网络编程第二次作业

吴桐
57119108

2022 年 9 月 20 日

1 TCP 并发服务器程序开发

TCP 并发服务器的核心思想，是在与客户端建立连接后，将相应的事务指派给子进程处理，而父进程继续监听端口等待新的连接请求。我们简单修改服务器的代码：在子进程中停止端口监听，完成服务器系统日期和时间的获取，将信息发送给客户端后，就直接终止子进程；在父进程中关闭建立的连接（该连接已经转交给子进程处理），继续监听相应的端口。客户端程序不变。

我们在子进程内使用 sleep 指令，暂缓子进程的退出，便于我们在终端查看到各个进程的状态。

```
for( ; ; )
{
    connfd = accept( listenfd , (struct sockaddr *)NULL , NULL );
    if ( ( pid = fork() ) == 0)
    {  // child
        close( listenfd );  // chlid stop listen
        time( &ticks );
        struct tm *p;
        p = gmtime( &ticks );

        snprintf( buff , sizeof( buff ) , "%.24s\r\n" , ctime( &ticks ) );
        write( connfd , buff , strlen( buff ) );
        close( connfd );   // close connection

        sleep(10);  // help us to observe the processes
        exit(0);
```

```
    }
    // father
    printf("create child process %d\n", pid);
    close( connfd );
}
```

我们在虚拟机上编译服务器和客户端的程序后，首先启动服务器程序。从图 1 中可以看到，此时服务器进程已经开始正常监听。

```
[09/20/22]seed@VM:~/.../chapter 2$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1784    1782  2  80   0 - 145417 ep_pol tty2     00:00:20 Xorg
0 S  1000    1841    1782  0  80   0 - 47714 poll_s tty2      00:00:00 gnome-session-b
0 S  1000    3078    2629  0  80   0 -   589 inet_c pts/0    00:00:00 server
0 R  1000    3079    2753  0  80   0 -  2853 -      pts/2    00:00:00 ps
```

图 1: 服务器进程开始正常监听

之后我们多次启动客户端程序，从图 2 中可以看到，此时服务器进程产生了多个子进程处理与客户端之间的通信。

```
[09/20/22]seed@VM:~/.../chapter 2$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1784    1782  2  80   0 - 145417 ep_pol tty2     00:00:20 Xorg
0 S  1000    1841    1782  0  80   0 - 47714 poll_s tty2      00:00:00 gnome-session-b
0 S  1000    3078    2629  0  80   0 -   589 inet_c pts/0    00:00:00 server
1 S  1000    3084    3078  0  80   0 -   622 hrtime pts/0    00:00:00 server
1 S  1000    3087    3078  0  80   0 -   622 hrtime pts/0    00:00:00 server
1 S  1000    3090    3078  0  80   0 -   622 hrtime pts/0    00:00:00 server
1 S  1000    3093    3078  0  80   0 -   622 hrtime pts/0    00:00:00 server
0 R  1000    3101    2753  0  80   0 -  2853 -      pts/2    00:00:00 ps
```

图 2: 服务器进程产生多个子进程

如图 3 所示，我们在一段时间后，重新查看当前活动的进程，发现子进程成为了僵尸进程（defunct）。

僵尸进程是一个早已死亡的进程，但在进程表中仍占了一个位置。在 Linux 系统中，一个进程结束了，但是其父进程没有等待（调用 wait/waitpid）它，那么它将变成一个僵尸进程。当用 ps 命令观察进程的执行状态时，看到这些进程的状态栏为 defunct。

但是如果该进程的父进程结束，那么该进程就不会变成僵尸进程。因为在每个进程结束的时候，系统都会扫描当前系统中所运行的所有进程，看看有没有哪个进程是刚刚结束的这个进程的子进程，如果是的话，就由 Init 进程来接管他，成为他的父进程，从而保证每个进程都会有一个父进程。而 Init 进程会自动 wait 其子进程，因此被 Init 接管的所有进程都不会变成僵尸进程。

为了解决僵尸进程问题，我们在服务器代码中添加以下内容。在这段代码中，我们使用信号量捕捉子进程的终止，之后调用 wait 回收子进程的资源。之所以不直接使用 wait 等待子进程，是因为 wait 是阻塞的，会影响父进程监听其他客户端的连接请求，所以我们这里使用信号量去触发 wait。

```
[09/20/22]seed@VM:~/.../chapter 2$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 R  1000    1784    1782  2  80   0 - 145417 -     tty2     00:00:20 Xorg
0 S  1000    1841    1782  0  80   0 - 47714 poll_s tty2     00:00:00 gnome-session-b
0 S  1000    3078    2629  0  80   0 -   589 inet_c pts/0    00:00:00 server
1 Z  1000    3084    3078  0  80   0 -     0 -      pts/0    00:00:00 server <defunct>
1 Z  1000    3087    3078  0  80   0 -     0 -      pts/0    00:00:00 server <defunct>
1 Z  1000    3090    3078  0  80   0 -     0 -      pts/0    00:00:00 server <defunct>
1 Z  1000    3093    3078  0  80   0 -     0 -      pts/0    00:00:00 server <defunct>
0 R  1000    3109    2753  0  80   0 -  2853 -      pts/2    00:00:00 ps
```

图 3: 僵尸进程

```
void sig_handler()
{
    printf("child process stop\n");
    wait(0);
}

if( signal(SIGCHLD, sig_handler) == SIG_ERR )
{
    printf("sigchld error\n");
    exit( 1 );
}
```

我们在虚拟机上重新编译代码后，启动服务器程序。之后，我们多次启动客户端程序，如图 4 所示。

```
[09/20/22]seed@VM:~/.../chapter 2$ client 0.0.0.0 4433
Tue Sep 20 02:50:30 2022
[09/20/22]seed@VM:~/.../chapter 2$ client 0.0.0.0 4433
Tue Sep 20 02:50:31 2022
[09/20/22]seed@VM:~/.../chapter 2$ client 0.0.0.0 4433
Tue Sep 20 02:50:32 2022
```

图 4: 多次启动客户端程序

如图 5 所示，我们持续监控当前活动的进程，发现子进程被依次正常回收。

如图 6 所示，服务器端也输出了相应的提示信息。

```
[09/20/22]seed@VM:~/.../chapter 2$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1784    1782  1  80   0 - 146817 ep_pol tty2     00:00:36 Xorg
0 S  1000    1841    1782  0  80   0 - 47714 poll_s tty2      00:00:00 gnome-session-b
0 S  1000    3724    2629  0  80   0 -   622 inet_c pts/0     00:00:00 server
1 S  1000    3726    3724  0  80   0 -   622 hrtime pts/0     00:00:00 server
1 S  1000    3729    3724  0  80   0 -   622 hrtime pts/0     00:00:00 server
1 S  1000    3732    3724  0  80   0 -   622 hrtime pts/0     00:00:00 server
0 R  1000    3761    2753  0  80   0 -  2853 -      pts/2     00:00:00 ps
[09/20/22]seed@VM:~/.../chapter 2$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1784    1782  1  80   0 - 146817 ep_pol tty2     00:00:36 Xorg
0 S  1000    1841    1782  0  80   0 - 47714 poll_s tty2      00:00:00 gnome-session-b
0 S  1000    3724    2629  0  80   0 -   622 inet_c pts/0     00:00:00 server
1 S  1000    3729    3724  0  80   0 -   622 hrtime pts/0     00:00:00 server
1 S  1000    3732    3724  0  80   0 -   622 hrtime pts/0     00:00:00 server
0 R  1000    3763    2753  0  80   0 -  2853 -      pts/2     00:00:00 ps
[09/20/22]seed@VM:~/.../chapter 2$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1784    1782  1  80   0 - 146817 ep_pol tty2     00:00:36 Xorg
0 S  1000    1841    1782  0  80   0 - 47714 poll_s tty2      00:00:00 gnome-session-b
0 S  1000    3724    2629  0  80   0 -   622 inet_c pts/0     00:00:00 server
1 S  1000    3732    3724  0  80   0 -   622 hrtime pts/0     00:00:00 server
0 R  1000    3765    2753  0  80   0 -  2853 -      pts/2     00:00:00 ps
[09/20/22]seed@VM:~/.../chapter 2$ ps -la
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1784    1782  1  80   0 - 146817 ep_pol tty2     00:00:36 Xorg
0 S  1000    1841    1782  0  80   0 - 47714 poll_s tty2      00:00:00 gnome-session-b
0 S  1000    3724    2629  0  80   0 -   622 inet_c pts/0     00:00:00 server
0 R  1000    3767    2753  0  80   0 -  2853 -      pts/2     00:00:00 ps
```

图 5: 子进程被正常回收

```
[09/20/22]seed@VM:~/.../chapter 2$ server 4433
create child process 3889
create child process 3892
create child process 3895
child process stop
child process stop
child process stop
```

图 6: 服务器端输出相应提示信息