

بسمه تعالی



گزارش تمرین عملی چهارم - سوال دوم - درس سیستم‌های عامل دکتر اسدی - نیمسال اول ۱۴۰۵-۱۴۰۴

نویسنده‌گان:

۱. سیداحمد موسوی‌اول - ۱۴۰۲۱۰۶۶۴۸

۲. عرفان تیموری - ۱۴۰۲۱۰۵۸۱۳

در این تمرین که ادامه تمرین قبلی است، باید تغییراتی برای دستور ps که در کانتینر اجرا می‌شود داشته باشیم تا درست باشد و دیگر ارتباطی با سیستم اصلی نداشته باشد.

ابتدا سوال‌ها را پاسخ می‌دهیم:

در مورد سوال اول: چرا دستور ps همچنان خوب کار نمی‌کند و تمامی فرایندها را نشان می‌دهد؟

دستور ps برای مشاهده لیست فرآیندها، اطلاعات خود را از سیستم فایل /proc می‌خواند. در مرحله اولیه، فضای نام mount جدیدی ایجاد نشده است. بنابراین، کانتینر همچنان از /proc مربوط به سیستم اصلی (host) استفاده می‌کند.

حال به سراغ بخش ج رفتیم و مورد گفته شده را با انجام مواردی که در لینک اشاره شده بود و با استفاده از mount انجام دادیم که نتایج در زیر مشهودند.

```
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ make clean
rm -f zocker
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ ls
main.c Makefile
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ nano main.c
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ ls
main.c Makefile
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ gcc -Wall -Wextra -std=c99 -o zocker main.c
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ sudo setcap cap_sys_admin+ep zocker
[sudo] password for z@ahmad-mousavi-awal:
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ ls
main.c Makefile zocker
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ ./zocker run --name test-container 'sh'
Running child with pid: 1 (in container namespace)
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$ ps
  PID TTY          TIME CMD
  1 pts/0    00:00:00 sh
  2 pts/0    00:00:00 sh
  3 pts/0    00:00:00 ps
z@ahmad-mousavi-melij8-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Faixxx:~/Desktop/zocker$
```

ولی به مشکلی که گفته شده خوردم و حال به سوال گفته شده پاسخ می‌دهیم:

دلیل این مشکل این است که به صورت پیش‌فرض در بسیاری از سیستم‌های لینوکس، نقاط اتصال (Mount Points) در حالت MS_SHARED قرار دارند. این بدان معناست که هرگونه عملیات Mount یا Unmount در داخل فضای نام جدید، به Host نیز منتشر می‌شود و می‌تواند سیستم Host را دچار مشکل کند. که برای این مشکل از MS_PRIVATE (Propagate)

در نتیجه باید از فلگ MS_PRIVATE استفاده می‌کردیم که در قسمت ه از آن استفاده کرده و نتایج به شرح زیر مشهودند:

```
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~$ cd Desktop/zockerTest2/zocker
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zockerTest2/zocker$ make clean
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zockerTest2/zocker$ nano main.c
gcc -Wall -Wextra -std=c99 -o zocker main.c
sudo setcap cap_sys_admin+ep zocker
[sudo] password for s-ahmed-mousavi-mal:
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zockerTest2/zocker$ ./zocker run --name test-container 'sh'
Running child with pid: 1 (in container namespace)
sh: 0: can't access tty; job control turned off
$ ps
 PID TTY      TIME CMD
 1 ?    00:00:00 sh
 2 ?    00:00:00 sh
 3 ?    00:00:00 ps
$ df
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/nvme0n1p5 75192448 53205888 18134524 75% /
tmpfs 7999944 95944 7904000 2% /dev/shm
tmpfs 1599992 2348 1597644 1% /run
tmpfs 5120 12 5100 1% /run/lock
tmpfs 7999944 8 7999944 0% /run/quiet
tmpfs 1599988 148 1599848 1% /run/user/1000
efivars 256 119 113 48% /sys/firmware/efi/efivars
/dev/nvme0n1p1 97318 81750 15500 85% /boot/efi
/dev/nvme0n1p4 94986680 50506448 44479568 54% /mnt/windows
tmpfs 1599988 148 1599848 1% /run/user/1000
$ exit
[Parent] Stopping...
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zockerTest2/zocker$ df
Filesystem 1K-blocks Used Available Use% Mounted on
tmpfs 1599992 2348 1597644 1% /run
/dev/nvme0n1p5 75192448 53205888 18134524 75% /
tmpfs 7999944 95944 7904000 2% /dev/shm
tmpfs 5120 12 5100 1% /run/lock
efivars 256 119 113 48% /sys/firmware/efi/efivars
tmpfs 7999944 8 7999944 0% /run/quiet
/dev/nvme0n1p1 97318 81750 15500 85% /boot/efi
/dev/nvme0n1p4 94986680 50506448 44479568 54% /mnt/windows
tmpfs 1599988 148 1599848 1% /run/user/1000
```

که دستور گفته شده درست کار می کند.

در ادامه برای بخش و باید در قسمت Unshare این را هم می زدیم: CLONE_NEWUTS که این را هم اضافه کرده و بنابراین برای هر کانتینر یک فضای نام به صورت جداگانه اختصاص پیدا می کند که این در عکس زیر مشهود است:

```
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~$ cd Desktop/
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop$ cd zockerTest2/
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zockerTest2$ ls
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zockerTest2$ cd zocker/
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zocker$ ls
main.c Makefile zocker
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zocker$ ./zocker run --name test-container2 'sh'
[ERROR] Failed to open /proc/self/timers_offsets: Permission denied
Running child with pid: 1 (in container namespace)
sh: 0: can't access tty; job control turned off
$ ps
 PID TTY      TIME CMD
 1 ?    00:00:00 sh
 2 ?    00:00:00 sh
 3 ?    00:00:00 ps
$ [ ]
```

که در آن فضای نام ها جدا اند.

در ادامه برای بخش ز نیاز بود از دستور setHostname گفته شده استفاده کنیم که با استفاده از آن توانستیم اسم را به کانتینر اختصاص دهیم مانند زیر:

```
s-ahmed-mousavi-malys-ahmed-mousavi-mal-Victus-by-HP-Gaming-Laptop-15-Faxxxi:~/Desktop/zockerTest2$ sudo ./zocker run --name isolated-box 'sh'
mounting child with pid: 1 (in container namespace)
sh: 0: can't access tty; job control turned off
# hostname
# hostname
isolated-box
#
```

همانطور که مشهود است دستور فوق در کانتینر و شل با یکدیگر فرق دارد.

برای بخش ح هم یک تابع جدید اضافه کردیم تا زمان را درست کند و عکس آن در زیر مشهود است که زمان از ۰ شروع شده که مشخصاً زمان

دستگاه آن نیست:

```
a-ahmed-mousavi-aval@a-ahmed-mousavi-aval-Victus-by-MS-Gaming-Laptop-15-Falxxx:/desktop/zocker/test$ sudo ./zocker run --name time-test 'cat /proc/uptime'
Running child with pid: 1 (in container namespace)
0.00 170950.98
[Parent] Stopping...
a-ahmed-mousavi-aval@a-ahmed-mousavi-aval-Victus-by-MS-Gaming-Laptop-15-Falxxx:/desktop/zocker/test$
```

کدهای زده شده به شرح زیر است که فایل main.c هم در کنار همین فایل قرار دارد و می‌توانید آن را ببینید

```
1. void setup_time_offsets() {
2.     struct timespec now;
3.
4.     if (clock_gettime(CLOCK_MONOTONIC, &now) != 0) {
5.         perror("clock_gettime failed");
6.         return;
7.     }
8.
9.     long sec = -(now.tv_sec + 1);
10.    long nsec = 1000000000L - now.tv_nsec;
11.
12.    FILE *f = fopen("/proc/self/timens_offsets", "w");
13.    if (f == NULL) {
14.        perror("[ERROR] Failed to open /proc/self/timens_offsets");
15.        return;
16.    }
17.
18.    fprintf(f, "monotonic %ld %ld\n", sec, nsec);
19.    fprintf(f, "boottime %ld %ld\n", sec, nsec);
20.
21.    fflush(f);
22.    fclose(f);
23. }
24.
25. int run_container(struct config cfg) {
26.     pid_t pid;
27.
28.     if (unshare(CLONE_NEWPID | CLONE_NEWNS | CLONE_NEWUTS |
29.                 CLONE_NEWTIME) != 0) {
30.         fprintf(stderr, "[ERR] Failed to unshare(2).");
31.         return 1;
32.     }
33.     setup_time_offsets();
34.
35.     pid = fork();
36.     if (pid < 0) {
37.         return 1;
38.     }
39.     if (sethostname(cfg.name, strlen(cfg.name)) != 0) {
40.         perror("Failed to set hostname");
41.         exit(1);
42.     }
```

```
43.
44.         if (mount(NULL, "/", NULL, MS_REC | MS_PRIVATE, NULL) != 0) {
45.             perror("Failed to make root mount private");
46.             exit(1);
47.         }
48.
49.         if (mount("proc", "/proc", "proc", 0, NULL) != 0) {
50.             perror("Failed to mount /proc");
51.             exit(1);
52.         }
53.
54.         setsid();
55.
56.         printf("Running child with pid: %d (in container namespace)\n",
57.                getpid());
58.         // Execute the command
59.         char *args[] = {"./bin/sh", "-c", cfg.command, NULL};
60.         execvp(args[0], args);
61.         perror("Execvp failed");
62.         exit(1);
63.     } else {
64.         waitpid(pid, NULL, 0);
65.         printf("[Parent] Stoping...\n");
66.     }
67.     return 0;
68. }
```