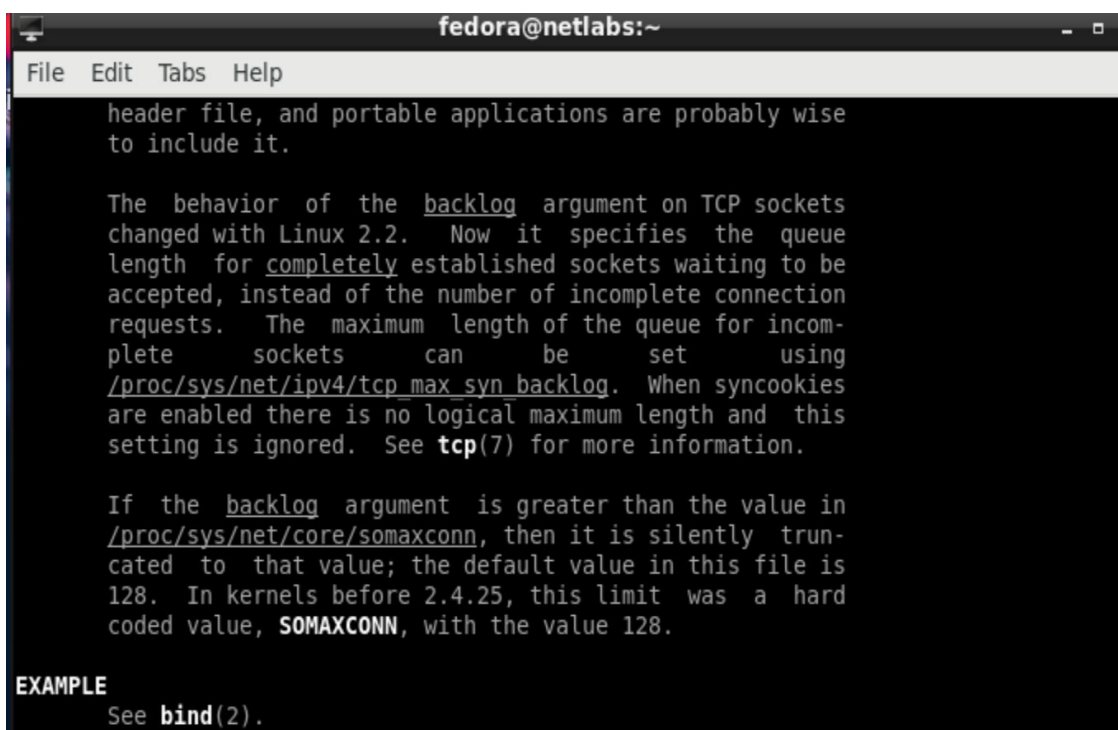


Atividade 3: Backlog e Processos Zumbis

Luana Felipe de Barros

RA: 201705

1. O valor backlog representa a soma do tamanho máximo de duas filas: fila de conexões já estabelecidas corretamente e fila de conexões pendentes/incompletas. Este valor é mantido pelo socket listening e o valor numérico do backlog pode representar outro número dependendo do sistema operacional. Sempre que um pedido de conexão é feito por um cliente, este vai para a lista de conexões pendentes. Após a finalização do 3-way handshake que esta conexão passa para a lista de conexões estabelecidas. Então, o backlog está relacionado ao número de conexões totais (estabelecidas e pendentes) que o socket listening pode suportar.
2. Utilizando o comando `man listen` obtive as informações mostradas na figura abaixo. É possível notar que na versão 2.2 no kernel Linux, o valor de backlog representa apenas o tamanho da fila de conexões corretamente estabelecidas esperando a ser aceitas. Além disso, na versão atual do kernel da minha máquina o valor padrão é 128.



```
fedora@netlabs:~
File Edit Tabs Help

header file, and portable applications are probably wise
to include it.

The behavior of the backlog argument on TCP sockets
changed with Linux 2.2. Now it specifies the queue
length for completely established sockets waiting to be
accepted, instead of the number of incomplete connection
requests. The maximum length of the queue for incom-
plete sockets can be set using
/proc/sys/net/ipv4/tcp_max_syn_backlog. When syncookies
are enabled there is no logical maximum length and this
setting is ignored. See tcp(7) for more information.

If the backlog argument is greater than the value in
/proc/sys/net/core/somaxconn, then it is silently trun-
cated to that value; the default value in this file is
128. In kernels before 2.4.25, this limit was a hard
coded value, SOMAXCONN, with the value 128.

EXAMPLE
See bind(2).
```

3. Para realizar estes testes, variei o backlog de 0 a 10 e utilizei um shell script para conectar 10 clientes da forma mais rápida possível. Através do comando netstat, consegui observar quantos clientes conseguiam se comunicar simultaneamente. Em meus testes, apareceu somente o status ESTABLISHED. Precisei colocar um sleep(3) entre as chamadas listen e accept no servidor.c, para que retardasse aceitar as conexões e conseguíssemos observar a influência do valor do backlog entre os 10 clientes. Sem este atraso, só consegui observar diferença a partir de 100 clientes. Também utilizei outro sleep(3) dentro do processo filho no servidor e no cliente, para segurar a conexão e conseguir visualizar os status com o netstat.

Backlog	Nº de Conexões
0	2
1	3
2	4
3	5
4	5
5	7
6	7
7	8
8	10
9	10
10	10

4. Atualmente, o código gera processos zumbi. Após criar um handler para tratar o sinal SIGCHLD, vemos que nenhum processo filho está atrelado ao pai, após a finalização do programa.

Podemos ver a árvore de processos, descobrindo inicialmente o pid do processo pai. (7448)

```
(base) luanabarrosguitarra:~$ ps -aux | grep './servidor 1024'
luanaba+ 7448  0.0  0.0  4520  808 pts/0    S+   13:22   0:00 ./servidor 1024 0
luanaba+ 7507  0.0  0.0  13144 1016 pts/7    S+   13:23   0:00 grep --color=auto ./servidor 1024
```

Agora, podemos ver que durante a execução do programa, vários pids de processos filhos aparecerem e sumiram da árvore, terminando todos.

```

(base) luanabarrosguitarra:~$ pstree -p 7448
servidor(7448)─servidor(7526)
               └─servidor(7527)
(base) luanabarrosguitarra:~$ pstree -p 7448
servidor(7448)─servidor(7529)
               └─servidor(7530)
(base) luanabarrosguitarra:~$ pstree -p 7448
servidor(7448)─servidor(7532)
(base) luanabarrosguitarra:~$ pstree -p 7448
servidor(7448)─servidor(7532)
(base) luanabarrosguitarra:~$ pstree -p 7448
servidor(7448)─servidor(7532)
(base) luanabarrosguitarra:~$ pstree -p 7448
servidor(7448)

```

E no código do servidor, vemos os processos filhos sendo terminados.

```

(base) luanabarrosguitarra:~/network-programming$ ./servidor 1024 0
IP para conectar: 0.0.0.0
Numero de porta para voce se conectar: 1024
Handling: 127.0.0.1:57464
Handling: 127.0.0.1:57466
child 7516 terminated
Handling: 127.0.0.1:57478
child 7521 terminated
Handling: 127.0.0.1:57474
child 7522 terminated
Handling: 127.0.0.1:57480
child 7523 terminated
Handling: 127.0.0.1:57470
child 7524 terminated
Handling: 127.0.0.1:57482
child 7526 terminated
Handling: 127.0.0.1:57476
child 7527 terminated
Handling: 127.0.0.1:57468
child 7529 terminated
Handling: 127.0.0.1:57472
child 7530 terminated
child 7532 terminated

```

5. Como executar os programas

- a. Compilar o servidor: `gcc servidor.c -o servidor -Wall`
- b. Compilar o cliente: `gcc cliente.c -o cliente -Wall`
- c. Rodar o servidor: `./servidor porta backlog`
- d. Monitorar a porta: `netstat -tpanc | grep porta`
- e. Executar o script que roda 10 clientes: `bash run_clientes.sh porta`