Test alsoNice()

For testing and debug, **procdump()** in proc.c is modified. Besides printing the processes listing, pocdump will also print last 32 timer ticks and corresponding running process history in that tick. Any time, hit **Ctrl+P**, the procdump will be called and print processes history.

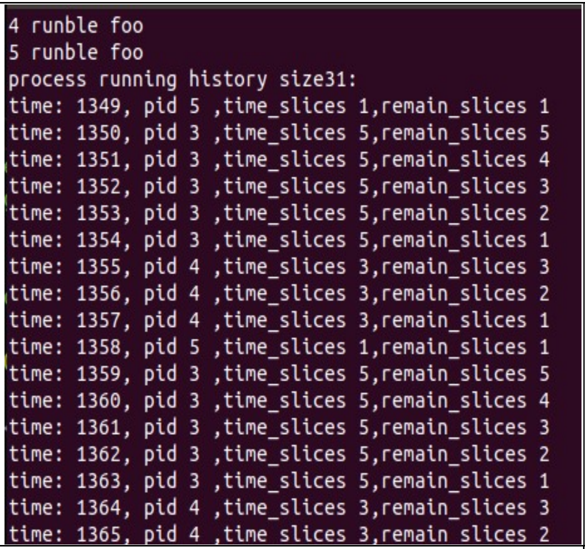
In xv6 shell, run the test program foo().
What Foo.c doing is given in the below chart.

Main process	Child process	Grand child process

alsoNice(5)
fork()
 	
Long for loop; 5 time slices
	
	starting
	alsoNice(3)
	fork()
	Long for loop; 3 time slices
		
		Starting
		Long for loop; 1 time slices

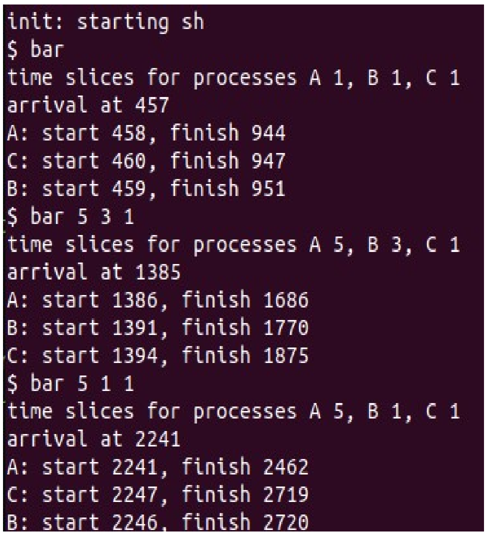
The picture on left, is a screenshot of **procdump** output. Here pid 3 is the main process above, and pid 4 is the child process, pid 5 is the grand child process. In the test we can easily see how many time slices the process has and how many slices remains. The first column is the timer ticks. After we changed the time slices, the scheduler works like,

A A A A A B B B C



```
4 runble foo
5 runble foo
process running history size31:
time: 1349, pid 5 ,time_slices 1,remain_slices 1
time: 1350, pid 3 ,time_slices 5,remain_slices 5
time: 1351, pid 3 ,time_slices 5,remain_slices 4
time: 1352, pid 3 ,time_slices 5,remain_slices 3
time: 1353, pid 3 ,time_slices 5,remain_slices 2
time: 1354, pid 3 ,time_slices 5,remain_slices 1
time: 1355, pid 4 ,time_slices 3,remain_slices 3
time: 1356, pid 4 ,time_slices 3,remain_slices 2
time: 1357, pid 4 ,time_slices 3,remain_slices 1
time: 1358, pid 5 ,time_slices 1,remain_slices 1
time: 1359, pid 3 ,time_slices 5,remain_slices 5
time: 1360, pid 3 ,time_slices 5,remain_slices 4
time: 1361, pid 3 ,time_slices 5,remain_slices 3
time: 1362, pid 3 ,time_slices 5,remain_slices 2
time: 1363, pid 3 ,time_slices 5,remain_slices 1
time: 1364, pid 4 ,time_slices 3,remain_slices 3
time: 1365, pid 4 ,time_slices 3,remain_slices 2
```

Test program **foo**, hit Ctrl+P to output



```
init: starting sh
$ bar
time slices for processes A 1, B 1, C 1
arrival at 457
A: start 458, finish 944
C: start 460, finish 947
B: start 459, finish 951
$ bar 5 3 1
time slices for processes A 5, B 3, C 1
arrival at 1385
A: start 1386, finish 1686
B: start 1391, finish 1770
C: start 1394, finish 1875
$ bar 5 1 1
time slices for processes A 5, B 1, C 1
arrival at 2241
A: start 2241, finish 2462
C: start 2247, finish 2719
B: start 2246, finish 2720
```

Test program **bar**

There is another program **bar** showing how alsoNice() will change average turnaround time and response time. Like **foo** we still use a long loop, but this time it can be finished in few seconds. Three processes will start one by one. If we call **bar(5,3,1)**, then three processes will have 5,3,1 timeslices. The picture on right above show 3 test cases.

Process	A	B	C
Test 1 arraive:457	Start: 458,end:944	Start: 459,end:951	Start: 460,end:947
Test 2 arrive: 1385	Start: 1386,end:1686	Start: 1391,end:1770	Start: 1394,end:1875
Test 3 arrive: 2241	Start: 2241,end:2462	Start: 2246,end:2720	Start: 2247,end:2719

(the time here is the value of **ticks** in xv6)

Let’s say processes A,B,C have same amount of work.
The process with bigger nice value will finish earlier, yet the time to finish all job won’t change. The last processes still finish at the same time. So the average turnaround time should decrease.
For average response time, it should increase after we change the times slices. Because, the first job comes still at the same time and its response time won’t change, yet the following job will start later, so the average response time should increase.

Below is the test result for **bar**

Turnaround Time	A	B	C	Average
Test1(1,1,1)	487	494	490	490.3
Test2(5,3,1)	301	385	490	392
Test3(5,1,1)	221	479	478	393

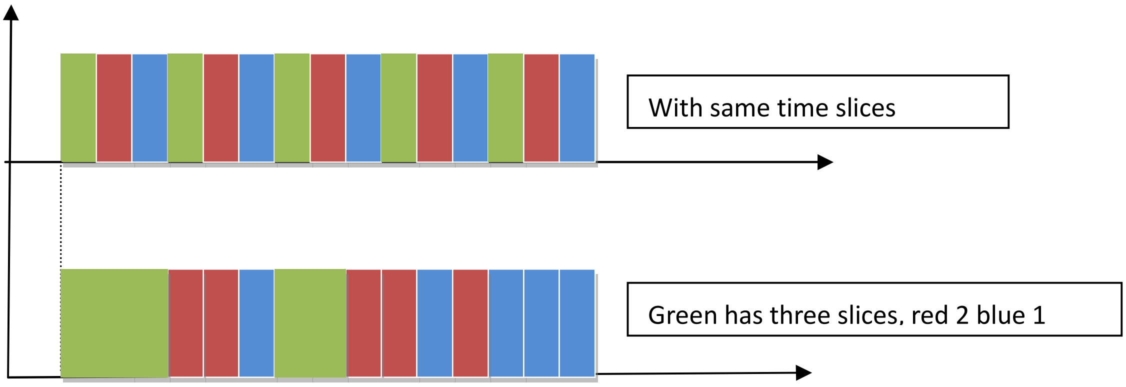
Average Turnaround Time

We can see if we increase times lices of one process, the average turnaround time will decreased.

Response Time	A	B	C	Average
Test1(1,1,1)	1	2	3	2
Test2(5,3,1)	1	6	10	5.67
Test3(5,1,1)	0	5	6	3.67

Average Response Time

After we increase the time slices of processes, we can see that average response time increased.


With same time slices
Green has three slices, red 2 blue 1