刘宇

2014213404

19 班

In this assignment, I finish both two projects.

Project 1—UNIX Shell and History Feature

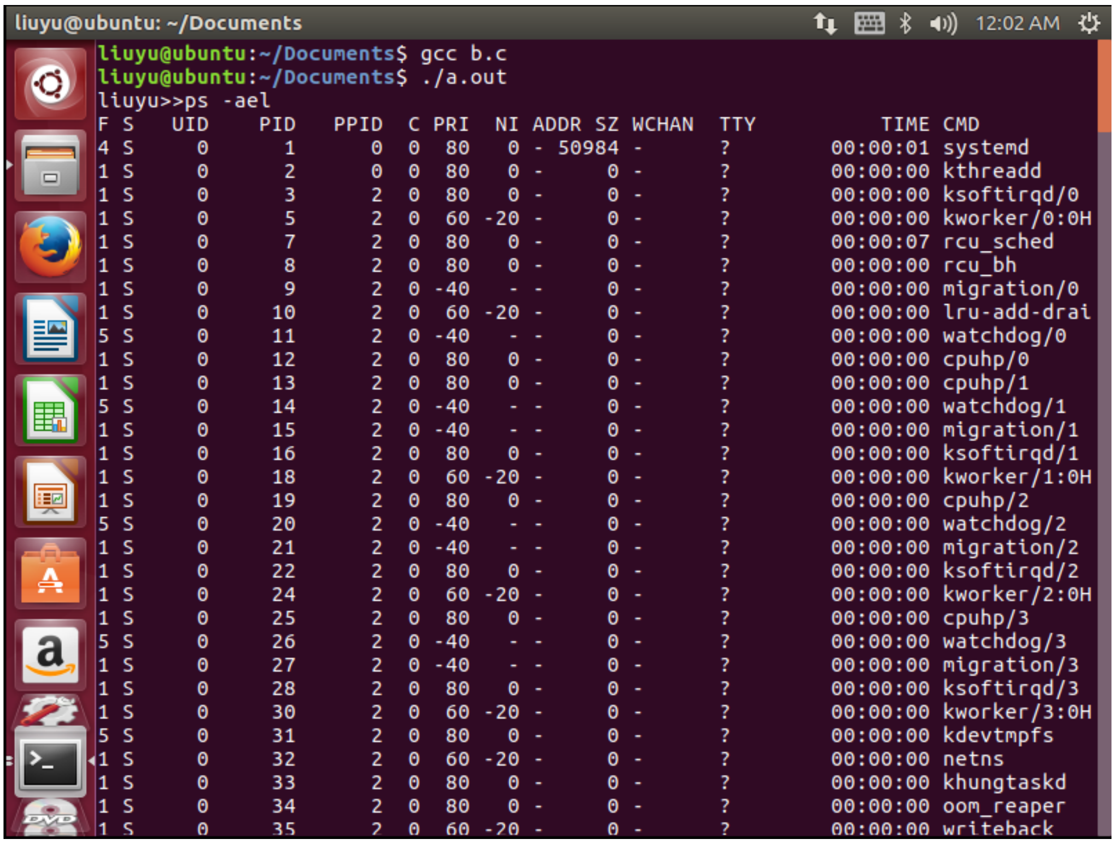
Part I— Creating a Child Process

In the part, I design a shell interface to accept the user commands and then executes each command in a separate process. First I define the pointer array char *args[MAX LINE/2 + 1], which can be used to accept the user commands(points to separate tokens that user has entered into). Then I initialize the pointer array by

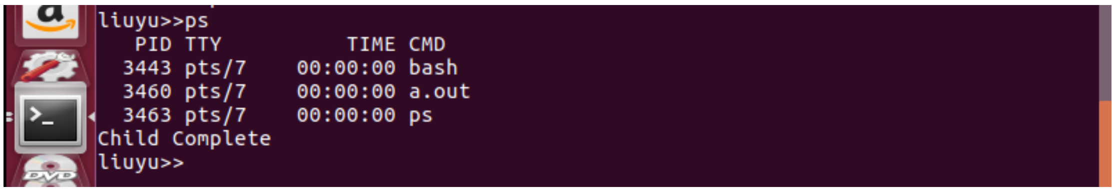
```
 for(i=0;i<MAX_LINE/2 + 1;i++)
{
 args[i]=(char *)malloc(20*sizeof(char));
 strcpy(args[i],"");
 }
```

Then I use while((c=getchar())!='\n') to traverse the user commands. I use char a[2]={c}; strcat(args[i],a); to add the character to specific location in args until the character is blank space. Finally, each separate token entered by user is stored in args. Then I use if statement to judge if args[0] is “exit”. If args[0] is “exit”, the should_run is zero, which means the program will be discontinued. And my code also judge if the last token in args Is “&”. If the last token is “&”, flag h is 1, otherwise h is 0. Then in the while(should_run) loop, I use pid=fork() to generate the child process. And I use if…else statement to judge whether the process is child process or parent process. In the parent process, I judge if command included &, parent will invoke wait(). You can see more details in my first C file called Project1-1.

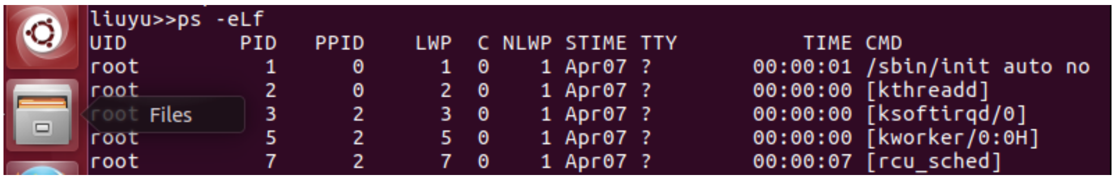
At the beginning, I show that the program can execute user command ps -ael, ps -eLf



```
liuyu@ubuntu:~/Documents$ gcc b.c
liuyu@ubuntu:~/Documents$ ./a.out
liuyu>>ps -ael
F S   UID    PID   PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0      1      0  0  80   0 - 50984 -      ?        00:00:01 systemd
1 S     0      2      0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 S     0      3      2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 S     0      5      2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H
1 S     0      7      2  0  80   0 -     0 -      ?        00:00:07 rcu_sched
1 S     0      8      2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0      9      2  0 -40   - -     0 -      ?        00:00:00 migration/0
1 S     0     10      2  0  60 -20 -     0 -      ?        00:00:00 lru-add-drai
5 S     0     11      2  0 -40   - -     0 -      ?        00:00:00 watchdog/0
1 S     0     12      2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
1 S     0     13      2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
5 S     0     14      2  0 -40   - -     0 -      ?        00:00:00 watchdog/1
1 S     0     15      2  0 -40   - -     0 -      ?        00:00:00 migration/1
1 S     0     16      2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/1
1 S     0     18      2  0  60 -20 -     0 -      ?        00:00:00 kworker/1:0H
1 S     0     19      2  0  80   0 -     0 -      ?        00:00:00 cpuhp/2
5 S     0     20      2  0 -40   - -     0 -      ?        00:00:00 watchdog/2
1 S     0     21      2  0 -40   - -     0 -      ?        00:00:00 migration/2
1 S     0     22      2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/2
1 S     0     24      2  0  60 -20 -     0 -      ?        00:00:00 kworker/2:0H
1 S     0     25      2  0  80   0 -     0 -      ?        00:00:00 cpuhp/3
5 S     0     26      2  0 -40   - -     0 -      ?        00:00:00 watchdog/3
1 S     0     27      2  0 -40   - -     0 -      ?        00:00:00 migration/3
1 S     0     28      2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/3
1 S     0     30      2  0  60 -20 -     0 -      ?        00:00:00 kworker/3:0H
5 S     0     31      2  0  80   0 -     0 -      ?        00:00:00 kdevtmpfs
1 S     0     32      2  0  60 -20 -     0 -      ?        00:00:00 netns
1 S     0     33      2  0  80   0 -     0 -      ?        00:00:00 khungtaskd
1 S     0     34      2  0  80   0 -     0 -      ?        00:00:00 oom_reaper
1 S     0     35      2  0  60 -20 -     0 -      ?        00:00:00 writeback
```



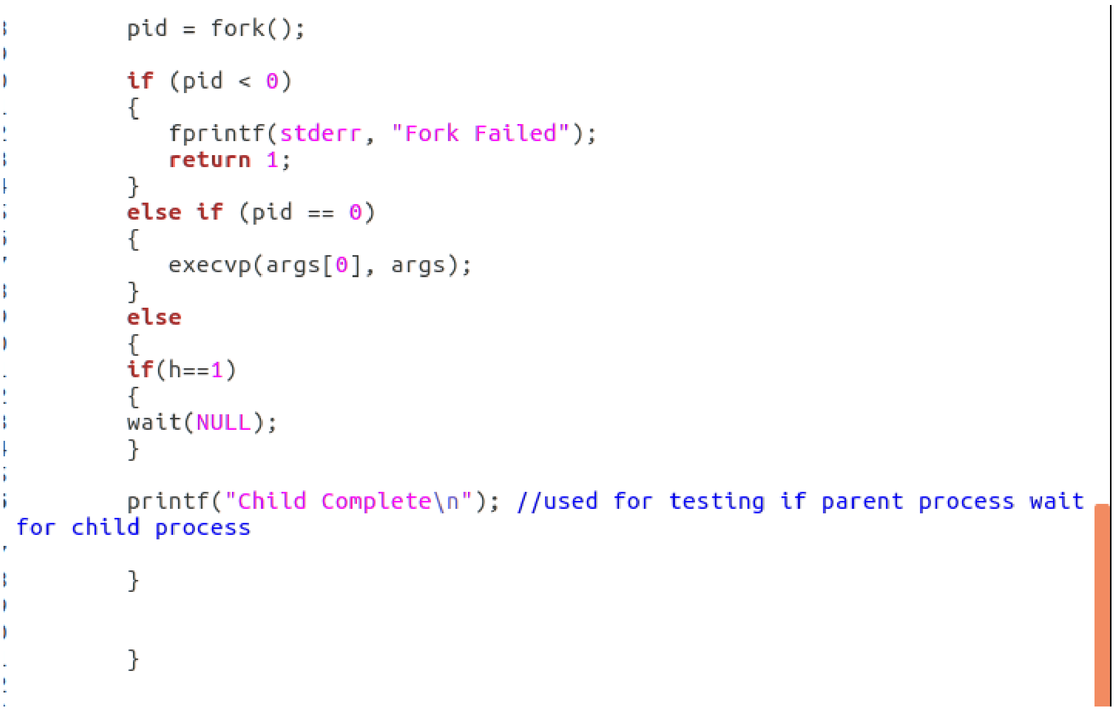
```
liuyu>>ps
   PID TTY          TIME CMD
  3443 pts/7    00:00:00 bash
  3460 pts/7    00:00:00 a.out
  3463 pts/7    00:00:00 ps
Child Complete
liuyu>>
```



```
liuyu>>ps -eLf
UID        PID  PPID   LWP  C NLWP STIME TTY          TIME CMD
root         1     0     1  0    1 Apr07 ?        00:00:01 /sbin/init auto no
root         2     0     2  0    1 Apr07 ?        00:00:00 [kthreadd]
root         3     2     3  0    1 Apr07 ?        00:00:00 [ksoftirqd/0]
root         5     2     5  0    1 Apr07 ?        00:00:00 [kworker/0:0H]
root         7     2     7  0    1 Apr07 ?        00:00:07 [rcu_sched]
```

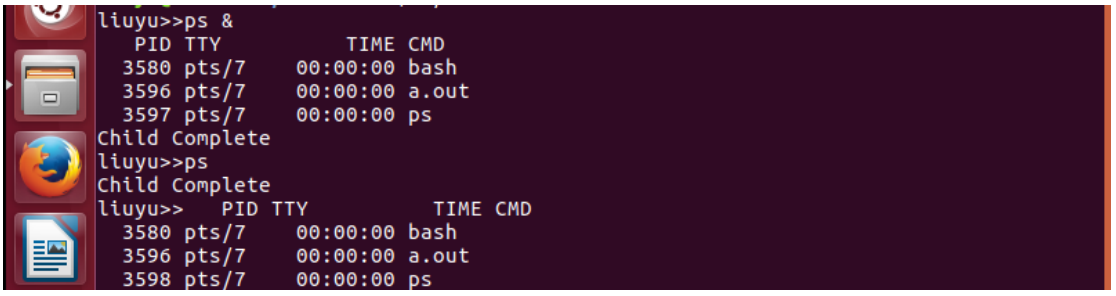
The program can also execute other user commands.

Next, I will show that the “&” can determine whether the parent process should wait child process.



```
        pid = fork();

        if (pid < 0)
        {
           fprintf(stderr, "Fork Failed");
           return 1;
        }
        else if (pid == 0)
        {
           execvp(args[0], args);
        }
        else
        {
        if(h==1)
        {
        wait(NULL);
        }

        printf("Child Complete\n"); //used for testing if parent process wait
 for child process

        }


        }
```

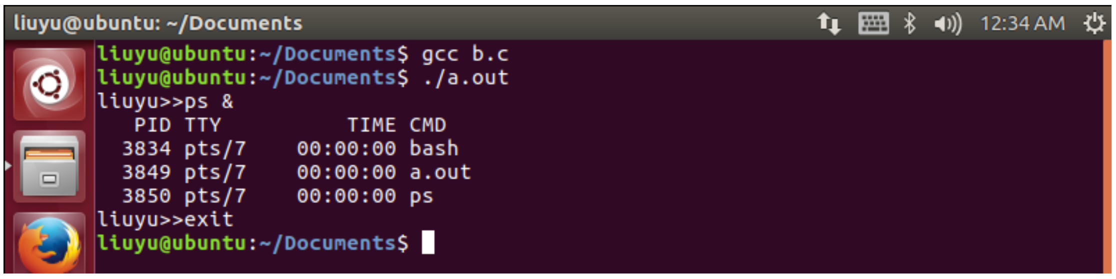
You can see from part of code, the parent process will execute printf("Child Complete\n")
If I enter the command ps &, the parent process will wait child process, and message “Child Complete” is printed after child process finish.
If I don’t enter &, the parent process won’t wait child process, and message “Child Complete” is printed before child process finish.



```
liuyu>>ps &
   PID TTY          TIME CMD
  3580 pts/7    00:00:00 bash
  3596 pts/7    00:00:00 a.out
  3597 pts/7    00:00:00 ps
Child Complete
liuyu>>ps
Child Complete
liuyu>>   PID TTY          TIME CMD
  3580 pts/7    00:00:00 bash
  3596 pts/7    00:00:00 a.out
  3598 pts/7    00:00:00 ps
```

From the picture, the message is printed in different places.

Finally, I should test “exit”.



```
liuyu@ubuntu: ~/Documents
liuyu@ubuntu:~/Documents$ gcc b.c
liuyu@ubuntu:~/Documents$ ./a.out
liuyu>>ps &
   PID TTY          TIME CMD
  3834 pts/7    00:00:00 bash
  3849 pts/7    00:00:00 a.out
  3850 pts/7    00:00:00 ps
liuyu>>exit
liuyu@ubuntu:~/Documents$
```

Part II—Creating a History Feature

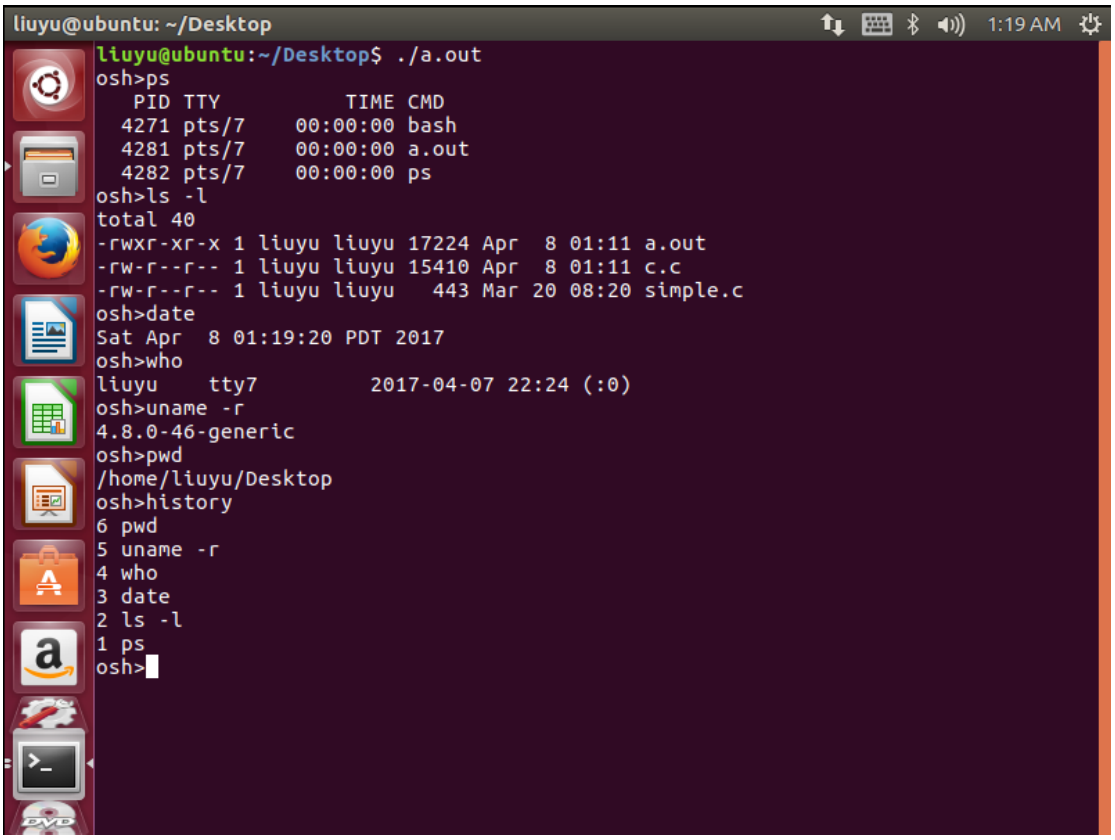
In this part, I create several pointer array named number1, number2 to accept the user command stored in args. Each pointer array will accept a token of user command. For example, number1[0] and number2[0] together represents the first user command.

As for the design of “history”, I assume that if the number of user commands is less than 10, I just print all user commands in combination of number1 and number2. If the number of user commands is greater than 11, I print 10 most recent commands in combination of number1 and number2.

As for the design of “!!”, I pass the number1[i] and number2[i] (i is the most recent command in the history) to args[0] and args[1]. Then use execvp(args[0], args). What’ more, I also consider the last element in args should be NULL.

As for the design of “!N”, I use num=args[0][1]-'0' to judge the value of N. And I use switch statement to consider all conditions.

Firstly, I can show the function of “history”. I enter ps, ls -l, date , who, uname -r, pwd then I input history.

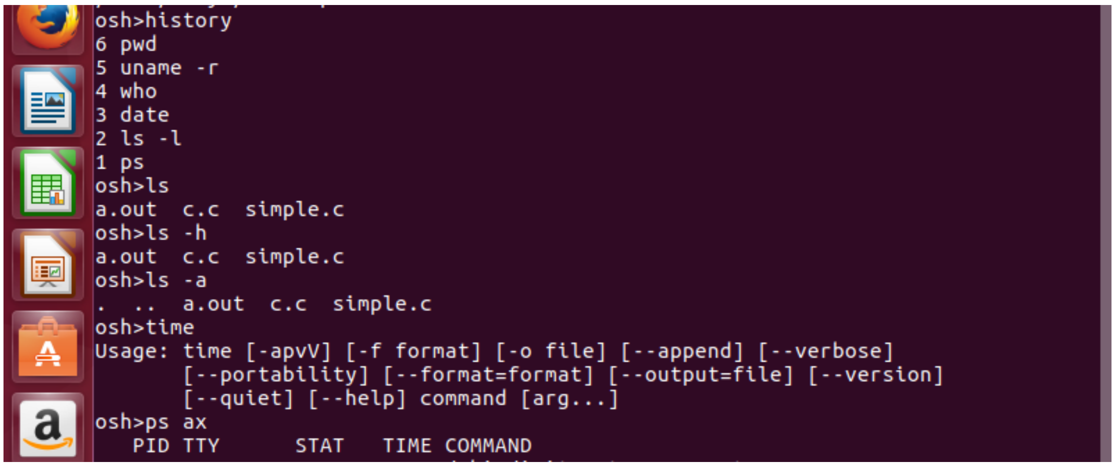


```
liuyu@ubuntu:~/Desktop$ ./a.out
osh>ps
   PID TTY          TIME CMD
  4271 pts/7    00:00:00 bash
  4281 pts/7    00:00:00 a.out
  4282 pts/7    00:00:00 ps
osh>ls -l
total 40
-rwxr-xr-x 1 liuyu liuyu 17224 Apr  8 01:11 a.out
-rw-r--r-- 1 liuyu liuyu 15410 Apr  8 01:11 c.c
-rw-r--r-- 1 liuyu liuyu   443 Mar 20 08:20 simple.c
osh>date
Sat Apr  8 01:19:20 PDT 2017
osh>who
liuyu    tty7         2017-04-07 22:24 (:0)
osh>uname -r
4.8.0-46-generic
osh>pwd
/home/liuyu/Desktop
osh>history
6 pwd
5 uname -r
4 who
3 date
2 ls -l
1 ps
osh>
```

You can see from the picture, these commands are printed in order.

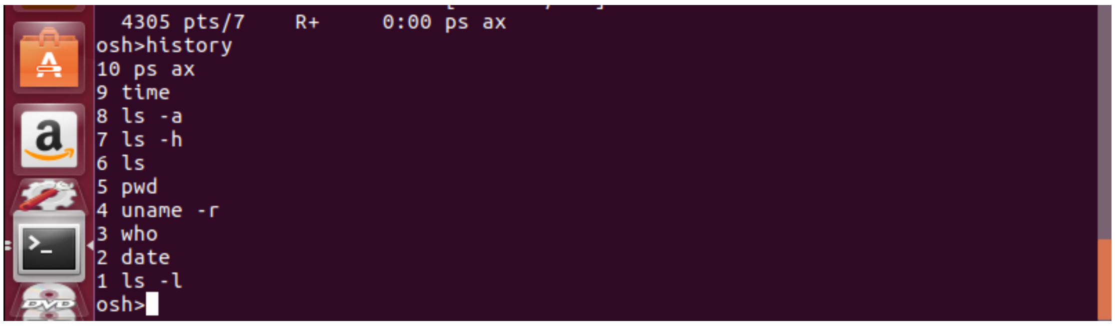
If I input more commands, the 10 most recent commands will be printed.

I continue to input ls, ls -h, ls -a, time, ps ax



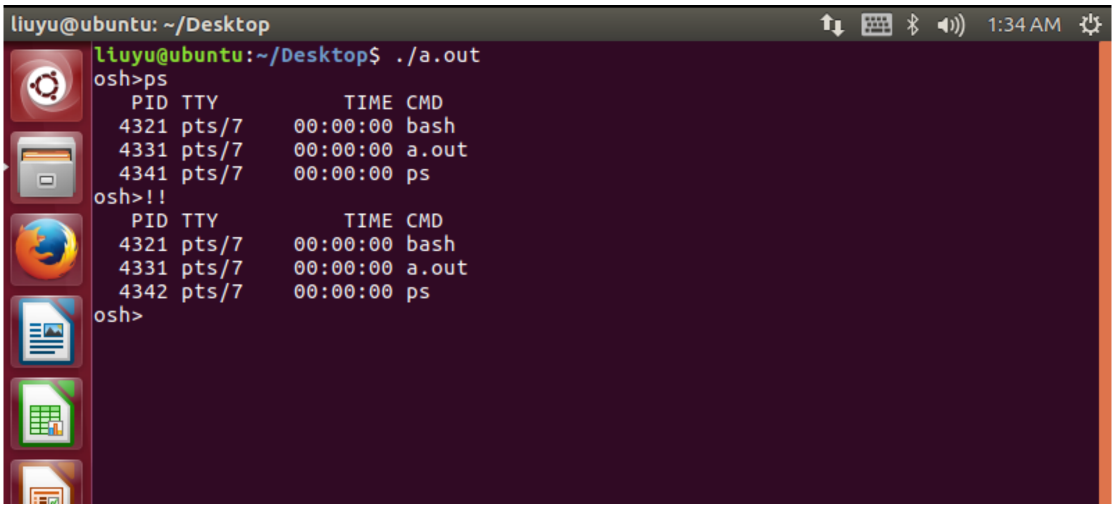
```
osh>history
6 pwd
5 uname -r
4 who
3 date
2 ls -l
1 ps
osh>ls
a.out  c.c  simple.c
osh>ls -h
a.out  c.c  simple.c
osh>ls -a
.  ..  a.out  c.c  simple.c
osh>time
Usage: time [-apvV] [-f format] [-o file] [--append] [--verbose]
       [--portability] [--format=format] [--output=file] [--version]
       [--quiet] [--help] command [arg...]
osh>ps ax
   PID TTY      STAT   TIME COMMAND
```

Then you can see only the 10 most recent commands are printed.



```
  4305 pts/7    R+     0:00 ps ax
osh>history
10 ps ax
9 time
8 ls -a
7 ls -h
6 ls
5 pwd
4 uname -r
3 who
2 date
1 ls -l
osh>
```

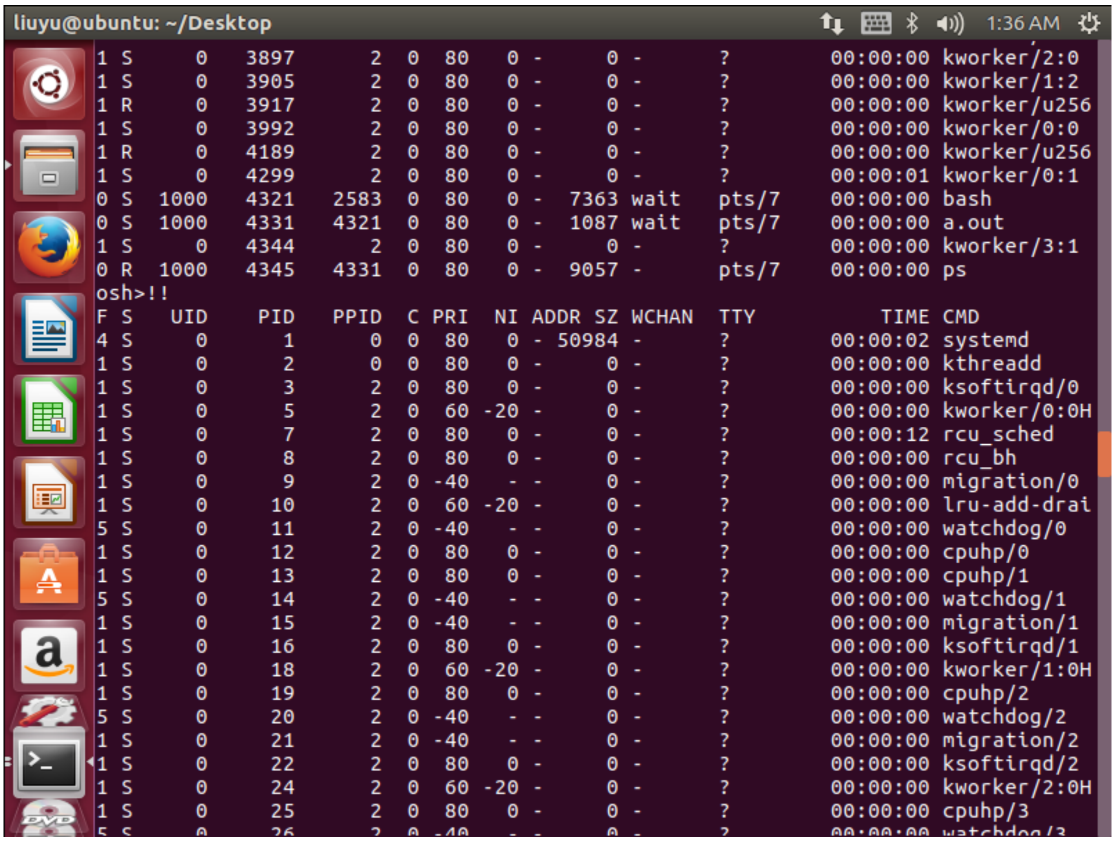
Then I check "!!"



```
liuyu@ubuntu: ~/Desktop                                    1:34 AM
liuyu@ubuntu:~/Desktop$ ./a.out
osh>ps
   PID TTY          TIME CMD
  4321 pts/7    00:00:00 bash
  4331 pts/7    00:00:00 a.out
  4341 pts/7    00:00:00 ps
osh>!!
   PID TTY          TIME CMD
  4321 pts/7    00:00:00 bash
  4331 pts/7    00:00:00 a.out
  4342 pts/7    00:00:00 ps
osh>
```

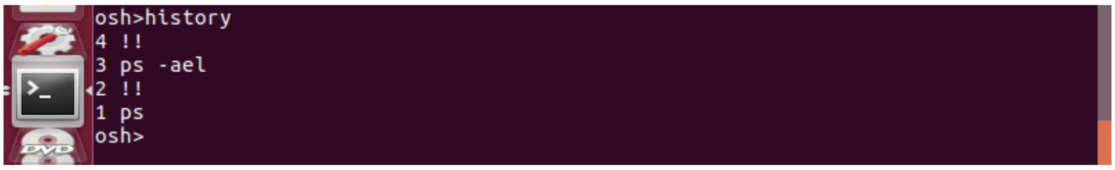
ps as the most recent command is executed when I enter !!.

I continue to enter ps -ael, and then I input !!. ps -ael as the most recent command is executed.



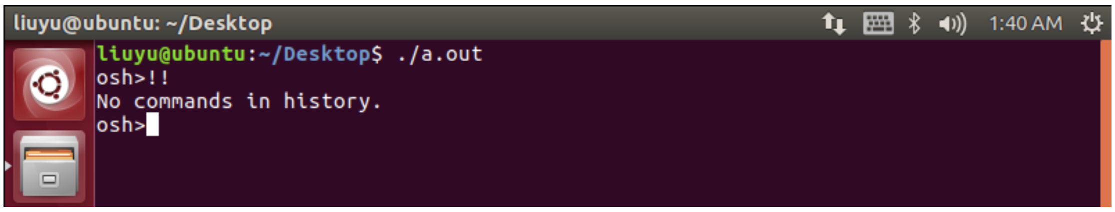
```
liuyu@ubuntu: ~/Desktop                                   1:36 AM
1 S     0  3897     2  0  80   0 -     0 -      ?        00:00:00 kworker/2:0
1 S     0  3905     2  0  80   0 -     0 -      ?        00:00:00 kworker/1:2
1 R     0  3917     2  0  80   0 -     0 -      ?        00:00:00 kworker/u256
1 S     0  3992     2  0  80   0 -     0 -      ?        00:00:00 kworker/0:0
1 R     0  4189     2  0  80   0 -     0 -      ?        00:00:00 kworker/u256
1 S     0  4299     2  0  80   0 -     0 -      ?        00:00:01 kworker/0:1
0 S  1000  4321  2583  0  80   0 -  7363 wait   pts/7    00:00:00 bash
0 S  1000  4331  4321  0  80   0 -  1087 wait   pts/7    00:00:00 a.out
1 S     0  4344     2  0  80   0 -     0 -      ?        00:00:00 kworker/3:1
0 R  1000  4345  4331  0  80   0 -  9057 -      pts/7    00:00:00 ps
osh>!!
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 50984 -      ?        00:00:02 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 S     0     3     2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 S     0     5     2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H
1 S     0     7     2  0  80   0 -     0 -      ?        00:00:12 rcu_sched
1 S     0     8     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0     9     2  0 -40   - -     0 -      ?        00:00:00 migration/0
1 S     0    10     2  0  60 -20 -     0 -      ?        00:00:00 lru-add-drai
5 S     0    11     2  0 -40   - -     0 -      ?        00:00:00 watchdog/0
1 S     0    12     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
1 S     0    13     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
5 S     0    14     2  0 -40   - -     0 -      ?        00:00:00 watchdog/1
1 S     0    15     2  0 -40   - -     0 -      ?        00:00:00 migration/1
1 S     0    16     2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/1
1 S     0    18     2  0  60 -20 -     0 -      ?        00:00:00 kworker/1:0H
1 S     0    19     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/2
5 S     0    20     2  0 -40   - -     0 -      ?        00:00:00 watchdog/2
1 S     0    21     2  0 -40   - -     0 -      ?        00:00:00 migration/2
1 S     0    22     2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/2
1 S     0    24     2  0  60 -20 -     0 -      ?        00:00:00 kworker/2:0H
1 S     0    25     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/3
5 S     0    26     2  0 -40   - -     0 -      ?        00:00:00 watchdog/3
```

You can also see my history buffer.



```
osh>history
4 !!
3 ps -ael
2 !!
1 ps
osh>
```

If the history buffer is empty, the message ""No commands in history." will be printed.

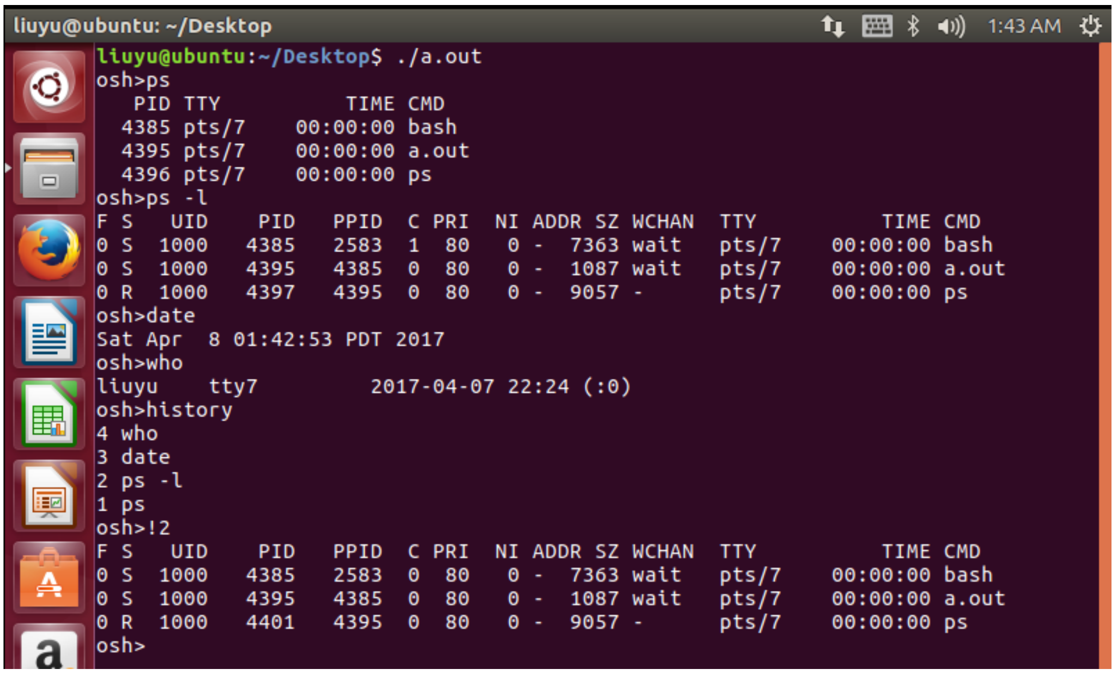


```
liuyu@ubuntu: ~/Desktop                                   1:40 AM
liuyu@ubuntu:~/Desktop$ ./a.out
osh>!!
No commands in history.
osh>
```

Next, I will test !N.

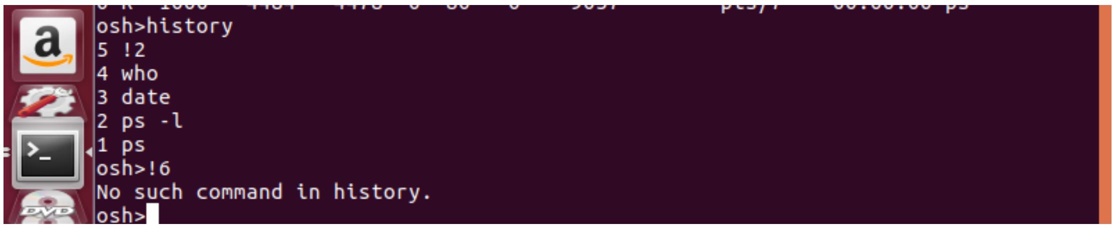
I input four commands ps, ps -l, date, who

Then in the history buffer, I find that ps -l is number 2, so I input !2, and ps -l is executed.



```
liuyu@ubuntu: ~/Desktop                                           1:43 AM
liuyu@ubuntu:~/Desktop$ ./a.out
osh>ps
   PID TTY          TIME CMD
  4385 pts/7    00:00:00 bash
  4395 pts/7    00:00:00 a.out
  4396 pts/7    00:00:00 ps
osh>ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  4385  2583  1  80   0 -  7363 wait   pts/7    00:00:00 bash
0 S  1000  4395  4385  0  80   0 -  1087 wait   pts/7    00:00:00 a.out
0 R  1000  4397  4395  0  80   0 -  9057 -      pts/7    00:00:00 ps
osh>date
Sat Apr  8 01:42:53 PDT 2017
osh>who
liuyu    tty7         2017-04-07 22:24 (:0)
osh>history
4 who
3 date
2 ps -l
1 ps
osh>!2
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  4385  2583  0  80   0 -  7363 wait   pts/7    00:00:00 bash
0 S  1000  4395  4385  0  80   0 -  1087 wait   pts/7    00:00:00 a.out
0 R  1000  4401  4395  0  80   0 -  9057 -      pts/7    00:00:00 ps
osh>
```

Currently, there are four elements in history buffer. When I enter i6 the message "No such command in history." will be printed.



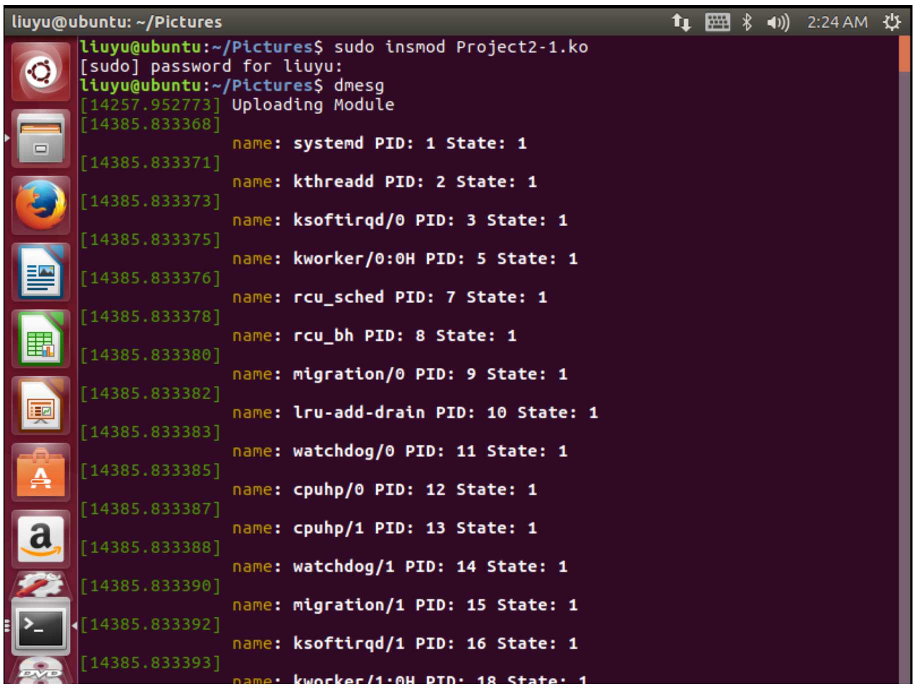
```
osh>history
5 !2
4 who
3 date
2 ps -l
1 ps
osh>!6
No such command in history.
osh>
```

And you can also input !N(N is from 1 to 9), the program can also work well(can use !! to execute the most recent command).

Project 2—Linux Kernel Module for Listing Tasks

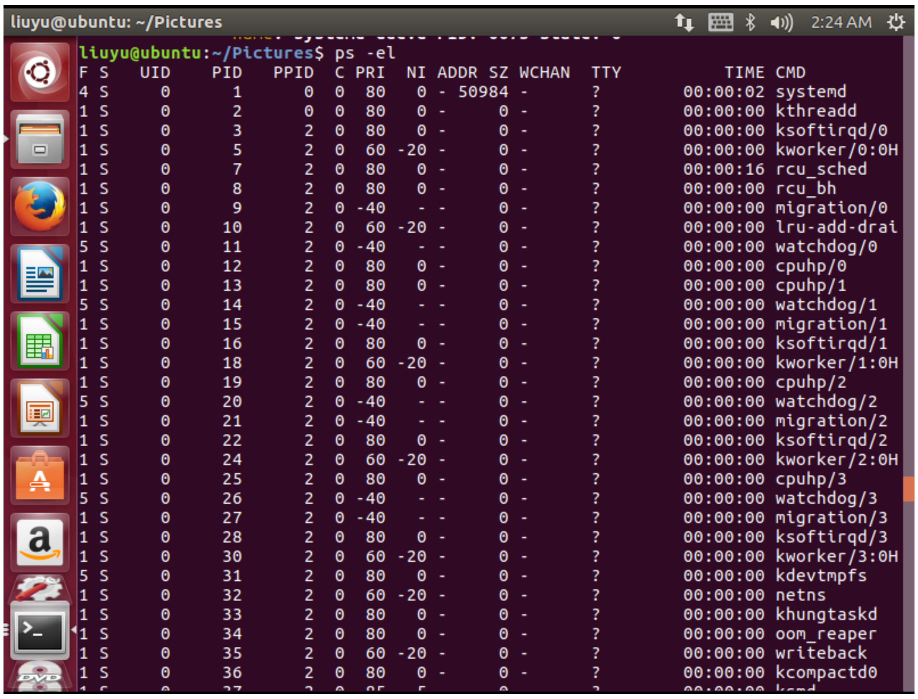
Part I—Iterating over Tasks Linearly

In this part, I use for_each_process() macro to iterate over all current tasks in the system.
comm is the name of process. pid is the process's pid. state is the process's state.



```
liuyu@ubuntu:~/Pictures$ sudo insmod Project2-1.ko
[sudo] password for liuyu:
liuyu@ubuntu:~/Pictures$ dmesg
[14257.952773] Uploading Module
[14385.833368]
               name: systemd PID: 1 State: 1
[14385.833371]
               name: kthreadd PID: 2 State: 1
[14385.833373]
               name: ksoftirqd/0 PID: 3 State: 1
[14385.833375]
               name: kworker/0:0H PID: 5 State: 1
[14385.833376]
               name: rcu_sched PID: 7 State: 1
[14385.833378]
               name: rcu_bh PID: 8 State: 1
[14385.833380]
               name: migration/0 PID: 9 State: 1
[14385.833382]
               name: lru-add-drain PID: 10 State: 1
[14385.833383]
               name: watchdog/0 PID: 11 State: 1
[14385.833385]
               name: cpuhp/0 PID: 12 State: 1
[14385.833387]
               name: cpuhp/1 PID: 13 State: 1
[14385.833388]
               name: watchdog/1 PID: 14 State: 1
[14385.833390]
               name: migration/1 PID: 15 State: 1
[14385.833392]
               name: ksoftirqd/1 PID: 16 State: 1
[14385.833393]
               name: kworker/1:0H PID: 18 State: 1
```

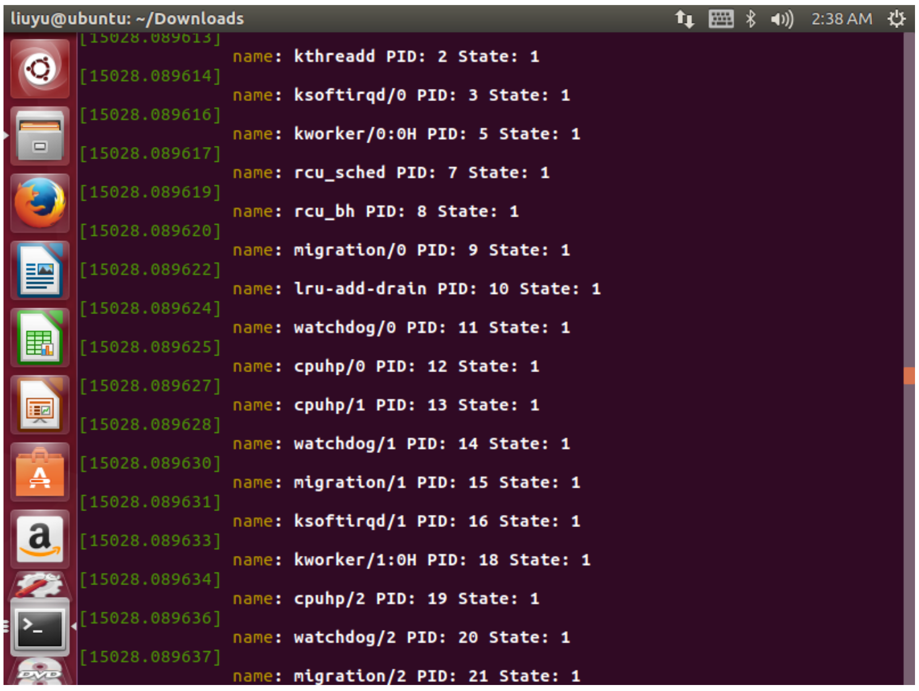
Then I use ps -el to compare results. And results are similar.



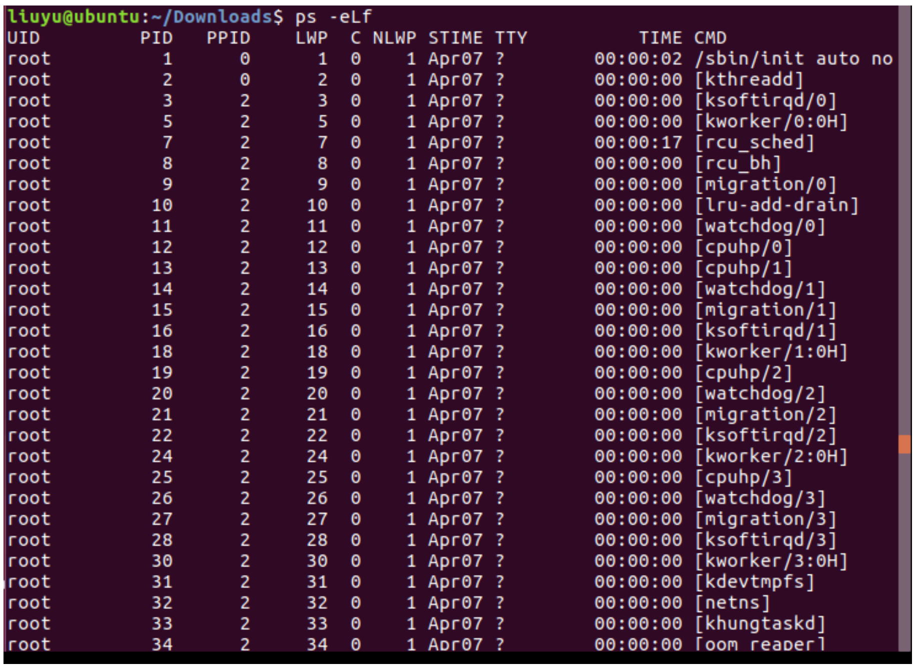
```
liuyu@ubuntu:~/Pictures$ ps -el
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0       1       0  0  80   0 - 50984 -      ?        00:00:02 systemd
1 S     0       2       0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 S     0       3       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/0
1 S     0       5       2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H
1 S     0       7       2  0  80   0 -     0 -      ?        00:00:16 rcu_sched
1 S     0       8       2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0       9       2  0 -40   - -     0 -      ?        00:00:00 migration/0
1 S     0      10       2  0  60 -20 -     0 -      ?        00:00:00 lru-add-drai
5 S     0      11       2  0 -40   - -     0 -      ?        00:00:00 watchdog/0
1 S     0      12       2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
1 S     0      13       2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
5 S     0      14       2  0 -40   - -     0 -      ?        00:00:00 watchdog/1
1 S     0      15       2  0 -40   - -     0 -      ?        00:00:00 migration/1
1 S     0      16       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/1
1 S     0      18       2  0  60 -20 -     0 -      ?        00:00:00 kworker/1:0H
1 S     0      19       2  0  80   0 -     0 -      ?        00:00:00 cpuhp/2
5 S     0      20       2  0 -40   - -     0 -      ?        00:00:00 watchdog/2
1 S     0      21       2  0 -40   - -     0 -      ?        00:00:00 migration/2
1 S     0      22       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/2
1 S     0      24       2  0  60 -20 -     0 -      ?        00:00:00 kworker/2:0H
1 S     0      25       2  0  80   0 -     0 -      ?        00:00:00 cpuhp/3
5 S     0      26       2  0 -40   - -     0 -      ?        00:00:00 watchdog/3
1 S     0      27       2  0 -40   - -     0 -      ?        00:00:00 migration/3
1 S     0      28       2  0  80   0 -     0 -      ?        00:00:00 ksoftirqd/3
1 S     0      30       2  0  60 -20 -     0 -      ?        00:00:00 kworker/3:0H
5 S     0      31       2  0  80   0 -     0 -      ?        00:00:00 kdevtmpfs
1 S     0      32       2  0  60 -20 -     0 -      ?        00:00:00 netns
1 S     0      33       2  0  80   0 -     0 -      ?        00:00:00 khungtaskd
1 S     0      34       2  0  80   0 -     0 -      ?        00:00:00 oom_reaper
1 S     0      35       2  0  60 -20 -     0 -      ?        00:00:00 writeback
1 S     0      36       2  0  80   0 -     0 -      ?        00:00:00 kcompactd0
```

Part II—Iterating over Tasks with a Depth-First Search Tree

I use recursion to traverse processes through DFS tree.


```
liuyu@ubuntu: ~/Downloads                                         2:38 AM
[15028.089613]
               name: kthreadd PID: 2 State: 1
[15028.089614]
               name: ksoftirqd/0 PID: 3 State: 1
[15028.089616]
               name: kworker/0:0H PID: 5 State: 1
[15028.089617]
               name: rcu_sched PID: 7 State: 1
[15028.089619]
               name: rcu_bh PID: 8 State: 1
[15028.089620]
               name: migration/0 PID: 9 State: 1
[15028.089622]
               name: lru-add-drain PID: 10 State: 1
[15028.089624]
               name: watchdog/0 PID: 11 State: 1
[15028.089625]
               name: cpuhp/0 PID: 12 State: 1
[15028.089627]
               name: cpuhp/1 PID: 13 State: 1
[15028.089628]
               name: watchdog/1 PID: 14 State: 1
[15028.089630]
               name: migration/1 PID: 15 State: 1
[15028.089631]
               name: ksoftirqd/1 PID: 16 State: 1
[15028.089633]
               name: kworker/1:0H PID: 18 State: 1
[15028.089634]
               name: cpuhp/2 PID: 19 State: 1
[15028.089636]
               name: watchdog/2 PID: 20 State: 1
[15028.089637]
               name: migration/2 PID: 21 State: 1
```

Then I use ps -eLf to compare results. And results are similar.


```
liuyu@ubuntu:~/Downloads$ ps -eLf
UID         PID   PPID    LWP  C NLWP STIME TTY          TIME CMD
root          1      0      1  0    1 Apr07 ?        00:00:02 /sbin/init auto no
root          2      0      2  0    1 Apr07 ?        00:00:00 [kthreadd]
root          3      2      3  0    1 Apr07 ?        00:00:00 [ksoftirqd/0]
root          5      2      5  0    1 Apr07 ?        00:00:00 [kworker/0:0H]
root          7      2      7  0    1 Apr07 ?        00:00:17 [rcu_sched]
root          8      2      8  0    1 Apr07 ?        00:00:00 [rcu_bh]
root          9      2      9  0    1 Apr07 ?        00:00:00 [migration/0]
root         10      2     10  0    1 Apr07 ?        00:00:00 [lru-add-drain]
root         11      2     11  0    1 Apr07 ?        00:00:00 [watchdog/0]
root         12      2     12  0    1 Apr07 ?        00:00:00 [cpuhp/0]
root         13      2     13  0    1 Apr07 ?        00:00:00 [cpuhp/1]
root         14      2     14  0    1 Apr07 ?        00:00:00 [watchdog/1]
root         15      2     15  0    1 Apr07 ?        00:00:00 [migration/1]
root         16      2     16  0    1 Apr07 ?        00:00:00 [ksoftirqd/1]
root         18      2     18  0    1 Apr07 ?        00:00:00 [kworker/1:0H]
root         19      2     19  0    1 Apr07 ?        00:00:00 [cpuhp/2]
root         20      2     20  0    1 Apr07 ?        00:00:00 [watchdog/2]
root         21      2     21  0    1 Apr07 ?        00:00:00 [migration/2]
root         22      2     22  0    1 Apr07 ?        00:00:00 [ksoftirqd/2]
root         24      2     24  0    1 Apr07 ?        00:00:00 [kworker/2:0H]
root         25      2     25  0    1 Apr07 ?        00:00:00 [cpuhp/3]
root         26      2     26  0    1 Apr07 ?        00:00:00 [watchdog/3]
root         27      2     27  0    1 Apr07 ?        00:00:00 [migration/3]
root         28      2     28  0    1 Apr07 ?        00:00:00 [ksoftirqd/3]
root         30      2     30  0    1 Apr07 ?        00:00:00 [kworker/3:0H]
root         31      2     31  0    1 Apr07 ?        00:00:00 [kdevtmpfs]
root         32      2     32  0    1 Apr07 ?        00:00:00 [netns]
root         33      2     33  0    1 Apr07 ?        00:00:00 [khungtaskd]
root         34      2     34  0    1 Apr07 ?        00:00:00 [oom_reaper]
```