies](#)

[Personal Pages](#)

[Upcoming Events](#)

References

[Mailing Lists](#)

KernelNewbies : Linux_5.5

Last updated at 2020-01-30 11:11:40

Linux 5.5 [was released](#) on 26 Jan 2020.

This release includes support in Btrfs for RAID1 with 3 and 4 copies and new checksum types; KUnit, a kernel unit testing framework; many improvements to io_uring(2) largely focused around networked I/O; Airtime Queue Limits for fighting bufferbloat on Wi-Fi and provide a better connection quality; support for mounting a CIFS network share as root filesystem and for using the same session across multiple network interfaces; support adding alternative names for network devices; many BPF improvements that allow to generate faster BPF code; and support for NFS cross device offloaded copy. As always, there are many other new drivers and improvements.

目录

1. [Coolest features](#)
 1. [Btrfs RAID1 with 3 and 4 copies and more checksum alternatives](#)
 2. [Many improvements to io_uring\(2\)](#)
 3. [KUnit, an unit testing framework for the kernel](#)
 4. [Airtime Queue Limits for a better Wi-Fi connection quality](#)
 5. [CIFS as root file system and multichannel support](#)
 6. [Support alternative names for network interfaces](#)
 7. [BPF improvements: type checking and BPF Trampoline](#)
 8. [NFS cross-device offloaded copy](#)
2. [Core \(various\)](#)
3. [File systems](#)
4. [Memory management](#)
5. [Block layer](#)
6. [Tracing, perf and BPF](#)
7. [Virtualization](#)
8. [Cryptography](#)
9. [Security](#)
10. [Networking](#)
11. [Architectures](#)
 1. [ARM](#)
 2. [x86](#)
 3. [PowerPC](#)

搜索



登录 ➞

5. Block layer



index : kernel/git/torvalds/linux.git

Linux kernel source tree

master

switch

Linus Torvalds

[about](#) [summary](#) [refs](#) [log](#) [tree](#) **commit** [diff](#) [stats](#)

log msg

search

author Eric Dumazet <edumazet@google.com> 2019-10-30 09:36:20 -0700
committer David S. Miller <davem@davemloft.net> 2019-10-31 14:01:40 -0700
commit [19f92a030ca6d772ab44b22ee6a01378a8cb32d4](#) (patch)
tree [29ef4d4bfa78c7b0e21b4adeb0ffc27691021e75](#)
parent [6d6f0383b697f004c65823c2b64240912f18515d](#) (diff)
download [linux-19f92a030ca6d772ab44b22ee6a01378a8cb32d4.tar.gz](#)

diff options

context:

space:

mode:

net: increase SOMAXCONN to 4096

SOMAXCONN is `/proc/sys/net/core/somaxconn` default value.

It has been defined as 128 more than 20 years ago.

Since it caps the `listen()` backlog values, the very small value has caused numerous problems over the years, and many people had to raise it on their hosts after being hit by problems.

Google has been using 1024 for at least 15 years, and we increased this to 4096 after TCP listener rework has been completed, more than 4 years ago. We got no complain of this change breaking any legacy application.

Many applications indeed setup a TCP listener with `listen(fd, -1)`; meaning they let the system select the backlog.

Raising SOMAXCONN lowers chance of the port being unavailable under even small SYNFLOOD attack, and reduces possibilities of side channel vulnerabilities.

Signed-off-by: Eric Dumazet <edumazet@google.com>

Cc: Willy Tarreau <wt@wt.eu>

Cc: Yue Cao <vcao009@ucr.edu>

Signed-off-by: Eric Dumazet <edumazet@google.com>
Cc: Willy Tarreau <wt@wt.eu>
Cc: Yue Cao <ycao009@ucr.edu>
Signed-off-by: David S. Miller <davem@davemloft.net>

Diffstat

```
-rw-r--r-- Documentation/networking/ip-sysctl.txt 4  
-rw-r--r-- include/linux/socket.h 2
```

2 files changed, 3 insertions, 3 deletions

```
diff --git a/Documentation/networking/ip-sysctl.txt b/Documentation/networking/ip-sysctl.txt  
index 49e95f4..0e66534 100644
```

```
--- a/Documentation/networking/ip-sysctl.txt  
+++ b/Documentation/networking/ip-sysctl.txt  
@@ -207,8 +207,8 @@ TCP variables:
```

```
    somaxconn - INTEGER  
                Limit of socket listen() backlog, known in userspace as SOMAXCONN.  
-           Defaults to 128. See also tcp_max_syn_backlog for additional tuning  
-           for TCP sockets.  
+           Defaults to 4096. (Was 128 before linux-5.4)  
+           See also tcp_max_syn_backlog for additional tuning for TCP sockets.
```

```
    tcp_abort_on_overflow - BOOLEAN  
                If listening service is too slow to accept new connections,
```

```
diff --git a/include/linux/socket.h b/include/linux/socket.h  
index fc0bed5..4049d97 100644
```

```
--- a/include/linux/socket.h  
+++ b/include/linux/socket.h  
@@ -263,7 +263,7 @@ struct ucred {  
    #define PF_MAX          AF_MAX
```

```
    /* Maximum queue length specifiable by listen. */  
-#define SOMAXCONN          128  
+#define SOMAXCONN          4096
```

```
    /* Flags we can use with send/ and recv.  
       Added those for 1003.1g not all are supported yet
```

1、选取补丁——补丁提示：

一般来说，打补丁的困难程度取决于以下因素：

- 补丁提交时间与我们的内核版本相近程度
- 改动的文件数量
- 改动的文件是否跨越了较多的模块
- 涉及的模块改动是否频繁
- 代码更改行数

1、选取补丁——示例介绍：

以Linux**5.4** 版本内核的网络模块中对参数SOMAXCONN的patch为例，简单介绍打补丁的一般流程

补丁名称： **net: increase SOMAXCONN to 4096**

补丁链接：

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=19f92a030ca6d772ab44b22ee6a01378a8cb32d4>

Diffstat

```
-rw-r--r-- Documentation/networking/ip-sysctl.txt 4   
-rw-r--r-- include/linux/socket.h 2 
```

2 files changed, 3 insertions, 3 deletions

```
diff --git a/Documentation/networking/ip-sysctl.txt b/Documentation/networking/ip-sysctl.txt  
index 49e95f4..0e66534 100644
```

```
--- a/Documentation/networking/ip-sysctl.txt
```

```
+++ b/Documentation/networking/ip-sysctl.txt
```

```
@@ -207,8 +207,8 @@ TCP variables:
```

```
    somaxconn - INTEGER
```

```
        Limit of socket listen() backlog, known in userspace as SOMAXCONN.
```

```
-        Defaults to 128.  See also tcp_max_syn_backlog for additional tuning  
-        for TCP sockets.
```

```
+        Defaults to 4096. (Was 128 before linux-5.4)
```

```
+        See also tcp_max_syn_backlog for additional tuning for TCP sockets.
```

```
    tcp_abort_on_overflow - BOOLEAN
```

```
        If listening service is too slow to accept new connections,
```

```
diff --git a/include/linux/socket.h b/include/linux/socket.h
```

```
index fc0bed5..4049d97 100644
```

```
--- a/include/linux/socket.h
```

```
+++ b/include/linux/socket.h
```

```
@@ -263,7 +263,7 @@ struct ucred {
```

```
    #define PF_MAX          AF_MAX
```

```
    /* Maximum queue length specifiable by listen.  */
```

```
-#define SOMAXCONN          128
```

```
+#define SOMAXCONN          4096
```

```
    /* Flags we can use with send/ and recv.
```

```
        Added those for 1003.1g not all are supported yet
```

1、选取补丁——示例介绍：

- 这个参数主要限制了接收新 TCP 连接侦听队列的大小，默认 128
- 对于一个经常处理新连接的高负载 web服务环境来说，128太小了。大多数环境这个值建议增加到 1024 或者更多，谷歌使用1024已经有15年了。
- 有实验显示将这个数字改为4096后，超过4年时间没有出现一次由于这个改变导致传统应用崩溃的申诉。大的侦听队列对防止拒绝服务 DoS 攻击也会有所帮助。

官网的补丁描述：

SOMAXCONN is /proc/sys/net/core/somaxconn default value.

It has been defined as 128 more than 20 years ago.

Since it caps the listen() backlog values, the very small value has caused numerous problems over the years, and many people had to raise it on their hosts after being hit by problems.

Google has been using 1024 for at least 15 years, and we increased this to 4096 after TCP listener rework has been completed, more than 4 years ago. We got no complain of this change breaking any legacy application.

Many applications indeed setup a TCP listener with `listen(fd, -1);` meaning they let the system select the backlog.

Raising SOMAXCONN lowers chance of the port being unavailable under even small SYNFLOOD attack, and reduces possibilities of side channel vulnerabilities.

1、选取补丁——准备（查看现状）

首先，我们在旧版本内核下查看默认的SOMAXCONN参数值（打印为128）

```
>> cat /proc/sys/net/core/somaxconn
```

我们尝试把补丁打到我们目前5.3的内核中。

2、制作补丁

2、制作补丁

找不同——

补丁添加**版本**的**相关代码**与现有版本相关代码的**不同**

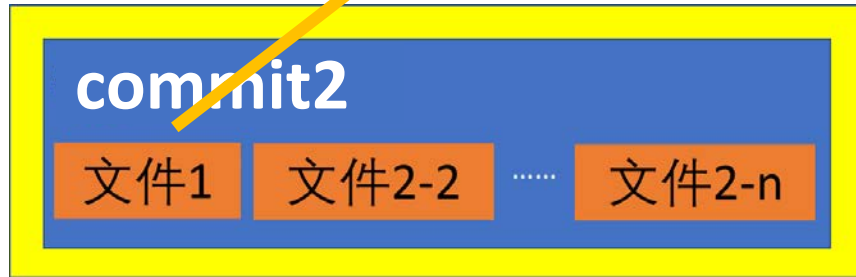
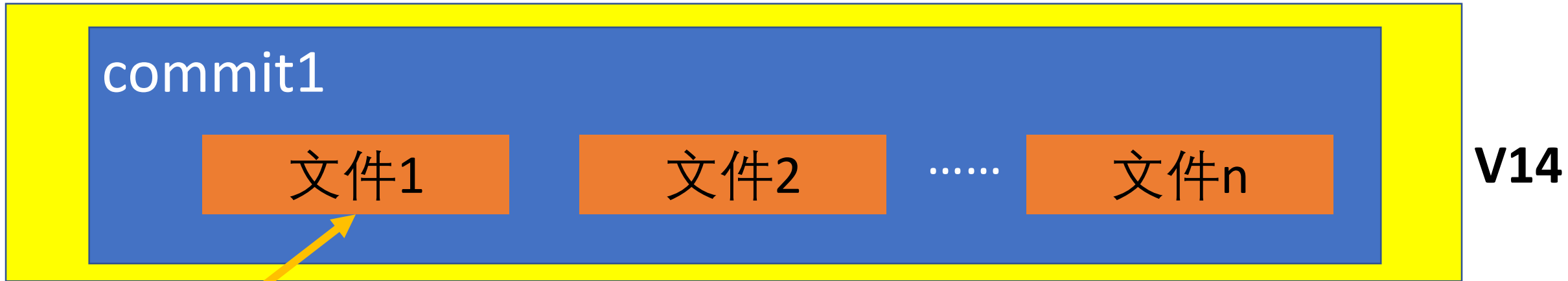
2、制作补丁

对补丁的代码进行分析，并找出与之相关的代码

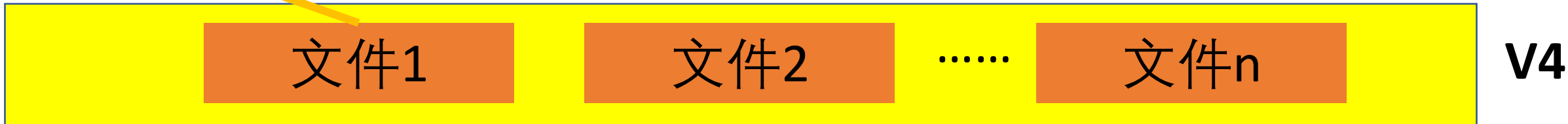
代码相关的复杂性：

- 跨越多个版本
- 涉及到多个commit
- 涉及到多个模块
- 涉及到多个文件

在选取补丁时最好选取自己熟悉的模块。



⋮



跨越多版本示例：

实验目的：

在kernel`3.4`上打上Autocorking补丁（kernel`3.14`），并验证这个补丁在大量发送小包的情况下的cpu性能改善情况

问题及解决1：

- Patch要求在tcp.h的第282行加入如下代码
`extern int sysctl_tcp_autocorking`

实际上应该在tcp.h的第254行增加此代码

- Patch要求在proc.c的第279行加入如下代码
`SNMP_MIB_ITEM("TCPAutoCorking",LINUX_MIB_TCPAUTOCORKING),`
实际上应该在文件的261行加入上述代码

问题及解决2

需要修改部分代码

```
net/ipv4/tcp.c:607:31: error: 'TSQ_THROTTLED' undeclared (first use in this function)
net/ipv4/tcp.c:607:31: note: each undeclared identifier is reported only once for each function it appears in
net/ipv4/tcp.c:607:49: error: 'struct tcp_sock' has no member named 'tsq_flags'
```

查看源码发现，这个判断里面会判断**tp**的某个标志位，并调用**NET_INC_STATS**对**autocorking**的调用次数进行了统计，而在其后验证过程中，并不需要用到这个统计值，也没有相关子系统的使用。所以将这一段代码删掉，即可成功编译。

2、制作补丁

挑选出相关文件之后

- 手动：筛选出相关代码并将其复制到源码相关处
 - 因为版本和分支不同的原因，多数情况下文件的行数是无法对应的，所以需要手工查找对应代码。
- 自动：用diff制作补丁——标识不同

2、制作补丁

通过diff指令制作补丁

1.对于单个文件

找出from-file与to-file差别，生成能够从 from-file到to-file升级的补丁

```
>>diff -uN from-file to-file > x.patch
```

2.对于文件夹中多个文件

找出from-directory与to-directory差别，生成能够从 from-directory到to-directory升级的补丁

```
>>diff -uNr from-directory to-directory > x.patch
```

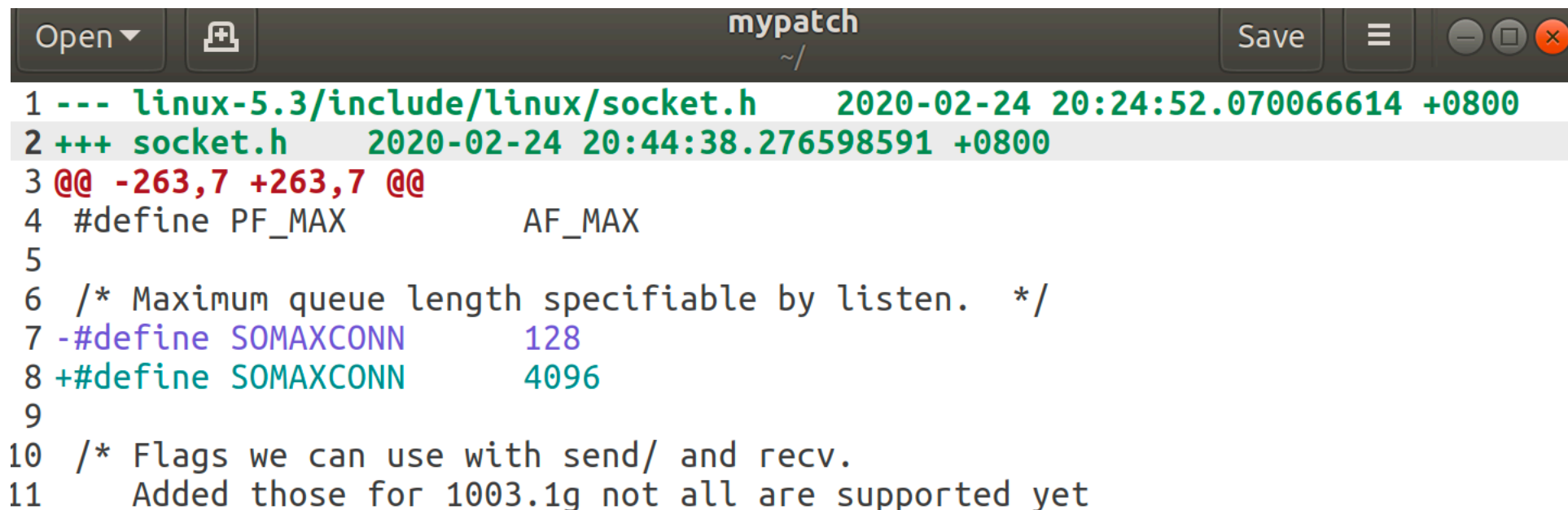
-r: --recursive(比较子目录); **-u: --unified**(将两个文件不同的地方合并显示);
-N: --new-file(显示仅出现在一个文件夹的文件); **-a: --text**(仅逐行比较文本)

2、制作补丁

利用diff命令生成补丁

```
>> diff -au linux-5.3/include/linux/socket.h socket.h > mypatch
```

查看patch内容



```
Open ▾  mypatch  Save  ≡  -  □  ×  
~/  
1 --- linux-5.3/include/linux/socket.h    2020-02-24 20:24:52.070066614 +0800  
2 +++ socket.h        2020-02-24 20:44:38.276598591 +0800  
3 @@ -263,7 +263,7 @@  
4  #define PF_MAX          AF_MAX  
5  
6  /* Maximum queue length specifiable by listen.  */  
7 -#define SOMAXCONN       128  
8 +#define SOMAXCONN       4096  
9  
10 /* Flags we can use with send/ and recv.  
11     Added those for 1003.1g not all are supported yet
```

```

diff -auNr -x '\.*' linux-4.19.23/net/ipv4/tcp.c linux-patch/net/ipv4/tcp.c
--- linux-4.19.23/net/ipv4/tcp.c      2019-02-15 16:09:54.000000000 +0800
+++ linux-patch/net/ipv4/tcp.c      2019-03-15 23:01:00.599811757 +0800
@@ -3889,8 +3889,8 @@
        init_net.ipv4.sysctl_tcp_wmem[2] = max(64*1024, max_wshare);

        init_net.ipv4.sysctl_tcp_rmem[0] = SK_MEM_QUANTUM;
-       init_net.ipv4.sysctl_tcp_rmem[1] = 87380;
-       init_net.ipv4.sysctl_tcp_rmem[2] = max(87380, max_rshare);
+       init_net.ipv4.sysctl_tcp_rmem[1] = 131072;
+       init_net.ipv4.sysctl_tcp_rmem[2] = max(131072, max_rshare);

        pr_info("Hash tables configured (established %u bind %u)\n",
                tcp_hashinfo.ehash_mask + 1, tcp_hashinfo.bhash_size);
diff -auNr -x '\.*' linux-4.19.23/net/ipv4/tcp_input.c linux-patch/net/ipv4/tcp_
input.c
--- linux-4.19.23/net/ipv4/tcp_input.c  2019-02-15 16:09:54.000000000 +0800
+++ linux-patch/net/ipv4/tcp_input.c      2019-03-15 23:04:28.709821680 +0800
@@ -426,26 +426,7 @@
        }

}

-/* 3. Tuning rcvbuf, when connection enters established state. */

```

补丁说明

---: 源文件

+++ : 目标文件

@@ -x,y +m,n @@:

- 源文件修改范围从第x行开始，共y行
- 修改之后对应的目标文件从m行开始，共n行

缩进: 表示该部分进行修改

+ : 增加一行

- : 减少一行

无符号: 表示引用这一行，不进行增加或减少

3、打补丁

3、打补丁 (patch)

通过 **patch** 指令打补丁

1.对于单个文件

将补丁x应用到源文件src中，生成dst文件

```
>>patch src < x.patch
```

2.对于文件夹中多个文件

将补丁x应用到源目录src中，src目录中多个文件被修改

```
>>patch -p0 < x.patch
```

pnum指定对x.patch文件里指定的文件路径的处理方式

如，p0：当前文件夹；pn：去掉前边n个"/"之后的路径

pnum说明

-pnum or **--strip=num**

Strip the smallest prefix containing num leading slashes from each file name found in the patch file. A sequence of one or more adjacent slashes is counted as a single slash. This controls how file names found in the patch file are treated, in case you keep your files in a different directory than the person who sent out the patch. For example, supposing the file name in the patch file was

/u/howard/src/blurfl/blurfl.c

setting **-p0** gives the entire file name unmodified, **-p1** gives

u/howard/src/blurfl/blurfl.c

without the leading slash, **-p4** gives

blurfl/blurfl.c

and not specifying **-p** at all just gives you **blurfl.c**. Whatever you end up with is looked for either in the current directory, or the directory specified by the **-d** option.

pnum说明

```
Open ▾ mypatch Save ≡
1 diff -auNr -x '\.*' linux-5.3/include/linux/0.c mylinux-patch/include/linux/0.c
2 --- linux-5.3/include/linux/0.c 1970-01-01 07:00:00.000000000 +0700
3 +++ mylinux-patch/include/linux/0.c 2020-02-24 20:30:41.871136304 +0800
4 @@ -0,0 +1,3 @@
5 +printf(1)
6 +printf(2)
7 +printf(3)
8 diff -auNr -x '\.*' linux-5.3/include/linux/socket.h mylinux-patch/include/linux/socket.h
9 --- linux-5.3/include/linux/socket.h 2020-02-24 20:24:52.070066614 +0800
10 +++ mylinux-patch/include/linux/socket.h 2020-02-24 18:45:25.348616069 +0800
11 @@ -263,7 +263,7 @@
12 #define PF_MAX AF_MAX
13
14 /* Maximum queue length specifiable by listen. */
15 -#define SOMAXCONN 128
16 +#define SOMAXCONN 4096
17
18 /* Flags we can use with send/ and recv.
19 Added those for 1003.1q not all are supported yet
```

对文件夹diff才会有这部分

patch命令中
pnum参数的作用路径


```
Open ▾ mypatch Save ≡ - □ ×
1 --- linux-5.3/include/linux/socket.h    2020-02-24 20:24:52.070066614 +0800
2 +++ socket.h    2020-02-24 20:44:38.276598591 +0800
3 @@ -263,7 +263,7 @@
4  #define PF_MAX          AF_MAX
5
6  /* Maximum queue length specifiable by listen.  */
7  -#define SOMAXCONN      128
8  +#define SOMAXCONN      4096
9
10 /* Flags we can use with send/ and recv.
11     Added those for 1003.1g not all are supported yet
```

>>cd linux-5.3/

>>patch -p1 <../mypatch

3、打补丁

>> patch linux-5.3/include/linux/socket.h < mypatch

```
amos@ubuntu:~$ patch linux-5.3/include/linux/socket.h < mypatch  
patching file linux-5.3/include/linux/socket.h
```

4、安装并验证补丁

4、安装并验证补丁

- 打完补丁后即生成了新的内核源码（自己查看一下源代码）
- 按照第一节课的内容对新的内核源码进行编译、安装

4、安装并验证补丁

确定更改后的代码是否生效：

- 使用内核提供的**printk**打印日志的方式验证更改的代码确实已经执行。由于内核进行大量的打印日志可能会造成日志存储区溢出，如果要对打印速度进行限制可以选择**printk_ratelimit**函数。
- 通过**proc**虚拟文件系统查看新内核的**somaxconn**参数值来验证，

>> cat /proc/sys/net/core/somaxconn

如果默认值已经变化为4096，则实验成功。

```
amos@ubuntu:~$ cat /proc/sys/net/core/somaxconn
4096
```

