به نام خدا

امیرحسین میرزائی
۴۰۲۱۰۶۶۶۱
محمدامین کوهی
۴۰۲۱۰۶۴۰۱

درس سیستم‌عامل
نیم‌سال اول ۰۴-۰۵
استاد: دکتر اسدی

دانشکده مهندسی کامپیوتر

تمرین عملی سری اول

سؤال دوم

(آ)

(ب) دستور خواسته شده را شل جدید اجرا می‌کنیم.

```
$ ps
fatal library error, lookup self
$ top
top - 13:51:42 up 13 min,  0 user,  load average: 0.82, 0.77, 0.64
Tasks:  24 total,   1 running,  23 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.4 us,  7.9 sy,  0.0 ni, 88.1 id,  1.5 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :  15996.0 total,    286.2 free,   2034.3 used,  14085.3 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.  13961.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2071 codespa+  20   0   52.8g 485028  66620 S   1.0   3.0   0:20.29 node
   2521 codespa+  20   0 1140560  77560  49024 S   0.3   0.5   0:01.28 node
      1 codespa+  20   0    1136    640    640 S   0.0   0.0   0:00.03 docker-init
      7 codespa+  20   0    2800   1408   1408 S   0.0   0.0   0:00.14 sh
    127 root      20   0   12016   2964   1920 S   0.0   0.0   0:00.00 sshd
    289 root      20   0 2043708  81320  56064 S   0.0   0.5   0:00.46 dockerd
    525 root      20   0 1876964  49880  34816 S   0.0   0.3   0:00.11 containerd
   1667 codespa+  20   0    2800   1664   1664 S   0.0   0.0   0:00.00 sh
   1738 root      20   0    2800   1536   1536 S   0.0   0.0   0:00.00 sh
   2012 codespa+  20   0    2808   1664   1664 S   0.0   0.0   0:00.00 sh
   2021 codespa+  20   0   11.3g 175376  54656 S   0.0   1.1   0:06.48 node
   2085 codespa+  20   0 1262800  60836  48640 S   0.0   0.4   0:00.19 node
   3028 codespa+  20   0   16712  12672   3584 S   0.0   0.1   0:00.22 bash
   7286 codespa+  20   0    2800   1536   1536 S   0.0   0.0   0:00.00 sh
   7351 root      20   0    2800   1664   1664 S   0.0   0.0   0:00.00 sh
   7581 codespa+  20   0   16580  12416   3456 S   0.0   0.1   0:00.09 bash
   8512 codespa+  20   0    2552   1152   1152 S   0.0   0.0   0:00.00 zocker
   8513 codespa+  20   0    2804   1664   1664 S   0.0   0.0   0:00.00 sh
   8514 codespa+  20   0    2804   1664   1664 S   0.0   0.0   0:00.00 sh
   9262 codespa+  20   0   12348   5376   3328 R   0.0   0.0   0:00.00 top
   9382 codespa+  20   0    6112   1792   1792 S   0.0   0.0   0:00.00 sleep
   9383 codespa+  20   0    2804   1664   1664 S   0.0   0.0   0:00.00 sh
   9384 codespa+  20   0    7744   3328   3072 S   0.0   0.0   0:00.00 cpuUsage.sh
   9387 codespa+  20   0    6116   1920   1920 S   0.0   0.0   0:00.00 sleep
```

سوال: در این بخش دستور ps با ارور برخورد می‌کند زیرا وقتی نیم‌اسپیس pid های آن را تغییر می‌دهیم دیگر دسترسی به libc سیستم را نداشته می‌شود و به ارور می‌خورد.

به عنوان دستور جایگزین ما top را اجرا کردیم. مشکل این است که دستورps اطلاعات پردازه‌ها را از فایل‌سیستم proc می‌خواند، نه از kernel . وقتی فقط CLONE_NEWPID استفاده می‌کنیم، فایل‌سیستم جدید ایجاد نمی‌شود، بنابراین proc همچنان به فایل‌سیستم host اشاره می‌کند.

(ج) پس اعمال تغییرات نتیجه به شکل زیر می‌شود:

```
@MohammadAminKoohi → .../Operating-Systems-Practical/HW4/P2/zocker (5abbe3f) $ ./zocker run
 --name test -container 'sh'
Running child with pid: 1
$ ps
    PID TTY          TIME CMD
      1 pts/3    00:00:00 sh
      2 pts/3    00:00:00 sh
      3 pts/3    00:00:00 ps
$
```

کد را به شکل زیر تغییر می‌دهیم

```c
if (pid == 0) {
    printf("Running child with pid: %d\n", getpid());

    if (mount("proc", "/proc", "proc", 0, NULL) != 0) {
        fprintf(stderr, "[ERR] Failed to mount proc filesystem.\n");
        return 1;
    }

    execl("/bin/sh", "sh", "-c", cfg.command, NULL);
```

(د) به مشکلی برخورد نکرد.

```
$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
overlay         32847680 11250676  19902908  37% /
tmpfs              65536        0     65536   0% /dev
shm                65536        0     65536   0% /dev/shm
/dev/root       30298176 22332988   7948804  74% /vscode
/dev/sdc1      123266624  2859240 114099648   3% /tmp
/dev/loop4      32847680 11250676  19902908  37% /workspaces
$
```

سوال: جدول مونت مستقل هر فضای نام مونت (mount namespace) جدول مونت مخصوص به خود را دارد. هنگامی که شما proc را در فضای فرزند مونت می‌کنید، این عمل تنها بر روی فضای نام جاری تأثیر می‌گذارد.

MS_SHARED در مقابل **MS_PRIVATE** به طور پیش‌فرض، انتشار مونت‌ها (mount propagation) روی حالت MS_PRIVATE تنظیم شده است که به این معنی است:

- مونت‌های انجام شده در فضای فرزند به فضای والد انتشار نمی‌یابند.
- مونت‌های موجود در فضای والد نیز به فضای فرزند منتشر نمی‌شوند.

این رفتار منجر به بروز مشکل «مونت‌های شکسته» (broken mounts) می‌شود.

سلسله‌مراتب مونت‌ها هسته لینوکس یک سلسله‌مراتب از نقاط مونت را نگهداری می‌کند. هنگامی که یک فضای نام را

بدون تنظیمات مناسب برای انتشار مونت‌ها ایزوله می‌کنید، مونت‌های زیردست (subordinate mounts) غیرقابل دسترس می‌شوند.

(ه) تنظیمات انتشار MS_PRIVATE | MS_REC بر روی فایل سیستم ریشه انجام می‌شود تا از بروز مشکلات انتشار مونت بین فضای‌های نام والد و فرزند جلوگیری شود.
این تنظیمات مانع از تأثیرگذاری مونت‌های درون کانتینر بر روی سیستم میزبان و بالعکس می‌شود.

```
if (mount(NULL, "/", NULL, MS_PRIVATE | MS_REC, NULL) != 0) {
    fprintf(stderr, "[ERR] Failed to set mount propagation.\n");
    return 1;
}
```

(و) برای این بخش کافی است فلگ CLONE_NEWUTS را به سیسکال Unshare اضافه کنیم. سپس با sethost-name برای این کانتینر یک اسم انتخاب می‌کنیم.

```
if (unshare(CLONE_NEWPID | CLONE_NEWNS | CLONE_NEWUTS) != 0) {
    fprintf(stderr, "[ERR] Failed to unshare(2).");
    return 1;
}

if (mount(NULL, "/", NULL, MS_PRIVATE | MS_REC, NULL) != 0) {
    fprintf(stderr, "[ERR] Failed to set mount propagation.\n");
    return 1;
}


if (sethostname(cfg.name, strlen(cfg.name)) != 0) {
    fprintf(stderr, "[ERR] Failed to set hostname.\n");
    return 1;
}
```