



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.

```
s-ahmad-nousavi-awalg@s-ahmad-nousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx: /Desktop/zocker/task2/zocker$ make clean
rm -f zocker
s-ahmad-nousavi-awalg@s-ahmad-nousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx: /Desktop/zocker/task2/zocker$ ls
main.c Makefile
s-ahmad-nousavi-awalg@s-ahmad-nousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx: /Desktop/zocker/task2/zocker$ nano main.c
s-ahmad-nousavi-awalg@s-ahmad-nousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx: /Desktop/zocker/task2/zocker$ ls
main.c Makefile
s-ahmad-nousavi-awalg@s-ahmad-nousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx: /Desktop/zocker/task2/zocker$ make
gcc -Wall -Wextra -std=c99 -o zocker main.c
[sudo] password for s-ahmad-nousavi-awal:
s-ahmad-nousavi-awalg@s-ahmad-nousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx: /Desktop/zocker/task2/zocker$ ls
main.c Makefile zocker
s-ahmad-nousavi-awalg@s-ahmad-nousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx: /Desktop/zocker/task2/zocker$ ./zocker run --name test-container 'sh'
Running child with pid: 1 (in container namespace)
$ 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
$
```

ولی به مشکلی که گفته شده خوردیم 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 (Propagate) می‌شود و می‌تواند سیستم Host را دچار مشکل کند. که برای این مشکل از MS\_PRIVATE استفاده می‌کنیم.

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

```

s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ make clean
rm -f zocker
s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ nano main.c
s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ make
gcc -Wall -Wextra -std=c99 -o zocker main.c
sudo setcap cap_sys_admin+ep zocker
[sudo] password for s-ahmad-mousavi-aa:
s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/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 53265888 18134524 75% /
tmpfs                 7999944    95944   7994000  2% /dev/shm
tmpfs                 1599992    2348   1597644  1% /run
tmpfs                 5120      12     5108    1% /run/lock
tmpfs                 7999944      0   7999944  0% /run/gem
tmpfs                 1599988    540   1599848  1% /run/user/1000
efivarfs              256       119     133    48% /sys/firmware/efi/efivars
/dev/nvme0n1p1        97318     81750   15568   85% /boot/efi
/dev/nvme0n1p4       9488888 5050048 44479568 54% /mnt/windows
$ exit
[Parent] Stopping...
s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ df
Filesystem            1K-blocks      Used Available Use% Mounted on
tmpfs                 1599992    2348   1597644  1% /run
/dev/nvme0n1p5        75192448 53265888 18134524 75% /
tmpfs                 7999944    95944   7994000  2% /dev/shm
tmpfs                 5120      12     5108    1% /run/lock
tmpfs                 7999944      0   7999944  0% /run/gem
tmpfs                 1599988    540   1599848  1% /run/user/1000
efivarfs              256       119     133    48% /sys/firmware/efi/efivars
/dev/nvme0n1p1        97318     81750   15568   85% /boot/efi
/dev/nvme0n1p4       9488888 5050048 44479568 54% /mnt/windows
tmpfs                 1599988    540   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 هرکانینر یک فضای نام به صورت جداگانه اختصاص پیدا می کند که این در عکس زیر مشهود است:

```

s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ ./zocker run --name test-container 'sh'
[ERROR] Failed to open /proc/self/timens_offsets: Permission denied
Running child with pid: 1 (in container namespace)
sh: 0: can't access tty: job control turned off
$ hostname
test-container
$ ps
  PID TTY          TIME CMD
   1 ?        00:00:00 sh
   2 ?        00:00:00 sh
   4 ?        00:00:00 ps
$
s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ cd zocker/
s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ ls
main.c  Makefile  zocker
s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ ./zocker run --name test-container2 'sh'
[ERROR] Failed to open /proc/self/timens_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 استفاده کنیم که با استفاده از آن توانستیم اسم را به کانینر اختصاص دهیم مانند زیر:

```

test-container
s-ahmad-mousavi-aa@test-container:~$ hostname
test-container
s-ahmad-mousavi-aa@test-container:~$
s-ahmad-mousavi-aa@ahmad-mousavi-aal-Victus-by-HP-Gaming-Laptop-15-fa1xxx:~/Desktop/zockerTask1/zocker$ sudo ./zocker run --name isolated-box 'sh'
Running child with pid: 1 (in container namespace)
sh: 0: can't access tty: job control turned off
# hostname
isolated-box

```

همانطور که مشهود است دستور فوق در کانینر 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 0 شروع شده که مشخصا زمان دستگاه آن نیست:

```
s-ahmad-mousavi-awal@s-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx:~/Desktop/dockerTask2/docker$ sudo ./zocker run --name time-test 'cat /proc/uptime'
Running child with pid: 1 (in container namespace)
0.00 170950.90
[Parent] Stopping...
s-ahmad-mousavi-awal@s-ahmad-mousavi-awal-Victus-by-HP-Gaming-Laptop-15-Fa1xxx:~/Desktop/dockerTask2/docker$
```

کدهای زده شده به شرح زیر است که فایل main.c هم در کنار همین فایل قرار دارد و می توانید 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 (pid == 0) {
40.         if (sethostname(cfg.name, strlen(cfg.name)) != 0) {
41.             perror("Failed to set hostname");
42.             exit(1);
43.         }
44.     }
45. }
```

```
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] Stopping...\n");
66. }
67. return 0;
68. }
```