

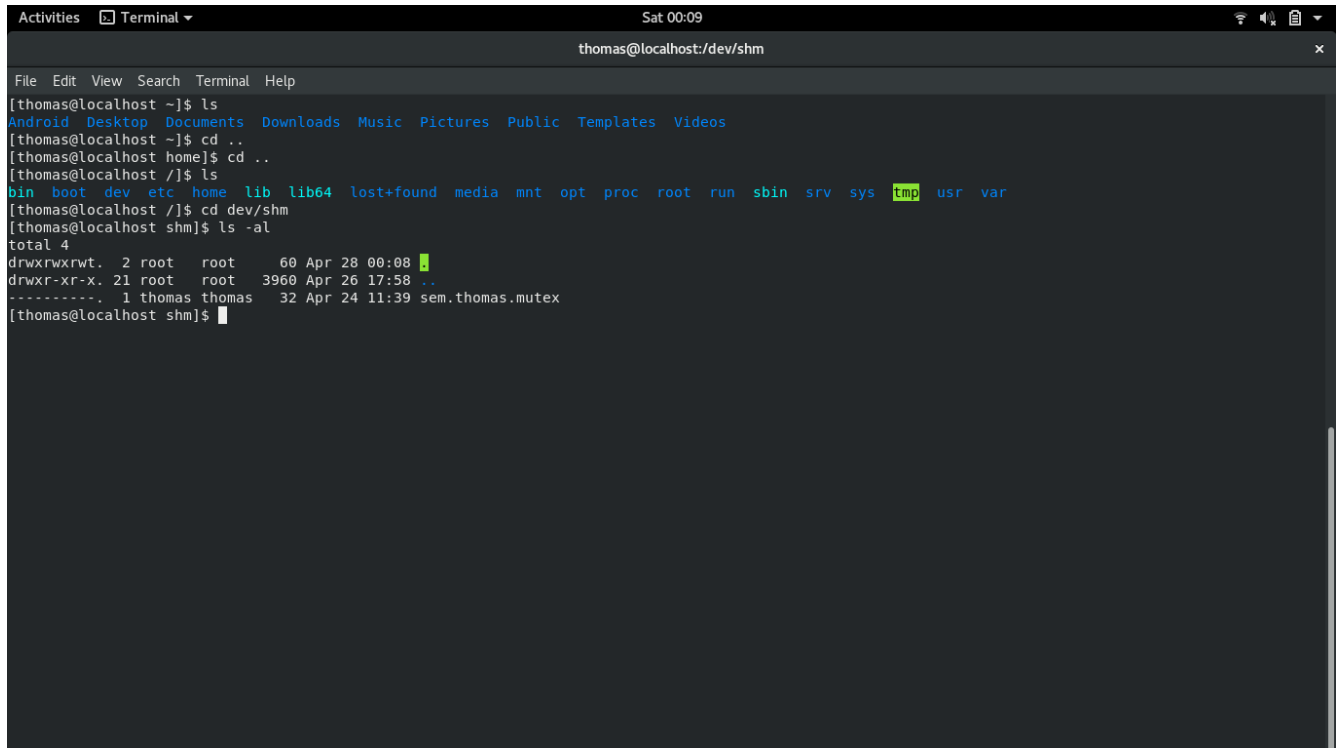
Thomas Pedraza

25 APR 2018

CSC 562

Assignment 6

`sem_unlink()` removes a specific semaphore. If the unlink call is absent from a program that uses semaphores, the mutex file with the given name will remain in the directory `dev/shm`. Figure 1 demonstrates the exclusion of `sem_unlink()`.



```
Activities Terminal Sat 00:09
thomas@localhost:/dev/shm

File Edit View Search Terminal Help

[thomas@localhost ~]$ ls
Android Desktop Documents Downloads Music Pictures Public Templates Videos
[thomas@localhost ~]$ cd ..
[thomas@localhost home]$ cd ..
[thomas@localhost /]$ ls
bin boot dev etc home lib lib64 lost+found media mnt opt proc root run sbin srv sys tmp usr var
[thomas@localhost /]$ cd dev/shm
[thomas@localhost shm]$ ls -al
total 4
drwxrwxrwt. 2 root root 60 Apr 28 00:08 .
drwxr-xr-x. 21 root root 3960 Apr 26 17:58 ..
----- 1 thomas thomas 32 Apr 24 11:39 sem.thomas.mutex
[thomas@localhost shm]$
```

Figure 1: `sem_unlink()` excluded from program

Figure 2 shows 3 different parameters in `prseq`, all which count sequentially from 0 to maxnum using semaphores.

```
Activities Terminal Sat 01:45
thomas@localhost:~/Documents/Assignment6
File Edit View Search Terminal Help
'Assignment 6.odt' prseq prseq.c
[thomas@localhost Assignment6]$ gcc -o prseq prseq.c -lpthread
[thomas@localhost Assignment6]$ ./prseq 10 5
Invalid use of command.
[thomas@localhost Assignment6]$ ./prseq 19
1st Child: 0
Parent: 1
2nd child: 2
1st Child: 3
Parent: 4
2nd child: 5
1st Child: 6
Parent: 7
2nd child: 8
1st Child: 9
Parent: 10
2nd child: 11
1st Child: 12
Parent: 13
2nd child: 14
1st Child: 15
Parent: 16
2nd child: 17
1st Child: 18
Parent: 19
[thomas@localhost Assignment6]$ ./prseq 5
1st Child: 0
Parent: 1
2nd child: 2
1st Child: 3
Parent: 4
2nd child: 5
[thomas@localhost Assignment6]$ ./prseq 2
1st Child: 0
Parent: 1
2nd child: 2
[thomas@localhost Assignment6]$

prseq.c — Assignment6 — ~/Documents/Assignment6 — Atom
File Edit View Selection Find Packages Help
prseq.c — Assignment5 prseq.c — Assignment6 unlinksems.c
56 printf("1st Child: %d\n", i);
57 sem_post(sem1);
58 sem_wait(sem3);
59 }
60 //Parent process
61 } else {
62 // Fork the second child
63 pid = fork();
64 //If the pid < 0 then an error has occurred
65 if (pid < 0) {
66 fprintf(stderr, "Second Fork Failed\n");
67 return 1;
68 //Second child
69 } else if (pid == 0) {
70 //Second child prints 2, 5, 8, ...
71 for (int i = 2; i <= maxnum; i = i + 3) {
72 //Prepare for critical section
73 sem_wait(sem2);
74 //Critical section
75 printf("2nd child: %d\n", i);
76 sem_post(sem3);
77 }
78 sem_post(sem3);
79 //Parent process
80 } else {
81 //Parent prints 1, 4, 7, ...
82 for (int i = 1; i <= maxnum; i = i + 3) {
83 //Prepare for critical section
84 sem_wait(sem1);
85 printf("1st Child: %d\n", i);
86 sem_post(sem3);
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
```

Figure 2: prseq parameters of 19, 5, and 2.